

# Hollywood 10.0

---

The Cross-Platform Multimedia Application Layer

Andreas Falkenhahn

---



# Inhaltsverzeichnis

<b>1</b>	<b>Allgemeines .....</b>	<b>1</b>
1.1	Einführung .....	1
1.2	Philosophie .....	4
1.3	Lizenzvereinbarung .....	4
1.4	Anforderungen .....	6
1.5	Danksagungen .....	7
1.6	Forum .....	8
1.7	Kontakt .....	8
<b>2</b>	<b>Erste Schritte .....</b>	<b>9</b>
2.1	Übersicht übers Hollywoodpaket .....	9
2.2	Die grafische Benutzeroberfläche (GUI) .....	10
2.3	Windows IDE .....	15
2.4	Mobile Plattformen .....	23
<b>3</b>	<b>Konsole benutzen .....</b>	<b>31</b>
3.1	Konsolenmodus .....	31
3.2	Konsolenargumente .....	33
3.3	Konsolenargumente ohne Konsole .....	55
<b>4</b>	<b>Compiler und Linker .....</b>	<b>57</b>
4.1	Kompilieren von Programmen .....	57
4.2	Kompilieren von Applets .....	58
4.3	Einbinden von Datendateien .....	58
4.4	Einbinden von Schriftarten .....	60
4.5	Einbinden von Plugins .....	62
4.6	Verwendung des Videorecorders .....	63
<b>5</b>	<b>Plugins .....</b>	<b>67</b>
5.1	Plugins .....	67
5.2	Installation .....	67
5.3	Anwendung .....	67
5.4	Vorhandene Plugins .....	68
5.5	Schreiben eigener Plugins .....	70
<b>6</b>	<b>Geschichte und Kompatibilität .....</b>	<b>73</b>
6.1	Geschichte .....	73
6.2	Hinweise zur Kompatibilität/API Änderungen .....	73
6.3	Zukunft .....	85

<b>7</b>	<b>Aufbau der Sprache</b>	<b>87</b>
7.1	Ihr erstes Hollywood Programm	87
7.2	Reservierte Bezeichnungen	89
7.3	Präprozessor-Anweisungen	90
7.4	Zeichenketten- und Zahlenumwandlung	91
7.5	Kommentare	92
7.6	Code einbinden/Includes	92
7.7	Fehlerbehandlung	94
7.8	Automatische ID-Zuweisung	96
7.9	Lade- und Adaptermodule	96
7.10	Benutzer-Tags	100
7.11	Gestaltungsrichtlinien (Styleguide)	101
<b>8</b>	<b>Datentypen</b>	<b>103</b>
8.1	Datentypen Überblick	103
8.2	Zahlen	103
8.3	Zeichenketten	104
8.4	Tabellen	106
8.5	Funktionen	109
8.6	Nil	110
<b>9</b>	<b>Ausdrücke und Operatoren</b>	<b>111</b>
9.1	Übersicht	111
9.2	Arithmetische Operatoren / Rechenzeichen	111
9.3	Vergleichsoperatoren	112
9.4	Logische Operatoren	113
9.5	Bit-Operatoren	114
9.6	Zeichenketten verketteten	115
9.7	Prioritäten der Operatoren	116
9.8	Metamethoden	117
9.8.1	Abweichende Metatabellen mit Binäroperatoren	120
9.8.2	Einschränkungen der relationalen Metamethoden	120
9.8.3	Erweiterte Metamethoden	120
<b>10</b>	<b>Variablen und Konstanten</b>	<b>123</b>
10.1	Variablen und Konstanten	123
10.2	Globale Variablen	123
10.3	Global-Anweisung	124
10.4	Lokale Variablen	124
10.5	Local-Anweisung	127
10.6	Speicherbereiniger/Garbage Collector	128
10.7	Konstanten	128
10.8	Const-Anweisung	129
10.9	Integrierte Konstanten	129
10.10	Zeichenkonstanten	130

<b>11</b>	<b>Kontrollstrukturen</b>	<b>133</b>
11.1	Skriptablaufkontrolle	133
11.2	If-EndIf-Anweisung	133
11.3	While-Wend-Anweisung	134
11.4	For-Next-Anweisung	135
11.5	Repeat-Until-Anweisung	138
11.6	Switch-Case-Anweisung	138
11.7	Break-Anweisung	140
11.8	Continue-Anweisung	141
11.9	Return-Anweisung	141
11.10	Block-EndBlock-Anweisung	142
11.11	Dim- und DimStr-Anweisung	142
<b>12</b>	<b>Funktionen</b>	<b>145</b>
12.1	Übersicht	145
12.2	Funktionen sind Variablen	146
12.3	Callback-Funktionen	147
12.4	Rückgabewerte (Return)	149
12.5	Rekursive Funktionen	151
12.6	Variable Anzahl von Argumenten (...)	151
12.7	Funktionen in Tabellenelemente	152
12.8	Lokale Funktionen	153
12.9	Methoden	154
<b>13</b>	<b>Unicode-Unterstützung</b>	<b>157</b>
13.1	Übersicht	157
13.2	Zeichencodierungen	158
<b>14</b>	<b>Fehlerbehebung</b>	<b>159</b>
14.1	Fehlerbehebung	159
14.2	Häufig gestellte Fragen (FAQ)	160
<b>15</b>	<b>Lernprogramme</b>	<b>165</b>
15.1	Tutorial	165
15.2	Animationstechniken	168
15.3	Zeitsteuerung des Skripts	170

## **16 Amiga-Bibliothek ..... 173**

16.1	Informationen über AmiDock.....	173
16.2	CloseAmigaGuide.....	173
16.3	CreateRexxPort.....	174
16.4	GetApplicationList.....	176
16.5	GetFrontScreen.....	176
16.6	GetPubScreens.....	177
16.7	HideScreen.....	178
16.8	OpenAmigaGuide.....	178
16.9	RunRexxScript.....	179
16.10	SendApplicationMessage.....	180
16.11	SendRexxCommand.....	181
16.12	SetScreenTitle.....	182
16.13	ShowRinghioMessage.....	182
16.14	ShowScreen.....	183

## **17 Animationsbibliothek ..... 185**

17.1	Übersicht.....	185
17.2	ANIM.....	185
17.3	BeginAnimStream.....	189
17.4	CloseAnim.....	191
17.5	CopyAnim.....	192
17.6	CreateAnim.....	192
17.7	DisplayAnimFrame.....	193
17.8	FinishAnimStream.....	194
17.9	FreeAnim.....	194
17.10	GetAnimFrame.....	195
17.11	IsAnim.....	195
17.12	IsAnimPlaying.....	196
17.13	LoadAnim.....	197
17.14	LoadAnimFrame.....	200
17.15	ModifyAnimFrames.....	202
17.16	MoveAnim.....	203
17.17	OpenAnim.....	203
17.18	PlayAnim.....	206
17.19	PlayAnimDisk.....	207
17.20	SaveAnim.....	208
17.21	ScaleAnim.....	211
17.22	SelectAnim.....	212
17.23	SetAnimFrameDelay.....	213
17.24	StopAnim.....	214
17.25	Vektoranimationen.....	214
17.26	WaitAnimEnd.....	214
17.27	WriteAnimFrame.....	215

<b>18</b>	<b>Anwendungsbibliothek</b>	<b>217</b>
18.1	APPAUTHOR	217
18.2	APPCOPYRIGHT	217
18.3	APPDESCRIPTION	218
18.4	APPENTRY	218
18.5	APPICON	219
18.6	APPIDENTIFIER	222
18.7	APPTITLE	222
18.8	APPVERSION	223
18.9	DeletePrefs	224
18.10	GetApplicationInfo	224
18.11	GetCommandLine	225
18.12	GetFileArgument	226
18.13	GetProgramInfo	227
18.14	GetRawArguments	227
18.15	LoadPrefs	228
18.16	SavePrefs	229
<b>19</b>	<b>Asynchrone Operationsbibliothek</b>	<b>231</b>
19.1	AsyncDrawFrame	231
19.2	CancelAsyncDraw	233
19.3	CancelAsyncOperation	233
19.4	ContinueAsyncOperation	234
19.5	FinishAsyncDraw	235
<b>20</b>	<b>Dateisystembibliothek (DOS)</b>	<b>237</b>
20.1	CanonizePath	237
20.2	ChangeDirectory	237
20.3	CloseDirectory	238
20.4	CloseFile	239
20.5	CompressFile	239
20.6	CopyFile	240
20.7	CountDirectoryEntries	245
20.8	CRC32	247
20.9	DecompressFile	247
20.10	DefineVirtualFile	248
20.11	DefineVirtualFileFromString	249
20.12	DeleteFile	250
20.13	DIRECTORY	254
20.14	DirectoryItems	257
20.15	Eof	258
20.16	Execute	259
20.17	Exists	261
20.18	FILE	262
20.19	FileAttributes	263

20.20	FileLength	264
20.21	FileLines	265
20.22	FilePart	266
20.23	FilePos	266
20.24	FileSize	267
20.25	FileToString	267
20.26	FlushFile	268
20.27	FullPath	269
20.28	GetCurrentDirectory	269
20.29	GetDirectoryEntry	270
20.30	GetEnv	271
20.31	GetFileAttributes	272
20.32	GetProgramDirectory	273
20.33	GetStartDirectory	274
20.34	GetTempFileName	274
20.35	GetVolumeInfo	275
20.36	GetVolumeName	276
20.37	HaveVolume	276
20.38	IsAbsolutePath	277
20.39	IsDirectory	277
20.40	MakeDirectory	278
20.41	MakeHostPath	279
20.42	MatchPattern	280
20.43	MD5	281
20.44	MonitorDirectory	281
20.45	MoveFile	283
20.46	NextDirectoryEntry	288
20.47	OpenDirectory	289
20.48	OpenFile	290
20.49	PathPart	292
20.50	ReadByte	293
20.51	ReadBytes	293
20.52	ReadChr	294
20.53	ReadDirectory	295
20.54	ReadFloat	295
20.55	ReadFunction	296
20.56	ReadInt	297
20.57	ReadLine	298
20.58	ReadShort	298
20.59	ReadString	299
20.60	Rename	300
20.61	RewindDirectory	301
20.62	Run	301
20.63	Schutzbits-Informationen	305
20.64	Seek	306
20.65	SetEnv	307
20.66	SetFileAttributes	307



20.67	SetFileEncoding.....	308
20.68	SetIOMode .....	309
20.69	StringToFile .....	310
20.70	UndefineVirtualStringFile .....	311
20.71	UnsetEnv .....	311
20.72	UseCarriageReturn.....	311
20.73	WriteByte .....	312
20.74	WriteBytes.....	313
20.75	WriteChr .....	313
20.76	WriteFloat .....	314
20.77	WriteFunction .....	315
20.78	WriteInt .....	316
20.79	WriteLine.....	317
20.80	WriteShort.....	317
20.81	WriteString.....	318
<b>21</b>	<b>Datum- und Zeitbibliothek .....</b>	<b>321</b>
21.1	CompareDates .....	321
21.2	DateToTimestamp .....	322
21.3	DateToUTC .....	323
21.4	GetDate .....	323
21.5	GetDateNum.....	324
21.6	GetTime .....	325
21.7	GetTimer .....	325
21.8	GetTimestamp .....	326
21.9	GetTimeZone .....	327
21.10	GetWeekday .....	328
21.11	MakeDate.....	328
21.12	ParseDate .....	329
21.13	PauseTimer.....	329
21.14	ResetTimer .....	330
21.15	ResumeTimer .....	331
21.16	SetTimerElapse .....	331
21.17	StartTimer.....	332
21.18	StopTimer .....	332
21.19	TimerElapsed .....	333
21.20	TimestampToDate .....	334
21.21	UTCToDate .....	334
21.22	ValidateDate.....	335
21.23	WaitTimer.....	335

<b>22</b>	<b>Debugbibliothek</b>	<b>337</b>
22.1	Assert	337
22.2	CloseResourceMonitor	337
22.3	DebugOutput	338
22.4	DebugPrint	338
22.5	DebugPrintNR	339
22.6	DebugPrompt	340
22.7	OpenResourceMonitor	340
22.8	WARNING	341
<b>23</b>	<b>Dialogbibliothek</b>	<b>343</b>
23.1	ColorRequest	343
23.2	FileRequest	344
23.3	FontRequest	346
23.4	ImageRequest	349
23.5	ListRequest	350
23.6	PathRequest	351
23.7	PermissionRequest	353
23.8	StringRequest	354
23.9	SystemRequest	356
<b>24</b>	<b>Displaybibliothek</b>	<b>359</b>
24.1	Übersicht	359
24.2	ActivateDisplay	360
24.3	BACKFILL	360
24.4	ChangeDisplayMode	362
24.5	ChangeDisplaySize	364
24.6	CloseDisplay	365
24.7	CreateDisplay	366
24.8	DISPLAY	370
24.9	FreeDisplay	387
24.10	GetDisplayModes	387
24.11	GetMonitors	388
24.12	HideDisplay	389
24.13	Mehrere Monitore	389
24.14	MoveDisplay	390
24.15	OpenDisplay	390
24.16	Palettenmodus-Displays	393
24.17	RefreshDisplay	394
24.18	SCREEN	394
24.19	SelectDisplay	396
24.20	SetDisplayAttributes	397
24.21	SetSubtitle	400
24.22	SetTitle	401
24.23	ShowDisplay	401
24.24	Skalierungssysteme	402

<b>25</b>	<b>Ebenenbibliothek</b>	<b>405</b>
25.1	Übersicht	405
25.2	AddMove	407
25.3	ClearMove	409
25.4	CopyLayer	409
25.5	CreateLayer	410
25.6	DisableLayers	412
25.7	DoMove	412
25.8	DumpLayers	413
25.9	EnableLayers	414
25.10	FreeLayers	415
25.11	GetLayerAtPos	415
25.12	GetLayerGroupMembers	416
25.13	GetLayerGroups	417
25.14	GetLayerPen	417
25.15	GetLayerStyle	418
25.16	GroupLayer	418
25.17	HideLayer	420
25.18	HideLayerFX	420
25.19	InsertLayer	422
25.20	LayerExists	423
25.21	LayerGroupExists	423
25.22	LayerToBack	424
25.23	LayerToFront	424
25.24	MergeLayers	425
25.25	ModifyLayerFrames	427
25.26	MoveLayer	428
25.27	NextFrame	429
25.28	PauseLayer	430
25.29	PlayLayer	430
25.30	RefreshLayer	430
25.31	RemoveLayer	431
25.32	RemoveLayerFX	431
25.33	RemoveLayers	433
25.34	RenderLayer	433
25.35	ResumeLayer	433
25.36	RotateLayer	434
25.37	ScaleLayer	434
25.38	SeekLayer	435
25.39	SelectLayer	436
25.40	SetLayerAnchor	438
25.41	SetLayerBorder	439
25.42	SetLayerDepth	440
25.43	SetLayerFilter	441
25.44	SetLayerName	446
25.45	SetLayerPalette	447

25.46	SetLayerPen .....	448
25.47	SetLayerShadow .....	448
25.48	SetLayerStyle .....	449
25.49	SetLayerTint .....	463
25.50	SetLayerTransparency .....	464
25.51	SetLayerTransparentPen .....	465
25.52	SetLayerVolume .....	465
25.53	SetLayerZPos .....	466
25.54	ShowLayer .....	466
25.55	ShowLayerFX .....	467
25.56	StopLayer .....	468
25.57	SwapLayers .....	469
25.58	TransformLayer .....	469
25.59	TranslateLayer .....	470
25.60	Undo .....	471
25.61	UndoFX .....	473
25.62	UngroupLayer .....	474
<b>26</b>	<b>Ereignisbibliothek .....</b>	<b>477</b>
26.1	BreakEventHandler .....	477
26.2	ChangeInterval .....	477
26.3	CheckEvent .....	477
26.4	CheckEvents .....	479
26.5	ClearInterval .....	479
26.6	ClearTimeout .....	479
26.7	CtrlCQuit .....	480
26.8	DeleteButton .....	480
26.9	DisableButton .....	481
26.10	EnableButton .....	481
26.11	EscapeQuit .....	481
26.12	InKeyStr .....	482
26.13	InstallEventHandler .....	483
26.14	IsKeyDown .....	503
26.15	IsLeftMouse .....	505
26.16	IsMidMouse .....	505
26.17	IsRightMouse .....	506
26.18	LeftMouseQuit .....	506
26.19	MakeButton .....	507
26.20	MouseX .....	511
26.21	MouseY .....	512
26.22	ResetKeyStates .....	512
26.23	Rohe Tasten (Raw Keys) .....	512
26.24	RunCallback .....	514
26.25	SetEventTimeout .....	514
26.26	SetInterval .....	515
26.27	SetTimeout .....	517

26.28	WaitEvent .....	518
26.29	WaitKeyDown .....	520
26.30	WaitLeftMouse .....	521
26.31	WaitMidMouse .....	521
26.32	WaitRightMouse .....	522
<b>27</b>	<b>Fehlerbehandlungsbibliothek .....</b>	<b>523</b>
27.1	ERROR .....	523
27.2	Error .....	523
27.3	ExitOnError .....	523
27.4	Fehlercodes .....	524
27.5	GetErrorName .....	563
27.6	GetLastError .....	563
27.7	RaiseOnError .....	564
<b>28</b>	<b>Grafikbibliothek .....</b>	<b>567</b>
28.1	ARGB .....	567
28.2	ARGB-Farben .....	567
28.3	BeginDoubleBuffer .....	568
28.4	BeginRefresh .....	569
28.5	Blue .....	571
28.6	ClearScreen .....	571
28.7	Collision .....	573
28.8	CreateClipRegion .....	574
28.9	DisablePrecalculation .....	575
28.10	DisableVWait .....	576
28.11	EnablePrecalculation .....	576
28.12	EnableVWait .....	577
28.13	EndDoubleBuffer .....	577
28.14	EndRefresh .....	577
28.15	Flip .....	578
28.16	FreeClipRegion .....	578
28.17	GetFPSLimit .....	579
28.18	GetRandomColor .....	579
28.19	GetRandomFX .....	580
28.20	GetRealColor .....	581
28.21	GrabDesktop .....	581
28.22	Green .....	582
28.23	Intersection .....	583
28.24	IsPicture .....	584
28.25	Matrix2D .....	586
28.26	MixRGB .....	586
28.27	Red .....	587
28.28	RGB .....	588
28.29	RGB-Farben .....	588
28.30	SaveSnapshot .....	589

28.31	SetClipRegion .....	592
28.32	SetDrawTagsDefault .....	593
28.33	SetFPSLimit .....	594
28.34	TransformBox .....	594
28.35	TransformPoint .....	595
28.36	VWait .....	596
<b>29</b>	<b>Grafikgrundelemente .....</b>	<b>597</b>
29.1	Arc .....	597
29.2	Box .....	598
29.3	Circle .....	600
29.4	Cls .....	601
29.5	Ellipse .....	601
29.6	GetFillStyle .....	602
29.7	GetFormStyle .....	604
29.8	GetLineWidth .....	604
29.9	Line .....	605
29.10	Plot .....	606
29.11	Polygon .....	607
29.12	ReadPixel .....	608
29.13	Richtungskonstanten .....	609
29.14	SetFillStyle .....	610
29.15	SetFormStyle .....	611
29.16	SetLineWidth .....	613
29.17	Standard-Tags zum Zeichnen .....	613
<b>30</b>	<b>Hintergrundbildbefehle .....</b>	<b>619</b>
30.1	Überblick .....	619
30.2	BGPIC .....	619
30.3	BrushToBGPic .....	622
30.4	CopyBGPic .....	622
30.5	CreateBGPic .....	623
30.6	CreateGradientBGPic .....	624
30.7	CreateTexturedBGPic .....	626
30.8	DisplayBGPic .....	627
30.9	DisplayBGPicPart .....	628
30.10	DisplayBGPicPartFX .....	629
30.11	DisplayTransitionFX .....	630
30.12	FreeBGPic .....	634
30.13	LoadBGPic .....	634
30.14	ScaleBGPic .....	638
30.15	SelectBGPic .....	639
30.16	Vektor-BGPic .....	642

<b>31</b>	<b>IPC-Bibliothek</b>	<b>643</b>
31.1	CreatePort	643
31.2	SendMessage	644
<b>32</b>	<b>Joystick-Bibliothek</b>	<b>647</b>
32.1	ConfigureJoystick	647
32.2	CountJoysticks	647
32.3	JoyAxisX	648
32.4	JoyAxisY	648
32.5	JoyAxisZ	649
32.6	JoyButton	649
32.7	JoyDir	650
32.8	JoyHat	652
<b>33</b>	<b>Konsolenbibliothek</b>	<b>653</b>
33.1	AllocConsoleColor	653
33.2	BeepConsole	653
33.3	ClearConsole	654
33.4	ClearConsoleStyle	654
33.5	CloseConsole	655
33.6	ConsolePrint	655
33.7	ConsolePrintNR	656
33.8	ConsolePrompt	656
33.9	CopyConsoleWindow	657
33.10	CreateConsoleWindow	658
33.11	DecomposeConsoleChr	659
33.12	DeleteConsoleChr	660
33.13	DeleteConsoleLine	660
33.14	DisableAdvancedConsole	660
33.15	DrawConsoleBorder	661
33.16	DrawConsoleBox	662
33.17	DrawConsoleHLine	663
33.18	DrawConsoleVLine	664
33.19	EnableAdvancedConsole	664
33.20	EraseConsole	665
33.21	FlashConsole	666
33.22	FormatConsoleLine	666
33.23	FreeConsoleColor	667
33.24	FreeConsoleWindow	667
33.25	GetAllocConsoleColor	668
33.26	GetConsoleBackground	668
33.27	GetConsoleChr	669
33.28	GetConsoleColor	669
33.29	GetConsoleControlChr	670
33.30	GetConsoleCursor	671

33.31	GetConsoleOrigin .....	671
33.32	GetConsoleSize .....	672
33.33	GetConsoleStr .....	672
33.34	GetConsoleStyle .....	673
33.35	GetConsoleWindow .....	673
33.36	HaveConsole .....	674
33.37	HideConsoleCursor .....	674
33.38	InitConsoleColor .....	675
33.39	InsertConsoleChr .....	676
33.40	InsertConsoleLine .....	677
33.41	InsertConsoleStr .....	677
33.42	MakeConsoleChr .....	678
33.43	MoveConsoleWindow .....	680
33.44	OpenConsole .....	681
33.45	ReadConsoleKey .....	682
33.46	ReadConsoleStr .....	683
33.47	RefreshConsole .....	684
33.48	ScrollConsole .....	684
33.49	SelectConsoleWindow .....	685
33.50	SetAllocConsoleColor .....	686
33.51	SetConsoleBackground .....	686
33.52	SetConsoleColor .....	687
33.53	SetConsoleCursor .....	688
33.54	SetConsoleOptions .....	689
33.55	SetConsoleStyle .....	690
33.56	SetConsoleTitle .....	692
33.57	ShowConsoleCursor .....	692
33.58	StartConsoleColorMode .....	692
33.59	TouchConsoleWindow .....	693

## **34 Lokalisierungsbibliothek..... 695**

34.1	Übersicht .....	695
34.2	CATALOG .....	696
34.3	CloseCatalog .....	697
34.4	FormatDate .....	697
34.5	GetCatalogString .....	699
34.6	GetCountryInfo .....	700
34.7	GetLanguageInfo .....	700
34.8	GetLocaleInfo .....	701
34.9	GetSystemCountry .....	702
34.10	GetSystemLanguage .....	707
34.11	OpenCatalog .....	712



<b>35</b>	<b>Mathebibliothek</b>	<b>715</b>
35.1	Abs	715
35.2	ACos	715
35.3	Add	716
35.4	ASin	716
35.5	ATan	717
35.6	ATan2	717
35.7	BitClear	717
35.8	BitComplement	718
35.9	BitSet	718
35.10	BitTest	719
35.11	BitXor	719
35.12	Cast	720
35.13	Ceil	721
35.14	Cos	721
35.15	Deg	721
35.16	Div	722
35.17	EndianSwap	722
35.18	Exp	723
35.19	Floor	723
35.20	Frac	724
35.21	FrExp	724
35.22	Hypot	724
35.23	Int	725
35.24	IsFinite	725
35.25	IsInf	726
35.26	IsNan	727
35.27	Ld	727
35.28	LdExp	728
35.29	Limit	728
35.30	Ln	729
35.31	Log	729
35.32	Max	730
35.33	Min	730
35.34	Mod	731
35.35	Mul	731
35.36	NearlyEqual	732
35.37	Pi	732
35.38	Pow	733
35.39	Rad	733
35.40	RawDiv	734
35.41	Rnd	734
35.42	RndF	735
35.43	RndStrong	735
35.44	Rol	736
35.45	Ror	737

35.46	Round .....	737
35.47	Rt .....	738
35.48	Sar .....	738
35.49	Sgn .....	739
35.50	Shl .....	740
35.51	Shr .....	740
35.52	Sin .....	741
35.53	Sqrt .....	741
35.54	Sub .....	742
35.55	Tan .....	742
35.56	Wrap .....	743
<b>36</b>	<b>Mauszeigerbibliothek .....</b>	<b>745</b>
36.1	CreatePointer .....	745
36.2	FreePointer .....	746
36.3	HidePointer .....	746
36.4	MovePointer .....	747
36.5	SetPointer .....	747
36.6	ShowPointer .....	748
<b>37</b>	<b>Menübibliothek .....</b>	<b>749</b>
37.1	CreateMenu .....	749
37.2	DeselectMenuItem .....	750
37.3	DisableMenuItem .....	751
37.4	EnableMenuItem .....	751
37.5	FreeMenu .....	752
37.6	IsMenuItemDisabled .....	753
37.7	IsMenuItemSelected .....	753
37.8	MENU .....	754
37.9	PopupMenu .....	758
37.10	SelectMenuItem .....	759
<b>38</b>	<b>Mobile-Geräte-Bibliothek .....</b>	<b>761</b>
38.1	CallJavaMethod .....	761
38.2	GetAsset .....	763
38.3	HideKeyboard .....	764
38.4	PerformSelector .....	764
38.5	ShowKeyboard .....	766
38.6	ShowToast .....	766
38.7	Vibrate .....	767

## **39    Netzwerkbibliothek ..... 769**

39.1	CloseConnection .....	769
39.2	CloseServer .....	769
39.3	CloseUDPObject .....	769
39.4	CreateServer .....	770
39.5	CreateUDPObject .....	771
39.6	DownloadFile .....	773
39.7	GetConnectionIP .....	779
39.8	GetConnectionPort .....	780
39.9	GetConnectionProtocol .....	781
39.10	GetHostName .....	782
39.11	GetLocalInterfaces .....	783
39.12	GetLocalIP .....	784
39.13	GetLocalPort .....	784
39.14	GetLocalProtocol .....	785
39.15	GetMACAddress .....	786
39.16	IsOnline .....	787
39.17	OpenConnection .....	787
39.18	ReceiveData .....	789
39.19	ReceiveUDPData .....	792
39.20	ResolveHostName .....	792
39.21	SendData .....	793
39.22	SendUDPData .....	794
39.23	SetNetworkProtocol .....	795
39.24	SetNetworkTimeout .....	796
39.25	ToHostName .....	796
39.26	ToIP .....	797
39.27	UploadFile .....	797

## **40    Objektbibliothek ..... 805**

40.1	Übersicht .....	805
40.2	ClearObjectData .....	807
40.3	CopyObjectData .....	807
40.4	GetAttribute .....	808
40.5	GetObjectData .....	836
40.6	GetObjects .....	836
40.7	GetObjectType .....	837
40.8	HaveObject .....	837
40.9	HaveObjectData .....	838
40.10	SetObjectData .....	839

<b>41</b>	<b>Palettenbibliothek</b>	<b>841</b>
41.1	Übersicht	841
41.2	ContrastPalette	842
41.3	CopyPalette	842
41.4	CopyPens	843
41.5	CreatePalette	844
41.6	CyclePalette	846
41.7	ExtractPalette	847
41.8	FreePalette	848
41.9	GammaPalette	849
41.10	GetBestPen	849
41.11	GetFreePen	850
41.12	GetPalettePen	850
41.13	GetPen	851
41.14	InvertPalette	852
41.15	LoadPalette	852
41.16	ModulatePalette	854
41.17	PALETTE	854
41.18	PaletteToGray	856
41.19	ReadPen	857
41.20	SavePalette	858
41.21	SelectPalette	859
41.22	SetBorderPen	860
41.23	SetBulletPen	860
41.24	SetCycleTable	861
41.25	SetDepth	862
41.26	SetDitherMode	863
41.27	SetDrawPen	864
41.28	SetGradientPalette	865
41.29	SetPalette	866
41.30	SetPaletteDepth	868
41.31	SetPaletteMode	868
41.32	SetPalettePen	869
41.33	SetPaletteTransparentPen	870
41.34	SetPen	870
41.35	SetShadowPen	872
41.36	SetStandardPalette	872
41.37	SetTransparentPen	874
41.38	SetTransparentThreshold	875
41.39	SolarizePalette	875
41.40	TintPalette	876
41.41	WritePen	876

## 42 Piktogramm-Bibliothek ..... 879

42.1	AddIconImage .....	879
42.2	ChangeApplicationIcon .....	880
42.3	CreateIcon .....	881
42.4	FreeIcon .....	884
42.5	GetIconProperties .....	885
42.6	ICON .....	888
42.7	LoadIcon .....	892
42.8	RemoveIconImage .....	894
42.9	SaveIcon .....	895
42.10	SetIconProperties .....	896
42.11	SetStandardIconImage .....	899
42.12	SetTrayIcon .....	900
42.13	SetWBIcon .....	901

## 43 Pinselbibliothek ..... 905

43.1	Überblick .....	905
43.2	ArcDistortBrush .....	905
43.3	BarrelDistortBrush .....	906
43.4	BGPicToBrush .....	907
43.5	BlurBrush .....	908
43.6	BRUSH .....	908
43.7	BrushToGray .....	911
43.8	BrushToMonochrome .....	912
43.9	BrushToPenArray .....	913
43.10	BrushToRGBArray .....	913
43.11	ChangeBrushTransparency .....	914
43.12	CharcoalBrush .....	915
43.13	ContrastBrush .....	916
43.14	ConvertToBrush .....	916
43.15	CopyBrush .....	919
43.16	CreateBorderBrush .....	920
43.17	CreateBrush .....	921
43.18	CreateGradientBrush .....	925
43.19	CreateShadowBrush .....	927
43.20	CreateTexturedBrush .....	928
43.21	CropBrush .....	928
43.22	DeleteAlphaChannel .....	929
43.23	DeleteMask .....	930
43.24	DisplayBrush .....	930
43.25	DisplayBrushFX .....	931
43.26	DisplayBrushPart .....	932
43.27	EdgeBrush .....	933
43.28	EmbossBrush .....	933
43.29	EndSelect .....	934
43.30	ExtendBrush .....	935

43.31	FlipBrush	936
43.32	FloodFill	936
43.33	FreeBrush	937
43.34	GammaBrush	938
43.35	GetBrushLink	939
43.36	GetBrushPen	940
43.37	Hardware-Pinsel	940
43.38	InvertAlphaChannel	941
43.39	InvertBrush	942
43.40	InvertMask	942
43.41	IsBrushEmpty	943
43.42	LoadBrush	944
43.43	Masken und Alphakanal	947
43.44	MixBrush	947
43.45	ModulateBrush	948
43.46	MoveBrush	948
43.47	OilPaintBrush	950
43.48	PenArrayToBrush	950
43.49	PerspectiveDistortBrush	952
43.50	PixelateBrush	953
43.51	PolarDistortBrush	953
43.52	QuantizeBrush	954
43.53	RasterizeBrush	956
43.54	ReadBrushPixel	956
43.55	ReduceAlphaChannel	957
43.56	RemapBrush	957
43.57	RemoveBrushPalette	958
43.58	ReplaceColors	959
43.59	RGBArrayToBrush	959
43.60	RotateBrush	961
43.61	SaveBrush	962
43.62	ScaleBrush	965
43.63	SelectAlphaChannel	966
43.64	SelectBrush	968
43.65	SelectMask	969
43.66	SepiaToneBrush	971
43.67	SetAlphaIntensity	972
43.68	SetBrushDepth	972
43.69	SetBrushPalette	973
43.70	SetBrushPen	974
43.71	SetBrushTransparency	975
43.72	SetBrushTransparentPen	976
43.73	SetMaskMode	976
43.74	SharpenBrush	977
43.75	SolarizeBrush	978
43.76	SwirlBrush	979
43.77	TintBrush	979

43.78	TransformBrush	980
43.79	TrimBrush	981
43.80	Vektorpinsel	982
43.81	WaterRippleBrush	982
43.82	WriteBrushPixel	983
<b>44</b>	<b>Pluginbibliothek</b>	<b>985</b>
44.1	DisablePlugin	985
44.2	EnablePlugin	985
44.3	GetPlugins	985
44.4	HavePlugin	988
44.5	LoadPlugin	989
44.6	REQUIRE	990
<b>45</b>	<b>Serialisierungsbibliothek</b>	<b>993</b>
45.1	DeserializeTable	993
45.2	GetSerializeMode	993
45.3	ReadTable	994
45.4	SerializeTable	995
45.5	SetSerializeMode	997
45.6	SetSerializeOptions	1000
45.7	WriteTable	1001
<b>46</b>	<b>Serielle Schnittstellenbibliothek</b>	<b>1005</b>
46.1	ClearSerialQueue	1005
46.2	CloseSerialPort	1005
46.3	FlushSerialPort	1005
46.4	GetBaudRate	1006
46.5	GetDataBits	1007
46.6	GetDTR	1007
46.7	GetFlowControl	1008
46.8	GetParity	1008
46.9	GetRTS	1009
46.10	GetStopBits	1009
46.11	OpenSerialPort	1010
46.12	PollSerialQueue	1013
46.13	ReadSerialData	1013
46.14	SetBaudRate	1014
46.15	SetDataBits	1015
46.16	SetDTR	1016
46.17	SetFlowControl	1016
46.18	SetParity	1017
46.19	SetRTS	1017
46.20	SetStopBits	1018
46.21	WriteSerialData	1018

<b>47</b>	<b>Soundbibliothek</b>	<b>1021</b>
47.1	Übersicht	1021
47.2	CloseAudio	1021
47.3	CloseMusic	1022
47.4	CopySample	1022
47.5	CreateMusic	1023
47.6	CreateSample	1024
47.7	FillMusicBuffer	1026
47.8	FlushMusicBuffer	1029
47.9	ForceSound	1029
47.10	FreeModule	1030
47.11	FreeSample	1030
47.12	GetChannels	1031
47.13	GetPatternPosition	1031
47.14	GetSampleData	1032
47.15	GetSongPosition	1032
47.16	HaveFreeChannel	1033
47.17	InsertSample	1033
47.18	IsChannelPlaying	1035
47.19	IsModule	1035
47.20	IsMusicPlaying	1036
47.21	IsMusic	1036
47.22	IsSamplePlaying	1037
47.23	IsSample	1038
47.24	IsSound	1038
47.25	LoadModule	1039
47.26	LoadSample	1040
47.27	MixSample	1041
47.28	MUSIC	1043
47.29	OpenAudio	1044
47.30	OpenMusic	1045
47.31	PauseModule	1046
47.32	PauseMusic	1047
47.33	PlayModule	1047
47.34	PlayMusic	1047
47.35	PlaySample	1048
47.36	PlaySubsong	1050
47.37	ResumeModule	1051
47.38	ResumeMusic	1051
47.39	SAMPLE	1051
47.40	SaveSample	1053
47.41	SeekMusic	1054
47.42	SetChannelVolume	1054
47.43	SetMasterVolume	1054
47.44	SetMusicVolume	1055
47.45	SetPanning	1055



47.46	SetPitch .....	1056
47.47	SetVolume .....	1056
47.48	StopChannel .....	1057
47.49	StopModule .....	1057
47.50	StopMusic .....	1058
47.51	StopSample .....	1058
47.52	WaitMusicEnd .....	1058
47.53	WaitPatternPosition .....	1059
47.54	WaitSampleEnd .....	1059
47.55	WaitSongPosition .....	1060
<b>48</b>	<b>Speicherblockbibliothek .....</b>	<b>1061</b>
48.1	AllocMem .....	1061
48.2	AllocMemFromPointer .....	1061
48.3	AllocMemFromVirtualFile .....	1062
48.4	CopyMem .....	1063
48.5	DecreasePointer .....	1063
48.6	DumpMem .....	1064
48.7	FillMem .....	1065
48.8	FreeMem .....	1065
48.9	GetMemPointer .....	1066
48.10	GetMemString .....	1066
48.11	IncreasePointer .....	1067
48.12	MemToTable .....	1067
48.13	Peek .....	1069
48.14	Poke .....	1070
48.15	ReadMem .....	1071
48.16	TableToMem .....	1071
48.17	WriteMem .....	1072
<b>49</b>	<b>Spritebibliothek .....</b>	<b>1075</b>
49.1	Übersicht .....	1075
49.2	CopySprite .....	1076
49.3	CreateSprite .....	1077
49.4	DisplaySprite .....	1078
49.5	FlipSprite .....	1078
49.6	FreeSprite .....	1079
49.7	LoadSprite .....	1079
49.8	MoveSprite .....	1082
49.9	RemoveSprite .....	1083
49.10	RemoveSprites .....	1083
49.11	ScaleSprite .....	1083
49.12	SetSpriteZPos .....	1084
49.13	SPRITE .....	1085

## 50 Systembibliothek ..... 1089

50.1	Beep.....	1089
50.2	CollectGarbage.....	1089
50.3	DisableLineHook.....	1090
50.4	ELSE.....	1091
50.5	ELSEIF.....	1091
50.6	EnableLineHook.....	1091
50.7	End.....	1092
50.8	ENDIF.....	1092
50.9	GCInfo.....	1093
50.10	GetConstant.....	1093
50.11	GetDefaultAdapter.....	1094
50.12	GetDefaultLoader.....	1094
50.13	GetMemoryInfo.....	1095
50.14	GetSystemInfo.....	1095
50.15	GetType.....	1097
50.16	GetVersion.....	1097
50.17	IF.....	1099
50.18	IIf.....	1101
50.19	INCLUDE.....	1102
50.20	IsNil.....	1103
50.21	IsUnicode.....	1103
50.22	LegacyControl.....	1104
50.23	LINKER.....	1105
50.24	OpenURL.....	1106
50.25	OPTIONS.....	1107
50.26	SetDefaultAdapter.....	1110
50.27	SetDefaultLoader.....	1111
50.28	SetVarType.....	1113
50.29	ShowNotification.....	1113
50.30	Sleep.....	1115
50.31	VERSION.....	1116
50.32	Wait.....	1116

## 51 Tabellenbibliothek ..... 1119

51.1	Concat.....	1119
51.2	CopyTable.....	1119
51.3	CreateList.....	1120
51.4	ForEach.....	1121
51.5	ForEachI.....	1122
51.6	GetItem.....	1123
51.7	GetMetaTable.....	1123
51.8	HaveItem.....	1124
51.9	InsertItem.....	1124
51.10	IPairs.....	1125
51.11	IsTableEmpty.....	1126

51.12	ListItems .....	1126
51.13	NextItem .....	1127
51.14	Pack .....	1128
51.15	Pairs .....	1129
51.16	RawEqual .....	1130
51.17	RawGet .....	1130
51.18	RawSet .....	1131
51.19	RemoveItem .....	1132
51.20	SetListItems .....	1132
51.21	SetMetaTable .....	1133
51.22	Sort .....	1134
51.23	TableItems .....	1134
51.24	Unpack .....	1135
<b>52</b>	<b>Textbibliothek .....</b>	<b>1137</b>
52.1	Übersicht .....	1137
52.2	AddFontPath .....	1137
52.3	AddTab .....	1138
52.4	Arbeiten mit Schriften .....	1139
52.5	CloseFont .....	1140
52.6	CopyTextObject .....	1140
52.7	CreateFont .....	1141
52.8	CreateTextObject .....	1143
52.9	DisplayTextObject .....	1145
52.10	DisplayTextObjectFX .....	1146
52.11	FONT .....	1147
52.12	FreeGlyphCache .....	1148
52.13	FreeTextObject .....	1149
52.14	GetAvailableFonts .....	1149
52.15	GetBulletColor .....	1151
52.16	GetCharMaps .....	1151
52.17	GetDefaultEncoding .....	1152
52.18	GetFontColor .....	1152
52.19	GetFontStyle .....	1153
52.20	GetKerningPair .....	1154
52.21	Locate .....	1155
52.22	MoveTextObject .....	1155
52.23	NPrint .....	1156
52.24	OpenFont .....	1156
52.25	Print .....	1157
52.26	ResetTabs .....	1158
52.27	RotateTextObject .....	1159
52.28	ScaleTextObject .....	1159
52.29	Schriftdeklaration .....	1160
52.30	SetBulletColor .....	1161
52.31	SetDefaultEncoding .....	1162

52.32	SetFont	1163
52.33	SetFontColor	1166
52.34	SetFontStyle	1166
52.35	SetMargins	1168
52.36	TextExtent	1169
52.37	Textformatierungen	1170
52.38	TextHeight	1171
52.39	TextOut	1172
52.40	TextWidth	1179
52.41	TransformTextObject	1180
52.42	UseFont	1181

## **53 Vektorgrafikbibliothek . . . . . 1183**

53.1	AddArcToPath	1183
53.2	AddBoxToPath	1184
53.3	AddCircleToPath	1184
53.4	AddEllipseToPath	1185
53.5	AddTextToPath	1186
53.6	AppendPath	1187
53.7	ClearPath	1187
53.8	ClosePath	1188
53.9	CopyPath	1188
53.10	CurveTo	1188
53.11	DrawPath	1189
53.12	ForcePathUse	1191
53.13	FreePath	1191
53.14	GetCurrentPoint	1192
53.15	GetDash	1192
53.16	GetFillRule	1193
53.17	GetLineCap	1193
53.18	GetLineJoin	1194
53.19	GetMiterLimit	1194
53.20	GetPathExtents	1194
53.21	IsPathEmpty	1195
53.22	LineTo	1195
53.23	MoveTo	1196
53.24	NormalizePath	1196
53.25	PathItems	1196
53.26	PathToBrush	1199
53.27	RelCurveTo	1201
53.28	RelLineTo	1201
53.29	RelMoveTo	1202
53.30	SetDash	1202
53.31	SetFillRule	1203
53.32	SetLineCap	1204
53.33	SetLineJoin	1204

53.34	SetMiterLimit .....	1205
53.35	SetVectorEngine .....	1205
53.36	StartPath .....	1206
53.37	StartSubPath .....	1207
53.38	TranslatePath .....	1207
53.39	Vektorgrafikplugin .....	1208
<b>54</b>	<b>Veraltete Befehle .....</b>	<b>1209</b>
54.1	ACTIVEWINDOW .....	1209
54.2	BreakWhileMouseOn .....	1209
54.3	ClearEvents .....	1210
54.4	CLOSEWINDOW .....	1211
54.5	CreateButton .....	1211
54.6	CreateKeyDown .....	1213
54.7	DisableEvent .....	1214
54.8	DisableEventHandler .....	1215
54.9	EnableEventHandler .....	1215
54.10	EnableEvent .....	1216
54.11	GetEventCode .....	1217
54.12	Gosub .....	1217
54.13	Goto .....	1218
54.14	INACTIVEWINDOW .....	1218
54.15	Label .....	1219
54.16	ModifyButton .....	1219
54.17	ModifyKeyDown .....	1220
54.18	MOVEWINDOW .....	1220
54.19	ONBUTTONCLICK .....	1220
54.20	ONBUTTONCLICKALL .....	1221
54.21	ONBUTTONOVER .....	1222
54.22	ONBUTTONOVERALL .....	1223
54.23	ONBUTTONRIGHTCLICK .....	1223
54.24	ONBUTTONRIGHTCLICKALL .....	1224
54.25	ONJOYDOWN .....	1224
54.26	ONJOYDOWNLEFT .....	1225
54.27	ONJOYDOWNRIGHT .....	1225
54.28	ONJOYFIRE .....	1225
54.29	ONJOYLEFT .....	1226
54.30	ONJOYRIGHT .....	1226
54.31	ONJOYUP .....	1226
54.32	ONJOYUPLEFT .....	1228
54.33	ONJOYUPRIGHT .....	1228
54.34	ONKEYDOWN .....	1228
54.35	ONKEYDOWNALL .....	1229
54.36	RemoveButton .....	1229
54.37	RemoveKeyDown .....	1230
54.38	Return .....	1230

54.39	SIZEWINDOW .....	1231
54.40	WhileKeyDown .....	1231
54.41	WhileMouseDown .....	1231
54.42	WhileMouseOn .....	1232
54.43	WhileRightMouseDown .....	1232
<b>55</b>	<b>Videobibliothek .....</b>	<b>1235</b>
55.1	Übersicht .....	1235
55.2	CloseVideo .....	1236
55.3	DisplayVideoFrame .....	1236
55.4	ForceVideoDriver .....	1237
55.5	GetVideoFrame .....	1237
55.6	IsVideo .....	1239
55.7	IsVideoPlaying .....	1240
55.8	OpenVideo .....	1240
55.9	PauseVideo .....	1241
55.10	PlayVideo .....	1242
55.11	ResumeVideo .....	1244
55.12	SeekVideo .....	1244
55.13	SetVideoPosition .....	1245
55.14	SetVideoSize .....	1246
55.15	SetVideoVolume .....	1246
55.16	StopVideo .....	1247
55.17	VIDEO .....	1247
<b>56</b>	<b>Windows-Bibliothek .....</b>	<b>1249</b>
56.1	CreateShortcut .....	1249
56.2	GetShortcutPath .....	1249
56.3	ReadRegistryKey .....	1250
56.4	WriteRegistryKey .....	1251
<b>57</b>	<b>Zeichenkettenbibliothek .....</b>	<b>1253</b>
57.1	AddStr .....	1253
57.2	ArrayToStr .....	1253
57.3	Asc .....	1254
57.4	Base64Str .....	1255
57.5	BinStr .....	1255
57.6	ByteAsc .....	1256
57.7	ByteChr .....	1256
57.8	ByteLen .....	1257
57.9	ByteOffset .....	1257
57.10	ByteStrStr .....	1258
57.11	ByteVal .....	1259
57.12	CharOffset .....	1260
57.13	CharWidth .....	1261

57.14	Chr.....	1262
57.15	CompareStr.....	1262
57.16	ConvertStr.....	1263
57.17	CountStr.....	1264
57.18	CRC32Str.....	1265
57.19	EmptyStr.....	1265
57.20	EndsWith.....	1266
57.21	Eval.....	1266
57.22	FindStr.....	1268
57.23	FormatNumber.....	1269
57.24	FormatStr.....	1270
57.25	HexStr.....	1271
57.26	IgnoreCase.....	1271
57.27	InsertStr.....	1272
57.28	IsAlNum.....	1273
57.29	IsAlpha.....	1274
57.30	IsCntrl.....	1274
57.31	IsDigit.....	1275
57.32	IsGraph.....	1276
57.33	IsLower.....	1276
57.34	IsPrint.....	1277
57.35	IsPunct.....	1278
57.36	IsSpace.....	1278
57.37	IsUpper.....	1279
57.38	IsXDigit.....	1280
57.39	LeftStr.....	1280
57.40	LowerStr.....	1281
57.41	MD5Str.....	1281
57.42	MidStr.....	1282
57.43	PadNum.....	1283
57.44	PatternFindStr.....	1283
57.45	PatternFindStrDirect.....	1285
57.46	PatternFindStrShort.....	1286
57.47	PatternReplaceStr.....	1286
57.48	RepeatStr.....	1290
57.49	ReplaceStr.....	1290
57.50	ReverseFindStr.....	1291
57.51	ReverseStr.....	1292
57.52	RightStr.....	1293
57.53	SplitStr.....	1293
57.54	StartsWith.....	1294
57.55	StripStr.....	1295
57.56	StrLen.....	1296
57.57	StrStr.....	1297
57.58	StrToArray.....	1297
57.59	ToNumber.....	1298
57.60	ToString.....	1298

57.61	ToUserData.....	1299
57.62	TrimStr.....	1300
57.63	UnleftStr.....	1300
57.64	UnmidStr.....	1301
57.65	UnrightStr.....	1302
57.66	UpperStr.....	1303
57.67	Val.....	1303
57.68	ValidateStr.....	1304
<b>58</b>	<b>Zwischenablagebibliothek.....</b>	<b>1305</b>
58.1	ClearClipboard.....	1305
58.2	GetClipboard.....	1305
58.3	PeekClipboard.....	1306
58.4	SetClipboard.....	1307
<b>Anhang A</b>	<b>Lizenzen.....</b>	<b>1309</b>
A.1	Lua license.....	1309
A.2	OpenCV license.....	1309
A.3	ImageMagick license.....	1309
A.4	GD Graphics Library license.....	1313
A.5	Bitstream Vera fonts license.....	1313
A.6	Pixman license.....	1314
A.7	LuaSocket license.....	1315
A.8	libs232 license.....	1315
A.9	UsbSerial license.....	1316
A.10	SDL license.....	1316
A.11	LGPL license.....	1317
<b>Index</b> .....	<b>1321</b>	



# 1 Allgemeines

## 1.1 Einführung

Hollywood ist eine Multimediaorientierte Programmiersprache, mit der sich ganz einfach Programme und Spiele erstellen lassen. Entworfen mit dem Paradigma, die Softwareerstellung so einfach wie möglich zu machen, ist Hollywood für Anfänger und Fortgeschrittene gleichermaßen geeignet. Hollywood verfügt über eine umfangreiche Funktionsbibliothek (mit über 1000 Befehlen), die die Erstellung von Programmen und Spielen erheblich vereinfacht. Es wird seit 2002 entwickelt und ist heute ein sehr ausgereiftes und stabiles Softwarepaket.

Hollywood ist eine Multimediaorientierte Programmiersprache, mit der Programme und Spiele ganz einfach realisiert werden können. Bei der Konzeption Hollywoods wurde großen Wert darauf gelegt, dem Programmierer möglichst viel Arbeit abzunehmen, ohne ihn in seiner Kreativität einzuschränken. Hollywood ist daher sowohl für Anfänger als auch für fortgeschrittene Programmierer geeignet. Hollywood besitzt eine umfangreiche Befehls-/Funktionsbibliothek (die über 1000 verschiedene Befehle umfasst), mit welcher die Erstellung von Spielen und Programmen zu einem großen Teil vereinfacht wird. Seit 2002 ist es ein sehr ausgereiftes und stabiles Softwarepaket.

Eines der Highlights von Hollywood ist der integrierte Cross-Compiler, mit dem Software auf vielen verschiedenen Plattformen erzeugt werden kann, ohne dass eine einzige Codezeile geändert werden muss. Der Cross-Compiler kann Programme für alle Plattformen von jeder Plattform aus kompilieren, auf der Hollywood läuft. Sie können beispielsweise macOS-Programmpakete mit der Windows-Version von Hollywood kompilieren und umgekehrt. Darüber hinaus gibt es auch Erweiterungen, mit denen Sie Ihre Hollywood-Projekte in native Programme für iOS und Android kompilieren können.

Hollywood ist eine leichte, aber dennoch leistungsstarke Programmiersprache, deren Kern nur etwa zwei Megabyte groß ist und keine externen Komponenten benötigt. Daher ist sie ideal zum Erstellen von Programmen, die sofort einsatzbereit sind. Tatsächlich laufen Hollywood-Programme ohne jegliche vorherige Installation perfekt von einem USB-Stick. Darüber hinaus kann die Kernfunktionalität von Hollywood durch viele frei verfügbare Plugins erheblich erweitert werden, sodass Sie beispielsweise über Hollywood auf OpenGL und SDL zugreifen können.

Die folgenden Plattform-Architekturen werden derzeit von Hollywood unterstützt:

- AmigaOS 3 (m68k)
- AmigaOS 4 (ppc)
- Android (arm)
- AROS (x86)
- iOS (arm)
- Linux (x86)
- Linux (x64)
- Linux (ppc)
- Linux (arm)
- macOS (arm)

- macOS (x86)
- macOS (x64)
- macOS (ppc)
- MorphOS (ppc)
- WarpOS (m68k/ppc)
- Windows (x86)
- Windows (x64)

## Merkmale von Hollywood

### Grafik:

- Sehr flexibles, ebenenbasiertes Grafiksystem
- Unterstützung für Alphakanäle
- Sprites jeglicher Größe können benutzt werden
- Umfangreiche Textunterstützung mit Formatierung, Zeilenumbruch und Rotation
- Plattformunabhängige TrueType-Unterstützung
- Videos können abgespielt werden
- Lädt echte Vektorformate wie SVG
- Unterstützt PDF-Import und -Export
- Viele grafische Grundelemente (Ellipsen, Kreisausschnitte, Linien, Rechtecke, Polygone...)
- Unterstützung für vektorbasiertes Zeichnen (Bezier-Kurven...)
- Antialias-Unterstützung für Text und Vektorgrafik
- Über 150 Überblendeffekte für Grafik und Text
- Dutzende eingebaute Bildbearbeitungsbefehle
- Mächtige Befehle zum Offscreen-Zeichnen inkl. Zeichnen auf Masken und Alphakanäle
- Unterstützung für Clip-Regionen (rechteckig und unregelmäßig)
- Hardwarebeschleunigte Doppelpuffer-Unterstützung
- Animationen können eingebunden werden
- Grafiken können als PNG-Bild oder sogar als AVI-Video exportiert werden
- Fenster können Alphatransparenz benutzen
- OpenGL 3D-Programmierung wird über ein Plugin unterstützt
- Video-Streaming unterstützt über ein spezielles Plugin

### Sound:

- Mehrkanalige Sound-Schnittstelle
- Unterstützung für Samples und Streams
- Protracker-Module können abgespielt werden
- Lautstärke und Tonhöhe kann während dem Abspielen moduliert werden
- Multikanal-Mixer zum Manipulieren von Samples

- Dynamisch generierte Audiodaten können abgespielt werden
- Audio-Streaming unterstützt über ein spezielles Plugin

**GUI:**

- Das RapaGUI-Plugin ermöglicht native GUI-Entwicklung mit Hollywood
- Native GUIs können für Windows, Linux (GTK), macOS und AmigaOS (MUI) erstellt werden
- GUI-Layouts werden bequem über XML-Dateien angelegt
- Unterstützung für über 40 verschiedene Bedienelemente (Widgets)
- Volle Flexibilität da Hollywood-Displays in GUIs eingebunden werden können
- Komplett plattformunabhängige GUI-Entwicklung - benutzen Sie denselben Code für alle Plattformen!

**Netzwerk:**

- Volle Internet- und Netzwerkunterstützung
- Erstellt Server- und Client-Verbindungen
- Datenübertragung über eine Vielzahl von Protokollen wie HTTP, FTP und SCP
- Serielle E/A-Unterstützung durch RS/232- oder USB-Adapter
- IPC-Unterstützung für die Kommunikation mit anderen Programmen
- Unterstützt IPv4- und IPv6-Schnittstellen
- Volle SSL/TLS-Unterstützung

**System:**

- Mächtige, aber einfach zu benutzende Programmiersprache
- Cross-Compiler für Amiga, Windows, macOS und Linux
- Android- und iOS-Unterstützung über den frei erhältlichen Hollywood Player
- APKs können über optional erhältlichen Compiler generiert werden
- Mit Hollywood kompilierte Programme benötigen keine externen Bibliotheken/DLLs
- Alle externen Daten (inkl. Schriften) können bequem in Programme einkompiliert werden
- Volle Unicode-Unterstützung
- Unterstützung für Fenster- und Vollbildmodi
- Mehrere Monitore können angesprochen werden
- Sandbox-Konzept: Hollywood-Programme können nicht abstürzen
- OS-native Menüleisten können benutzt werden
- Ereignisbasiertes Programmiermodell, um die CPU zu schonen
- Interval- und Timerbefehle mit niedriger Latenzzeit
- Umfangreiche DOS-Bibliothek für Dateioperationen
- Unterstützt ZIP und andere Archivierungsprogramme
- Einfacher Zugriff auf die Zwischenablage

- Zugriff auf Systemdialoge (Datei auswählen, Texteingabe, etc.)
- Drag'n'drop-Unterstützung
- Unterstützt Datenbankverwaltung über SQL
- Große Mathe- und Zeichenketten-Bibliotheken
- Mauszeiger kann einfach ausgetauscht werden
- Bequeme Serialisierung von Daten in und aus JSON und XML
- Datum- und Zeitbefehle
- Einfache Internationalisierung über das Katalog-System
- Joystick-Unterstützung

### Plugin:

- Sehr mächtiges, plattformübergreifendes Plugin-System
- Öffentlich verfügbares SDK mit über 500 Seiten Dokumentation und Beispielen
- Gesamter Grafiktreiber kann komplett auf alternative Grafiksysteme (z.B. OpenGL...) umgeleitet werden
- Gesamter Audiotreiber kann komplett auf alternative Audiosysteme umgeleitet werden
- Plugins können Lade- und Speichermodule für zusätzliche Bild-, Ton-, Animations-, Schriftarten- und Videoformate bereitstellen
- Sämtliche Dateioperationen können abgefangen und durch eigene Routinen ersetzt werden
- Hollywoods Sprachumfang kann über Plugins erweitert werden

## 1.2 Philosophie

Die Philosophie hinter Hollywood ist, eine einfache aber leistungsfähige Plattform zu bieten, die verwendet werden kann, um atemberaubende Programme in sehr kurzer Zeit zu schreiben. Die verwendete Sprache, um mit Hollywood zu programmieren, ist nicht nur für den Anfänger sehr einfach, sondern hat auch genug Features, die erfahrene Programmierer lieben. Sie können mit Hollywood auf einem sehr einfachen Basic ähnlichem Level programmieren. Aber es ist auch möglich, mit Hollywood in eine voll objektorientierte Welt hinein zu tauchen. Der umfangreiche Befehlssatz mit mehr als 1000 eingebauten Befehle bietet dem Programmierer alle notwendigen Werkzeuge, um erstaunliche Software mit Hollywood zu erstellen. Es gibt fast nichts, was man mit Hollywood nicht tun könnte. Hinzu kommt, dass Hollywood ein cross-platform multimedia application layer ist. Das bedeutet, Sie können Ihre Programme auf vielen verschiedenen Plattformen ohne Änderung einer einzigen Codezeile ausführen. Es ist sogar möglich, quer kompilierte Programme für diese Plattformen zu erstellen. Zum Beispiel können Sie macOS-Anwendungen von Ihrer AmigaOS 3.1-Installation kompilieren. All dies macht Hollywood zum ultimativen Werkzeug für die Multimedia-Programmierer.

## 1.3 Lizenzvereinbarung

Hollywood (im Folgenden als "das Programm" bezeichnet) ist © Copyright 2002-2023 von Andreas Falkenhahn (im folgenden "der Autor" genannt). Alle Rechte vorbehalten.

Das Programm wird zur Verfügung gestellt "wie es ist" und der Autor kann für keinerlei Schäden, welcher Natur sie auch immer sein mögen, verantwortlich gemacht werden. Sie benutzen dieses Programm völlig auf eigene Gefahr und eigenes Risiko. Der Autor gibt keinerlei Garantien in Verbindung mit der Benutzung dieses Programmes, nicht einmal die Garantie der Funktionstüchtigkeit.

Dies ist eine Einzelbenutzerlizenz. Das Programm darf ausschließlich von der Person benutzt werden, die diese Lizenz erworben hat. Der Name dieser Person wird im Programm angezeigt.

Dieses Programm darf ohne schriftliche Genehmigung des Autors nicht weitergegeben oder verbreitet werden.

Jegliche Art von Wrapper-Programmen, die Hollywood-Befehle oder Hollywood-Funktionalität für andere Programmiersprachen oder den Endbenutzer zugänglich machen, sind verboten. Es darf zudem keine Mittlersoftware geschrieben werden, die Hollywood-Befehle oder Hollywood-Funktionalität über diese Mittlersoftware zugänglich macht.

Ohne Einwilligung des Autors sind keine Änderungen am Programm erlaubt.

Dieses Programm verwendet Lua von Roberto Ierusalimsky, Waldemar Celes und Luiz Henrique de Figueiredo. Siehe [Abschnitt A.1 \[Lua Lizenz\]](#), [Seite 1309](#), für Details.

Dieses Programm verwendet libjpeg von der Independent JPEG Group.

Dieses Programm verwendet libpng von der PNG Development Group und zlib von Jean-loup Gailly und Mark Adler.

Dieses Programm verwendet PTPlay © Copyright 2001, 2003, 2004 bei Ronald Hof, Timm S. Mueller, Per Johansson.

Dieses Programm benutzt die OpenCV-Bibliothek der Intel Corporation. Siehe [Abschnitt A.2 \[OpenCV-Lizenz\]](#), [Seite 1309](#), für Details.

Dieses Programm verwendet ImageMagick von ImageMagick Studio LLC. Siehe [Abschnitt A.3 \[ImageMagick-Lizenz\]](#), [Seite 1309](#), für Details.

Dieses Programm verwendet die GD Graphics Library von Thomas Boutell. Siehe [Abschnitt A.4 \[GD Graphics Library-Lizenz\]](#), [Seite 1313](#), für Details.

Dieses Programm benutzt die Pixman-Bibliothek. Siehe [Abschnitt A.6 \[Pixman-Lizenz\]](#), [Seite 1314](#), für Details.

Teile dieses Programms sind Copyright © 2010 The FreeType Project (<http://www.freetype.org>). Alle Rechte vorbehalten.

Hollywood benutzt die Bitstream Vera-Schriftarten. Siehe [Abschnitt A.5 \[Bitstream Vera-Lizenz\]](#), [Seite 1313](#), für Details.

Die Linux-Version von Hollywood benutzt glibc und die Advanced Linux Sound Architecture (ALSA). All diese Komponenten stehen unter der LGPL-Lizenz. Siehe [Abschnitt A.11 \[LGPL-Lizenz\]](#), [Seite 1317](#), für Details.

Die Android-Version von Hollywood verwendet den Simple DirectMedia Layer (SDL) von Sam Lantinga. Siehe [Abschnitt A.10 \[SDL-Lizenz\]](#), [Seite 1316](#), für Details.

Dieses Programm benutzt die codesets.library von Alfonso Ranieri und die codesets.library vom Open Source Team. Siehe [Abschnitt A.11 \[LGPL-Lizenz\]](#), [Seite 1317](#), für Details.

Dieses Programm verwendet LuaSocket von Diego Nehab. Siehe [Abschnitt A.7 \[LuaSocket-Lizenz\]](#), [Seite 1315](#), für Details.

Dieses Programm verwendet libs232 von Petr Stetiar. Siehe [Abschnitt A.8 \[libs232-Lizenz\]](#), [Seite 1315](#), für Details.

Dieses Programm verwendet UsbSerial von Felipe Herranz. Siehe [Abschnitt A.9 \[UsbSerial-Lizenz\]](#), [Seite 1316](#), für Details.

Amiga ist ein eingetragenes Warenzeichen der Amiga, Inc. Alle anderen Warenzeichen gehören ihren jeweiligen Besitzern.

FÜR DIESES PROGRAMM GIBT ES KEINE GARANTIE, SOWEIT ES DIE ANZUWENDENDEN GESETZE ZULASSEN. SOFERN ANDERSWO NICHTS GEGENTEILIGES GESCHRIEBEN STEHT STELLEN DER AUTOR UND/ODER DRITTE DAS PROGRAMM "SO WIE ES IST" ZUR VERFÜGUNG, OHNE IRGEND-EINE GARANTIE, WEDER DIREKT NOCH INDIREKT. DIES BEINHÄLTET, IST ABER NICHT DARAUF BESCHRÄNKT, VERKÄUFLICHKEIT UND EIGNUNG FÜR EINEN BESTIMMTEN VERWENDUNGSZWECK. DAS VOLLSTÄNDIGE RISIKO DER QUALITÄT UND AUSFÜHRBARKEIT DES PROGRAMMS LIEGT BEIM ANWENDER. SOLLTE SICH DAS PROGRAMM ALS DEFECT HERAUS-STELEN, LIEGEN ALLE KOSTEN FÜR SERVICE, INSTANDSETZUNG ODER NACHBESSERUNG BEIM ANWENDER.

KEIN COPYRIGHT-INHABER ODER DRITTER, DER DAS PROGRAMM WIE OBEN ERLAUBT WEITERVERKAUFT, KANN FÜR SCHÄDEN IRGENDWELCHER ART HAFTBAR GEMACHT WERDEN (DIES BEINHÄLTET, IST ABER NICHT BE-SCHRÄNKT AUF, DATENVERLUST INFOLGE UNFÄHIGKEIT DES PROGRAMMS, MIT ANDEREN PROGRAMMEN ZUSAMMENZUARBEITEN), SELBST WENN EIN SOLCHER INHABER ODER DRITTER AUF DIE MÖGLICHKEIT EINES SOLCHEN SCHADENS HINGEWIESEN WURDE, AUSSER ES BESTEHT EINE SCHRIFTLICHE EINWILLIGUNG ODER WIRD VOM GESETZ VERLANGT.

## 1.4 Anforderungen

### Windows Version:

- mindestens Windows 2000

### macOS Version:

- arm64 version: Mindestens macOS 11.0 (Big Sur)
- Intel Version: Mindestens macOS 10.6 (Snow Leopard)
- PowerPC Version: Mindestens macOS 10.4 (Tiger)

### Linux Version:

- mindestens ein X11 Server und glibc
- optional: ALSA Library für die Soundausgabe
- optional: GTK für Dialogboxen Unterstützung

- optional: XFree86 video mode extension library, Xfixes, Xrender, Xcursor, Xrandr, Xss für weitere fortgeschrittene Funktionalität

### AmigaOS/MorphOS/AROS Version:

- Kickstart 3.0 (V39)
- 68020+ oder PowerPC
- CyberGraphX oder Picasso96
- `codesets.library` auf AmigaOS 3 und 4, WarpOS und AROS
- `charsets.library` auf MorphOS
- optional: AHI bei Martin Blom für Soundausgabe
- optional: `reqtools.library` für die Befehle `StringRequest()` (außer OS4) und `ColorRequest()` (auch OS4)
- optional: `popupmenu.library` für die `PopupMenu()` Befehle (außer auf MorphOS)

### iOS Version:

- iOS 8 oder höher

### Android Version:

- Android 4.0 oder höher
- ARM CPU (32-bit or 64-bit)

## 1.5 Danksagungen

Hollywood wurde von Andreas Falkenhahn geschrieben. Aber ich wäre nicht ohne die Hilfe von vielen Personen so weit gekommen, denen ich hier danken möchte.

Zuerst geht ein besonderer Dank an Timm S. Müller für seine wesentlichen Hinweise in der frühen Phase der Konzeptualisierung von Hollywood.

Zweiter großer Dank an das Team von Lua 5.0.2 für die Herstellung dieser leistungsstarken leichte Sprache: Roberto Ierusalimsky, Waldemar Celes und Luiz Henrique de Figueiredo.

Danke an Frank Wille für das ständige Verbessern des wunderbaren vbcc Compilers, der für die Erstellung von den Hollywoodversionen 68k und WarpOS notwendig sind.

Ein besonderer Dank muss an Dominic Widmer, Helmut Haake und Alexander Pfau für die Übersetzung der riesigen Dokumentation ins Deutsche gehen. Alexander Pfau war der erste, der an einer deutschen Übersetzung arbeitete. Er pflegte das deutsche Hollywood-Handbuch bis zur Version 1.9. Seine Arbeit wurde später in einer schweizerisch-deutschen Gemeinschaftsarbeit von Dominic Widmer und Helmut Haake fortgesetzt, die die deutsche Hollywood-Übersetzung bis heute pflegen. Fehler oder Verbesserungsvorschläge bzgl. des deutschen Handbuchs bitte an das Übersetzungsteam richten, welches unter [handbuch@gmx.ch](mailto:handbuch@gmx.ch) oder <https://amiga-resistance.info> erreicht werden kann.

Bedanken möchte ich mich auch bei Grzegorz Kraszewski, Martin Blom, Tomasz Wiskowski, Kimmo Pekkola, Olaf Barthel, Thomas Richter, Christoph Gutjahr, Jean-Yves Auger, Ralph Schmidt, Detlef Würkner, Stephan Rupprecht, Frank Mariak, Jacek Piszczek, Torgeir

Vee Christoph Pözl, Fabio Falcucci, Michael Jurisch, Petteri Valli sowie allen Betatestern und bei all denen, die hier stehen sollten, aber vergessen wurden.

Hollywood wurde unter SAS/C 6,58 (Amiga 68k Version), VBCC (WarpOS-Version), GCC 4.4.4 (MorphOS-Version) und GCC 4.0.2 (AmigaOS4-Version) entwickelt. Zusätzlich wurden die folgenden Programme verwendet: GoldED Studio AIX, Directory Opus 4, PPaint 7, Cyberguard, CyberGraphX 4 und MUI. Die Hauptentwicklung wurde auf einem Pegasos 2 mit einem 1 GHz-G4-CPU und MorphOS 1.4.5 durchgeführt. Die weitere Entwicklung auf einem Amiga 1200 mit einem Phase5 Blizzard PPC 603e 240MHz mit SCSI, eine 68060 CPU, einer Phase5 BVision PPC Grafikkarte und 82 Megabyte RAM. Hollywood wurde auf CyberGraphX 3 und 4, Picasso96, MorphOS, AmigaOS4, AROS DraCo, Amithlon und WinUAE getestet. Hollywood greift in keiner Weise direkt auf die Hardware zu. Hollywood respektiert alle Systemstyleguides und verwendet nur systemfreundliche Befehle.

Die macOS-Version wurde auf einem 1,5 Ghz Mac Mini mit macOS 10.4 (Tiger) und auf einem Intel-iMac 2,4 GHz mit macOS 10.5 (Leopard) entwickelt. Der Code wurde mit Allan Odgaards flexiblen Texteditor TextMate geschrieben. Hollywood wurde mit gcc kompiliert, die mit Apples macOS SDK geliefert wird.

Die Windows-Version wurde auf einem 2,6 Ghz Pentium IV mit Win XP Home Edition und den neuesten Service Packs entwickelt. Der Code wurde mit dem berühmten UltraEdit von IDM Comp geschrieben. Hollywood wurde mit Microsoft Visual C kompiliert.

Die Linux-Version wurde mit openSUSE 11.2 auf einem 2.6 Ghz Pentium IV entwickelt.

## 1.6 Forum

Das offizielle Hollywood-Forum ist unter <https://www.hollywood-mal.de/> online. Fühlen Sie sich frei und sprechen Sie mit anderen Hollywood-Benutzern. Dies ist der perfekte Ort, um Hilfe von anderen Benutzern zu bitten.

Neben dem Forum haben wir auch einen Newsletter, der über Ankündigungen informiert. Wenn Sie über neue Veröffentlichungen, Hollywood-Plugins, Updates und alles andere rund um das Hollywood-Projekt benachrichtigt werden möchten, zögern Sie nicht, sich für den Newsletter auf dem offiziellen Hollywood-Portal unter <http://www.hollywood-mal.com> anzumelden.

Hollywood-Benutzer aus dem deutschsprachigen Raum sollten einen Blick auf die "Hollywood User Page" werfen, die von Helmut Haake verwaltet werden. Hier finden Sie viele Quellcodes sowie Workshops und es hat auch ein Forum, in dem Sie Ihre Fragen stellen können. Hier ist der Link: <https://forum.amiga-resistance.info/viewforum.php?f=38>

## 1.7 Kontakt

Wenn Sie mich kontaktieren möchten, können Sie entweder eine E-Mail an [andreas@airsoftsoftwair.de](mailto:andreas@airsoftsoftwair.de) schicken oder verwenden Sie das Kontaktformular auf <https://www.hollywood-mal.de/>.

Fehler oder Verbesserungsvorschläge bzgl. des deutschen Hollywood-Handbuchs bitte an das Übersetzungsteam richten, welches unter [handbuch@gmx.ch](mailto:handbuch@gmx.ch) oder <https://amiga-resistance.info> erreicht werden kann.



## 2 Erste Schritte

### 2.1 Übersicht übers Hollywoodpaket

Hier ist ein Überblick über die wichtigsten Komponenten, die Sie bei der Programmierung mit Hollywood kennen müssen:

#### Interpreter:

Das Hollywood-Hauptprogramm ist der "Interpreter". Er liest Ihre Quellcode-Dateien (die Skripte) und übersetzt diese in den benutzerdefinierten Hollywood-Bytecode. Er kann auch Ihre Quellcodes in eigenständige ausführbare Programme oder Applets kompilieren und Datendateien in diese Programme oder Applets einbinden. Sobald Sie Ihre Quellcodes in ausführbare Programme oder Applets kompiliert haben, können Sie sie verteilen. Wenn Sie Ihre Projekte als Applets verteilen, müssen Ihre Benutzer zuerst den frei verfügbaren Hollywood-Player installieren, bevor sie Ihr Projekt ausführen können. Wenn Sie Ihre Projekte als eigenständige ausführbare Programme verteilen möchten, sind keine weiteren Komponenten erforderlich. Der Interpreter ist ein Konsolenprogramm und hat keine GUI. Hollywoods GUI und die Windows IDE werden einfach den Interpreter starten, wenn Sie ein Skript ausführen wollen.

#### Player:

Der Hollywood-Player kann Applets (siehe unten), aber keine Quellcodes ausführen. Der Hollywood-Player ist nicht Teil der kommerziellen Distribution von Hollywood, sondern steht zum kostenlosen Download auf dem offiziellen Hollywood-Portal <https://www.hollywood-mal.de/> zur Verfügung. Im Gegensatz zum Interpreter ist der Hollywood-Player frei verteilbar. Wenn Sie sich dafür entscheiden, Ihre Projekte als Hollywood-Applets zu verteilen, müssen Ihre Benutzer zuerst den Hollywood-Player herunterladen und installieren, bevor sie Ihre Projekte ausführen können. Wenn Sie Ihre Projekte als eigenständige Programme verteilen wollen, ist der Hollywood-Player nicht nötig, weil er bereits in Ihr ausführbares Programm eingebunden ist. Beachten Sie, dass der Hollywood-Player für Android nicht über das Hollywood-Portal erhältlich ist, sondern bei Google Play: <http://play.google.com/store/apps/details?id=com.airsoftsoftwair.hollywood>

#### GUI:

Da der Hollywood-Interpreter nur ein Konsolenprogramm ist, wird ein separates GUI-Programm mitgeliefert, mit dem Sie den Interpreter bequem nutzen können. Unter Windows ist die Hollywood-GUI eine vollwertige IDE, während es auf allen anderen Systemen (Amiga, Linux, macOS) nur eine grafische Benutzeroberfläche ist, mit der Sie Skripte starten und kompilieren können. Aber diese GUI erlaubt es nicht, Skripte zu bearbeiten. Auf Amiga, Linux und macOS müssen Sie Ihren Lieblings-Texteditor für die Bearbeitung von Hollywood-Skripten verwenden. Aber da der Interpreter ein Konsolenprogramm ist, ist es ganz einfach, ihn in Ihre Lieblings-IDE zu integrieren.

#### Skripts:

Quellcodes in der Hollywood-Sprache heißen Hollywood-Skripte. Hollywood-Skripte sind einfache Textdateien, die eine Reihe von Anweisungen enthalten, die Hollywood verstehen

kann. So müssen sie bestimmten syntaktischen Regeln folgen, die in dieser Dokumentation erläutert werden. Hollywood-Skripte verwenden die Erweiterung `*.hws`. Es wird empfohlen, UTF-8-Codierung für alle Ihre Skripte zu verwenden.

### Applets:

Hollywood-Applets sind Binärdateien, die den Bytecode eines Skripts enthalten sowie zusätzliche Daten wie Bilder, Sounds, Schriften usw. Applets können vom Hollywood-Player oder dem Hollywood-Interpreter ausgeführt werden. Hollywood-Applets verwenden die Erweiterung `*.hwa`. Die Wahl, Ihr Projekt als Applet zu verteilen, kann viel Platz sparen, da Applets meist sehr klein sind. Sie enthalten nicht den Hollywood-Player. Siehe [Abschnitt 4.2 \[Applets\]](#), [Seite 58](#), für Details.

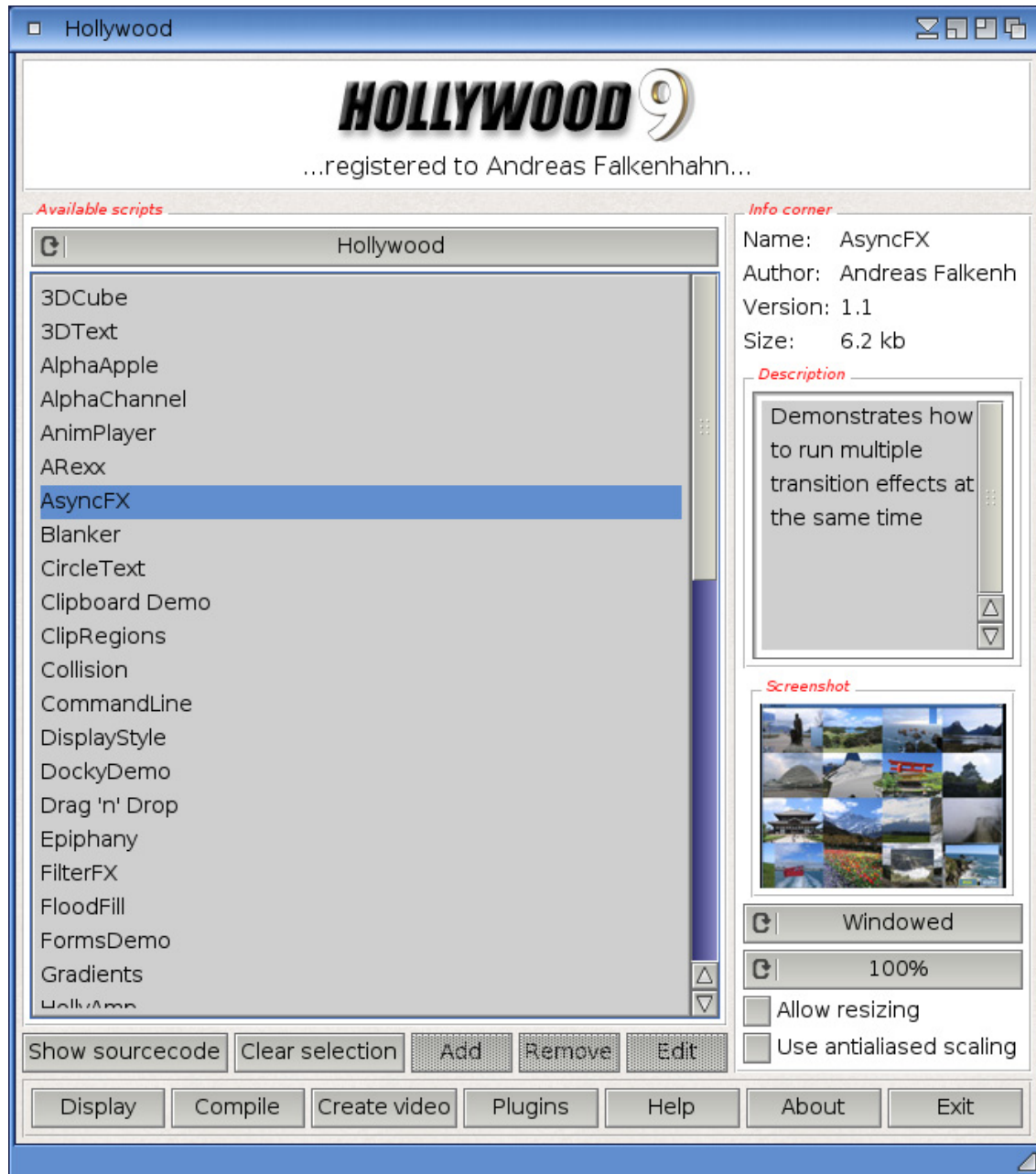
### Plugins:

Bei allen Systemen mit Ausnahme von AmigaOS und kompatiblen Systemen müssen Plugins in einem Verzeichnis mit dem Namen "Plugins" gespeichert werden, das sich im selben Verzeichnis wie das Hollywood-Programm befindet. Bei AmigaOS und kompatiblen Systemen müssen Plugins im Verzeichnis `LIBS:Hollywood` installiert werden. Unter macOS muss sich das Verzeichnis "Plugins" im Verzeichnis "Resources" des Programmpakets befinden, d.h. im Verzeichnis `HollywoodInterpreter.app/Contents/Resources`. Beachten Sie, dass `HollywoodInterpreter.app` im Programmpaket `Hollywood.app` selbst unter `Hollywood.app/Contents/Resources` gespeichert ist. Bei der Verteilung eines kompilierten Hollywood-Programms müssen Plugins, die von Ihrem Programm benötigt werden, einfach in das gleiche Verzeichnis wie Ihr Programm kopiert werden. Beim Kompilieren von Programmpaketen für macOS müssen die Plugins im Verzeichnis "Ressourcen" des Programmpakets abgelegt werden, d.h. in `MyProject.app/Contents/Resources`. Alternativ können Sie auch Plugins in Ihr Programm einbinden. Hollywood-Plugins verwenden die Erweiterung `*.hwp`. Siehe [Abschnitt 5.1 \[Plugins\]](#), [Seite 67](#), für Details.

## 2.2 Die grafische Benutzeroberfläche (GUI)

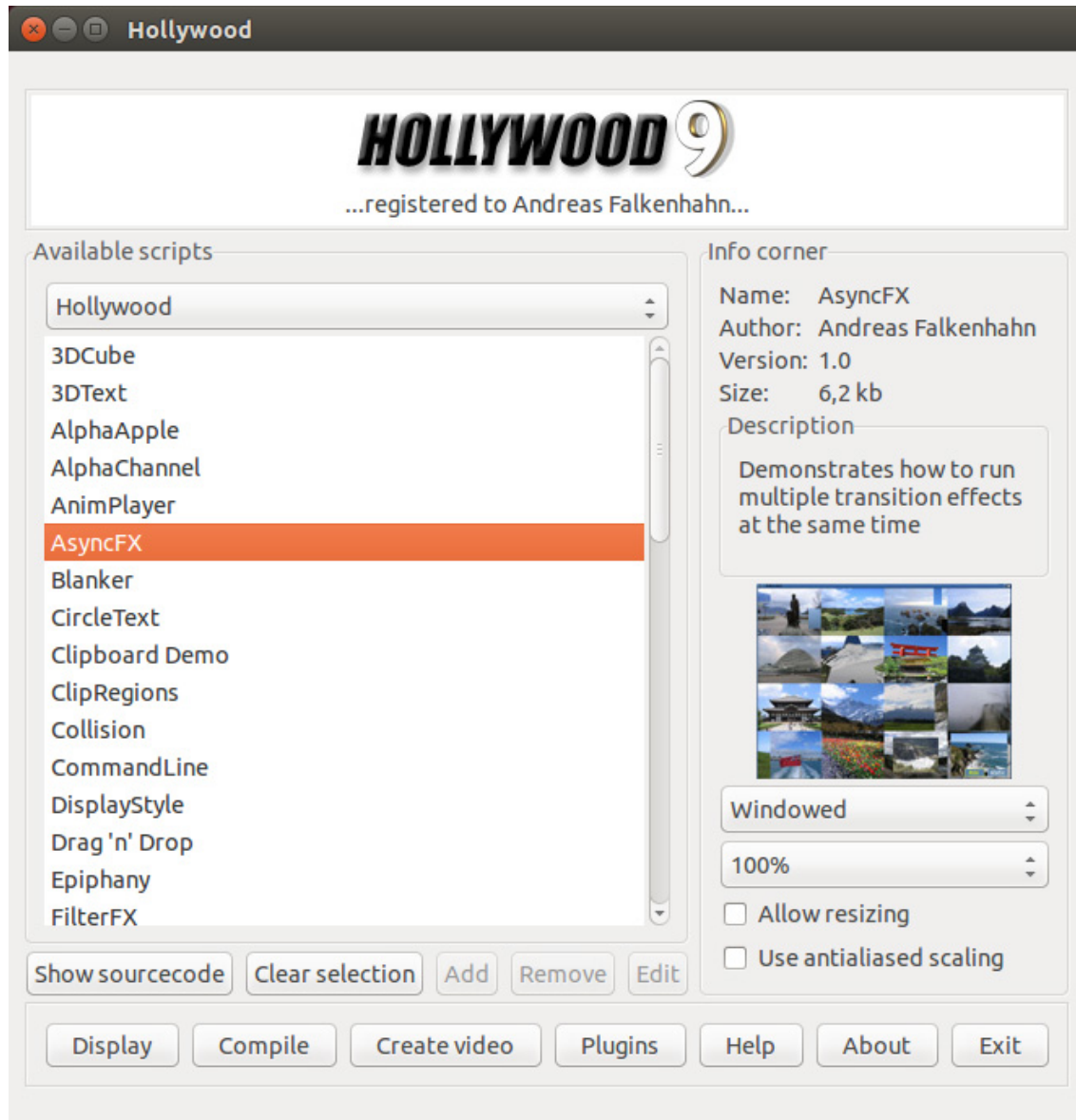
Hollywood selbst ist ein Konsolenprogramm. Aber man kann auch über die Hollywood-GUI Skripte starten, ohne irgendwelche Konsolenargumente auswendig zu können.

Hier ist ein Bildschirmfoto der Hollywood-GUI für Amiga-Systeme:



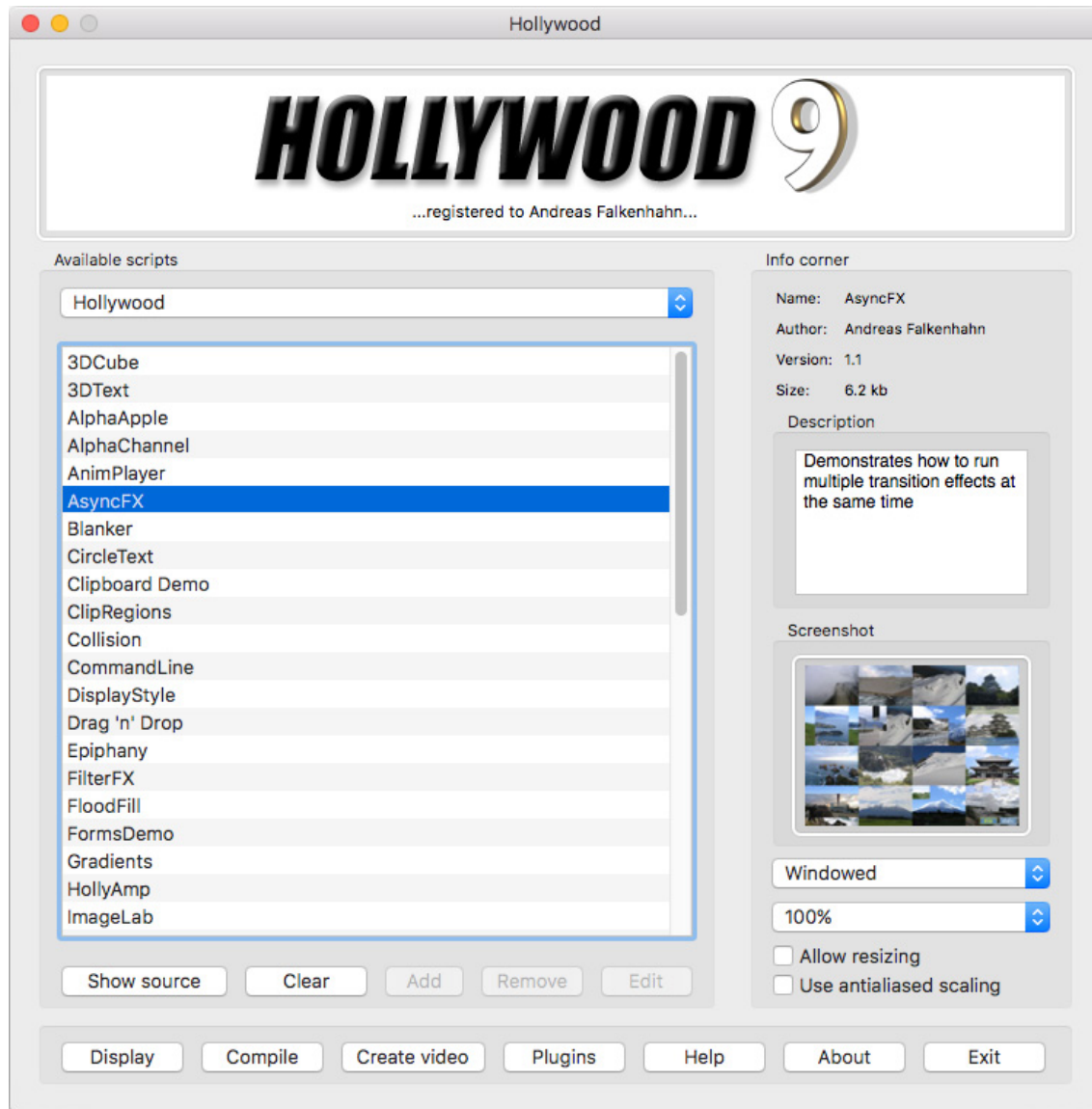
Die beiden GUIs von Linux und macOS sind vollkommen gleich wie die GUI vom Amiga. Wie Sie sehen können, wird die GUI in drei Bereiche geteilt: Dem Schaufenster (Listview) mit den Hollywoodbeispielen, der Info-Ecke mit den Informationen zu den Programmen sowie dem Bildschirmfoto des aktuell ausgewählten Projekt und schließlich eine Auswahl an Knöpfen für die meisten üblichen Handlungen.

So sieht das Hollywood GUI unter Linux aus:



Nach dem Starten wird Hollywood automatisch das Verzeichnis **Examples** scannen und die gefundenen Skripte nach Themen sortieren. Mit dem Wahlknopf können Sie zwischen den Gruppen wechseln. Die letzte Gruppe ist immer "Meine Projekte". Die hier aufgeführten Skripte sollten nicht im Verzeichnis **Examples** gespeichert sein, da sie sonst bei einem Update von Hollywood gelöscht werden.

Hier ist ein Bildschirmfoto von der Hollywood-GUI auf macOS:



Die folgenden Tasten sind in der GUI verfügbar:

**Quelltext anzeigen / Show sourcecode:**

Drücken Sie diese Taste, um den Quellcode des gegenwärtig ausgewählten Hollywood-Projektes zu zeigen. Unter dem Menüpunkt Einstellungen-Hollywood können Sie den von Ihnen bevorzugten Anzeiger wählen.

**Auswahl löschen / Clear selection:**

Löschen Sie das gegenwärtig ausgewählte Projekt. Dies ist notwendig, wenn Sie gerne ein Projekt laufen lassen wollen, das nicht in einem der Projektschaufenster (Listviews) erscheint. Falls kein Projekt ausgewählt ist, öffnet sich ein Dateiauswahlfenster. Jetzt kann ein anderes Hollywoodprojekt ausgewählt werden, welches auf keinem der Bereiche des Schaufensters aufgeführt ist.

**Hinzufügen / Add:**

Diese Schaltfläche wird nur aktiviert, wenn **Meine Projekte** als Gruppe im Wahlknopf ausgewählt wurde. In diesem Fall können Sie diese Taste verwenden, um ein neues Projekt in die Liste der Projekte hinzuzufügen.

**Entfernen / Remove:**

Diese Schaltfläche wird nur aktiviert, wenn **Meine Projekte** als Gruppe im Wahlknopf ausgewählt wurde und mindestens ein Projekt hinzugefügt wurde. In diesem Fall können Sie diese Taste verwenden, um ein Projekt aus der Liste von Projekten zu entfernen.

**Bearbeiten / Edit:**

Diese Schaltfläche wird nur aktiviert, wenn **Meine Projekte** als Gruppe im Wahlknopf ausgewählt und mindestens ein Projekt hinzugefügt wurde. In diesem Fall können Sie diese Taste verwenden, um das ausgewählte Projekt in Ihrem bevorzugten Texteditor zu öffnen. Unter dem Menüpunkt Einstellungen-Hollywood können Sie den von Ihnen bevorzugten Texteditor wählen.

**Skript starten / Display:**

Klicken Sie auf diese Schaltfläche, um das ausgewählte Projekt auszuführen. Wenn kein Projekt markiert ist, öffnet sich ein Dateiauswahlfenster, um ein Projekt auszuwählen.

**Kompilieren / Compile:**

Sie können diese Taste verwenden, um ein Hollywood-Skript in eine ausführbares Programm oder Hollywood-Applet zu kompilieren. Sie werden >aufgefordert, ein Hollywood-Skript, eine Ausgabedatei und eine Zielplattform zu wählen. Anschließend wird der Hollywood-Compiler aufgerufen und die ausführbare Datei oder das Applet zu kompilieren. Siehe [Abschnitt 4.1 \[Verwendung von Compiler & Linker\]](#), Seite 57, für Details.

**Video erstellen / Create video:**

Mit dieser Taste können Sie ein Video von einem Hollywood-Projekt aufnehmen. Sie werden aufgefordert, ein Hollywood-Skript, eine Ausgabedatei sowie das Videoformat zu wählen. Anschließend wird der Hollywood-Recorder ein Video von Ihrem Skript erstellen. Siehe [Abschnitt 4.6 \[Verwendung des Video-recorders\]](#), Seite 63, für Details.

**Erweiterungen (Plugins):**

Drücken Sie diese Taste, um einen Dialog zu öffnen, der alle derzeit verfügbaren Hollywood-Plugins und einige Informationen über sie zeigt.

**Hilfe / Help:**

Klicken Sie auf diese Schaltfläche, um diese Dokumentation zu öffnen.

**Über / About:**

Zeigt das Urheberrecht und die Lizenzinformationen sowie eine Übersicht über alle verfügbaren Hollywood-Befehle.

**Verlassen / Exit:**

Schließt die GUI.

Es gibt einige weitere Optionen unter dem Bildschirmfoto:

**Anzeigemodus / Display mode:**

Mit diesem Wahlknopf können Sie den gewünschten Anzeigemodus für das Projekt auswählen. Dies kann entweder **Im Fenster**, **Vollbild** oder **Vollbild (skaliert)** sein. Wenn Sie **Vollbild** wählen, ändert der Monitor seine physikalische Auflösung in die Auflösung des Skripts, während **Vollbild (skaliert)** die physikalische Auflösung des Monitors beibehält und nur das Skript skaliert, um die aktuelle Auflösung des Monitors zu benutzen.

**Skalierungsfaktor / Scale factor:**

Sie können mit diesem Cycleknopf die automatische Skalierung von Hollywood einstellen. Wenn Sie eine andere Einstellung als 100% wählen, wird automatisch um den angegebenen Faktor skaliert werden. Siehe [Abschnitt 24.24 \[Skalierungssysteme\]](#), [Seite 402](#), für Details.

**Größe verändern erlauben / Allow resizing:**

Wenn Sie dieses Kästchen ankreuzen wird Hollywood das Skript als größenveränderlich öffnen. Der Benutzer wird dann in der Lage sein, die Dimensionen des Fensters zu ändern. Lassen Sie dieses Feld unmarkiert, wenn Sie dies nicht wünschen.

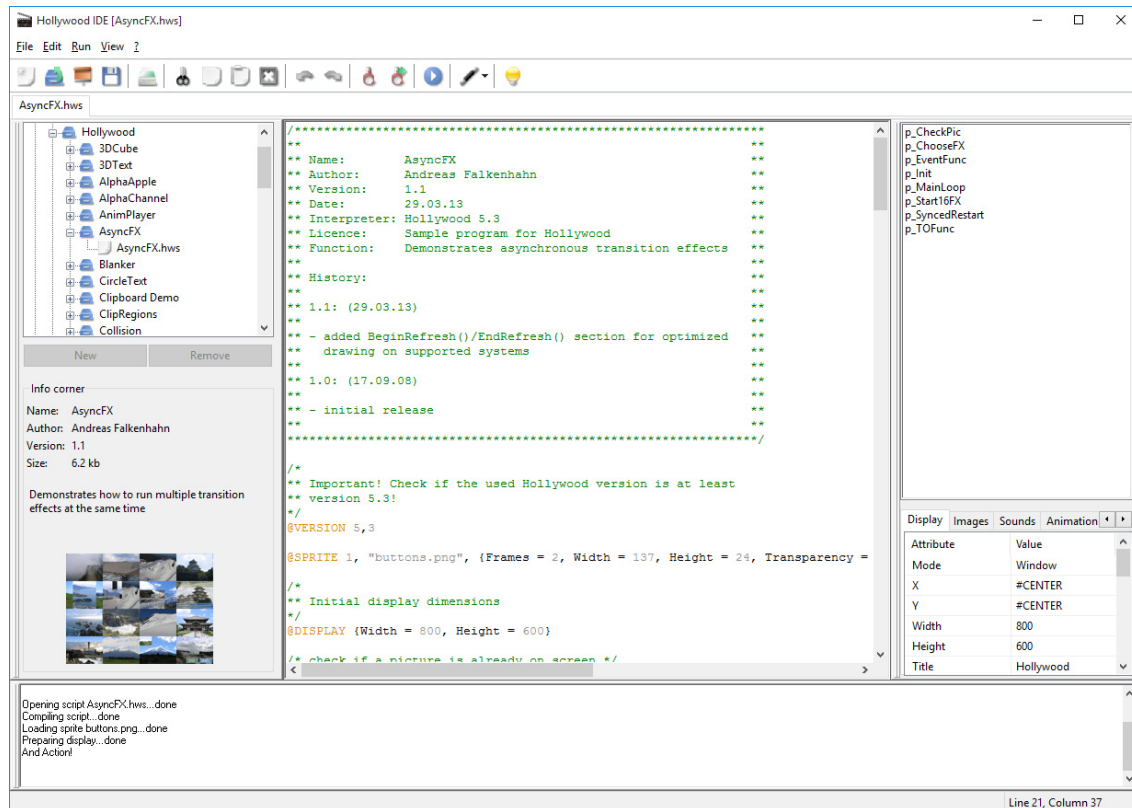
**Antialias-Skalierung benutzen / Use antialiased scaling:**

Wenn die Skalierung aktiviert ist, können Sie mit dieser Option festlegen, ob Antialias-Skalierung verwendet werden soll oder nicht. Antialias-Skalierung sieht besser aus, ist aber auch viel langsamer als die normale Skalierung.

## 2.3 Windows IDE

Auf der Windows-Plattform verfügt Hollywood über eine voll funktionsfähige integrierte Entwicklungsumgebung (IDE), die verwendet werden kann, um sehr leicht Hollywood-Projekte zu erstellen. Die IDE verfügt über einen Texteditor mit Syntax-Hervorhebung, Live-Hilfe während der Eingabe und einem Funktion-Browser. Eine komfortable Übersicht über externe Daten, welche durch die Präprozessor-Anweisungen in Ihrem Skript verwendet werden, ist auch vorhanden. Hier ist ein Bildschirmfoto von Hollywoods Windows-IDE:





Wie Sie sehen können, besteht die IDE aus sechs verschiedenen Teilen:

1. Auf der oberen linken Seite der IDE gibt es den **Projekt-Browser** als Baumstruktur. Dieser ist in die beiden Bäume "Beispiele" und "Meine Projekte" geteilt. Alle "Beispiele", die mit Hollywood geliefert werden, sind in dem ersten Baum aufgeführt, während Ihre eigenen Projekte im zweiten Baum erscheinen. Um Objekte in den Baum "Meine Projekte" hinzuzufügen, drücken Sie einfach die Schaltfläche "Hinzufügen", die sich unterhalb der Baumansicht befindet. Um Elemente aus dem Baum "Meine Projekte" zu entfernen, verwenden Sie die Schaltfläche "Entfernen".
2. Auf der unteren linken Seite der IDE befindet sich die **Info-Ecke**. In diesem Bereich werden einige Informationen über das gerade aktive Hollywood-Projekt angezeigt. Dies ist derzeit nur für die Beispielpunkte verfügbar, welche mit Hollywood ausgeliefert werden. Wenn Sie eines Ihrer eigenen Projekte auswählen, wird die Info-Ecke nichts anzuzeigen.
3. Der **Editor** bildet das Herz und befindet sich in der Mitte der IDE. Der Editor zeigt automatisch die Live-Hilfe in der Statusleiste der IDE an, wenn ein bekannter Hollywood-Befehl erkannt wird. Es werden auch alle Schlüsselwörter sowie Kommentare markiert und automatisch Funktionsnamen im Funktions-Browser (siehe unten) hinzugefügt. Wenn Sie mit der rechten Maustaste im Editor draufklicken, öffnet sich ein Kontextmenü, mit dem Sie zur Definition der Funktion springen. Sie können auch die kontext-sensitive Live-Hilfe auf diese Weise aufrufen.
4. Auf der oberen rechten Seite der IDE finden Sie den **Funktion-Browser**. Diese Listenansicht enthält die Namen aller Funktionen, die von dem derzeit aktiven Projekt definiert



wurden. Die Liste wird vor zu aktualisiert, sobald Sie Ihr Skript bearbeiten. Durch einen Doppelklick auf eine Funktion in dieser Liste springen Sie automatisch in die Funktionsdeklaration.

5. Auf der rechten unteren Seite der IDE sind die **Präprozessor-Anweisungswerkzeuge**. In der ersten Registerkarte können Sie verschiedene Attribute der Präprozessor-Anweisung `@DISPLAY` konfigurieren, die das Aussehen Ihres Bildschirms steuert. Sie können zum Beispiel die Anfangsposition des Displays auf dem Desktop-Bildschirm festlegen, ob es im Vollbildmodus geöffnet werden soll oder ob das Display eine Grenze und eine bestimmte Größe haben sollte. Sie können auch den Titel des Displays (Standard: "Hollywood") setzen. Natürlich können Sie alle diese Attribute direkt im Editor mit der Präprozessor-Anweisung `@DISPLAY` konfigurieren. Wenn Sie diese Attribute mit dem Präprozessor-Anweisungswerkzeug bearbeiten, wird sofort die entsprechende Präprozessor-Anweisung `@DISPLAY` aktualisiert. Wenn es keinen gibt, wird die Präprozessor-Anweisung Ihrem Skript hinzugefügt. Die anderen Registerkarten vom Präprozessor-Anweisungswerkzeug enthalten alle externen Dateien, die vom derzeit aktiven Skript verwendet werden. Wenn Sie auf eine der Dateien klicken, wird die IDE zu dem Punkt springen, wo diese Datei im Skript deklariert wurde. Wenn Sie auf eine der Dateien doppelklicken, wird die Hollywood-IDE die Datei öffnen und anzuzeigen.
6. Auf der Unterseite der IDE finden Sie das **Hollywood-Ausgabefenster**. Immer wenn die Hollywood-IDE startet, wird die Ausgabe an dieses Fenster umgeleitet werden. Sie können auch mit dem Befehl `DebugPrint()` in dieses Fenster drucken. Wenn Sie das Kontextmenü mit der rechten Maustaste öffnen und "Clear" wählen, dann wird der Inhalt in diesem Fenster gelöscht.

Die meisten der oben vorgestellten Bestandteile der IDE sind als Dockfenster implementiert. Dies bedeutet, dass Sie diese nach Ihren persönlichen Vorlieben neu ordnen können: Sie können sie an einen anderen Ort im IDE-Fenster ziehen oder sogar aus dem IDE-Fenster bewegen. In diesem Fall werden sie als Toolbox-Fenster erscheinen. Schließlich ist es auch möglich, sie zu verstecken, wenn Sie die Funktionen nicht benötigen. Um eines dieser Fenster zu verstecken, benutzen Sie entweder das Menü "Ansicht" oder ziehen die Docks aus dem Fenster, um dann das Toolbox-Fenster zu schließen.

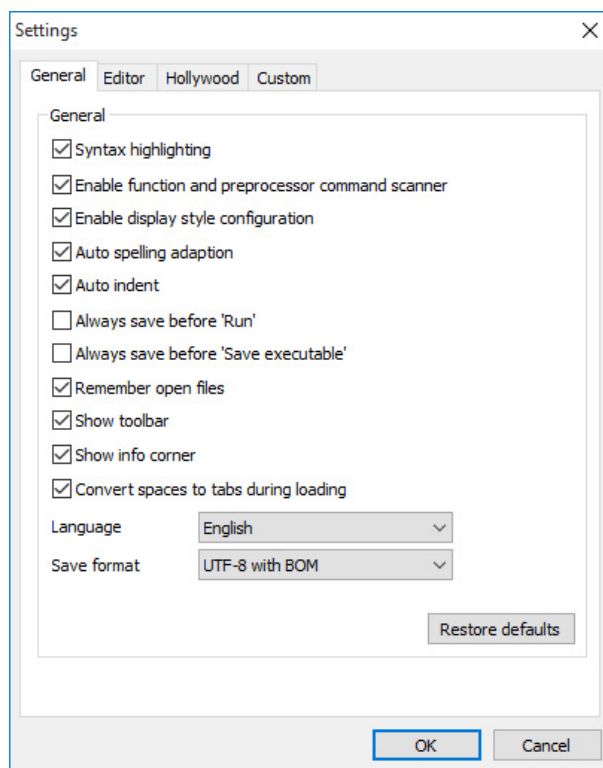
Die IDE kann entweder über die Symbolleiste gesteuert werden, die im oberen Teil des Fensters erscheint oder über das Menü. Es gibt auch Tastenkombinationen, die Sie verwenden können. Live-Hilfe wird in der Statusleiste angezeigt, wenn die Maus über eine Symbolleiste oder Menüpunkt ist. Sie können die Symbolleiste und das Menü verwenden, um das aktuelle Projekt zu testen, kompilieren, ein Video davon zu erstellen oder es an den Drucker senden. Darüber hinaus sind mehrere Standardfunktionen im Menü und in der Symbolleiste zugänglich wie Speichern ("Save"), Speichern unter ("Save as"), Suchen ("Find"), Suchen und Ersetzen ("Replace"), Kopieren ("Copy"), Ausschneiden ("Cut"), Einfügen ("Paste"), Rückgängig ("Undo"), Wiederherstellen ("Redo") und so weiter.

Die IDE kann auch mit Tastaturbefehle gesteuert werden. Hier sind einige Abkürzungen:

- F1:           Öffnet die kontextsensitive Hilfe für die Funktion/das Schlüsselwort, die/das sich an der aktuellen Cursorposition befindet.
- F2:           Springt auf die Definition der Funktion an der aktuellen Cursorposition.
- F4:           Öffnet die Datei, dessen Name sich an der aktuellen Cursorposition befindet.

- F5:** Das aktuelle Hollywood Projekt wird ausgeführt.
- Ctrl-F:** Öffnet ein Dialogfenster zum Suchen nach einer Zeichenkette im aktuellen Projekt.
- Ctrl-G:** Springt zur genannten Zeile im aktuellen Projekt.
- Ctrl-S:** Speichert das aktuelle Projekt.

Mehrere IDE-Einstellungen sind vom Benutzer konfigurierbar. Wählen Sie den Eintrag "IDE-Einstellungen ..." aus dem Menü "Datei" ("IDE-settings" aus dem Menü "File"). Es öffnet sich ein Dialogfenster, in dem Sie verschiedene Einstellungen nach Ihrem persönlichen Geschmack anpassen können. Der Einstellungen-Dialog besteht aus vier Seiten: Allgemein (General, Editor, Hollywood und Benutzerdefiniert (Custom). Hier ist ein Bildschirmfoto von der ersten Seite:



Folgendes kann auf diesen Seiten konfiguriert werden:

#### Syntax highlighting

Markieren Sie dieses Kästchen, um die Syntax-Hervorhebung im Skript-Editor der IDE zu ermöglichen. Normalerweise sollte diese Einstellung aktiviert sein, weil so Ihr Code viel besser lesbar ist. Bei sehr langsamen Systemen oder wenn Sie mit extrem großen Skripte arbeiten, könnte es notwendig sein, aus Performance-Gründen dies auszuschalten.

#### Enable function and preprocessor command scanner

Wenn Sie diese Option aktivieren, wird die IDE automatisch alle Funktionsnamen dem Funktionsbrowser hinzufügen. Außerdem werden alle Dateien von

Präprozessor-Anweisungen an die Präprozessor-Anweisungswerkzeuge weitergeleitet. Auf sehr langsamen Systemen oder mit sehr großen Skripten kann es sinnvoll sein, dies zu deaktivieren.

#### **Enable display style configuration**

Wenn diese Option aktiviert ist, können Sie die Parameter der Präprozessor-Anweisung **@DISPLAY** mit dem Präprozessor-Anweisungswerkzeug in der rechten unteren Seite der IDE zu konfigurieren. Wenn Sie das nicht wollen, lassen Sie dieses Feld unmarkiert.

#### **Auto spelling adaptation**

Mit diesem Feld können Sie festlegen, ob die IDE die Schreibweise der Befehle und Schlüsselwörter automatisch anpassen soll. Zum Beispiel, wenn Sie **waitleftmouse** schreiben und diese Option aktiviert ist, wird die Rechtschreibung diesen Befehl automatisch in **WaitLeftMouse** korrigieren.

#### **Auto indent**

Markieren Sie dieses Kästchen, wenn Sie die IDE anweisen wollen, den Code automatisch nach Kontrollstrukturen einrücken zu lassen, die einen neuen Code-Abschnitt einleiten (z.B. **If**, **While**, **Function** etc). Wenn Sie diese Option aktivieren, wird die IDE die nächste Zeile durch ein Tab-Zeichen einrücken. Dies ist sehr nützlich für die Lesbarkeit des Codes und sollte nicht ausgeschaltet werden.

#### **Always save before Run**

Wenn Sie dieses Kästchen ankreuzen, wird die IDE immer automatisch das aktuelle Projekt speichern, wenn Sie es starten/ausführen. Seien Sie vorsichtig mit dieser Option, weil Sie einige wichtige Änderungen verlieren könnten.

#### **Always save before Compile**

Wenn Sie dieses Kästchen ankreuzen, wird die IDE immer das aktuelle Projekt automatisch speichern, wenn Sie es in eine ausführbare Datei kompilieren.

#### **Remember open files**

Markieren Sie dieses Kästchen um der IDE mitzuteilen, dass es sich an alle Registerkarten erinnern soll, die in der vorherigen Sitzung geöffnet waren. In der nächsten Sitzung werden wieder alle Projekte geöffnet.

#### **Show toolbar**

Sie können diese Option verwenden, um zu konfigurieren, ob die Symbolleiste angezeigt werden soll.

#### **Show info corner**

Sie können diese Option verwenden, um zu konfigurieren, ob die Info-Ecke des Bildschirms angezeigt werden soll.

#### **Convert spaces to tabs during loading**

Mit dieser Option können Sie festlegen, ob vor dem Laden Leerzeichen in Tabulatoren umgewandelt werden sollen. Dies ist eine nützliche Option, weil es viel einfacher ist, den Code mit Tabs zu strukturieren als mit Leerzeichen. Aber seien Sie vorsichtig, denn es könnte das Layout Ihres Codes zerstören, wenn Sie eine abweichende Einstellung auf der Registerkarte zwischen der Hollywood

IDE und anderen Texteditoren verwenden. Sie können die Tabulatorbreite auf der Registerkarte "Editor" konfigurieren.

#### Keep help window on top

Wenn Sie diese Option aktivieren, erscheint Hollywoods Hilfefenster immer vorne vor allen anderen Fenstern, die zur IDE gehören.

**Language** Mit diesem Wahlknopf können Sie die Sprache von der IDE ändern (falls überhaupt andere Sprachen vorhanden sind). Sie müssen die IDE neu starten, damit diese Änderung wirksam wird.

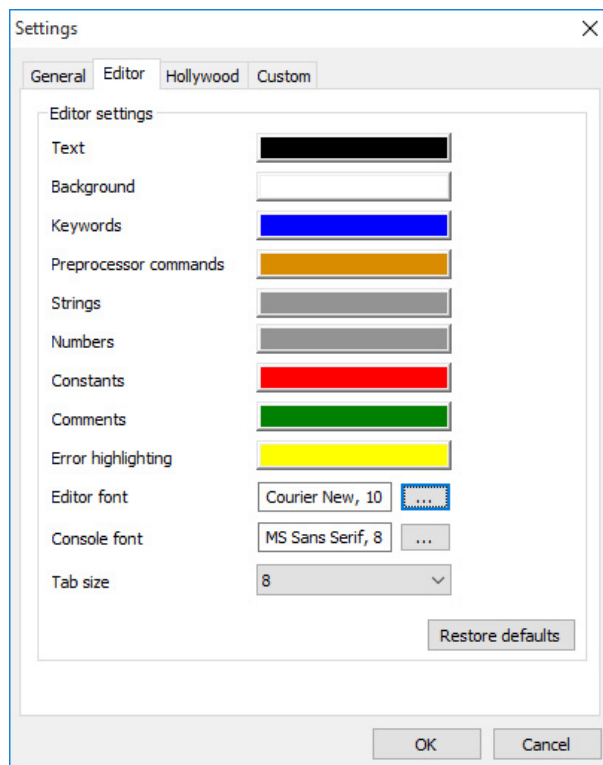
#### Save format

Hier können Sie das Ausgabeformat des Skripts einstellen. Normalerweise sollte dies immer UTF-8 mit oder ohne BOM sein. ISO 8859-1 sollte nicht mehr verwendet werden, da es bei Systemen mit einer anderen Sprach-/Ländereinstellung zu Kompatibilitätsproblemen kommen kann.

#### Restore defaults

Verwenden Sie diese Taste, um alle Einstellungen auf dieser Seite auf die Standardwerte zurückzusetzen.

Auf der zweiten Seite können Sie das Aussehen des Editors konfigurieren:



**Colors** Sie können diese Tasten verwenden, um die Farben anzupassen, die der Editor für die Syntaxhervorhebung verwendet.

**Font** Die hier angegebene Schriftart wird der Editor verwenden. Sie müssen hier eine Schrift mit fester Breite (nonporpotional/monospace) verwenden. Andernfalls gerät das Layout durcheinander.

**Console font**

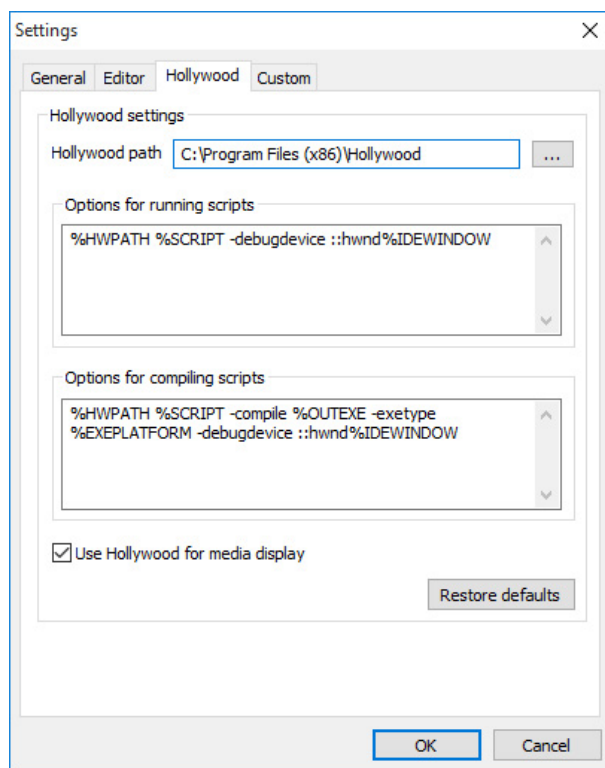
Die Schriftart, die Sie hier angeben, wird von den Konsolenfenstern verwendet. Sie können hier eine Monospace- oder Proportionalschrift verwenden.

**Tab size** Hier können Sie die Breite angeben, die vom Editor für die Tabs verwendet werden soll. Sie haben die Auswahl zwischen einer Breite von 2, 4, 6 oder 8 Leerzeichen.

**Restore defaults**

Verwenden Sie diese Taste, um alle Einstellungen auf dieser Seite auf die Standardwerte zurückzusetzen.

Auf der dritten Seite können Sie die Hollywood-Schnittstelle konfigurieren:

**Hollywood path**

Dieses Feld muss auf den Pfad festgelegt werden, in dem sich Hollywood befindet. Immer, wenn die IDE Hollywood aufruft, wird es in dem hier angegebenen Pfad suchen. Normalerweise befindet sich Hollywood im gleichen Pfad wie die IDE.

**Options for running scripts**

Die Vorlage in diesem Textfeld wird von der IDE verwendet, um ein Hollywood-Skript auszuführen. Normalerweise müssen Sie hier nichts ändern. Siehe weiter unten, was für Platzhalter hier verwendet werden können.

**Options for compiling scripts**

Die Vorlage in diesem Textfeld wird von der IDE verwendet, um ein Hollywood-Skript zu kompilieren. Normalerweise müssen Sie hier nichts ändern. Siehe weiter unten, was für Platzhalter hier verwendet werden können.

**Use Hollywood for media display**

Markieren Sie dieses Kästchen, wenn Sie die Hollywood-IDE verwenden wollen, um alle externen Mediendateien wie Bilder und Animationen zu zeigen. Es wird empfohlen, diese Option zu aktivieren. Hollywood unterstützt mehrere exotische Formate wie IFF ILBM und Protracker-Module, die nicht von den Standard-Media-Viewer unterstützt werden, die Windows benutzt.

**Restore defaults**

Verwenden Sie diese Taste, um alle Einstellungen auf dieser Seite auf die Standardwerte zurückzusetzen.

Folgende Platzhalter können als Vorlage festgelegt werden, die verwendet werden, wenn Hollywood ein Skript ausführt oder kompiliert:

**%HWPATH:** Diesen Platzhalter wird durch den Pfad vom ausführenden Programm ersetzt.

**%SCRIPT:** Diesen Platzhalter wird vom Namen des Hollywood-Skripts festgelegt, das kompiliert werden soll.

**%IDEWINDOW:**

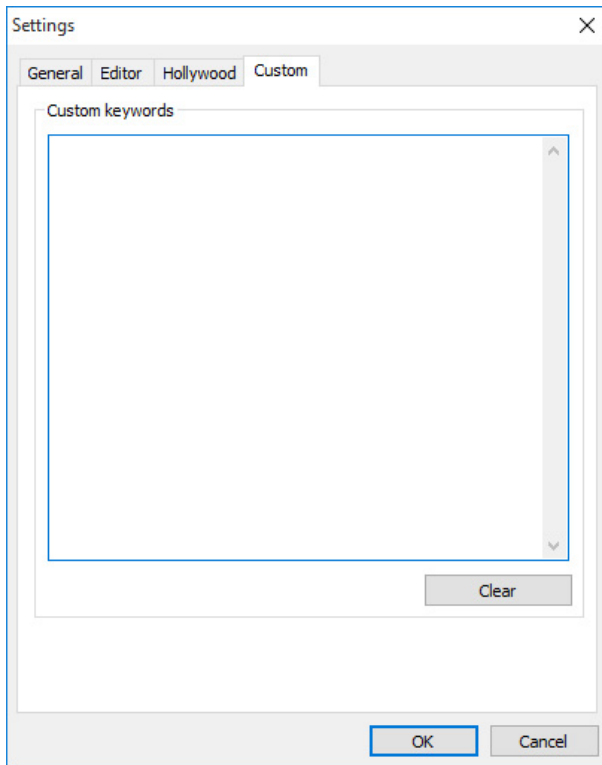
Diesen Platzhalter wird durch den Namen des IDE-Fensters ersetzt.

**%OUTEXE:** Verweist auf die Zieldatei, die vom Compiler erstellt werden soll.

**%EXEPLATFORM:**

Wird ersetzt durch die Namen von einer oder mehreren Plattformen, für die der Compiler Programme erstellen soll.

Die vierte Seite können Sie benutzerdefinierte Schlüsselwörter hinzufügen, die der Editor für die Syntax-Hervorhebung verwenden soll:

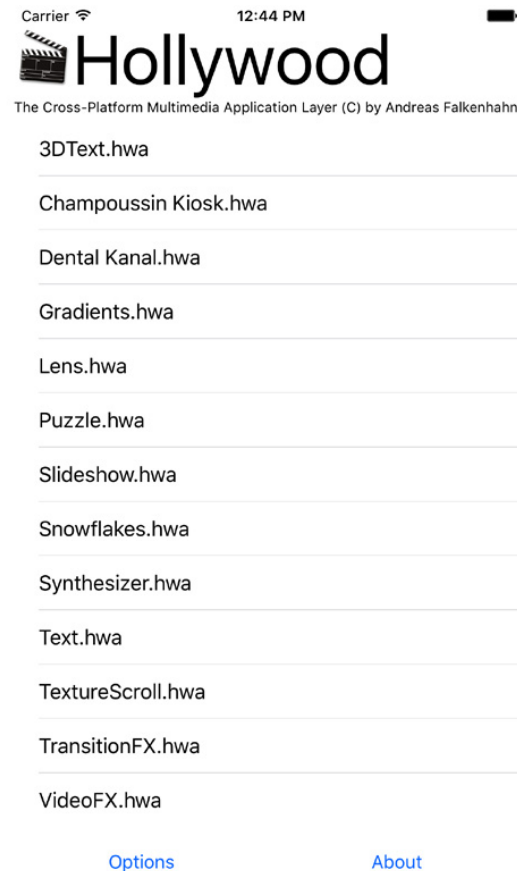
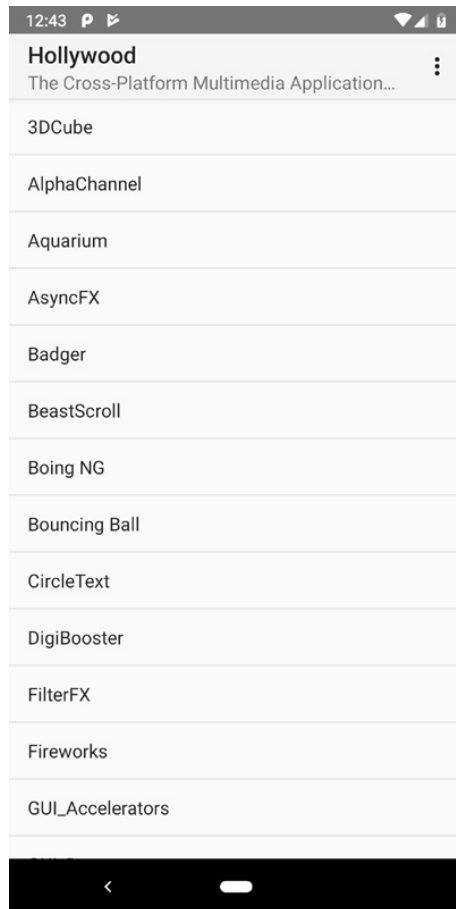


Wenn der Editor andere als die vordefinierten Schlüsselwörter markieren soll, können Sie diese Schlüsselwörter hier hinzufügen. Bitte beachten Sie, dass Schlüsselwörter immer mit einem alphabetischen Buchstaben oder einem Unterstrich beginnen. Sie können Zahlen enthalten, aber nicht am Anfang. Als einzige Sonderzeichen sind der Unterstrich, das Dollar-Zeichen und das Ausrufezeichen erlaubt. Andere Zeichen sind nicht gestattet. Es darf nur ein Schlüsselwort pro Zeile vorhanden sein.

## 2.4 Mobile Plattformen

Hollywood ist auch als Nur-Player-Version für Android und iOS verfügbar, die es Ihnen erlauben, Ihre Hollywood-Applets auf Smartphones und Tablets auszuführen. Der Hollywood-Player für Android benötigt mindestens Android 4.0 und ist bei Google Play unter dieser URL frei verfügbar: <http://play.google.com/store/apps/details?id=com.airsoftsoftwair.hollywood>. Leider ist der Hollywood-Player für iOS derzeit nicht im App Store verfügbar, da seine Möglichkeiten mit den App Store-Regeln von Apple in Konflikt geraten.

Hier sind Screenshots des Hollywood-Players für Android (links) und iOS (rechts):



Wenn Sie Ihre Hollywood-Projekte auf Ihrem mobilen Gerät ausführen wollen, müssen Sie diese zuerst als Hollywood-Applet kompilieren, sie auf Ihr Gerät kopieren und sie dann mit dem Hollywood-Player für Ihre mobile Plattform ausführen.

Alternativ gibt es auch ein Add-On namens Hollywood-APK-Compiler, mit dem Sie Ihre Hollywood-Projekte in eigenständige APK-Dateien für Android kompilieren können. Bitte besuchen Sie das Hollywood-Portal für weitere Informationen über den Hollywood-APK-Compiler: <https://www.hollywood-mal.de/>

Wenn Sie den frei verfügbaren Hollywood-Player nutzen möchten, lesen Sie diese Schritt-für-Schritt-Anleitung, um Ihre Hollywood-Projekte auf Ihr Mobilgerät zu bekommen:

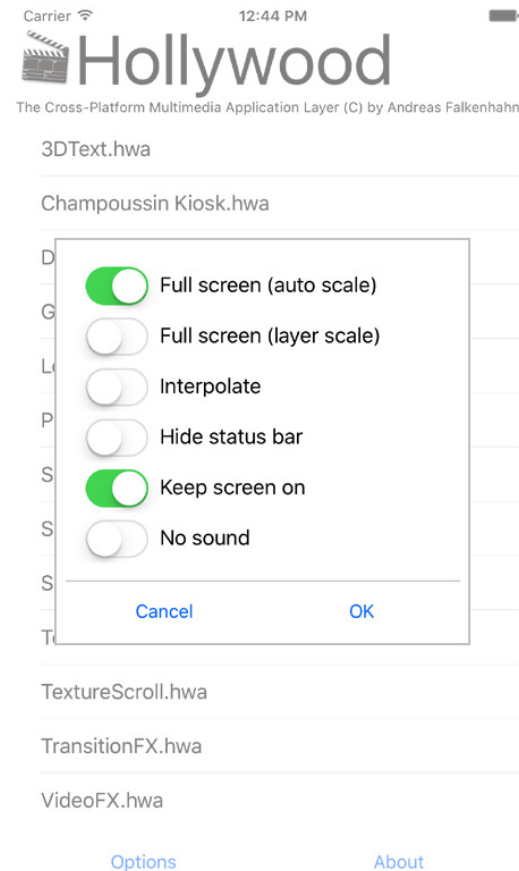
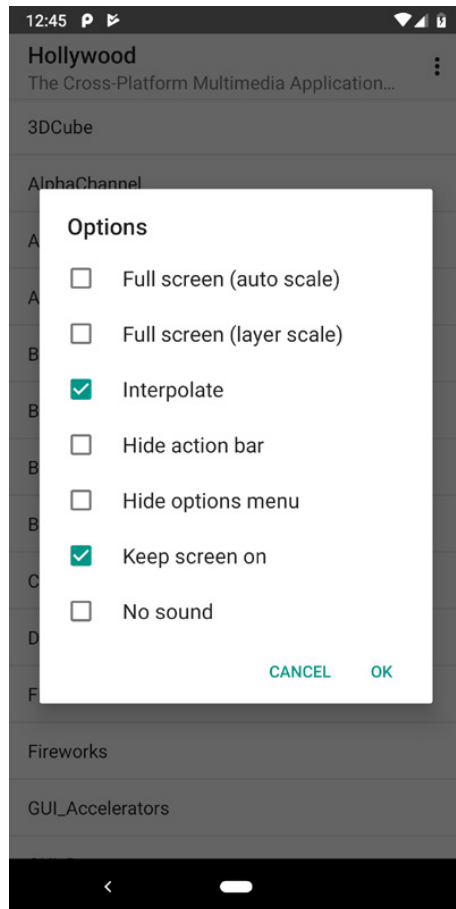
1. Erstellen Sie Ihre Hollywood-Projekt als Hollywood-Applet (\*.hwa) in der Hollywood-GUI, im Designer oder von der Kommandozeile aus.
2. Auf Android-Systeme: Kopieren Sie die \*.hwa Datei in das Verzeichnis **Hollywood** der SD-Karte auf Ihrem Android-Gerät. Das Verzeichnis **Hollywood** wird automatisch erstellt, wenn Sie zum ersten Hollywood auf Ihrem Android-Gerät starten. Wenn es nicht vorhanden ist, erstellen Sie es von Hand. Es muss im Stammverzeichnis der SD-Karte sein.
3. Auf iOS-Systeme: Verwenden Sie iTunes, um die Datei \*.hwa in das Dokumentenverzeichnis vom App Hollywood-Player zu kopieren.



4. Wenn das Applet ohne externen Dateien geladen wird oder alle Dateien ins Applet eingebunden wurden, dann lesen Sie bei Punkt 7 weiter.
5. Wenn Ihr Applet externen Dateien lädt, erstellen Sie ein neues Verzeichnis für Ihr Applet innerhalb des **Hollywood**-Verzeichnisses (auf Android) oder innerhalb der APP des Hollywood-Player im Dokumentverzeichnis (auf iOS). Der Verzeichnisname muss mit dem Namen Ihres Applets ohne die Dateierweiterung (**\*.hwa**) identisch sein. Z.B. wenn Ihr Applet **My cool game.hwa** heißt, müssen Sie das Verzeichnis **My cool game** innerhalb des Verzeichnisses Hollywood oder Dokumentenverzeichnis erstellen.
6. Kopieren Sie alle für Ihr Applet erforderlichen Dateien in das neu erstellte Verzeichnis.
7. Starten Sie den Hollywood-Player auf Ihrem Gerät. Eine Liste aller gefundenen Applets wird angezeigt. Wählen Sie Ihr Applet und berühren Sie dann die Play-Taste in der Steuerleiste am unteren Rand des Bildschirms.
8. Nun sollte Ihr Applet starten!

Alternativ können Sie auf Android auch die Hollywood-Applet zu einem beliebigen Ort auf der SD-Karte kopieren und starten, indem Sie einfach die **\*.hwa** Datei direkt von Ihrem Dateimanager auswählen. In diesem Fall werden alle externen Dateien vom Applet aus der Position gesucht werden, in dem sich die **\*.hwa** Datei befindet. Hollywood wechselt zum Laden der externen Dateien nicht automatisch in das Unterverzeichnis mit demselben Namen wie die **\*.hwa** Datei, wenn sich das Applet außerhalb des **/sdcard/Hollywood** Ordner befindet.

Sie können einige Optionen konfigurieren, indem Sie die Taste "Options" drücken. Den Options-Bildschirm von Android sehen Sie links, den von iOS rechts:



Folgende Optionen stehen zur Verfügung:

**Full screen (auto scale):**

Führt das Skript mit dem Autoskalierungs-System im Vollbildmodus aus. Dies ist die empfohlene Methode um sicherzustellen, dass Skripte, die für eine andere Auflösung entworfen wurden, den gesamten Bildschirm Ihres Mobilgeräts abdecken. Wenn Ihr Skript Ebenen und Vektorgrafiken verwendet, sollten Sie stattdessen das Ebenenskalierungs-System verwenden, um in jeder Auflösung perfekt scharfe Grafiken zu erhalten (siehe unten).

**Full screen (layer scale):**

Führt das Skript mit dem Ebenenskalierungs-System im Vollbildmodus aus. Dies wird nur für Skripte unterstützt, die Ebenen verwenden. Es ist langsamer als das Autoskalierungs-System, kann aber zu perfekt scharfen Grafiken in jeder Auflösung führen, wenn Ihr Skript nur Vektorgrafiken verwendet. Für Skripte, die keine Ebenen verwenden, sollten Sie das Autoskalierungs-System verwenden (siehe oben). Beachten Sie, dass Sie auch das Autoskalierungs-System für Skripte verwenden können, die Ebenen verwenden, aber dann ist die Qualität nicht so gut wie bei der Ebenenskalierung, wenn Ihr Skript Vektorgrafiken verwendet. Aus diesem Grund empfiehlt es sich, die Ebenenskalierung nur für Skripte zu verwenden, die Ebenen und Vektorgrafiken verwenden. Für Skripte, die Ebe-

nen und Grafiken im Pixelformat verwenden, sollten Sie die Autoskalierung verwenden, weil sie schneller ist.

**Interpolate:**

Verwendet beim Skalieren Antialiasing-Interpolation. Dies wird empfohlen, um eine glattere Grafik mit weniger Skalierungsartefakten zu erhalten. Dies ist zwar langsamer, sieht aber besser aus.

**Hide action bar**

Blendet die Aktionsleiste (ActionBar) auf Android aus. Dies wird für Skripte empfohlen, die keine Benutzerinteraktion über Optionen in der Android-Aktionsleiste benötigen.

**Hide status bar:**

Wenn eine Titelleiste unter iOS vorhanden ist, wird diese ausgeblendet. Die Aktivierung dieser Option wird einen Vollbildschirm zur Folge haben.

**Hide options menu:**

Über das Optionsmenü werden keine Änderungen mehr an der Display-Konfiguration über das Optionsmenü in der Aktionsleiste zugelassen. Wenn Sie dies aktivieren, ist in der Aktionsleiste kein Optionsmenü mehr sichtbar. Dies wird nur auf Android unterstützt. Auf iOS unterstützt Hollywood das Ändern von Optionen nicht, während ein Skript läuft.

**Keep screen on:**

Zwingt das mobile Gerät seinen Bildschirm anzulassen, selbst wenn keine Benutzeraktivitäten stattfinden. Normalerweise schalten mobile Geräte nach einer gewissen Inaktivitätszeit automatisch in den Batteriesparmodus. Wenn Sie diese Option auswählen, bleibt der Bildschirm an.

**No sound:** Wenn Sie diese Option aktivieren, wird Hollywood stumm gestartet.

Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), [Seite 33](#), für detailliertere Informationen zu den oben aufgeführten Optionen.

Die mobilen Versionen von Hollywood haben im Vergleich zu den Desktop-Versionen einige Besonderheiten. Hier ist eine Liste für Android und iOS:

1. Auf Android und iOS hat Hollywood native Unterstützung für Ogg Vorbis und Ogg Theora. Sie brauchen keine Plugins fürs Laden von diesen Dateiformaten.
2. Sie können eine virtuelle Tastatur anzeigen, indem Sie den Befehle `ShowKeyboard()` benutzen. Mit `HideKeyboard()` können Sie die virtuelle Tastatur schließen. Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), [Seite 483](#), für Details.
3. Sie können mit dem Ereignis-Handler `OrientationChange` herausfinden, wann der Benutzer das Gerät dreht. Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), [Seite 483](#), für Details.
4. Sie können auch eine Ausrichtung festlegen, die Ihr Skript verwenden soll, indem Sie den Tag "Orientation" angeben, das von der Präprozessoranweisung `@DISPLAY` unterstützt wird. In diesem Fall reagiert Hollywood nicht auf Orientierungsänderungen, wenn der Benutzer das Gerät dreht. Stattdessen behält es den Orientierungsmodus, den Sie im Tag "Orientation" angegeben haben. Siehe [Abschnitt 24.8 \[DISPLAY\]](#), [Seite 370](#), für Details.

5. Bei Android können Sie auch zwischen Vollbild- und Nicht-Vollbildmodus wechseln, indem Sie das Optionsmenü in der Aktionsleiste verwenden.

Natürlich gibt es einige Einschränkungen in den mobilen Versionen von Hollywood. Im Gegensatz zu den Desktop-Versionen von Hollywood unterstützt die iOS-Version von Hollywood nicht mehrere Displays, sondern nur ein einziges. Auch können unter iOS Displays keine Menüs verwenden. Außerdem ist es nicht möglich, den Mauszeiger zu wechseln, da es typischerweise keine auf Android und iOS gibt. Es ist auch nicht möglich, transparente BGPics zu verwenden und das Feld `Desktop` wird in der Anweisung `@DISPLAY` ebenfalls nicht unterstützt.

Hier ist eine Liste mit Befehlen, die nicht in den mobilen Versionen von Hollywood unterstützt werden:

- `ActivateDisplay()` (nur unter iOS nicht unterstützt)
- `ChangeDisplayMode()` (nur unter iOS nicht unterstützt)
- `CloseDisplay()` (nur unter iOS nicht unterstützt)
- `CreatePointer()`
- `CreatePort()`
- `DebugPrompt()`
- `FontRequest()`
- `FreePointer()`
- `GetEnv()`
- `GetFileArgument()`
- `GetRawArguments()`
- `HideDisplay()` (nur unter iOS nicht unterstützt)
- `HidePointer()`
- `MovePointer()`
- `OpenDisplay()` (nur unter iOS nicht unterstützt)
- `SelectDisplay()` (nur unter iOS nicht unterstützt)
- `SendMessage()`
- `SetPointer()`
- `SetEnv()`
- `ShowDisplay()` (nur unter iOS nicht unterstützt)
- `ShowPointer()`
- `UnsetEnv()`

Es gibt auch einige Befehle, die exklusiv für die mobilen Versionen von Hollywood verfügbar sind:

- `CallJavaMethod()` (nur Android)
- `GetAsset()` (nur Android)
- `HideKeyboard()`
- `ImageRequest()` (nur Android)
- `PerformSelector()` (nur iOS)
- `PermissionRequest()` (nur Android)
- `ShowKeyboard()`
- `ShowToast()`
- `Vibrate()` (nur Android)

Die Android-Version von Hollywood unterstützt auch Hollywood-Plugins. Sie müssen sie in das Verzeichnis `Hollywood/Plugins` auf Ihrer SD-Karte kopieren. Hollywood wird diesen Standort bei jedem Start scannen und alle Plugins von dort laden. Die iOS-Version von Hollywood unterstützt derzeit keine Plugins.

Wenn Ihr Skript auf einer mobilen Plattform extrem langsam läuft, müssen Sie Ihre Zeichnungstechnik ändern. Das Auffrischen des Bildschirms kann auf einem mobilen Gerät ziemlich viel Ressourcen in Anspruch nehmen, weshalb Sie versuchen sollten, die Häufigkeit der Bildschirmaktualisierungen zu minimieren. Der beste Weg, dies zu tun, ist das doppelt gepufferte Zeichnen mit den Befehlen `BeginDoubleBuffer()` und `Flip()`. Denn viele Zeichnungsoperationen, die nur kleine Bereiche betreffen, verlangsamen die Grafikmodule der mobilen Geräten. So sollten Sie versuchen, so viele Zeichnungsoperationen wie möglich in einem einzigen Aufruf zu kombinieren. Ein Doppelpuffer ist die beste Lösung für dieses Problem. Wenn Sie aus irgendeinem Grund keinen Doppelpuffer verwenden können (z.B. weil Sie Sprites oder Ebenen verwenden), dann kapseln Sie alle Zeichnungsbefehle, die zum Zeichnen eines Einzelbildes benötigt werden, in einem Abschnitt mit den Befehlen `BeginRefresh()` und `EndRefresh()`. Dann sollte die Leistung bei mobilen Geräten viel besser werden. Siehe [Abschnitt 28.4 \[BeginRefresh\]](#), [Seite 569](#), für Details.



## 3 Konsole benutzen

### 3.1 Konsolenmodus

Hollywood kann auch von der Konsole aus verwendet werden. Tatsächlich sind die GUIs, die mit den Amiga-, Linux- und macOS-Versionen geliefert werden, sowie die IDE, die mit der Windows-Version von Hollywood geliefert wird, nur der Oberbau (Frontends) für den Hollywood-Interpreter, der ein Konsolenprogramm ist. Somit können Sie Hollywood auch von der Konsole aus nutzen und natürlich auch Konsolenprogramme mit Hollywood entwickeln, da Hollywood über eine umfangreiche Konsolenbibliothek verfügt. Hier ist eine Übersicht, wo Sie den Hollywood-Interpreter finden, der von der Konsole aus gestartet werden kann:

- AmigaOS: Auf AmigaOS und kompatiblen Systemen wird der Hollywood-Interpreter in `Hollywood:System/Hollywood` installiert. Da das Hollywood-Installationsprogramm `Hollywood:System` zu Ihrem Pfad hinzufügt, können Sie den Hollywood-Interpreter von der Konsole aus starten, indem Sie einfach `Hollywood` eingeben.
- Linux: Unter Linux heißt der Hollywood-Interpreter einfach `Interpreter` und befindet sich im Hauptverzeichnis Ihrer Hollywood-Installation. Es gibt auch eine Datei namens `Hollywood` in diesem Verzeichnis, aber das ist nur das GUI-Frontend für den Hollywood-Interpreter. Der Hollywood-Interpreter ist einfach als `Interpreter` bekannt.
- macOS: Unter macOS finden Sie den Hollywood-Interpreter im Verzeichnis `Contents/Resources` von `Hollywood.app`. Der Hollywood-Interpreter wird als eigenes App-Paket mit dem Namen `HollywoodInterpreter.app` gespeichert. Um den Hollywood-Interpreter auf macOS von der Konsole aus zu starten, müssten Sie etwa so vorgehen:

```
cd Hollywood.app/Contents/Resources
./HollywoodInterpreter.app/Contents/MacOS/Hollywood test.hws
```

- Windows: Unter Windows sind die Dinge etwas komplizierter, da Windows zwischen Konsolenprogrammen und Nicht-Konsolenprogrammen unterscheidet. Somit ist Hollywood eigentlich in zwei Varianten unter Windows verfügbar: In einer Version, die für die Verwendung von der Konsole gedacht ist, und in einer Version, die für die Verwendung ohne Konsole gedacht ist. Die Konsolenversion von Hollywood heißt `Hollywood_Console.exe` und Sie finden sie in dem Verzeichnis, in dem Sie Hollywood installiert haben, normalerweise `C:/Program Files/Hollywood` oder `C:/Programme/Hollywood`. Darüber hinaus ist auch eine Nicht-Konsolenversion von Hollywood für Windows verfügbar. Diese Version heißt `Hollywood.exe` und befindet sich im selben Verzeichnis wie `Hollywood_Console.exe`.

Die Hollywood-IDE verwendet immer die Nicht-Konsolenversion von Hollywood, d.h. `Hollywood.exe`. Wenn Sie jedoch eine Konsolenausgabe von Hollywood haben möchten, müssen Sie die Konsolenversion von Hollywood verwenden, d.h. `Hollywood_Interpreter.exe`. Sie können das manuell von einer Windows-Konsole aus wie folgt starten:

```
cd "C:/Program Files/Hollywood"
Hollywood_Console.exe test.hws
```

Die Kenntnis des Unterschieds zwischen Konsolen- und Nicht-Konsolenprogrammen unter Windows ist auch wichtig, wenn es um die Verteilung Ihres Programms geht: Da Windows zwischen Konsolen- und Nicht-Konsolenprogrammen unterscheidet, kann Hollywood auch zwei verschiedene Arten kompilieren: Windows-Executables, die Konsolenprogramme sind und Windows-Executables, die keine Konsolenprogramme sind. Wenn Sie ein Konsolenprogramm für Windows kompilieren möchten, müssen Sie das Argument `-consolemode` an den Compiler übergeben. Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), [Seite 33](#), für Details. Beachten Sie, dass diese Option derzeit nicht in der Hollywood-IDE oder den Hollywood GUI-Frontends verfügbar ist. Wenn Sie ein Windows-Konsolenprogramm kompilieren möchten, müssen Sie Hollywood von der Konsole aus ausführen oder Argument `-consolemode` manuell zur IDE- oder Hollywood-GUI-Konfiguration hinzufügen.

Beachten Sie außerdem, dass auch Nicht-Konsolenprogramme eine Konsole unter Windows öffnen können. Dies kann mit dem Befehl `OpenConsole()` erfolgen. Siehe [Abschnitt 33.44 \[OpenConsole\]](#), [Seite 681](#), für Details.

Sobald Sie wissen, wie man Hollywood von der Konsole aus startet, können Sie eine Liste aller verfügbaren Optionen ausgeben, indem Sie ihm das Argument `-help` übergeben. Unter Windows könnte dies folgendermaßen erfolgen:

```
cd "C:/Program Files/Hollywood"
Hollywood_Console.exe -help
```

Auf Linux so:

```
cd <Hollywood-installation-directory>
./Interpreter -help
```

Auf macOS wie hier:

```
cd /Applications/Hollywood.app/Contents/Resources
./HollywoodInterpreter.app/Contents/MacOS/Hollywood -help
```

Auf AmigaOS hingegen wie dieses Beispiel:

```
Hollywood -help
```

Wenn Sie `-help` an Hollywood übergeben, wird eine umfassende Liste aller verfügbaren Konsolenargumente ausgegeben. Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), [Seite 33](#), für Details. Wenn Sie das Argument `-help` weglassen, öffnet Hollywood einen Dateidialog, der Sie auffordert, ein Hollywood-Skript oder -Applet auszuwählen, das ausgeführt werden soll.

Wenn Sie ein Hollywood-Skript von der Konsole aus starten möchten, können Sie unter Windows die folgenden Befehle verwenden:

```
cd "C:/Program Files/Hollywood"
Hollywood_Console.exe script.hws
```

Das Gleiche ist auf den anderen Plattformen möglich, siehe oben.

Es ist wichtig zu wissen, dass alle Hollywood-Funktionen der GUIs auch in der Konsole zur Verfügung stehen. Schließlich sind die GUIs/IDEs von Windows, Amiga, Linux und macOS nur Verbindungsprogramme für das konsolenbasierte Hauptprogramm von Hollywood. So können Sie auch alles über die Befehlszeile erledigen. Zum Beispiel sieht die Zeile so aus, wenn Sie mit Linux die Datei `test.hws` in ein AmigaOS3 Programm kompilieren würden:

```
cd <Hollywood-installation-directory>
```



```
./Interpreter test.hws -compile ~/MyTest_AmigaOS3 -exetype classic
```

Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), [Seite 33](#), für eine detaillierte Beschreibung aller Befehlszeilenparameter.

Beachten Sie, dass Hollywood auf AmigaOS und kompatiblen Systemen bei der Installation automatisch als Pfad hinzugefügt wird. So können Sie einfach Hollywood in die Konsole eingeben und Hollywood wird gestartet - egal wo Sie das Programm installiert haben.

## 3.2 Konsolenargumente

Wenn Sie aus irgendeinem Grund die GUI nicht verwenden wollen, können Sie Hollywood auch von der Konsole aus benutzen. Tippen Sie Hollywood ohne Argumente in die Konsole, wird ein Dateiauswahlfenster geöffnet, damit Sie ein Skript oder ein Applet auswählen können. Brauchen Sie eine Liste mit den von Hollywood unterstützten Konsolenargumente, dann geben Sie `-help` ein. Die Schablone ist wie folgt:

```
Hollywood file.hws [-alldisplays] [-askdisplaymode] [-audiodevice name]
[-autofullscreen] [-autoscale] [-backfill type] [-borderless]
[-brush brush] [-brushfile file] [-compile file] [-compress]
[-consolemode] [-cxkey hotkey] [-debugoutput] [-depth depth]
[-disableblanker] [-dpiaware] [-encoding enc] [-endcolor color]
[-exetype type] [-exportcommands file] [-exportconstants file]
[-exporthelpstrings] [-exportplugins file] [-exportpreprocs file]
[-fakefullscreen] [-fixed] [-fitscale] [-forceflush] [-forcesound]
[-formaterror] [-fullscreen] [-fullscreenscale] [-globalplugins] [-help]
[-hideoptionsmenu] [-hidepointer] [-hidetitlebar] [-icon16x16 file]
[-icon24x24 file] [-icon32x32 file] [-icon48x48 file] [-icon64x64 file]
[-icon96x96 file] [-icon128x128 file] [-icon256x256 file]
[-icon512x512 file] [-icon1024x1024 file] [-keepproportions]
[-keepscreenon] [-layerfullscreen] [-layerscale] [-legacyaudio]
[-linkfiles file] [-linkfonts file] [-linkplugins list] [-locksettings]
[-mastervolume vol] [-maximized] [-moderequester] [-monitor num]
[-nativeunits] [-nobackfill] [-nochdir] [-nocommodity] [-nodebug]
[-nodocky] [-nohardwarescale] [-nohide] [-nokeepproportions]
[-nolegacyaudio] [-noliveresize] [-nomodeswitch] [-nomousehook]
[-noscaleengine] [-noscaleswitch] [-nosmoothscale] [-nosound]
[-nostyleoverride] [-numchannels chans] [-overrideplacement]
[-overwrite] [-pictrans transparency] [-picxpos x] [-picypos y]
[-prnterror] [-pubscreen name] [-quiet] [-requireplugins list]
[-requiretags tags] [-resourcemonitor] [-scalefactor s] [-scalepicture]
[-scaleswitch] [-scalewidth width] [-scaleheight height]
[-scrwidth width] [-scrheight height] [-setconstants list] [-sizeable]
[-skipplugins mask] [-smoothscale] [-softtimer] [-softwarerenderer]
[-startcolor color] [-stayactive] [-systemscales] [-tempdir path]
[-usequartz] [-usewpa] [-videofps fps] [-videoout file] [-videopointer]
[-videoquality quality] [-videostategy strategy] [-vsync] [-window]
[-winwidth width] [-winheight height] [-wpamode mode] [-xserver name]
```

**Wichtige Informationen:** Die Mehrheit dieser Argumente wird auch von jedem von Hollywood kompilierten Programmen unterstützt. Wenn Sie wollen, dass Ihr kompiliertes Programm diese Argumente nicht benutzt, verwenden Sie das `-locksettings` Argument.

**Wichtige Informationen #2:** Die meisten der Argumente können auch ohne Konsole verwendet werden. Siehe [Abschnitt 3.3 \[Konsolenargumente ohne Konsole\]](#), Seite 55, für Details.

Hier sind nun die ausführlichen Beschreibungen für jedes Argument:

**-alldisplays:**

Voreingestellt werden Befehlsargumente wie `-borderless` oder `-sizeable` nur auf das erste Display angewandt. Wenn Sie `-alldisplays` verwenden, werden alle Hollywood Displays durch Befehlsargumente, welche den Displaystil ändern, beeinflusst.

**-askdisplaymode:**

Wenn Sie dieses Argument angeben, wird Hollywood ein Dialogfenster öffnen und Sie fragen, ob Sie das Programm im Fenster- oder im Vollbildmodus betreiben möchten.

**-audiodevice name:**

Mit diesem Argument wird das "ALSA Sound Device" angegeben, welches Hollywood für Audioausgaben benutzen soll. Sie müssen dieses Argument nur benutzen, wenn Sie ein anderes als das Standard "ALSA Sound Device" verwenden wollen. Normalerweise müssen Sie dieses Argument gar nicht gebrauchen. [Nur Linux]

**-autofullscreen:**

Dadurch wird das Display mithilfe des automatischen Skalierungssystems in den Vollbildmodus versetzt, anstatt die Auflösung des Monitors zu ändern, aber nur, wenn Hollywood auf Systemen ausgeführt wird, die GPU-beschleunigte Skalierung unterstützen. Auf allen anderen Plattformen wird ein normaler Vollbildmodus verwendet, d.h. Hollywood passt die Auflösung des Monitors an die aktuellen Bildschirmgröße an. Derzeit wird die GPU-beschleunigte Skalierung unter Windows, macOS, Android und iOS unterstützt, was bedeutet, dass auf diesen Plattformen keine Änderung der Monitorauflösung stattfindet, da Hollywood die Grafiken einfach skalieren kann, um sie an die aktuelle Bildschirmgröße anzupassen. Auf AmigaOS-kompatiblen Systemen und Linux wird es bei diesem Modus jedoch immer noch eine Änderung der Monitorauflösung geben, da Hollywood auf diesen Plattformen keine GPU-beschleunigte Skalierung unterstützt.

**-autoscale:**

Wenn die automatische Skalierung aktiviert ist, wird Hollywood Ihrem Skript vorgeben, dass es immer noch in seiner gewohnten Auflösung läuft, aber in Wirklichkeit wird es hoch- oder herunterskaliert (abhängig von der gewählten Skalierungsauflösung). Sie können die anfängliche automatische Skalierungsauflösung angeben, indem Sie die Argumente `-scalewidth` und `-scaleheight` oder `-scalefactor` oder `-systemscales` verwenden. Die Skalierungsauflösung kann vom Benutzer jederzeit durch Größenänderung des Hollywood-Fensters geändert werden (vergessen Sie nicht, die Skalierbarkeit des Fensters mit der

Präprozessor-Anweisung **@DISPLAY** oder dem Argument **-sizeable** einzuschalten). Wenn Sie beim Starten nicht **-scalewidth** und **-scaleheight** oder eines der beiden Argumente **-scalefactor** oder **-systemscales** angeben, wird das Skript ohne automatische Skalierung gestartet. Aber die automatische Skalierung wird aktiviert, sobald der Benutzer die Fenstergröße ändert. Möchten Sie beim Skalieren die Kantenglättung (Antialias) aktivieren, benutzen Sie dafür das Argument **-smoothscale** (ist aber langsamer). Hollywood unterstützt noch ein zweites automatisches Skalierungssystem, welches durch Angabe des Arguments **-layerscale** aktiviert werden kann (siehe unten). Siehe [Abschnitt 24.24 \[Informationen über das Skalierungssystem\]](#), Seite 402, für Details.

**-backfill type:**

Dieses Argument erlaubt Ihnen, einen Hintergrund für die Displays von Hollywood anzugeben. Falls Sie dieses Argument angeben, wird Hollywood den gesamten Bildschirm füllen. **type** kann eines der folgenden Schlüsselworte sein:

- color**      füllt den Hintergrund mit der in **-startcolor** Argument definierten Farbe.
- picture**    stellt den in **-brush** oder **-brushfile** Argument definierten Pinsel als zentriertes Hintergrundbild dar. Falls Sie zusätzlich das Argument **-startcolor** angeben, wird der Hintergrund mit dieser Farbe ausgefüllt. Wenn Sie außerdem noch das Argument **-endcolor** angeben, wird als Hintergrund ein Farbverlauf zwischen **-startcolor** und **-endcolor** benutzt.
- gradient**   stellt einen Farbverlauf dar (mit einem Übergang von der Farbe, die in **-startcolor** angegeben ist bis zu der in **-endcolor** angegebenen Farbe)
- texture**    stellt den Pinsel, der im Argument **-brush** oder **-brushfile** angegeben ist, als Textur dar.

Wie Sie sehen, erfordern alle Hintergrundtypen ein zusätzliches Argument als Parameter. Sie müssen die Argumente **-brush**, **-brushfile**, **-startcolor** und **-endcolor** dafür verwenden (wie oben beschrieben).

**-borderless:**

Wenn Sie dieses Argument setzen, wird Hollywood ein rahmenloses Fenster öffnen. Dies ist besonders für durchsichtige (transparente) Fenster nützlich.

**-brush id:**

Nur erforderlich in Verbindung mit **-backfill**. Gibt die Nummer des Pinsels an, der für **texture** oder **picture** benutzt werden soll. Sie können stattdessen auch das Argument **-brushfile** angeben (falls Sie den Pinsel nicht in Ihrem Skript angeben wollen).

**-brushfile file:**

Nur erforderlich in Verbindung mit **-backfill**. Gibt den Dateinamen des Pinsels an, der für **texture** oder **picture** benutzt werden soll. Sie können stattdessen auch das Argument **-brush** angeben.

**-compile file:**

Wenn Sie dieses Argument angeben, wird Hollywood Ihr Skript in ein ausführbares Programm übersetzen. Sie müssen dieses Argument verwenden, wenn Sie Ihr Skript veröffentlichen wollen. Ihr Skript wird nicht ausgeführt, sondern kompiliert und als die Datei gespeichert, die Sie in `file` angegeben haben. Verwenden Sie das Argument `-exetype`, um die Plattform anzugeben, für die Sie Ihr Skript speichern möchten.

**-compress:**

Damit können Sie Hollywood Projekte, sei es Applets oder Programme, komprimieren. Dieses Argument kann nur im Zusammenhang mit `-compile` verwendet werden.

**-consolemode:**

Wenn Sie dieses Argument angeben, kompiliert Hollywood ein Programm, welches im Konsolenmodus unter Windows ausgeführt wird. Unter Windows wird zwischen Konsolen- und Nicht-Konsolenprogrammen unterschieden. Wenn Sie also ein Programm für die Konsole kompilieren möchten, müssen Sie Hollywood explizit dazu auffordern. Sie können dies tun, indem Sie dieses Argument übergeben. Beachten Sie, dass dieses Argument offensichtlich nur verarbeitet wird, wenn ebenfalls `-compile` auch angegeben wird. Beachten Sie noch, dass `-consolemode` auch in der Präprozessor-Anweisung `@OPTIONS` verfügbar ist. Siehe [Abschnitt 50.25 \[OPTIONS\]](#), Seite 1107, für Details. Siehe [Abschnitt 3.1 \[Konsolenmodus\]](#), Seite 31, für Details.

**-cxkey hotkey:**

Dieses Argument kann benutzt werden, um einen systemweiten Hotkey für Ihr Programm zu installieren. Jedesmal wenn der Benutzer die definierte Tastenkombination drückt, wird ein Ereignis ausgelöst, welches mit der definierten Funktion in `InstallEventHandler()` bearbeitet wird. [Nur Amiga OS]

**-debugoutput:**

Mit diesem Argument schalten Sie die Debugausgabe an. Es hat die gleiche Wirkung, wie wenn Sie den Befehl `DebugOutput()` am Skriptanfang aufrufen.

**-disableblanker:**

Dieses Argument kann dafür benutzt werden, um den Bildschirmschoner auszuschaalten, während Hollywood am laufen ist.

**-dpiaware:**

Dieses Argument wird nur unter Windows unterstützt. Wenn Sie dieses Argument übergeben, startet Hollywood im DPI-Awareness-Modus. Das bedeutet, dass das Betriebssystem Hollywood nicht automatisch an die DPI-Werte des Monitors anpassen muss. Wenn `-dpiaware` nicht angegeben ist, wendet Hollywood die Skalierung auf Monitoren mit hoher DPI automatisch an, damit das Display auf ihnen nicht zu klein erscheint. Ein Display mit 640 x 480 Pixeln wird beispielsweise auf einem Monitor mit hoher DPI-Zahl wirklich sehr klein erscheinen. Durch die automatische Anpassung der Displays an die DPI des Monitors versucht Hollywood dies zu vermeiden. Diese Skalierung kann jedoch dazu führen, dass Displays auf Monitoren mit hohem DPI unscharf erscheinen. Wenn

Sie das nicht möchten, übergeben Sie das Argument `-dpiaware`. Beachten Sie jedoch, dass Sie dann darauf achten müssen, dass Ihr Display auf Monitoren mit hohen DPI-Werten korrekt angezeigt wird. Dies können Sie beispielsweise durch Setzen des Tags `SystemScale` in der Präprozessor-Anweisung `@DISPLAY` tun. Beachten Sie zusätzlich, dass `-dpiaware` auch in der Präprozessor-Anweisung `@OPTIONS` verfügbar ist. Siehe [Abschnitt 50.25 \[OPTIONS\]](#), [Seite 1107](#), für Details.

**-encoding enc:**

Dieses Argument kann verwendet werden, um die Zeichencodierung des Skripts festzulegen. `enc` kann eines der folgenden Schlüsselworte sein:

**utf8:** Die Zeichencodierung vom Skript ist UTF-8 (mit oder ohne BOM). Dies ist auch der Standard und sollte wo immer möglich verwendet werden.

**iso8859\_1:**

Die Zeichencodierung des Skripts ist ISO 8859-1. Beachten Sie, dass aus historischen Gründen Hollywood auf AmigaOS und kompatiblen nicht die ISO 8859-1 Zeichencodierung verwendet wird, sondern die System-Standard-Zeichencodierung. `iso8859_1` wird Hollywood in den Legacy-Modus versetzen und somit sollte Ihr Skript voll kompatibel mit Hollywood-Versionen vor 7.0 sein. Da jedoch der ISO 8859-1-Modus mehrere Nachteile aufweist, wird nicht empfohlen, diesen Legacy-Modus dauerhaft zu verwenden. Stattdessen sollten Sie Ihre Skripte anpassen, um im Unicode-Modus korrekt zu arbeiten.

Beachten Sie, dass nicht empfohlen wird, `iso8859_1` zu verwenden, da Hollywood nur dann korrekt auf Systemen läuft, die mit westeuropäischen Sprachen kompatibel sind. Sie sollten immer `utf8` verwenden, denn das wird Hollywood in den Unicode-Modus versetzen und dafür sorgen, dass Hollywood korrekt auf allen Systemen läuft.

Die hier angegebene Codierung wird automatisch als Standardcodierung für die Text- und Zeichenketten-Bibliothek mit dem Befehl `SetDefaultEncoding()` gesetzt. Dies bedeutet, dass alle Befehle der Zeichenketten- und Textbibliotheken auf diese Codierung zurückgreifen.

**-endcolor color:**

Nur erforderlich wenn beim Argument `-backfill` der Type auf `gradient` gesetzt wurde. `color` ist eine im **RGB-Format** angegebene Farbe (z.B. `$FF0000` für Rot). Kann auch mit `-backfill` Type `picture` verwendet werden, was dazu führt, dass hinter dem Bild ein Farbverlauf zwischen `-startcolor` und `-endcolor` erstellt wird.

**-exetype type:**

Dieses Argument muss nur in Verbindung mit `-compile` angegeben werden. Es legt das Ausgabeformat für das vom Compiler zu erzeugende Programm fest. Für `Type` können sie folgendes angeben:

`amigaos4` AmigaOS 4 (PowerPC)

<b>android</b>	Hollywood-Applet hat die Plattform-spezifischen Konstanten für Android gesetzt (siehe unten).
<b>aros</b>	AROS (x86)
<b>classic</b>	AmigaOS 3.x (68020+)
<b>classic881</b>	AmigaOS 3.x (68020+) mit Coprozessor (68881/2 oder 68040/68060)
<b>ios</b>	Hollywood-Applet hat die Plattform-spezifischen Konstanten für iOS gesetzt (siehe unten).
<b>linux</b>	Linux (x86)
<b>linuxarm</b>	Linux (arm)
<b>linuxppc</b>	Linux (PowerPC)
<b>macos</b>	macOS Programmpaket (PowerPC)
<b>macosarm64</b>	macOS Programmpaket (arm64)
<b>macos86</b>	macOS Programmpaket (x86)
<b>macos64</b>	macOS Programmpaket (x64)
<b>morphos</b>	MorphOS PowerPC)
<b>warpos</b>	WarpOS Mixed-Binary (68040/PowerPC)
<b>win32</b>	Windows (x86)
<b>win64</b>	Windows (x64)
<b>applet</b>	Universal Hollywood Applet, welches auf jedem System mit einem Hollywood Player läuft

Dieses Argument ist in der 68k-Version auf **classic** voreingestellt, in der WarpOS-Version auf **warpos** und so weiter.

Beachten Sie, dass die Ausgabeformate **applet**, **android** und **ios** alle plattformunabhängige Applets kompilieren, die mit dem Hollywood-Player auf jeder Plattform ausgeführt werden können. Der Unterschied zwischen **applet**, **android** und **ios** ist, dass Hollywood bei der Kompilierung von **android** oder **ios** die jeweiligen plattformspezifischen Konstanten setzen wird (d.h. **#HW\_ANDROID** für Android und **#HW\_IOS** für iOS). Wenn Sie das **applet** als Ausgabeformat angeben, wird jedoch keine der plattformspezifischen Konstanten gesetzt. Siehe [Abschnitt 50.17 \[IF\]](#), [Seite 1099](#), für Details. Natürlich werden die mit **applet** kompilierten Applets auch auf Android und iOS arbeiten. Die Skripte werden einfach nicht wissen, dass sie für Android oder iOS kompiliert wurden. Dies kann nur erkannt werden, wenn Sie ausdrücklich **ios** oder **android** als Ausgabeformat angeben. Umgekehrt werden die mit **android** oder **ios** kompilierten Applets auch auf Nicht-Android- und Nicht-iOS-Geräten ausgeführt.

Sie können auch sofort für mehrere Plattformen kompilieren. In diesem Fall müssen Sie mit einem senkrechten Balken (|) getrennte Plattformnamen schreiben. Wenn Sie z.B. `test.hws` für AmigaOS 3 und MorphOS kompilieren wollen, benutzen Sie folgende Codezeile:

```
Hollywood test.hws -compile test -exetype classic|morphos
```

Wenn Sie mehrfach Zielplattformen in `-compile` angeben, wird der Ausgabe-datei im Namen die Plattform angehängt. Das obige Beispiel wird die beiden Dateien `test_OS3` und `test_MOS` erzeugen.

**-exportcommands file:**

Dieses Argument kann benutzt werden, um eine Liste verfügbarer Befehle in die angegebene Datei zu exportieren. Die Liste wird nach Bibliotheken und den dazugehörigen Befehlen ausgegeben. Weder die Befehle noch die Bibliotheken sind alphabetisch sortiert. Es werden nur interne Hollywood-Befehle berücksichtigt. Externe von Plugins bekommen Sie mit dem Argument `-exportplugins`. Diese Möglichkeit wird bei den meisten Benutzern keine Verwendung finden, aber für Autoren von Hollywood-IDEs könnte es hilfreich sein.

**-exportconstants file:**

Mit diesem Argument können Sie eine Liste von Hollywood-Konstanten unsortiert in die angegebene Datei ausgeben. Konstanten von Plugins erhalten Sie mit dem Argument `-exportplugins`. Für Autoren von Hollywood-IDEs kann diese Möglichkeit hilfreich sein, für normale Benutzer eher weniger.

**-exporthelpstrings:**

Wenn dieses Argument zusammen mit dem Argument `-exportplugins` verwendet wird, schreibt Hollywood zwei anstelle einer Zeile für jeden Plugin-Befehl in die Datei, die im Argument `-exportplugins` angegeben ist. Die erste Zeile ist der Name des Befehls, die zweite Zeile enthält die Befehlsschablone sowie ihr Hilfstext des Befehls aus der Dokumentation für das Plugin. Diese Informationen sind nützlich für IDEs, die Hilfe für Plugin-Befehle zur Verfügung stellen möchten. Beachten Sie, dass die zweite Zeile gekürzt oder gar leer sein kann, wenn das Plugin keine Befehlsschablone und/oder Hilfstext für den Befehl exportiert.

**-exportplugins file:**

Hiermit wird eine Liste von verfügbaren Plugins und deren Befehle und/oder Konstanten in die angegebene Datei exportiert. Normale Benutzer werden diese Möglichkeit kaum gebrauchen, aber für Autoren von Hollywood-IDEs könnte dies nützlich sein. Wenn Sie auch das Argument `-exporthelpstrings` angeben (siehe oben), wird Hollywood auch die Befehlsschablonen und Hilfstexte für alle Plugin-Befehle exportieren. Siehe oben für Details.

**-exportpreprocs file:**

Damit werden alle von Hollywood unterstützten Präprozessor-Anweisungen als Liste in die angegebene Datei exportiert. Die Liste wird unsortiert und ohne das Präfix "@" ausgegeben. Eher für Programmierer von Hollywood-IDEs als für normale Benutzer gedacht.

**-fakefullscreen:**

Dieses Argument erlaubt es Ihnen, Hollywood in einem unechten vollen Bildschirmmodus auszuführen. Das bedeutet, Hollywood öffnet ein Hintergrundfenster, das den ganzen Desktop abdecken wird. So bekommt der Benutzer den Eindruck, das Programm läuft auf einem separaten Screen/Bildschirm, obwohl es auf dem Desktop läuft.

**-fitscale:**

Dieses Argument kommt nur zum Zuge, wenn **-layerscale** oder **-autoscale** verwendet werden. Somit wird **-fitscale** die Auflösung des Skripts auf die Auflösung des aktuellen Bildschirms setzen, damit immer der ganze Bildschirm ausgefüllt ist. Im Grunde ist es dasselbe, als würden Sie mit den Argumenten **-scalewidth/-scaleheight** die aktuellen Bildschirmmaße übergeben. Da Sie aber die Bildschirmauflösungen der Benutzer nicht kennen können, übernimmt Ihnen **-fitscale** diese Arbeit. Beachten Sie, dass **-fitscale** Ihr Skript verzerrt darstellen könnte, da diverse Bildschirmauflösungen verschiedene Seitenverhältnisse haben. Um diese Verzerrungen zu vermeiden, benutzen Sie **-keepproportions** (siehe weiter unten).

**-fixed:** Wenn Sie dieses Argument angeben, wird das Hollywood-Fenster auf dem Bildschirm fixiert, d.h. Sie können es nicht mehr bewegen. Dies ist nützlich, wenn Hollywood im Vollbildmodus öffnet.

**-forceflush:**

Mit diesem Argument wird der Puffer geleert, nachdem Hollywood in ein Debug-Gerät eine einzelne Zeile geschrieben hat. Das ist nur nützlich, wenn das Gerät (Device) eine Datei oder Leitung ist. Bei einer Konsole wird nach jeder Zeile immer der Puffer geleert.

**-forcesound:**

Normalerweise, wenn ein Skript versucht eine Musik abzuspielen und die Audio-Hardware nicht zugeordnet werden kann, läuft Hollywood normal ohne Musik weiter. Wenn Sie das nicht wollen, übergeben Sie dieses Argument. In diesem Fall wird Hollywood einen Fehler auslösen, falls die Audio-Hardware nicht zugeordnet werden kann.

**-formaterror:**

Dieses Argument weist Hollywood an, seine Fehlermeldungen auf eine bestimmte Weise zu formatieren, damit sie leicht von anderen Konsolenausgaben unterschieden werden können. Beachten Sie, dass dieses Argument nur berücksichtigt wird, wenn die Option **-printerror** ebenfalls aktiv ist. In diesem Fall werden Fehler wie folgt in der Konsole protokolliert:

```
@_hwerror<line>:<file>*<message>
```

Beachten Sie, dass **<file>** möglicherweise in doppelte Anführungszeichen eingeschlossen ist und leer sein kann, wenn der Fehler nicht mit einer Skriptdatei zusammenhängt. In diesem Fall ist **<line>** 0.

**-fullscreen:**

Dieses Argument wird Hollywood im Vollbildmodus betreiben. Es wird die Anzeigemodi Ihres Monitors scannen, um die beste Auflösung für Ihr Skript zu



bestimmen und schaltet dann den Monitor in diese Auflösung und führt das Skript aus. Wenn Sie Ihr Skript im Vollbildmodus ohne Umschalten der Monitoraufösung ausführen möchten, verwenden Sie stattdessen `-fullscreenscale` (siehe unten).

**-fullscreenscale:**

Dies ist ein spezieller Vollbildmodus, der die Auflösung Ihres Monitors nicht verändert. Stattdessen wird die Größe des Hollywood-Displays an die Abmessungen Ihres Monitors angepasst. Darüber hinaus aktiviert dieser Vollbildmodus das Autoskalierungs-System, so dass Ihr Display automatisch an die Abmessungen Ihres Monitors angepasst wird. Standardmäßig verwendet `-fullscreenscale` das Autoskalierungs-System. Wenn Sie hingegen die Ebenenskalierung verwenden möchten, müssen Sie auch die Option `-layerscale` übergeben. `-fullscreenscale` ist besonders auf Mobilgeräten nützlich, deren Display-Hardware eine fest codierte Auflösung hat und keine Auflösungsänderungen unterstützt, wie dies bei einem externen Monitor der Fall ist, der an einen Desktop-Computer angeschlossen ist. Der Nachteil von `-fullscreenscale` ist, dass er langsamer ist, da Hollywood alle Wiedergabe-Operationen auf die Abmessungen des Monitors skalieren muss.

**-globalplugins:**

Unter AmigaOS und kompatiblen Geräten können Plugins auch global in `LIBS:Hollywood` installiert werden. Von Hollywood kompilierte Programme laden jedoch nur die Plugins, die neben dem Programm in ihrem Verzeichnis gespeichert sind. Wenn Ihre Programme auch alle Plugins in `LIBS:Hollywood` laden soll, können Sie dieses Argument angeben. Alternativ können Sie auch `GlobalPlugins` in der Präprozessor-Anweisung `@OPTIONS` auf `True` setzen. [Nur AmigaOS]

**-help:** Gibt eine Liste der unterstützen Konsolenargumente aus.

**-hideoptionsmenu:**

Wenn der Benutzer das Optionen-Menü auf Android-Geräten öffnet, wird Hollywood dem Benutzer erlauben, mehrere optische Displayparameter zu konfigurieren. So ist es möglich, Auto- oder Ebenenskalierung ein- bzw. auszuschalten. Soll dies nicht möglich sein, benutzen Sie das Argument `-hideoptionsmenu`. [Nur Android]

**-hidepointer:**

Bei diesem Argument wird der Mauszeiger automatisch versteckt, sobald Hollywood den vollen (full screen) oder unechten vollen Bildschirmmodus (fake full screen) öffnet. Gegenüber dem Befehl `HidePointer()` hat dieses Argument den Vorteil, dass nur bei vollen Bildschirmmodi den Mauszeiger verschwinden lässt. Im Fenstermodus wird der Mauszeiger sichtbar bleiben, weil ein versteckter Mauszeiger im Fenstermodus den Benutzer verwirren kann.

**-hidetitlebar:**

Dieses Argument verbirgt die Titelzeile des benutzten Screen. Es hat nur Auswirkungen, wenn Hollywood auf einem eigenen Bildschirm (Screen) geöffnet wird oder wenn Sie die Option `-backfill` auf dem Workbench Screen benut-

zen. Bei mobilen Geräten versteckt diese Option die Statusleiste (iOS) oder Aktionsleiste (Android). [Nur Amiga OS, macOS und Android]

**-icon16x16 file:**

Dieses und alle andere **-iconXXX** Argumente können benutzt werden, um die Piktogramme für Ihre Programme anzugeben. Auf Windows, macOS und Linux werden diese Piktogramme im Fensterrahmen erscheinen, aber auch z.B. in der Taskbar von Windows. Diese Piktogramme werden in dem Hollywood Applet oder kompilierten Programm eingebunden. Voreingestellt ist das Hollywood-Piktogramm (Filmklappe). Sie können Ihr eigenes Piktogramm verwenden, indem Sie eines, mehrere oder alle dieser **-iconXXX** Argumente angeben. Für das beste Ergebnis sollten Sie für alle Größen ein von Hand bearbeitetes Piktogramm erstellen. Hollywood unterstützt gegenwärtig die Piktogrammgrößen 16x16, 24x24, 32x32, 48x48, 64x64, 96x96, 128x128, 256x256, 512x512 und 1024x1024. Nicht jede Piktogrammgröße wird auf allen Plattformen berücksichtigt, aber Sie sollten sicherstellen, dass alle Größen von Ihnen ein Piktogramm erhalten. Lassen Sie eine Größe aus, könnte Hollywood für diese oder gar für alle Größen auf das Standard-Piktogramm (Filmklappe) zurückgreifen. Die Bilddatei für das Piktogramm muss ein PNG-Bild mit Alphakanal sein. Bilder ohne Alphakanal werden auch unterstützt, aber wird nicht empfohlen. Mit Alphakanal sieht das Piktogramm optisch besser aus. Als Alternative können Sie auch die Präprozessor-Anweisung **@APPICON** benutzen.

**-icon24x24 file:**

Gleich wie **-icon16x16**, aber bettet ein Piktogramm der Größe 24x24 ein.

**-icon32x32 file:**

Gleich wie **-icon16x16**, aber bettet ein Piktogramm der Größe 32x32 ein.

**-icon48x48 file:**

Gleich wie **-icon16x16**, aber bettet ein Piktogramm der Größe 48x48 ein.

**-icon64x64 file:**

Gleich wie **-icon16x16**, aber bettet ein Piktogramm der Größe 64x64 ein.

**-icon96x96 file:**

Gleich wie **-icon16x16**, aber bettet ein Piktogramm der Größe 96x96 ein.

**-icon128x128 file:**

Gleich wie **-icon16x16**, aber bettet ein Piktogramm der Größe 128x128 ein.

**-icon256x256 file:**

Gleich wie **-icon16x16**, aber bettet ein Piktogramm der Größe 256x256 ein.

**-icon512x512 file:**

Gleich wie **-icon16x16**, aber bettet ein Piktogramm der Größe 512x512 ein.

**-icon1024x1024 file:**

Gleich wie **-icon16x16**, aber bettet ein Piktogramm der Größe 1024x1024 ein.

**-keepproportions:**

Dieses Argument kommt nur zum Zuge, wenn **-layerscale** oder **-autoscale** verwendet werden. Das Argument **-keepproportions** wird die Auflösung vom

Skript nicht verzerren, wenn der Benutzer die Fenstergröße ändert. Stattdessen werden schwarze Rahmen benutzt, um die nicht-proportionalen Fenstergebiete zu füllen. Die optische Anzeige selbst wird immer im gleichen Seitenverhältnis dargestellt. Dies ist sehr nützlich für Skripte, die nicht verzerrt werden sollen.

**-keepscreenon:**

Wenn Sie dieses Argument verwenden, wird der Batterie-Einsparungsmodus auf mobilen Geräten ausgeschaltet. Somit wird der Bildschirm des Gerätes nie verdunkelt. Das ist nützlich für Skripts, die keine Benutzereingaben verlangen. [Nur Android und iOS]

**-layerfullscreen:**

Dadurch wird das Display mithilfe der Ebenenskalierung in den Vollbildmodus versetzt, anstatt die Auflösung des Monitors zu ändern, aber nur, wenn Hollywood auf Systemen ausgeführt wird, die GPU-beschleunigte Skalierung unterstützen. Auf allen anderen Plattformen wird ein normaler Vollbildmodus verwendet, d.h. Hollywood passt die Auflösung des Monitors an die aktuelle Bildschirmgröße an. Derzeit wird die GPU-beschleunigte Skalierung unter Windows, macOS, Android und iOS unterstützt, was bedeutet, dass auf diesen Plattformen keine Änderung der Bildschirmauflösung stattfindet, da Hollywood die Grafiken einfach skalieren kann, um sie an die aktuellen Bildschirmabmessungen anzupassen. Auf AmigaOS-kompatiblen Systemen und Linux wird es bei diesem Modus jedoch immer noch eine Änderung der Bildschirmauflösung geben, da Hollywood auf diesen Plattformen keine GPU-beschleunigte Skalierung unterstützt.

**-layerscale:**

Wenn Sie dieses Argument angeben, wird die Ebenenskalierung eingeschaltet. Das bedeutet, dass Ihr Skript in jeder Auflösung gezeigt werden kann, die Sie definieren. Es funktioniert vollkommen automatisch und Sie müssen keine Änderungen an Ihrem Code vornehmen. Aber wie der Name bereits andeutet, wird die Ebenenskalierung nur funktionieren, wenn die **Ebenentechnik eingeschaltet** wurde. Im Modus Ebenenskalierung werden alle Ebenen automatisch an die neue Auflösung angepasst und Hollywood gaukelt Ihrem Skript vor, es wird immer noch in der originalen Auflösung ausgeführt. So wird sicher gestellt, dass Ihr Skript auf die genau gleiche Weise wie ohne Ebenenskalierung ausgeführt wird. Der Vorteil der Ebenenskalierung ist, dass Vektorebenen (z.B. Grafikgrundelemente, TrueType-Text, Vektorpinsel, Vektoranims) ohne Qualitätsverluste der Größe angepasst werden, auch wenn Ihr Skript anstelle von 320x240 mit 1280x1024 ausgeführt wird. Die Startauflösung können Sie mit den Argumenten **-scalewidth** und **-scaleheight** oder **-scalefactor** oder **-systemscale** festlegen (siehe unten). Die Auflösung des Fensters kann jederzeit vom Benutzer durch ändern der Fenstergröße gewechselt werden. Vergessen Sie nicht, Ihrem Hollywoodfenster die Größenänderung mit der Präprozessor-Anweisung **@DISPLAY** oder dem Argument **-sizeable** zu erlauben. Wenn Sie die Argumente **-scalewidth** und **-scaleheight** oder eines der beiden Argumente **-scalefactor** oder **-systemscale** beim Starten nicht angeben, läuft das Skript am Anfang ohne Ebenenskalierung. Sobald Sie aber die Fenstergröße ändern, wird die Ebenenskalierung eingeschaltet. Wollen Sie Kantenglättung

(Antialias) verwenden, benutzen Sie das Argument `-smoothscale` (ist langsamer). Hollywood unterstützt noch eine weitere Skaliermethode, welche mit dem Argument `-autoscale` aktiviert wird. Siehe [Abschnitt 24.24 \[Informationen über das Skalierungssystem\]](#), [Seite 402](#), für Details.

**-legacyaudio:**

Mit Hollywood 6.0 wurde bei den AmigaOS-Versionen ein neuer Audiotreiber mitgeliefert. Der alte Audiotreiber wird immer noch unterstützt und kann mit diesem Argument aktiviert werden. Beachten Sie, dass bei Amiga OS 3.x aus Leistungsgründen der alte Treiber voreingestellt ist. Möchten Sie hingegen den neuen Audiotreiber mit AmigaOS 3.x verwenden, dann benutzen Sie das Argument `-nolegacyaudio` [Nur AmigaOS]

**-linkfiles file:**

Dieses Argument wird nur berücksichtigt, wenn `-compile` auch angegeben wurde. Damit können Sie Dateien in Ihr Applet oder Programm einbinden. Ihr Skript wird dann diese Dateien automatisch laden. Möchten Sie in Ihrem Skript keine Präprozessor-Anweisungen für das Einbinden von Dateien benutzen, verwenden Sie als alternative das Argument `-linkfiles`. Als andere Möglichkeit können Sie auch die Präprozessor-Anweisung `@LINKER` verwenden, um eine Liste von Dateien anzugeben, die in Ihr Applet oder Ihrem Programm eingebunden werden sollen. Siehe [Abschnitt 4.3 \[Einbinden von Datendateien\]](#), [Seite 58](#), für Details.

**-linkfonts file:**

Dieses Argument wird nur berücksichtigt, wenn `-compile` auch angegeben wurde. Damit können Sie Zeichensätze (Fonts) in Ihr Applet/Programm einbinden. Ihr Skript wird diese Zeichensätze automatisch aus Ihrem Applet/Programm laden, sobald Sie diese mit `SetFont()` aufrufen. Somit ist `-linkfonts` eine Alternative zur Präprozessor-Anweisung `@Font()`. Normalerweise ist das Benutzen von `@Font()` viel einfacher. Darum sollten Sie `-linkfonts` nur aus guten Gründen verwenden. Als weitere Möglichkeit können Sie auch die Präprozessor-Anweisung `@LINKER` verwenden, um eine Liste von Dateien anzugeben, die in Ihr Applet oder Ihrem Programm eingebunden werden sollen. Siehe [Abschnitt 4.4 \[Einbinden von Schriftarten\]](#), [Seite 60](#), für Details.

**-linkplugins list:**

Dieses Argument wird nur berücksichtigt, wenn Sie `-compile` auch angegeben haben. Sie können dieses Argument benutzen, um Plugins in Ihr Programm einzubinden. Beim Starten Ihres Programms werden dann diese Plugins automatisch geladen und Sie haben alles in einer einzelnen Datei. Es ist nicht mehr notwendig, die Plugins für Ihr Programm im selben Verzeichnis zu installieren. Wichtig: Plugins können nur in Programme, aber nicht in Applets eingebunden werden, weil Applets Plattformunabhängig sind, Programme hingegen nicht. Sie müssen mit diesem Argument eine Liste von Plugins übergeben. Wenn Sie mehr als ein Plugin einbinden wollen, dann trennen Sie jedes einzelne Plugin mit einem senkrechten Strich (`|`). Wenn Sie also z.B. "plugin1.hwp" und

"plugin2.hwp" in Ihr Skript einbinden wollen, müssten Sie "plugin1|plugin2" schreiben.

**Wichtig:** Lesen Sie vorher die Lizenz jedes Plugin, das Sie in Ihre Programmen einbinden möchten. Wenn Sie z.B. ein Plugin unter LGPL-Lizenz in Ihr Programm einbinden, dann wird Ihr vollständiges Projekt auch automatisch LGPL und Sie müssen alle Quellen und Dateien zur Verfügung stellen. Darum studieren Sie die Pluginlizenzen, bevor Sie sie ins Programm einbinden. Beachten Sie außerdem, dass Sie vor der Verwendung der Option `-linkplugins` zuerst die Infrastruktur für den Plugin-Linker einrichten müssen. Siehe [Abschnitt 4.5 \[Einbinden von Plugins\]](#), [Seite 62](#), für Details.

**-locksettings**

Dieses Argument wird nur berücksichtigt, wenn `-compile` auch angegeben wurde. Sie können `-locksettings` benutzen, um die optischen Anzeigeparameter Ihres Skriptes unveränderlich festzulegen. Wenn Sie normalerweise ein Applet oder Programm mit Hollywood binden, kann der Benutzer das Aussehen nach seinen Wünschen verändern. Möchten Sie das unterbinden, verwenden Sie das Argument `-locksettings` und Hollywood berücksichtigt immer nur die Parameter, welche im Skript mit der Präprozessor-Anweisung `@DISPLAY` festgelegt wurden. Wenn Sie zum Beispiel `Mode=FullScreen` mit `@DISPLAY` definiert haben und mit `-locksettings` kompilieren, kann der Benutzer das Programm nicht im Fenstermodus öffnen. Überlegen Sie zweimal, bevor Sie `-locksettings` benutzen, da es die Flexibilität Ihres Programms einschränkt.

**-mastervolume vol:**

Dieses Argument erlaubt Ihnen, die Master-Lautstärke festzulegen, die Hollywood benutzen soll. Benutzen Sie es nur, wenn sie Verzerrungen bemerken, während Hollywood Sounds abspielt. Normalerweise sollten Sie hier keinen Wert setzen müssen. Der Wertebereich reicht von 0 bis 64. [Nur AmigaOS]

**-maximized:**

Öffnen Sie das Display im maximierten Modus. Dieses Display muss größenveränderlich sein, damit dieser Parameter wirksam wird. [Nur Windows]

**-moderequester:**

Wenn Sie dieses Argument zusammen mit `-fullscreen` benutzen, wird Hollywood ein Fenster öffnen, wo der Benutzer eine Auflösung für den eigenen Bildschirm wählen kann.

**-monitor num:**

Dieses Argument erlaubt es Ihnen, den Monitor zu festzulegen, den Ihr Skript benutzen soll. Monitore werden von 1 an gezählt. Monitor 1 ist auch der primäre Monitor.

**-nativeunits:**

Wenn dieses Konsolenargument angegeben wird, verwendet Hollywood den nativen Koordinatenraum des Hostsystems und deren Einheiten anstelle von Pixeln. Dies hat derzeit nur einen Einfluss auf macOS und iOS, da beide Betriebssysteme benutzerdefinierte Einheiten anstelle von Pixeln verwenden, wenn sie auf einem Retina-Gerät laufen. Standardmäßig wird Hollywood die Verwendung von Pixeln auf Retina Macs und iOS-Geräten für plattformübergreifende

Kompatibilitätsgründe erzwingen. Aber Sie können diese Einstellung mit diesem Konsolenargument ändern.

**-nobackfill:**

Voreingestellt ist, dass Hollywood für den Vollbildmodus immer einen Hintergrund für Ihr Display anzeigt. Wenn Sie das nicht wollen, benutzen Sie das Argument `-nobackfill`. Hollywood wird dann einen Vollbildschirm öffnen, aber den Bereich um das Display nicht abdecken. Dies ist auf Amiga-Systemen nützlich, wenn Sie einen neuen Bildschirm öffnen wollen, aber nicht die sichtbaren Objekte wie Titelleiste und Hintergrunddekoration des Benutzers verstecken möchten. Falls Sie `-nobackfill` benutzen, dann lesen Sie auch noch die Erklärungen zum `-nostyleoverride` Argument. [Nur AmigaOS]

**-nochdir:**

Voreingestellt ist, dass Hollywood immer in das aktuelle Verzeichnis vom laufenden Skript wechselt. Übergeben Sie dieses Argument, wenn Sie dieses Verhalten unterbinden möchten. In diesem Fall wird Hollywood das aktuelle Verzeichnis nicht wechseln, wenn Sie das Skript ausführen.

**-nocommodity:**

Auf AmigaOS-Systemen wird Hollywood, wenn dieses Argument angegeben wird, nicht in der Liste der Commodities des Systems eingetragen. [Nur AmigaOS]

**-nodebug:**

Wenn Sie dieses Argument angeben, werden die Befehle `DebugPrint()`, `DebugPrintNR()`, `Assert()`, `DebugOutput()` und `@WARNING` übersprungen, wenn das Skript oder Applet abläuft. Das erlaubt Ihnen, die Debugbefehle mit einem einzelnen Aufruf völlig auszuschalten.

**-nodocky:**

Auf AmigaOS 4-Systemen wird Hollywood, wenn dieses Argument angegeben wird, die Anwendung im AmiDock nicht anzeigen. Dieser Tag ist nützlich, wenn Sie eine unsichtbare Anwendung haben möchten, die alle Anwendungsfunktionen wie den Nachrichtenmechanismus und Ringhio nutzen kann, aber nicht im AmiDock erscheint. Dieser Tag wird nur berücksichtigt, wenn `RegisterApplication` in `@OPTIONS` auf `True` gesetzt wurde. [Nur AmigaOS 4]

**-nohardwarescale: VERALTET**

Um die Leistung/Geschwindigkeit bei Android zu steigern, wird Hollywood bei eingeschalteter Autoskalierung auf die Grafikhardware zugreifen. Einige Geräte haben keine Grafikhardware und es kann zu seltsamen Ergebnissen führen, wenn Sie Autoskalierung benutzen. Ist dies der Fall, dann verwenden Sie das Argument `-nohardwarescale` und schauen, ob das hilft. Dieses Argument ist seit Hollywood 8.0 veraltet. Hollywood wird jetzt immer die hardwarebeschleunigte Skalierung verwenden. [Nur Android]

**-nohide:**

Wenn Sie dieses Argument angeben, wird es dem Benutzer nicht mehr möglich sein, das Display mittels Tastenkombination zu minimieren. Dieses Argument hat keinen Einfluss auf die beiden Befehle `ShowDisplay()` sowie `HideDisplay()`.

Ihr Skript kann also immer noch das Display minimieren (Piktogramm auf der Workbench) oder wieder hervorholen, nur dem Benutzer bleibt dies verwahrt.

**-nokeepproportions:**

Wenn der Benutzer mit dem Tastaturkürzel **CMD+RETURN** (unter Windows ist es **ALT+RETURN**) zwischen Fenster- und Vollbildmodus wechselt und Hollywood das Display auf die Auflösung des aktuellen Monitors skaliert, werden die Ränder falls erforderlich aufgefüllt, um die Proportionen des Displays beizubehalten. Wenn Sie das nicht wollen, übergeben Sie dieses Argument.

**-nolegacyaudio:**

Mit Hollywood 6.0 wurde für die Amigaversionen von Hollywood ein neues Audiotreiber eingeführt. Aber dieser Treiber ist bei AmigaOS 3.x aus Leistungsgründen nicht aktiviert. Wenn Sie trotzdem diesen Treiber verwenden wollen, benutzen Sie dieses **-nolegacyaudio** Argument. [Nur AmigaOS]

**-noliveresize:**

Auf vielen Plattformen wird Hollywood Live-Größenänderung verwenden, wenn der Benutzer das Displays verändert. Dies bedeutet, dass der Inhalt des Displays skaliert wird, während es der Benutzer verkleinert. Wenn Sie das nicht wollen, können Sie dieses Konsolenargument setzen.

**-nomodeswitch:**

Wenn Sie **-nomodeswitch** angeben, wird ein Wechseln mit der Tastaturkombination **CMD+RETURN** (macOS und Amiga) oder **ALT+RETURN** (Windows) zwischen Fenster- und Vollbildmodus nicht möglich sein. Das bedeutet, Hollywood wird immer im anfänglichen Anzeigemodus bleiben.

**-nomousehook:**

Wenn Sie dieses Argument benutzen, wird Hollywood kein Programmteil installieren, der ständig die Mausposition abfragt. Dies ist dann auf Linux nützlich, wenn die Verbindung zum X-Server sehr langsam ist. Das Argument **-nomousehook** könnte in dem Fall einen Leistungsschub bringen. Aber der Nachteil ist, dass Sie nicht mehr über **OnMouseMove** Ereignissen benachrichtigt werden, wenn sie außerhalb der Grenzen des Fensters vorkommen. [Nur Linux]

**-noscaleengine:**

Dieses Konsolenargument wird nur behandelt, wenn Sie auch das Argument **-fullscreenscale** übergeben. In diesem Fall wird Hollywood kein Skalierungssystem verwenden, sondern wird einfach Ihr Display in den gleichen Dimensionen wie die Auflösung des Monitors öffnen. Ihr Skript muss sich dann manuell an die Auflösung des Monitors anpassen. Dies ermöglicht es Ihnen Skripte zu schreiben, die sich dynamisch an verschiedene Auflösungen anpassen können, ohne einfach ihre Grafiken zu skalieren.

**-noscaleswitch:**

Wenn der Benutzer mit dem Tastaturkürzel **CMD+RETURN** (unter Windows ist es **ALT+RETURN**) zwischen Fenster- und Vollbildmodus wechselt, kann Hollywood das Display auf die aktuelle Auflösung des Monitors skalieren, anstatt die physikalische Auflösung des Monitors umzuschalten. Wenn Sie nicht möchten, dass Hollywood den Vollbildmodus simuliert, indem Sie einfach das Display

auf die aktuelle Auflösung des Monitors skaliert, übergeben Sie dieses Konsoleargument. In diesem Fall wird durch Drücken des Tastaturkürzels für den Moduswechsel immer die physikalische Auflösung des Monitors geändert.

**-nosmoothscale:**

Wenn der Benutzer mit dem Tastaturkürzel **CMD+RETURN** (unter Windows ist es **ALT+RETURN**) zwischen Fenster- und Vollbildmodus wechselt und Hollywood das Display auf die Auflösung des aktuellen Monitors skaliert, wird standardmäßig Antialiasing-Interpolation verwendet, um die Skalierung glatter zu gestalten. Wenn Sie das nicht wollen, übergeben Sie dieses Argument.

**-nosound:**

Dieses Argument schaltet alle Soundbefehle von Hollywood ab. Wenn Sie es angeben, wird Hollywood komplett stumm laufen.

**-nostyleoverride:**

Wenn Hollywood Ihr Skript im Vollbildmodus ausführt, wird das Fenster rahmenlos, fixiert und die Fensterdekorationen werden modifiziert. Wenn Sie das nicht möchten, benutzen Sie dieses Argument, um Hollywood zu zwingen, das Fenster unverändert zu lassen. Dieses Argument wird meistens zusammen mit **-nobackfill** verwendet. [Nur AmigaOS]

**-numchannels chans:**

Hollywood hat 8 Audiokanäle voreingestellt. Diese gehen allerdings aus, wenn Ihr Skript mehr als 8 verschiedene Samples, Musik oder Videos zur gleichen Zeit abspielen muss. Wenn Sie aus irgendeinem Grund mehr als 8 Kanäle brauchen, können Sie mit diesem Argument Hollywood die Anzahl mitzuteilen.

**-overrideplacement:**

Wenn Sie dieses Argument angeben, wird Hollywood die gespeicherten Informationen über Position und Größe des Displays, welche mit dem Tag **RememberPosition True** festgelegt wurden, ignorieren. Stattdessen werden die voreingestellte Größe und Position verwendet.

**-overwrite:**

Wenn Sie dieses Argument angeben, wird Hollywood automatisch bestehende Dateien beim Kompilieren mit dem Argument **-compile** überschreiben. Normalerweise wird Sie Hollywood fragen, ob eine bestehende Datei überschrieben werden darf.

**-pictrans transparency:**

Nur möglich, falls **-backfill** auf **picture** gesetzt wurde. Dieses Argument erlaubt es Ihnen, dem Bild eine transparente Farbe zuzuweisen. Standardwert ist hier **#NOTRANSARENCY**.

**-picxpos x:**

Nur möglich, falls **-backfill** auf **picture** gesetzt wurde. Dieses Argument legt die x-Position für das Bild fest. Der Standardwert ist **#CENTER**.

**-picypos y:**

Nur möglich, falls **-backfill** auf **picture** gesetzt wurde. Dieses Argument legt die y-Position für das Bild fest. Der Standardwert ist **#CENTER**.



**-printerror:**

Wenn dieses Argument angegeben ist, zeigt Hollywood keine Skriptfehler in einem Dialogfeld an, sondern gibt sie einfach an die Konsole aus. Dies kann bei der Integration von Hollywood in IDEs nützlich sein. Wenn Sie die Fehlermeldungen von Hollywood parsen müssen, sollten Sie auch das Argument **-formaterror** angeben, um Hollywood zu zwingen, Fehler in einem Format auszugeben, das geparkt werden kann, d.h. in seine einzelnen Bestandteile wie Skriptdatei, Zeilennummer, Fehlermeldung aufgeteilt.

**-pubscreen name:**

Veranlasst Hollywood, auf dem öffentlichen Bildschirm mit dem angegebenen Namen zu erscheinen. [Nur AmigaOS]

**-quiet:** Falls angegeben, wird Hollywood "still" starten und keine Informationen ausgeben, während es das Skript lädt.

**-requireplugins list:**

Dieses Argument erlaubt es Ihnen, eine Liste von Plugins zu erstellen, die Ihr Skript ausdrücklich verlangt. Wenn Sie mehr als ein Plugin angeben müssen, werden diese mit dem senkrechten Strich (|) getrennt. Wenn zum Beispiel Ihr Skript "plugin1.hwp" und "plugin2.hwp" braucht, schreiben Sie "plugin1|plugin2" in die Liste. Dieses Argument hat die gleiche Wirkung, als würden Sie in Ihrem Skript die Präprozessor-Anweisung **@REQUIRE** verwenden. Siehe [Abschnitt 44.6 \[REQUIRE\]](#), [Seite 990](#), für Details. Wenn Sie Argumente der Initialisierungsroutine des Plugins übergeben müssen, dann benutzen Sie das Argument **-requiretags**.

**-requiretags tags:**

Dieses Konsolenargument erlaubt es Ihnen, der Initialisierungsroutine von Plugins zusätzliche Argumente zu übergeben. Normalerweise werden diese dem Plugin mit der Präprozessor-Anweisung **@REQUIRE** weitergereicht. Wenn Sie aber **-requiretags** benutzen, müssen Sie nicht immer Ihr Skript abändern/speichern/ausführen, damit Sie verschiedene Argumente und/oder Werte testen können. Da Sie mehrere Plugins zusätzliche Argumente weitergeben können, sieht das Format der Zeichenkette so aus:

```
name1[tag1=value1,...,tagN=valueN]name2[...]...nameN[...]
```

Hier ist ein Beispiel:

```
-requiretags testplugin[User='admin',Pwd='secret',Len=64]
```

Mit dieser Zeile wird der Initialisierungsroutine des Plugin **testplugin.hwp** drei zusätzliche Felder übergeben: **User**, **Pwd** und **Len**. **User** wird "admin", **Pwd** wird "secret" als Zeichenkette und **Len** wird 64 als Ganzzahl zugewiesen. Bitte beachten Sie, dass Sie auch **-requireplugins** als Argument benutzen müssen, ansonsten wird der Initialisierungscode des Plugins überhaupt nicht ausgeführt werden.

**-resourcemonitor:**

Mit diesem Argument wird Hollywood beim Starten von Ihrem Skript den Ressourcenmonitor aufrufen. Der Ressourcenmonitor ist nützlich, um während dem

Ausführen von Ihrem Skript die Ressourcen im Auge zu behalten. Bitte lesen Sie die Dokumentation von `OpenResourceMonitor()` für mehr Informationen.

**-scalefactor s:**

Dieses Argument kann entweder in Verbindung mit `-autoscale` oder `-layerscale` verwendet werden, um einen globalen Skalierungsfaktor auf Ihr ganzes Skript anzuwenden. Der Skalierungsfaktor muss als Bruchzahl angegeben werden, die den gewünschten Skalierungskoeffizienten angibt. Z.B. ein Wert von 0.5 verkleinert alles auf die Hälfte, während ein Wert von 2.0 alles auf das Doppelte seiner Größe skaliert. Beachten Sie, dass die Einstellung `-scalefactor` dazu führt, dass sich das Skript geringfügig anders verhält als mit der Einstellung in `-scalewidth` und `-scaleheight`. Letzteres erzwingt eine feste Display-Größe für das Skript, die nie geändert wird, ausser der Benutzer ändert die Display-Größe manuell mit der Maus. Wenn Sie `-scalefactor` einstellen, wird der Skalierungsfaktor jedoch auf alle neuen BGPics und Display-Größen angewendet, so dass sich die Display-Größe ändern kann, wenn sich die BGPic-Größe ändert oder das Skript die Display-Größe ändert. Die Verwendung von `-scalefactor` ist daher perfekt für die Skalierung eines Skripts für eine hohe dpi-Anzeige, da es sicherstellt, dass sich das Skript genau gleich verhält, aber einfach größer (oder kleiner, wenn Sie wollen!) erscheint. Sie können das Argument `-systemscale` auch verwenden, um den Skalierungsfaktor des Hostsystems automatisch auf Ihre Displays anzuwenden (siehe unten). Bitte lesen Sie auch die Dokumentation von den Argumenten `-autoscale/-layerscale`, um mehr über die Skalierungssysteme von Hollywood zu erfahren. Siehe [Abschnitt 24.24 \[Skalierungssysteme\]](#), [Seite 402](#), für Details.

**-scalepicture:**

Nur möglich falls `-backfill` auf `picture` gesetzt wurde. Dieses Argument veranlasst Hollywood, das angegebene Bild auf die Größe des Hintergrunds zu skalieren, so dass es ihn komplett ausfüllt.

**-scaleswitch:**

Wenn der Benutzer mit dem Tastaturkürzel `CMD+RETURN` (unter Windows ist es `ALT+RETURN`) zwischen Fenster- und Vollbildmodus wechselt, ändert Hollywood möglicherweise nicht den Bildschirmmodus des Monitors, sondern simuliert nur den Vollbildmodus durch die Skalierung des Displays auf die aktuelle Auflösung des Monitors. Dies geschieht nur, wenn das System, auf dem Hollywood läuft, die hardwarebeschleunigte Skalierung unterstützt. Auf älteren Systemen oder Plattformen, die keine hardwarebeschleunigte Skalierung unterstützen, schaltet Hollywood stattdessen den Monitor auf die neue Auflösung um. Wenn Sie möchten, dass Hollywood immer den Vollbildmodus simulieren soll, indem Sie das Display einfach auf die aktuelle Auflösung des Monitors skalieren, anstatt den physikalischen Modus zu ändern, übergeben Sie dieses Argument.

**-scalewidth width:**

Dieses Argument kann zusammen mit `-autoscale` oder `-layerscale` benutzt werden, um die Anfangsgröße anzugeben. Sie können einen numerischen

Pixel- (z.B. `-scalewidth 1280 -scaleheight 1024`) oder Prozentwert (z.B. `-scalewidth 200%, -scaleheight 200%`) angeben. Für mehr Informationen lesen Sie bitte die Dokumentation von `-autoscale/-layerscale` durch. Siehe [Abschnitt 24.24 \[Skalierungssysteme\]](#), Seite 402, für Details.

**`-scaleheight height:`**

Gleich wie `-scalewidth`, gibt aber die Höhe an, auf die das Display skaliert werden soll.

**`-scrdepth d:`**

Mit diesem Argument kann die Farbtiefe gesetzt werden, wenn Hollywood einen Vollbildschirm öffnet. Gültige Tiefen sind 15, 16, 24 und 32. Wenn Sie dieses Argument nicht gebrauchen, wird Hollywood dieselbe Farbtiefe wie der Desktopbildschirm benutzen.

**`-scrwidth width:`**

Dieses Argument kann im Zusammenhang mit `-fullscreen` benutzt werden. Es gibt die Breite des Schirms im Vollbildmodus an. Wenn Sie dieses Argument nicht setzen, wird Hollywood den Bildschirm öffnen, der am besten zu Ihrem Programm passt. Wenn Sie `-scrwidth` und `-scrheight` mit 0 angeben, wird Hollywood dieselben Maße für den Vollbildschirm nutzen wie der Desktop.

**`-scrheight height:`**

Gleich wie `-scrwidth`, gibt aber die Schirmhöhe an.

**`-setconstants list:`**

Sie können dieses Argument verwenden, um eine oder mehrere Konstanten zu deklarieren. Normalerweise werden Konstanten mit der [Const-Anweisung](#) deklariert. Manchmal kann es aber auch möglich sein, Konstanten auch von der Kommandozeile zu deklarieren. Dies ist besonders nützlich bei der Verwendung der Präprozessor-Anweisung [@IF](#). Sie müssen diesem Argument eine Zeichenkette übergeben, die eine oder mehrere Konstanten enthält. Wenn Sie mehrere Konstanten deklarieren möchten, müssen Sie die einzelnen Konstanten mit dem vertikalen Balkenzeichen (|) trennen. Sie können das Gleichheitszeichen (=) verwenden, um einer Konstanten einen Wert zuzuweisen. Wenn Sie das Gleichheitszeichen auslassen, wird der Konstanten automatisch ein Wert von 1 gegeben. Beachten Sie, dass der Name der Konstante das Hash-Tag-Präfix nicht enthalten darf, sondern nur den Namen der Konstante. Hier ist ein Beispiel für eine Zeichenkette: "MYCONSTANT|MYCONSTANT2=1000". Wenn Sie diese Zeichenfolge auf `-setconstants` übergeben, wird `#MYCONSTANT` als 1 und `#MYCONSTANT2` als 1000 definiert. Wenn Sie eine Zeichenketten-Konstante definieren müssen, muss die Zeichenfolge in eckige Klammern eingeschlossen werden, z.B.: "MYSTRINGCONSTANT = [Test123]". Wenn Sie eckige Klammern innerhalb einer Zeichenketten-Konstante speichern wollen, müssen Sie sie einfach duplizieren (z.B. `[[ ]]`), damit sie nicht mit den Zeichenketten-Konstanten-Trennzeichen verwechselt werden können.

**`-sizeable:`**

Geben Sie dieses Argument an, damit das Hollywood-Fenster ein Größensymbol an der rechten unteren Ecke erhält. Falls Sie ein rahmenloses Fenster eingestellt haben wird es unsichtbar sein, aber es ist dennoch vorhanden.

**-skipplugins mask:**

Mit diesem Argument wird definiert, welche Plugins beim Aufstarten nicht geladen werden sollen. Mehrfache Plugins werden mit einem senkrechten Strich (|) untereinander getrennt. Sie können mit einem Stern (\*) Hollywood anweisen, gar keine Plugins zu laden. Späteres laden von Plugins ist mit der Präprozessor-Anweisung **@REQUIRE** und dem Befehl **LoadPlugin()** möglich.

**-smoothscale:**

Wenn **-autoscale** oder **-layerscale** aktiv ist und Sie **-smoothscale** angeben, wird Antialias für die Kantenglättung benutzt. Dies sieht besser aus, ist aber auch langsamer. Bitte lesen Sie die Dokumentation von **-autoscale/-layerscale** für mehr Informationen.

**-softtimer:**

Mit diesem Argument wird Hollywood den Software-Timer mit geringerer Genauigkeit als den Hardware-Timer benutzen. Das ist manchmal notwendig, da der Timer von älterer (XP) Hardware ein unerwartetes Verhalten an den Tag legen kann. Bei neueren Hardware- und Windows-Versionen müssen Sie **-softtimer** niemals nutzen. [Nur Windows]

**-softwarerenderer:**

Geben Sie dieses Argument an, um Hollywoods GPU-beschleunigte Direct2D-Wiedergabe auf Windows-Systemen zu deaktivieren. Wenn **-softwarerenderer** festgelegt ist, verwendet Hollywood seine CPU-basierte Wiedergabe für maximale Kompatibilität. [Nur Windows]

**-startcolor color:**

Nur erforderlich wenn beim Argument **-backfill** der Type auf **gradient** oder **color** gesetzt wurde. **color** ist eine im **RGB-Format** angegebene Farbe (z.B. **\$00FF00** für Grün). Kann auch mit **-backfill** Type **picture** verwendet werden, was dazu führt, dass der Hintergrund mit dieser Farbe gefüllt wird. Wenn **-endcolor** auch einen Wert erhalten hat, entsteht hinter dem Bild ein Farbverlauf zwischen **-startcolor** und **-endcolor**.

**-stayactive:**

(wurde in Hollywood 2.0 entfernt)

**-systemscales:**

Wenn Sie dieses Argument setzen, wird der Skalierungsfaktor des Hostsystems automatisch auf Ihr Display angewendet. Dies kann auf Systemen mit hohen DPI-Monitoren nützlich sein. Wenn sich Ihr Bildschirm beispielsweise normalerweise in 640x480 Pixeln öffnet und Sie ihn auf einem Monitor mit doppelt so vielen Punkte pro Zoll (DPI) verwenden, wird die Angabe der Option **-systemscales** Ihr Skript automatisch auf 1280x960 Pixel skalieren, damit es nicht winzig aussieht, nur weil das System einen hohen DPI-Monitor verwendet. Standardmäßig aktiviert **-systemscales** die Autoskalierung. Wenn Sie möchten, dass es stattdessen die Ebenenskalierung verwendet, übergeben Sie einfach das Argument **-layerscales**. Beachten Sie, dass **-systemscales** intern den gleichen Skaliermodus verwendet wie **-scalefactor**, so dass sich Skripte, die **-systemscales** verwenden, so verhalten, als ob **-scalefactor** angegeben wurde. Es ist sogar möglich, das Argument **-scalefactor** zusätzlich zur

`-systemsca` zu verwenden. In diesem Fall wird der in `-scalefactor` angegebene Wert mit dem Standard-Skalierungsfaktor des Hostsystems multipliziert. Bitte lesen Sie auch die Dokumentation von `-autoscale/-layersca` für weitere Informationen zu den Skalierungssystemen von Hollywood. Siehe [Abschnitt 24.24 \[Skalierungssysteme\]](#), [Seite 402](#), für Details. Beachten Sie, dass Sie unter Windows in der Präprozessor-Anweisung `@OPTIONS` auch den Tag `DPIAware` auf `True` setzen müssen, um `-systemsca` zu verwenden. Siehe [Abschnitt 50.25 \[OPTIONS\]](#), [Seite 1107](#), für Details.

**-tempdir path:**

Dieses Argument definiert den Pfad, wo Hollywood seine temporären Dateien abspeichern soll. Dies ist besonders auf AmigaOS und kompatible Systemen nützlich, da sonst die temporären Dateien in der RAM-Disk gespeichert werden. Das kann auf Systemen mit wenig RAM oder großen Projekten zu Problemen führen. Um den aktuellen Pfad für die temporären Dateien zu benutzen, geben Sie bei path "" ein.

**-usequartz:**

Mit `-usequartz` als Argument wird Hollywood für alle Grafikausgaben Quarz 2D benutzen. Voreingestellt ist QuickDraw, weil es viel schneller ist. Aber falls auf macOS irgendwelche Grafikprobleme auftauchen, können Sie vielleicht mit diesem Argument beseitigt werden. Beachten Sie, dass dieses Argument nur von den PowerPC-Version von Hollywood unterstützt wird. Die x86/x64-Versionen von Hollywood für macOS verwenden immer Quarz 2D. [Nur macOS]

**-usewpa:** Mit diesem Argument wird Hollywood Device Independent Bitmaps anstelle von Standard OS Bitmaps benutzen. Device Independent Bitmaps sind normalerweise langsamer als Standard OS Bitmaps. Ausnahmen sind WinUAE und AROS, die beide ineffizient mit Standard OS Bitmaps umgehen können. Darum wird auf WinUAE und AROS automatisch `-usewpa` aktiviert, um Hollywood zu beschleunigen. Möchten Sie das ausschalten, setzen Sie `-wpamode` auf 0 (Null). Beachten Sie: `-usewpa` ist ein Lowlevel-Argument, das hauptsächlich zum Testen gedacht ist. Normalerweise werden Sie dieses Argument nicht benutzen. [Nur AmigaOS]

**-videofps fps:**

Wird nur zusammen mit `-videoout` benutzt. Ist `-videoout` aktiv, wird mit `-videofps` Hollywood mitgeteilt, wie viele Einzelbilder pro Sekunde (frames per seconds fps) der Videorecorder aufzeichnen soll. Wird dieses Argument nicht definiert, sind 50 Einzelbilder in der Sekunde voreingestellt. Siehe [Abschnitt 4.6 \[Verwendung des Videorecorders\]](#), [Seite 63](#), für Details.

**-videoout file:**

Dieses Argument schaltet den Hollywood internen Videorecorder an. Ihr Skript wird nun als ein AVI-Video aufgezeichnet, das Sie später z.B. auf eine DVD brennen können. Siehe [Abschnitt 4.6 \[Verwendung des Videorecorders\]](#), [Seite 63](#), für Details.

**-videopointer:**

Wird nur zusammen mit `-videoout` benutzt. Wenn Sie `-videopointer` angeben, wird der Mauszeiger immer im Video aufgezeichnet werden. Voreingestellt

ist, dass kein Mauszeiger aufgezeichnet wird. Wenn Sie z.B. dem Benutzer die Bedienung eines Programms mit der Maus erklären wollen, geben Sie dieses Argument an. Siehe [Abschnitt 4.6 \[Verwendung des Videorecorders\]](#), [Seite 63](#), für Details.

**-videoquality quality:**

Wird nur zusammen mit `-videoout` benutzt. Ist `-videoout` aktiv, kann `-videoquality` die Kompression des Videos definieren. Quality wird in Prozent von 0 bis 100 angegeben. Voreingestellt ist 90 für eine hohe Qualität und es braucht nicht mehr so viel Plattenspeicher. Um die Videodatei noch kleiner zu halten, wählen Sie eine niedrigere Zahl für quality. Siehe [Abschnitt 4.6 \[Verwendung des Videorecorders\]](#), [Seite 63](#), für Details.

**-videostrategy strategy:**

Wird nur zusammen mit `-videoout` benutzt. Ist `-videoout` aktiv, können Sie mit diesem Argument die Aufnahmestrategie festlegen. Gegenwärtig können sie für strategy `wait` und `raw` festlegen. Standardmäßig ist `wait` festgelegt. Siehe [Abschnitt 4.6 \[Verwendung des Videorecorders\]](#), [Seite 63](#), für Details.

**-vsync:** Auf Windows-Systemen kann dieses Argument verwendet werden, um Hollywoods Wiedergabe zu zwingen, die Bildwiederholrate des Monitors zu drosseln. Dies bedeutet, dass Sie keine Befehle wie `VWait()` mehr verwenden müssen, um das Zeichnen zu drosseln. Beachten Sie jedoch, dass Sie, wenn Sie dies auf `True` setzen, sicherstellen müssen, dass nur Vollbilder gezeichnet werden, da sonst das Zeichnen extrem langsam wird. Eine Vollbildzeichnung kann entweder mit einem Doppelpuffer oder mit `BeginRefresh()` und `EndRefresh()` erreicht werden. Beachten Sie auch, dass `VSyn` derzeit nur unter Windows unterstützt wird und auch nur, wenn Hollywood seine Direct2D-Wiedergabe verwendet. Direct2D ist vor Windows Vista SP2 nicht verfügbar. [Nur Windows]

**-window:** Geben Sie dieses Argument an, um Hollywood in einem Fenster auf der Workbench öffnen zu lassen, anstatt auf seinem eigenen Bildschirm. Das ist auch der voreingestellte Wert, wenn `-fullscreen` nicht gesetzt wurde.

**-winwidth width:**

Mit diesem Argument können Sie die Anfangsdisplaybreite einstellen, ohne eine der Skalierungssysteme zu aktivieren. Dies hat die gleiche Wirkung, wie wenn der Benutzer das Display auf die angegebene Breite verkleinert hat. Aus diesem Grund erhält Ihr Skript auch ein "SizeWindow"-Ereignis, nachdem Hollywood gestartet wurde und Sie dieses Konsolenargument verwendet haben. Beachten Sie, dass kein Skalierungssystem aktiviert wird. Somit muss Ihr Skript fähig sein, sich selbst an die neuen Dimensionen anzupassen.

**-winheight height:**

Gleich wie `-winwidth`, gibt aber die Displayhöhe an.

**-wpamode mode:**

Wenn `-usewpa` aktiv ist, kann mit diesem Argument der Device Independent Bitmapmodus festgelegt werden. Mit 0 wird `-usewpa` ausgeschaltet, mit 1 wird 32-bit DIB und mit 2 der Workbench Compliant DIB Modus benutzt. Voreingestellt ist 1, was die beste Leistung erbringen sollte. Beachten Sie: `-usewpa` ist ein



Lowlevel-Argument, das hauptsächlich zum Testen gedacht ist. Normalerweise werden Sie dieses Argument nicht benutzen. [Nur AmigaOS]

**-xserver name:**

Mit diesem Argument definieren Sie den X-Server, mit dem sich Hollywood in Verbindung setzen soll. Voreingestellt benutzt Hollywood den X-Server, der in der `DISPLAY` Umgebungsvariable angegeben ist. [Nur Linux]

### 3.3 Konsolenargumente ohne Konsole

Hollywood und kompilierte Programme kennen viele Konsolenargumente, die benutzt werden können, um verschiedene Funktionen zu kontrollieren. Meistens werden Sie aber Hollywood oder das kompilierte Programm nicht von der Konsole aus starten. Wie können Sie somit Argumente übergeben, wenn Sie keine Konsole benutzen? Da das von Plattform zu Plattform unterschiedlich ist, folgt nun ein Überblick, wie die Konsolenargumente übergeben werden können.

Beachten Sie, dass Voreingestellt alle von Hollywood kompilierten Programme mit Konsolenargumente umgehen und die angegebenen Argumente das Aussehen und Ausführen des Programms beeinflussen können. Möchten Sie diese Beeinflussung durch den Benutzer nicht zulassen, müssen Sie beim Kompilieren Ihres Programmes das **-locksettings** Argument benutzen.

Hier nun die Übersicht über die Möglichkeiten, Konsolenargumente ohne Konsole zu übergeben:

#### 1) AmigaOS:

Erstellen Sie einfach ein Piktogramm für Ihr Programm und fügen Sie die Konsolenargumente als Tooltypes hinzu. Z.B. wenn Sie Ihrem Programm `RAM:MyCoolProg` einen Farbverlauf von Schwarz nach Blau verpassen wollen, erstellen Sie ein Piktogramm mit dem Namen `RAM:MyCoolProg.info` und fügen folgende Tooltypes hinzu:

```
BACKFILL=GRADIENT
STARTCOLOR=$000000
ENDCOLOR=$0000ff
BORDERLESS
FIXED
(FULLSCREEN)
```

Beachten Sie, dass in Klammern eingeschlossene Tooltypes wie im oben erwähnten Beispiel `FULLSCREEN` ignoriert werden.

#### 2) Windows:

Unter Windows können Sie die Konsolenargumente in eine `*.ini` Datei schreiben. Nehmen wir an, Sie haben ein Programm `MyCoolProg.exe` kompiliert und installierten es in

```
C:/Program Files/MyCoolProg/MyCoolProg.exe
```

Sie können jetzt eine `*.ini` Datei mit weitere Optionen für Ihr Programm erstellen. Diese `*.ini` Datei muss den gleichen Namen wie Ihr Programm tragen, ansonsten merkt es nicht, dass die `*.ini` Datei existiert:

```
C:/Program Files/MyCoolProg/MyCoolProg.ini
```

Und nun benutzen Sie einen Texteditor. Um z.B. im Hintergrund einen Farbverlauf von Schwarz nach Blau zu plazieren, können Sie folgendes in die \*.ini Datei einfügen:

```
Backfill=Gradient
StartColor=$000000
EndColor=$0000ff
Borderless=True
Fixed=True
```

Für mehr Infos lesen Sie die **Konsolenargumente** durch, damit Sie wissen, was Sie alles in der \*.ini Datei angeben können.

### 3) macOS:

Unter macOS haben Sie zwei Möglichkeiten, Konsolenargumente ohne Konsole zu übergeben:

1. Öffnen Sie die Datei Info.plist, welche in Ihrem Applications-Ordner gespeichert ist:

```
/Programs/MyCoolProg.app/Contents/Info.plist
```

Öffnen Sie die Datei mit einem Texteditor und suchen Sie die Zeile mit dem Eintrag CFExecutableArgs. Hier können Sie nun die von Ihnen benötigten Konsolenargumente eintragen. Zum Beispiel:

```
<key>CFBundleExecutableArgs</key>
<string>
-backfill gradient -startcolor $000000 -endcolor $0000ff
-borderless -fixed
</string>
```

Mit diesen Konsolenargumente bekommt Ihr Programm einen Farbverlauf von Schwarz nach Blau.

2. Als alternative Möglichkeit können Sie wie unter Windows eine \*.ini Datei erstellen (siehe oben unter Windows). Der einzige Unterschied ist, dass Sie unter macOS die \*.ini Datei im Resources Verzeichnis von Ihrem Applications-Ordner speichern müssen. Wenn z.B. Ihre Applikation unter folgendem Pfad abgespeichert wurde:

```
/Programs/MyCoolProg.app
```

Dann speichern Sie Ihre \*.ini Datei hier:

```
/Programs/MyCoolProg.app/Contents/Resources/MyCoolProg.ini
```

Der Rest ist genau das Gleiche wie unter Windows.

### 4) Linux:

Unter Linux können Sie alle Konsolenargumente auch in eine \*.ini Datei hineinschreiben. Siehe unter Windows für die Details.



## 4 Compiler und Linker

### 4.1 Kompilieren von Programmen

Der Hollywood Compiler kann entweder von der **GUI** aus oder mit dem **-compile** Argument über die Konsole verwendet werden.

Einmal aufgerufen, liest der Compiler die angegebene Skriptdatei, kompiliert und verknüpft sie in eine spezielle nur ausführbare Version von Hollywood. Alle externen Daten, die mit **Präprozessor-Anweisungen** deklariert wurden, werden eingebunden (außer es wird ausdrücklich deklariert, dass eine Datei nicht eingebunden werden soll). Die Plattform für die ausführbare Datei wird durch die Angabe des Arguments **-exetype** definiert, welches auf die folgenden ausführbaren Typen gesetzt werden kann:

<b>amigaos4</b>	AmigaOS 4 (PowerPC)
<b>aros</b>	AROS (x86)
<b>classic</b>	AmigaOS 3.x (68020+)
<b>classic881</b>	AmigaOS 3.x (68020+) mit Coprozessor (68881/2 oder 68040/68060)
<b>linux</b>	Linux (x86)
<b>linux64</b>	Linux (x64)
<b>linuxarm</b>	Linux (arm)
<b>linuxppc</b>	Linux (PowerPC)
<b>macos</b>	macOS Programmpaket (PowerPC)
<b>macosarm64</b>	macOS Programmpaket (arm64)
<b>macos86</b>	macOS Programmpaket (x86)
<b>macos64</b>	macOS Programmpaket (x64)
<b>morphos</b>	MorphOS (PowerPC)
<b>warpos</b>	WarpOS Mixed-Binary (68040/PowerPC)
<b>win32</b>	Windows (x86)
<b>win64</b>	Windows (x64)
<b>applet</b>	Universal Hollywood-Applet, welches auf jedem System mit einem Hollywood Player läuft

In der 68k-Version von Hollywood ist bei **-exetype** standardmäßig **classic**, bei der AmigaOS4 Version **-exetype** **amigaos4** und so weiter eingestellt.

Wenn Ihr Skript eine Menge externe Daten verwendet, kann die ausführbare Datei sehr groß werden, da Hollywood alle mit Präprozessor-Anweisungen deklarierten Dateien einbinden wird. Wenn Sie das nicht wollen, können Sie das Argument **Link** benutzen, das alle Präprozessor-Anweisungen unterstützen. Damit wird dem Linker mitgeteilt, bestimmte

Dateien nicht einzubinden. Alternativ können Sie die Dateien auch mit einem normalen Hollywood-Befehle anstelle von Präprozessor-Anweisungen laden.

Die erzeugte ausführbare Datei wird die gleichen **Konsolenargumente** wie das Hollywood-Hauptprogramm akzeptieren. Daher können Sie es zum Beispiel mit einem randlosen Fenster starten, indem Sie es mit dem **-borderless** Argument aufrufen.

Sie können auch Plugins mit dem Argument **-linkplugins** in die ausführbaren Dateien einbinden. Aber man muss die Plugin-Lizenz sehr vorsichtig durchlesen, da nach dem Einbinden Ihr ganzes Projekt der Plugin-Lizenz unterstehen kann. Siehe **Abschnitt 3.2 [Konsolenargumente]**, **Seite 33**, für Details.

## 4.2 Kompilieren von Applets

Neben eigenständige Programme können Sie aus Ihren Skripten Hollywood-Applets erstellen. Diese sind sehr viel kleiner, weil sie nicht den Hollywood-Player enthalten. Hollywood-Applets können mit dem Interpreter oder den frei verfügbaren Hollywood-Player gestartet werden. Der Vorteil von Applets ist, dass man viel Platz sparen kann. Stellen Sie sich vor, Sie müssten Ihr Skript für alle Plattformen kompilieren, die Hollywood unterstützt (AmigaOS3, AmigaOS4, WarpOS, MorphOS, AROS, Windows, macOS, Linux). Die Player für alle diese Plattformen nehmen alleine mehr als 50 Megabyte ein, so würden Sie ein ziemlich großes Archiv verwalten müssen (und das nur für ein Projekt). In diesem Fall ist es eine viel bessere Idee, das Skript zu einem Hollywood-Applet zu kompilieren. Anschließend kann der Benutzer einfach die frei verfügbaren Hollywood-Player für seine Plattform von <https://www.hollywood-mal.de/> verwenden, um das Applet auszuführen. Und Sie müssen nur Ihr Applet verteilen.

Um Applets mit Hollywood zu kompilieren, benutzen Sie den Parameter **APPLET** mit dem Argument **-exetype** oder verwenden Sie die GUI. Hollywood-Applets tragen die Endung **\*.hwa**.

## 4.3 Einbinden von Datendateien

Standardmäßig verknüpft der Linker von Hollywood automatisch alle mit den Präprozessor-Anweisungen deklarierten externen Datendateien in das ausführbare Programm oder dem Applet. Somit wird z.B. die Datei **test.jpg** automatisch mit Ihrem ausführbaren Programm oder Applet verknüpft:

```
@BGPIC 1, "test.jpg"
WaitLeftMouse
End
```

Wenn Sie dies nicht möchten, können Sie den Tag **Link** setzen, welches von allen Präprozessor-Anweisungen für Dateien mit **False** akzeptiert wird. In diesem Fall wird die von der Präprozessor-Anweisung verwendete Datei nicht eingebunden. Der Code sieht dann so aus:

```
@BGPIC 1, "test.jpg", {Link = False}
WaitLeftMouse
End
```

Manchmal können Sie auch Dateien verknüpfen, die von Ihrem Skript zur Laufzeit in Ihr ausführbares Programm oder Ihr Applet geladen werden. Betrachten Sie zum Beispiel den folgenden Code:

```
LoadBGPic(1, "test.jpg")
LoadBrush(1, "title.png")
DisplayBGPic(1)
DisplayBrush(1, #CENTER, #CENTER)
WaitLeftMouse
End
```

Standardmäßig werden `test.jpg` und `title.png` nicht mit Ihrem ausführbaren Programm oder Applet verknüpft, da sie nicht mit einer Präprozessor-Anweisung deklariert wurden, sondern stattdessen zur Laufzeit mit `LoadBGPic()` und `LoadBrush()` geladen wurden. Dennoch ist es möglich, `test.jpg` und `title.png` mit Ihrem ausführbaren Programm oder Applet zu verlinken. Dies kann entweder mit der Compiler-Option `-linkfiles` oder der Präprozessor-Anweisung `@LINKER` erreicht werden.

Sie müssen eine Datenbankdatei an `-linkfiles` übergeben. Die Datenbankdatei ist eine einfache UTF-8 Textdatei, die eine Liste von Dateien enthält, die in das Applet oder ausführbaren Programm eingebunden werden, die von Hollywood kompiliert wird. Sie müssen nur eine Datei pro Zeile in der Datenbankdatei angeben. Die Dateiangabe muss mit der Angabe übereinstimmen, die Sie in Ihrem Skript verwenden.

Wenn Sie das Konsolenargument `-linkfiles` verwenden, müssen Sie eine Datenbankdatei übergeben. Die Datenbankdatei ist eine einfache UTF-8-Textdatei, die eine Liste von Dateien enthält, die in das Applet oder dem ausführbare Programm eingebunden werden sollen, das von Hollywood kompiliert wird. Sie dürfen in der Datenbankdatei nur eine Datei pro Zeile angeben. Die Dateiangabe muss mit der Angabe identisch sein, die Sie in Ihrem Skript verwenden. Wenn zum Beispiel der Befehl `LoadBrush(1, "data/menu.png")` in Ihrem Skript steht und Sie die Datei `data/menu.png` in ihrem Applet/Programm einbinden möchten, müssen Sie sie in die Datenbank schreiben, die Sie an `-linkfiles` übergeben. Aber Sie müssen die gleichen Angaben verwenden, d.h. Sie müssen `data/menu.png` verwenden! Die Angabe von `MyScripts/CoolGame/data/menu.png` in der Datenbank funktioniert nicht! Die Angabe, die in der Linkdatei-Datenbank und im Skript verwendet wird, muss dasselbe sein, da Hollywood sonst nicht feststellen kann, welche Datei geladen werden muss.

Um die Dateien `test.jpg` und `title.png` mit unserem ausführbaren Programm oder Applet zu verknüpfen, muss die Datenbankdatei, die wir an `-linkfiles` übergeben, so aussehen:

```
test.jpg
title.png
```

Das ist alles! Der Hollywood-Linker verknüpft dann `test.jpg` und `title.png` mit dem ausführbaren Programm oder dem Applet und der Aufruf von `LoadBGPic()` und `LoadBrush()` im oben vorgestellten Skript lädt `test.jpg` und `title.png` jeweils direkt aus dem ausführbaren Programm oder Applet statt aus einer externen Quelle.

Dasselbe kann mit der Präprozessor-Anweisung `@LINKER` erreicht werden. Der einzige Unterschied besteht darin, dass die zu verknüpfenden Dateien nicht in einer externen Datenbankdatei an Hollywood übergeben werden müssen. Sie müssen stattdessen direkt in Ihrem Skript als Teil der Präprozessor-Anweisung `@LINKER` gespeichert werden. Alle anderen Regeln sind die gleichen wie bei `-linkfiles`. Wenn Sie `-linkfiles` nicht wie oben

beschrieben verwenden möchten, können Sie Ihrem Skript auch die folgende Zeile hinzufügen und dasselbe erreichen:

```
@LINKER {Files = {"test.jpg", "title.png"}}
```

Sie können dem Tag **Files** der Präprozessor-Anweisung **@LINKER** beliebig viele Dateien hinzufügen, die mit Ihrem Applet oder dem ausführbaren Programm verknüpft werden sollen. Stellen Sie einfach sicher, dass die Pfadangabe der Dateien, die Sie an **@LINKER** übergeben, mit der Pfadangabe übereinstimmt, die später im Code verwendet wird, damit Hollywood die verknüpften Dateien den einzelnen im Skript verwendeten Dateien korrekt zuordnen kann.

Wenn Sie viele Dateien mit Ihrem Applet oder Ihrem ausführbaren Programm verknüpfen müssen, können Sie alle diese Dateien in einem Verzeichnis ablegen und dann Hollywood anweisen, alles in diesem Verzeichnis mit dem Applet oder dem ausführbaren Programm zu verknüpfen. Dies geschieht mit der Präprozessor-Anweisung **@DIRECTORY**. In der folgenden Zeile wird beispielsweise angegeben, dass Hollywood alle Dateien im Verzeichnis **data** mit dem Applet oder dem ausführbaren Programm verknüpft:

```
@DIRECTORY 1, "data"
```

Wenn Sie dies getan haben, können Sie mit dem Befehl **GetDirectoryEntry()** auf die einzelnen Dateien im Verzeichnis **data** zugreifen. Um beispielsweise die Dateien **data/test.jpg** und **data/title.png** mit **LoadBGPic()** und **LoadBrush()** zu laden, würden Sie folgenden Code schreiben:

```
LoadBGPic(1, GetDirectoryEntry(1, "data/test.jpg"))
LoadBrush(1, GetDirectoryEntry(1, "data/title.png"))
```

Die Präprozessor-Anweisung **@DIRECTORY** ist sehr flexibel, da sie die gesamte Verzeichnisstruktur in einem Applet oder einem ausführbaren Programm archiviert. Somit ist es möglich, dieses archivierte Verzeichnis (und alle seine Unterverzeichnisse!) so zu durchlaufen, als wären sie normal auf dem Datenträger. Siehe [Abschnitt 20.13 \[DIRECTORY\]](#), [Seite 254](#), für Details.

## 4.4 Einbinden von Schriftarten

Standardmäßig wird der Linker von Hollywood automatisch alle mit der Präprozessor-Anweisung **@FONT** deklarierten Schriftarten in das ausführbare Programm oder Applet eingebunden. Somit wird z.B. die Schriftart **Arial** automatisch mit Ihrem ausführbaren Programm oder Applet verknüpft:

```
@FONT 1, "Arial", 36
WaitLeftMouse
End
```

Wenn Sie dies nicht möchten, können Sie den Tag **Link** auf **False** setzen, der von der Präprozessor-Anweisung **@FONT** akzeptiert wird. In diesem Fall wird die im Präprozessor-Anweisung angegebene Schriftart nicht verknüpft. Der Code sieht dann so aus:

```
@FONT 1, "Arial", 36, {Link = False}
WaitLeftMouse
End
```

Manchmal möchten Sie vielleicht auch Fonts verknüpfen, die von Ihrem Skript zur Laufzeit in Ihr ausführbares Programm oder Applet geladen werden. Betrachten Sie zum Beispiel den folgenden Code:

```
SetFont("Arial", 36)
WaitLeftMouse
End
```

Standardmäßig wird die Schriftart **Arial** nicht in Ihr ausführbares Programm oder Applet eingebunden, da sie nicht mit einer Präprozessor-Anweisung deklariert, sondern stattdessen zur Laufzeit mit dem Befehl **SetFont()** geladen wird. Dennoch ist es möglich, die Schriftart **Arial** mit Ihrem Programm oder Applet zu verknüpfen. Dies kann durch die Verwendung der Compiler-Option **-linkfonts** oder des Präprozessor-Anweisung **@LINKER** erreicht werden.

Wenn Sie die Compileroption **-linkfonts** verwenden, müssen Sie eine Datenbankdatei übergeben. Die Datenbankdatei ist eine einfache UTF-8-Textdatei, die eine Liste von Schriftarten enthält, die mit dem Applet oder dem Programm verknüpft werden sollen, die von Hollywood erstellt wird. Sie müssen nur eine Schriftart pro Zeile in der Datenbankdatei angeben. Die Schriftart kann entweder der Name einer Schriftart oder ein Pfad zu einer \*.ttf- oder \*.otf-Datei sein. Die Schriftart kann entweder der Name einer Schriftart oder ein Pfad zu einer \*.ttf- oder \*.otf-Datei sein. Beachten Sie, dass Sie beim direkten Übergeben von Pfaden zu \*.ttf- oder \*.otf-Dateien das integrierte Schriftartenmodul verwenden müssen, da nur diese Schriftarten aus Dateien laden kann. Eine Schriftdatenbank könnte wie folgt aussehen:

```
Arial
"Times New Roman"
FuturaL
helvetica
data/arial.ttf
```

Vergessen Sie nicht, Anführungszeichen zu verwenden, wenn Sie Schriftnamen mit Leerzeichen angeben!

Dasselbe kann mit der Präprozessor-Anweisung **@LINKER** erreicht werden. Der einzige Unterschied besteht darin, dass die zu verknüpfenden Schriftarten nicht in einer externen Datenbankdatei an Hollywood übergeben werden müssen. Sie müssen stattdessen direkt in Ihrem Skript als Teil der Präprozessor-Anweisung **@LINKER** gespeichert werden. Alle anderen Regeln sind die gleichen wie bei **-linkfonts**. Wenn Sie **-linkfonts** nicht wie oben beschrieben verwenden möchten, können Sie Ihrem Skript auch die folgende Zeile hinzufügen und dasselbe erreichen:

```
@LINKER {Fonts = {"Arial", "Times New Roman", "FuturaL", "helvetica",
                  "data/arial.ttf"}}
```

Sie können dem Tag **Fonts** der Präprozessor-Anweisung **@LINKER** beliebig viele Schriftarten hinzufügen, die mit Ihrem Applet oder Programm verknüpft werden sollen.

**Wichtiger Hinweis:** Bitte beachten Sie, dass die meisten Schriften urheberrechtlich geschützt sind und es nicht erlaubt ist, sie in Ihre Programme einzubinden/zu verknüpfen, ohne eine Lizenz zu erwerben. So stellen Sie sicher, dass Sie die Lizenz der Schriftart überprüfen, die Sie in Ihr Programm einbinden werden! Wenn Sie nicht für Schriftlizenzen bezahlen möchten, ist es ratsam, eine kostenlose Schriftart wie DejaVu oder Bitstream Vera zu verwenden oder

eine der TrueType-Schriftarten zu verwenden, die in Hollywood eingebaut sind (`#SANS`, `#SERIF`, `#MONOSPACE`, vgl. `SetFont()`)

## 4.5 Einbinden von Plugins

Im Gegensatz zu Datendateien und Schriftarten werden Plugins nicht automatisch mit Ihrem ausführbaren Programm verknüpft, wenn Sie sie mit der Präprozessor-Anweisung deklarieren. Der folgende Code wird zum Beispiel den Linker nicht anweisen, "jpeg2000" in Ihr ausführbares Programm einzubinden:

```
@REQUIRE "jpeg2000"
```

Wenn Sie `jpeg2000.hwp` in Ihr Programm einbinden möchten, müssen Sie den Tag `Link` auf `True` setzen. Der Code sieht dann so aus:

```
@REQUIRE "jpeg2000", {Link = True}
```

In diesem Fall wird `jpeg2000.hwp` mit Ihrem ausführbaren Programm verknüpft und der Benutzer muss keine Kopie von `jpeg2000.hwp` behalten, da sie bereits im Programm eingebunden ist.

Alternativ können Sie auch das Konsolenargumente `-linkplugins` verwenden, um Plugins mit Ihrem Programm zu verknüpfen. Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), Seite 33, für Details.

Beachten Sie, dass Plugins nur mit ausführbaren Programmen und nicht mit Applets verknüpft werden können, da Applets plattformunabhängig sind und Plugins nicht.

Bevor Sie den Plugin-Linker nutzen können, müssen Sie zunächst die Plugins, die Sie verknüpfen möchten, in ein Verzeichnis mit dem Namen `LinkerPlugins` kopieren. Auf AmigaOS und kompatiblen Systemen muss dieses Verzeichnis im Installationsverzeichnis von Hollywood erstellt werden, d.h. Sie müssen hier das Verzeichnis `Hollywood:LinkerPlugins` erstellen. Auf allen anderen Systemen müssen Sie das Verzeichnis `LinkerPlugins` in dem Verzeichnis erstellen, in dem Hollywood installiert wurde, d.h. neben dem Programm Hollywood. Denken Sie daran, dass auf macOS dies im Programmpaket sein wird, d.h. in `HollywoodInterpreter.app/Contents/Resources/LinkerPlugins`. Außerdem müssen Sie im Verzeichnis `LinkerPlugins` folgende Unterverzeichnisse erstellen:

```
arm-android-v7a
arm64-android-v8a
arm-ios
arm-linux
m68k-amigaos
m881-amigaos
ppc-amigaos
ppc-linux
ppc-macos
ppc-morphos
ppc-warppup
x86-aros
x86-macos
x86-linux
x86-windows
```

```
x86-windows-console  
x64-linux  
x64-macos  
x64-windows  
x64-windows-console
```

Danach müssen Sie die Plugins kopieren, die Sie mit diesen Unterverzeichnissen verknüpfen möchten. Sie müssen Plugins für alle Plattformen kopieren, für die Sie ein ausführbares Programm kompilieren möchten. Wenn Sie das nicht tun, kann der Linker die Plugins nicht verknüpfen. Beachten Sie, dass der Linker nur Plugins im Verzeichnis `LinkerPlugins` sucht. Es wird nicht irgendwo anders nachschauen, vor allem nicht in den Standard-Plugins-Verzeichnissen.

Beachten Sie, dass bei der Erstellung von ausführbaren Programmen für die Plattform `m881-amigaos` der Linker auch nach Plugins im Verzeichnis `m68k-amigaos` sucht, da beide Plattform-Architekturen vollständig kompatibel sind. Dasselbe gilt für den `ppc-warpup`, der auch die Plattform-Architekturen `m68k-Amigaos` und `m881-Amigaos` berücksichtigen wird. Außerdem sind `x86-windows-console` und `x86-windows` kompatibel zu `x64-windows-console` und `x64-windows`.

**Wichtiger Hinweis:** Achten Sie darauf, sorgfältig die Lizenz von jedem Plugin zu lesen, die Sie mit Ihrem Programm verknüpfen möchten, weil viele Lizenzen sehr restriktiv, wenn es um statische Verknüpfung geht. Zum Beispiel wenn Sie ein Plugin verknüpfen, das unter der LGPL-Lizenz lizenziert ist, wird Ihr komplettes Projekt automatisch auch LGPL und Sie müssen alle Quellen und Datendateien bereitstellen. So studieren Sie die Plugin-Lizenzen, bevor Sie sie mit Ihren Programmen verknüpfen.

## 4.6 Verwendung des Videorecorders

Beginnend mit Hollywood 4.0 ist es möglich, Hollywood-Skripte als AVI-Video dateien zu speichern. Dies ist nützlich, wenn Sie zum Beispiel DVDs Ihrer Hollywood-Skripte erstellen möchten oder einfach Ihr Programm auf einer derzeit nicht unterstützten Plattform zeigen wollen. Als Video können Sie die Datei auch in Video-Editing-Software zur Weiterverarbeitung oder in Formatkonvertierer importieren.

Hollywoods Videorecorder wurde mit der Idee konzipiert, das genaue Verhalten des Hollywood-Skriptes in einer Videodatei zu reproduzieren. So werden Sie je nach Art des Skriptes höchstwahrscheinlich keinen Unterschied zwischen der Videodatei und dem tatsächlichen Hollywoodskript bemerken. Hollywoods Videorecorder versucht das Skript so aufzunehmen, wie es im Echtzeitmodus erscheinen würde. Daher ist es kein Problem für den Videorecorder mit Skripten umzugehen, die genaues Timing erfordern - zum Beispiel für die Synchronisierung mit der Musik. Der Videorecorder wirft ein besonderes Augenmerk darauf und versucht, alles in der richtigen Zeit aufzunehmen.

Um den Videorecorder-Modus zu aktivieren, müssen Sie einfach das Argument `-videoout` zusammen mit einem Dateinamen für das Video angeben. Hollywood wird dann im Aufnahmefmodus starten und Grafiken sowie Sounds werden nun in den Videostrom umgeleitet werden. Somit werden im Aufnahmefmodus keine Töne gespielt, weil die Tondaten sofort in den Videostrom wiedergegeben werden. Bitte beachten Sie, dass bestimmte Optionen außer Acht gelassen werden, wenn Hollywood sich im Video-Aufnahmefmodus befindet. Zum Beispiel wird Hollywood im Videoaufnahmefmodus nie im Vollbildmodus öffnen, auch wenn



Sie es deklariert haben, sondern immer im Fenstermodus. Auch wird das Fenster nicht skalierbar sein usw.

Die durch den Videorekorder geschriebene Videodatei wird im Format AVI Open DML (AVI 2.0) abgespeichert, somit sind Dateien größer als 2GB möglich. Hollywood verwendet derzeit den Motion-JPEG-Codec, um die Videobilder zu komprimieren. Audiodaten werden ohne Kompression in die Videodatei geschrieben. Sie können die Qualität der Motion-JPEG mit dem `-videoquality` Argument einstellen.

Um das beste Ergebnis erzielen zu können, müssen Sie einige Parameter im Videorecorder einstellen:

1. Erstens ist es ratsam, dass Sie den Videorecorder mitteilen, wie viele Bilder pro Sekunde aufgezeichnet werden sollen. Dies erreichen Sie mit dem `-videofps` Argument. Der Wert, den Sie hier angeben, sollte mit der Frequenz Ihrer Hauptschleife im Skript identisch sein. Wenn Ihre Hauptschleife mit 25 Einzelbilder pro Sekunde läuft, z.B. mit dem folgenden Code:

```
SetInterval(1, p_MainLoop, 1000\25)
```

Dann sollten Sie Ihre Video-Datei auch mit 25 Einzelbildern pro Sekunde laufen lassen. Das würden Sie mit

```
-videofps 25
```

in der Kommandozeile dem Videorecorder angeben müssen.

2. Sie können für die Videodatei eine Skalierung angeben. Denken Sie daran, dass sich die Videoauflösung nicht ändern kann, sondern während des gesamten Videos statisch ist. Hollywood kann jederzeit die Auflösung der Anzeige ändern, aber für Videodateien ist dies nicht möglich. Wenn Hollywood die Anzeigegröße ändert, während das Programm Videoaufnahme-Modus ist, werden die Grafiken auf die Videoauflösung skaliert werden. Standardmäßig wird die Videoauflösung auf die Auflösung des ersten Hintergrundbildes eingestellt. Wenn Sie eine andere Auflösung möchten, müssen Sie jedoch das mit den Argumenten `-scalewidth` und `-scaleheight` angeben.
3. Ihr Hollywood-Skript muss einem bestimmten Muster folgen, um es als Videodatei zu speichern. Insbesondere muss der Recorder wissen, wann sein Einzelbild-Puffer in die Videodatei gespeichert werden soll. Normalerweise geschieht dies, wenn er einer der Wartebefehle antrifft. Zum Beispiel:

```
VWait()  
Wait()  
WaitEvent()  
WaitTimer()  
etc.
```

Daher ist es notwendig, dass Sie in Ihrem Skript einen der obengenannten Befehle verwenden! Ihr Skript muss ein Zeitmechanismus verwenden, sonst kann es nicht richtig in eine Videodatei umgewandelt werden. Empfohlene Zeitmechanismen sind entweder eine Intervallfunktion, die eine bestimmte Anzahl Mal pro Sekunde oder mit `WaitTimer()` oder `VWait()` aufgerufen wird. Siehe [Abschnitt 15.3 \[Zeitsteuerung des Skripts\]](#), [Seite 170](#), für mehr Informationen über Verwendung eines richtigen Skriptzeitsteuerungs-Mechanismus.

Das Leeren/Abspeichern des Einzelbild-Puffer, wenn ein Wartebefehl auftritt, ist die Standardeinstellung. Normalerweise sollte diese "Wartestrategie" für alle Zwecke ge-



eignet sein. Mit zeitlich richtig getakteten Skripte, liefert die Wartestrategie die besten Ergebnisse. In einigen sehr seltenen Fällen - oder für die Fehlersuche - möchten Sie vielleicht stattdessen die "Rohstrategie" verwenden. Wenn `-videostrategy` auf `raw` gesetzt ist, wird der Videorecorder jedes Einzelbild im Strom gleich behandeln, egal ob gewartet wird oder nicht. In den meisten Fällen führt dies zu falsch getakteten Videos. So werden Sie wahrscheinlich nie die Rohstrategie verwenden.

4. Schließlich müssen Sie entscheiden, ob der Mauszeiger im Videostrom wiedergegeben werden soll. Standardmäßig ist dieser deaktiviert, weil dies nur Sinn in besonderen Situationen macht, z.B. wenn Sie ein Demo-Video erstellen, auf dem Benutzereingabe sichtbar sein sollen. Mit dem `-videopointer` Argument schalten Sie den Mauszeiger für die Aufnahme ein. Alle Mauszeiger Bewegungen werden dann in der Videodatei aufgezeichnet.



## 5 Plugins

### 5.1 Plugins

Hollywood-Funktionalität kann über Plugins erheblich erweitert werden. Plugins können das Laden und Speichern von zusätzlichen Video-, Audio-, Bild- und Sample-Formate unterstützen, können den Befehlssatz der Hollywood Sprache erweitern sowie echte Vektorgrafiken verwenden. Es ist sogar möglich, Plugins zu schreiben, die Teile von Hollywoods Kern wie den eingebauten Display und Audio-Treiber ersetzen. Es ist auch möglich Plugins zu schreiben, die Projektdateien von anderen Anwendungen wie Scala oder Powerpoint in Hollywood-Skripte konvertieren, so dass Hollywood direkt diese Projektdateien ausführen kann, obwohl sie nicht im Format `*.hws` sind.

### 5.2 Installation

Hollywood-Plugins verwenden als Endung `*.hwp`. Bei allen Systemen mit Ausnahme von AmigaOS und kompatiblen müssen Plugins in einem Verzeichnis mit dem Namen "Plugins" gespeichert werden, das sich im selben Verzeichnis wie das Hollywood-Programm befindet. Bei AmigaOS und kompatiblen Systemen müssen Plugins in `LIBS:Hollywood` installiert werden. Unter macOS muss sich das Verzeichnis "Plugins" im Verzeichnis "Resources" des Programmpakets befinden, d.h. im Verzeichnis `HollywoodInterpreter.app/Contents/Resources`. Beachten Sie, dass `HollywoodInterpreter.app` im `Hollywood.app` Programmpaket selbst gespeichert ist, nämlich in `Hollywood.app/Contents/Resources`.

Bei der Verteilung eines kompilierten Hollywood-Programms müssen Plugins, die von Ihrem Programm benötigt werden, einfach in das gleiche Verzeichnis wie Ihr Programm gespeichert werden. Beim Kompilieren vom Programmpaket für macOS müssen Plugins in das Verzeichnis "Resources" des Programmpakets eingefügt werden, d.h. in `MyProject.app/Contents/Resources`. Alternativ können Sie auch Plugins in Ihr Programm einbinden.

Die Android-Version von Hollywood unterstützt auch Hollywood-Plugins. Sie müssen sie in das Verzeichnis `Hollywood/Plugins` auf Ihrer SD-Karte kopieren. Hollywood wird diesen Ort bei jedem Start scannen und alle Plugins von dort laden.

### 5.3 Anwendung

Plugins werden beim Starten automatisch von Hollywood geladen. Wenn Sie das nicht wollen, können Sie das automatische Laden deaktivieren, indem Sie das Plugin umbenennen: Plugins, deren Dateinamen mit einem Unterstrichzeichen (`'_'`) beginnen, werden beim Starten nicht automatisch von Hollywood geladen. Alternativ können Sie auch das Konsolenargument `-skipplugins` verwenden, um das automatische Laden bestimmter Plugins zu überspringen. Plugins, die beim Start nicht geladen wurden, können später mit der Präprozessor-Anweisung `@REQUIRE` oder dem Befehl `LoadPlugin()` geladen werden. Siehe [Abschnitt 44.6 \[REQUIRE\]](#), Seite 990, für Details.

Bitte beachten Sie, dass obwohl Hollywood alle Plugins beim Start automatisch lädt, viele Plugins mit der Präprozessor-Anweisung `@REQUIRE` aufgerufen werden müssen, bevor sie

verwendet werden können. Dies liegt daran, dass diese Plugins benutzerdefinierten Initialisierungscode benötigen, welcher nur ausgeführt wird, wenn Sie sie explizit mit `@REQUIRE` aufrufen. Zum Beispiel werden Plugins, die einen Display-Adaptermodul installieren, nicht aktiviert, wenn Sie sie nicht mit `@REQUIRE` aufrufen. Plugins, die nur ein zusätzliches Dateiformat zum Laden oder Speichern hinzufügen, werden jedoch automatisch aktiviert, selbst wenn Sie `@REQUIRE` nicht benutzen.

## 5.4 Vorhandene Plugins

Viele Plugins sind im offiziellen Hollywood-Portal erhältlich, welches unter <https://www.hollywood-mal.de/> online ist. Hier ist ein Überblick über Plugins, die derzeit vom offiziellen Hollywood-Portal heruntergeladen werden können:

**AHX:** Ermöglicht es Ihnen, AHX- und HivelyTracker-Module mit Hollywood zu laden und abzuspielen.

**AIFF:** Ermöglicht es Ihnen, AIFF-Samples mit Hollywood zu laden und abzuspielen.

**APNG Anim:** Ermöglicht es Ihnen, APNG-Animationen mit Hollywood zu laden und zu speichern. Dies ist nützlich, weil das APNG-Format Anims mit Alphakanal unterstützt.

**AVCodec:** Fügt Lademodule für viele Video- und Audioformate hinzu, die von FFmpeg zur Verfügung gestellt werden. Dies ist für das Abspielen von modernen Video- und Audioformaten sehr nützlich. Aber seien Sie vorsichtig, weil viele dieser Formate patentiert sind und verlangen, dass Sie Lizenzgebühren bezahlen, wenn Sie sie in Ihren Produkten verwenden.

**DigiBooster:** Lädt und speichert DigiBooster-Module mit Hollywood.

**FLIC Anim:** Lädt FLI- und FLC-Animationen mit Hollywood.

**GL Galore:** OpenGL<sup>®</sup> Wrapper für Hollywood. Dieses Plugin ermöglicht es Ihnen, mit Hollywood in OpenGL zu programmieren. Es unterstützt auch hardwarebeschleunigte 2D-Zeichnung, d.h. es unterstützt Hardware-Doppelpuffer und Hardware-Pinsel. So ist es sehr nützlich für hardwarebeschleunigtes Zeichnen auf Windows, macOS und Linux, weil Hollywood standardmäßig nur hardwarebeschleunigtes Zeichnen auf AmigaOS und kompatibel unterstützt.

**HTTP Streamer:** Mit diesem Plugin können Sie Daten aus HTTP-Quellen als normale Dateien laden. Dies bedeutet, dass sobald dieses Plugin installiert ist, Sie einfach URLs an Befehle wie `LoadBrush()` übergeben und die Dateien werden von dort geladen. Dieses Plugin kann auch für Video- und Audio-Streaming aus HTTP-Quellen verwendet werden.

**Iconic:** Iconic ist das ultimative Plugin zum Laden und Speichern von Piktogrammen für Hollywood. Es kann eine Vielzahl verschiedener Piktogrammformate laden

und speichern. Derzeit werden die folgenden Piktogrammformate von Iconic unterstützt: AmigaOS 1.x, AmigaOS 2.x/3.x, AmigaOS 3.5-Icons (GlowIcons), AmigaOS 4.0, macOS (\*.icns-Format), MagicWB-Icons, MorphOS/PowerIcons (PNG), NewIcons, Windows (\*.ico-Format).

**hURL:** Dieses Plugin ermöglicht den Datentransfer über viele verschiedene Protokolle, so z.B. DICT, FILE, FTP, FTPS, Gopher, HTTP, HTTPS, IMAP, IMAPS, LDAP, LDAPS, POP3, POP3S, RTMP, RTSP, SCP, SFTP, SMB, SMBS, SMTP, SMTPS, Telnet und TFTP. Darüber hinaus unterstützt hURL SSL-Zertifikate, HTTP POST, HTTP PUT, FTP-Uploads, HTTP-Upload mittels Forms, Proxies, HTTP/2, Cookies, Benutzernamen/Password-Authentifizierung, Wiederaufnahme von Datentransfers, Proxytunneling und mehr.

**JPEG2000:**  
Lädt und speichert Bilder im Format JPEG2000.

**Malibu:** Mit diesem Plugin können Sie Scala-Skripte mit Hollywood ausführen. Malibu unterstützt alle Amiga-Versionen von Scala bis Scala MM400 und Scala InfoChannel 500.

**Movie Setter:**  
Öffnet und spielt Animationen im Gold Disk's MovieSetter-Format ab.

**MUI Royale:**  
Erstellen Sie MUI-GUIs mit Hollywood. Dies ist ein leistungsstarkes Plugin, das fast die komplette MUI-API nach Hollywood verpackt und Ihnen erlaubt, MUI-GUIs bequem über XML zu erstellen.

**Ogg Theora:**  
Lädt und spielt Videos im Format Ogg Theora ab.

**Ogg Vorbis:**  
Lädt und spielt das Audioformat Ogg Vorbis mit Hollywood ab.

**PCX:** Lädt Bilder im Format PCX mit Hollywood.

**Plananarama:**  
Verwenden Sie Hollywood auf Amigas ohne Grafikkarte. Wenn Sie Hollywood auf einem Paletten-basierten Bildschirm ausführen, können Sie das mit diesem Plugin machen.

**Polybios:** Polybios ist ein Plugin für Hollywood, mit dem Sie einfach PDF-Dokumente aus Hollywood-Skripten erstellen können. Darüber hinaus kann Polybios auch vorhandene PDF-Dokumente öffnen und ihre Seiten in Hollywood-Pinsel umwandeln. Tatsächlich erstellt Polybios beim Konvertieren von PDF-Seiten in Hollywood-Pinsel Vektorpinsel für Sie, die ohne Qualitätsverluste skaliert, gedreht und transformiert werden können (es sei denn, es sind Bitmap-Grafiken in das PDF-Dokument eingebettet).

**RapaGUI:** Cross-Plattform-GUI-Toolkit für Hollywood. Dies ist ein sehr leistungsfähiges Plugin, das Hollywood zu einem kompletten Cross-Plattform-GUI-Toolkit macht, welches es Ihnen ermöglicht, GUI-basierte Anwendungen sehr bequem zu erstellen, indem Sie das GUI-Layout in einer XML-Datei definieren.

Darüber hinaus können Hollywood-Displays in Ihrer GUI als Widgets eingebettet werden, mit denen Sie auch alle leistungsstarken Grafikbefehle von Hollywood in deiner GUI-Anwendung nutzen können.

**RebelSDL:**

Dies ist ein Hollywood-Wrapper-Plugin für die beliebte SDL-Bibliothek. Der große Vorteil ist, dass es hardwarebeschleunigte 2D-Zeichnung unterstützt, d.h. es unterstützt Hardware-Doppelpuffer und Hardware-Pinsel. So ist es sehr nützlich für Hardware-beschleunigtes Zeichnen auf Windows, macOS und Linux, weil standardmäßig Hollywood nur Hardware-beschleunigtes Zeichnen auf AmigaOS und kompatiblen unterstützt.

**SID:** Spielt SID-Dateien mit Hollywood ab.

**SQLite3:** Erstellen und Ändern von SQL-Datenbanken mit Hollywood.

**SVG Image:**

Lädt Vektorbilder im Format SVG mit Hollywood.

**TIFF:** Lädt und speichert TIFF-Bilder mit Hollywood.

**Vectorgraphics:**

Zeichnen Sie echte vektorbasierte Grafiken mit Hollywood.

**XAD:** Mit diesem Plugin können Sie viele verschiedene Archiv-Formate wie LhA, LZX, RAR, ZIP, TAR usw. entpacken. Sehr nützlich.

**XLSX:** Mit dem XLSX-Plugin können Sie bequem XLSX-Dokumente aus Hollywood-Skripten lesen und schreiben. Es bietet eine Vielzahl von Befehlen zum Festlegen und Abrufen von Zellwerten, Zelltypen, Zellformeln, Dokument-/Arbeitsblatteigenschaften und mehreren anderen Attributen. Es bietet auch eine Iteratorfunktion für eine leistungsstarke Iteration einer großen Anzahl von Zellen.

**XML Parser:**

Plugin für bequemes Parsen von XML-Dokumenten.

**XMP:** Dieses Plugin kann viele verschiedene Tracker-Formate abspielen.

**YAFA:** Lädt und spielt Animationen im Format YAFA ab (erstellt mit Wildfire).

**ZIP:** Mit diesem Plugin können Hollywood-Skripte Zip-Archive lesen und schreiben. Auch Dateien innerhalb von Zip-Archiven können direkt geöffnet werden, ohne vorher in eine externe Datei entpackt zu werden.

## 5.5 Schreiben eigener Plugins

Falls Sie einen bestimmten Befehl oder Funktionalität in Hollywood vermissen, können Sie ein Plugin schreiben, um das Vermisste der Sprache hinzuzufügen. Das Schreiben Ihres eigenen Plugins kann auch hilfreich sein, wenn Ihr Skript bestimmte CPU-intensive Berechnungen durchführen muss, die am besten in nativem Code für eine optimale Leistung implementiert sind.

Das Schreiben von Plugins ist wirklich einfach. Die Plugin-Schnittstelle von Hollywood ist öffentlich, vollständig dokumentiert und alle notwendigen Dateien stehen zum kostenlosen

Download auf dem offiziellen Hollywood-Portal zur Verfügung. Das Hollywood-SDK enthält über 300 Seiten detaillierter Dokumentation und einigen Beispielen, die Ihnen helfen, mit der Plugin-Entwicklung zu beginnen.

Bitte besuchen Sie das offizielle Hollywood-Portal unter <https://www.hollywood-mal.de/>, um das neueste Hollywood-SDK herunterzuladen. Es enthält alle Entwickler-Materialien, die Sie für den Aufbau Ihrer eigenen Hollywood-Plugins benötigen.





## 6 Geschichte und Kompatibilität

### 6.1 Geschichte

Für ein vollständiges Änderungsprotokoll von Hollywood beachten Sie bitte die auf englisch verfasste Datei `history.txt`.

### 6.2 Hinweise zur Kompatibilität/API Änderungen

#### Hollywood 9.0 API-Änderungen

In Hollywood 9.0 wurden einige API-Änderungen vorgenommen. Höchstwahrscheinlich müssen Sie Ihre Skripte nicht anpassen, damit sie mit der Version 9.0 funktionieren. Überprüfen Sie einfach die folgenden Hinweise, um festzustellen, ob für Ihr Skript eine Anpassung erforderlich ist.

- Unter AmigaOS und kompatiblen Systemen werden in `LIBS:Hollywood` installierte Plugins nicht mehr automatisch von allen mit Hollywood kompilierten Programmen geladen. Von Hollywood kompilierte Programme laden jetzt nur die Plugins, die neben dem Programm im gleichen Verzeichnis gespeichert sind. Wenn Sie möchten, dass Ihre Programme auch alle Plugins in `LIBS:Hollywood` lädt, müssen Sie den Tag `GlobalPlugins` in der Präprozessor-Anweisung `@OPTIONS` auf `True` setzen. Sie können auch das Konsolenargument oder Merkmal (ToolType) `-globalplugins` für das Programm verwenden.
- Unter Windows wurde Hollywoods Grafik-Modul komplett neu geschrieben und verwendet jetzt Direct2D. Dies hat viele Vorteile und ermöglicht auch die Nutzung der GPU, wenn das Autoskalierungs-System aktiv ist. In einigen seltenen Fällen kann Ihr Skript jedoch mit Direct2D langsamer sein als vorher, je nachdem, wie Sie Ihre Grafik erstellen. Wenn Ihr Skript mit Hollywood 9.0 langsamer ausgeführt wird, können Sie Hollywood entweder zwingen, das alte Grafik-Modul zu verwenden (welches aus Kompatibilitätsgründen mit der Windows-Versionen noch unterstützt wird, die Direct2D nicht unterstützen (d.h. alle Windows-Versionen vor Vista SP2)) oder die Art und Weise anpassen, wie Ihr Skript seine Grafiken zeichnet. Um Hollywoods altes Grafik-Modul zu aktivieren, setzen Sie einfach den Tag `SoftwareRenderer` der Präprozessor-Anweisung `@DISPLAY` auf `True`. Alternativ können Sie auch die Art und Weise anpassen, wie Ihr Skript seine Grafiken zeichnet, um die Leistung mit Direct2D zu verbessern. Bei Direct2D führt jeder Zeichnungsvorgang immer zu einer vollständigen Aktualisierung des Displays, auch wenn nur ein einziges Pixel gezeichnet wird! Daher muss Ihr Skript seine Grafiken so zeichnen, dass die Aktualisierung des gesamten Displays minimiert wird. Dies kann entweder durch Verwendung eines Doppelpuffers oder durch Verwendung von `BeginRefresh()` und `EndRefresh()` erreicht werden. Siehe [Abschnitt 28.4 \[BeginRefresh\]](#), [Seite 569](#), für Details.
- Unter Windows skaliert Hollywood Ihre Skripte jetzt automatisch so, dass sie auf hochauflösenden Monitoren an die DPI des Monitors angepasst werden. Dies garantiert, dass sie auch auf hochauflösenden Monitoren in der richtigen Größe angezeigt werden. Wenn Sie dies nicht möchten, setzen Sie den neuen Tag `DPIAware` in der Präprozessor-Anweisung `OPTIONS` auf `True`. Dies wird insbesondere für GUI-Anwendungen emp-

fohlen, da diese viel besser aussehen, wenn das Programm sich der Auflösung anpassen kann.

- Lader und Adapter werden jetzt in der Reihenfolge berücksichtigt, wie sie in der Zeichenkette an ein Tabellenargument von `Loader` oder `Adapter` übergeben werden. Wenn Sie beispielsweise die Zeichenkette `digibooster|xmp` an `OpenMusic()` übergeben, wird zuerst versucht, die Musik mit `digibooster.hwp` zu öffnen. Und nur wenn dies fehlschlägt, wird als nächstes mit `xmp.hwp` versucht, die Datei zu öffnen. Auf diese Weise können Sie auch generische Lader wie `Native`, `Inbuilt` und `Plugin` priorisieren. Wenn Sie z.B. möchten, dass native und interne Lader vor Plugin verwendet werden sollen, können Sie die Zeichenkette `native|inbuilt|plugin` übergeben, um dies zu erreichen. Beachten Sie, dass es sich zwar um eine API-Änderung handelt, es aber wahrscheinlich keine Skripte gibt, die wirklich vom alten Verhalten abhängen, da die Reihenfolge meist zufällig war.
- Wenn ein nicht vorhandener Pfad an `ChangeDirectory()` übergeben wurde, schlug der Befehl stillschweigend fehl und zeigte keinen Fehler an. Dies wurde jetzt geändert. `ChangeDirectory()` löst jetzt einen Fehler aus, wenn ein nicht vorhandener Pfad an ihn übergeben wird.
- `SaveAnim()`, `BeginAnimStream()` und `WriteAnimFrame()` unterstützen die Tags `Transparency` und `UseAlpha` nicht mehr, da sie einfach keinen Sinn ergeben. Die Transparenz- und Alpha-Einstellungen werden jetzt ausschließlich durch das Format der Quelldaten bestimmt, die an diese Befehle übergeben werden.
- Wenn der Tag `KeepProportions` für ein Display aktiv ist und `DisplayBGPic()` oder `ChangeDisplaySize()` aufgerufen wird, werden die Skalierungsdimensionen jetzt neu berechnet, um das Seitenverhältnis der neuen BGPic- oder Display-Größe anzupassen. Diese Änderung wirkt sich auch auf den Befehl `SetDisplayAttributes()` aus, wenn sie mit den Parametern `Width` und `Height` aufgerufen wird.
- Wenn der Tag `Monitor` nicht im optionalen Tabellenargument gesetzt ist, erkennt `ChangeDisplayMode()` jetzt immer automatisch den Monitor, auf dem sich das Display befindet. Wenn es sich also auf einem zweiten Monitor befindet und Sie `ChangeDisplayMode(#DISPMODE_FULLSCREEN)` aufrufen, wird dieser Monitor in den Vollbildmodus versetzt. Bisher wurde immer der erste Monitor verwendet (außer wenn der Tag `Monitor` für das Display festgelegt wurde).
- Videos, die vom nativen Video-Modul des Betriebssystems verarbeitet werden (z.B. `DirectShow` und `Media Foundation` unter Windows, `AVFoundation` und `QuickTime` unter macOS), werden jetzt automatisch in der Größe geändert und neu positioniert, wenn die Größe des Displays vom Benutzer geändert wird, auch wenn kein Skalierungssystem aktiv ist. Dies geschieht rein aus ästhetischen Gründen, da es sonst sehr hässlich aussieht. Sie können dieses Verhalten unterbinden, indem Sie den Tag `NoLiveResize` in `@DISPLAY` setzen. Beachten Sie, dass Videos, die mit Hollywoods integriertem Video-Modul verwaltet werden, nicht automatisch in der Größe geändert und neu positioniert werden. Dies geschieht nur bei Videos, die vom Betriebssystem verwaltet werden.

## Hollywood 8.0 API-Änderungen

Für Hollywood 8.0 wurden keine Änderungen an der API vorgenommen.

## Hollywood 7.1 API-Änderungen

Es gab einige kleine API-Änderungen in Hollywood 7.1. Meist müssen Sie wahrscheinlich Ihre Skripte nicht anpassen, um mit 7.1 arbeiten zu können. Überprüfen Sie einfach die folgenden Hinweise, um zu sehen, ob Ihr Skript eine Anpassung erfordert.

- Es gab einige kleinere Änderungen am plattformunabhängigen Sprachenkatalogformat von Hollywood. Zeilen, die mit einem Semikolon/Strichpunkt (;) beginnen, werden jetzt als Kommentare betrachtet und ignoriert. Wenn Sie eine Katalogzeichenfolge definieren müssen, die mit einem Semikolon beginnt, müssen Sie dem Semikolon einen umgekehrten Schrägstrich (\) voranstellen. Außerdem werden jetzt leere Zeilen ignoriert und Zeichenketten, die mit einem einzigen umgekehrten Schrägstrich enden, gelten als mehrzeilige Zeichenkette. Es könnte notwendig sein, Ihre Sprachkataloge so zu ändern, dass sie mit dem neuen Format kompatibel sind.

## Hollywood 7.0 API-Änderungen

### Neue Plugin- und Keyfile-Speicherorte unter Windows, macOS, Linux und Android

Zuerst sollten Benutzer von Windows-, macOS- und Linux beachten, dass in Hollywood 7.0 Plugins nun in einem Unterverzeichnis **Plugins** gespeichert sein müssen, das im selben Verzeichnis wie die Hollywood-Datei unter Windows und Linux sein muss. Unter macOS muss das **Plugins**-Verzeichnis innerhalb des Programmpakets gespeichert werden, d.h. im Verzeichnis `HollywoodInterpreter.app/Contents/Resources/Plugins`. Beachten Sie, dass `HollywoodInterpreter.app` im Programmpaket `Hollywood.app` gespeichert ist, nämlich in `Hollywood.app/Contents/Resources`. Auf Android müssen Plugins jetzt im Verzeichnis `Hollywood/Plugins` auf Ihrer SD-Karte gespeichert werden (statt in `Hollywood/_Plugins` wie in früheren Versionen). Bei AmigaOS und kompatiblen müssen Plugins wie gewohnt in `LIBS:Hollywood` kopiert werden.

Beachten Sie, dass ausführbare Dateien, die von Hollywood kompiliert werden, immer noch Plugins aus demselben Verzeichnis wie die ausführbare Datei laden (außer auf macOS, wo sie sich im Verzeichnis des Dateispeichers **Plugins** befinden müssen). Hollywood selbst wird jedoch jetzt unter Windows, macOS und Linux die Plugins in einem Unterverzeichnis **Plugins** wie oben beschrieben benötigen.

macOS Benutzer beachten bitte auch, dass die Datei `Hollywood.key` nun auch ins Verzeichnis `HollywoodInterpreter.app/Contents/Resources` kopiert werden muss. Es muss sich nicht mehr im Verzeichnis `HollywoodInterpreter.app/Contents/MacOS` befinden.

### Wichtige Unicode-Hinweise und andere API-Änderungen

Da Hollywood 7.0 nun voll Unicode unterstützt, kann es einige Kompatibilitätsprobleme mit Ihren alten Skripten geben. Wenn Sie Ihre Skripte nicht anpassen möchten, können Sie sie einfach im Nicht-Unicode-Modus ausführen, indem Sie Unicode wie folgt deaktivieren:

```
@OPTIONS {Encoding = #ENCODING_ISO8859_1}
```

Wenn Sie dies als allererste Zeile in Ihrem Skript hinzufügen, wird Hollywood Ihr Skript im Legacy-Modus laufen lassen und es sollte keine Kompatibilitätsprobleme geben. Allerdings läuft Ihr Skript im Modus ISO 8899-1, was bedeutet, dass es nicht korrekt auf nicht-westeuropäischen Systemen läuft.

Deshalb wird empfohlen, dass Sie Ihr Skript nicht im Legacy-Modus ausführen, sondern immer den Unicode-Modus benutzen. Die meisten Skripte werden wahrscheinlich nur aus der Box laufen, ohne irgendwelche Probleme zu bereiten und ohne irgendeine Notwendigkeit etwas anzupassen. Wenn Ihr Skript Kompatibilitätsprobleme mit Hollywood 7.0 zeigt, lesen Sie bitte die folgende Liste der API-Änderungen von Hollywood 7.0, um zu erfahren, wie Sie Ihre Skripte anpassen können.

- Zuerst einmal stellen Sie sicher, jetzt alle Ihre Skripte mit UTF-8-Codierung zu speichern. Wenn Sie Ihre alten Skripte mit Hollywood 7.0 ausführen, wird Hollywood zunächst prüfen, ob sie nur gültige UTF-8-Zeichen enthalten. Ist dies nicht der Fall wird Hollywood davon ausgehen, dass sich die Skripte in der ISO 8859-1-Codierung (oder der Standard-Codierung des Systems auf Amiga) befinden und sie automatisch in UTF-8 umwandeln. Da diese automatische Konvertierung zu Problemen mit Skripten führen kann, die eine andere Codierung als ISO 8859-1 verwenden, empfiehlt es sich, alle Skripte jetzt in UTF-8 zu speichern.
- Da Hollywood 7.0 standardmäßig im Unicode-Modus läuft, wird nun auch die Standard-Zeichenketten-Codierung auf `#ENCODING_UTF8` gesetzt. Dies bedeutet, dass Sie Probleme haben, wenn Ihr Skript versucht, die Zeichenkettenbibliotheks-Befehle zu verwenden, um auf die rohen Binärdaten von Zeichenfolgen zuzugreifen. Wenn die Zeichenfolgencodierung auf `#ENCODING_UTF8` gesetzt ist, können sich die Befehle der Zeichenkettenbibliothek nur mit Zeichenfolgen befassen, die gültigen UTF-8-Text enthalten. In Hollywood können Zeichenfolgen aber auch binäre Daten enthalten. Zum Beispiel können Sie eine Datei mit dem Befehl `DownloadFile()` in eine Zeichenfolge herunterladen und dann ihre Länge mit `StrLen()` herausfinden. Dies wird nicht funktionieren, wenn Hollywood im Unicode-Modus ist (d.h. wenn die Standard-Zeichenketten-Codierung auf `#ENCODING_UTF8` gesetzt ist), da `StrLen()` dann gültige UTF-8-Daten erwartet. Um dieses Problem zu umgehen, müssen Sie `#ENCODING_RAW` an `StrLen()` übergeben, damit die Zeichenkette, die Sie übergeben haben, rohe Binärdaten anstelle des gültigen UTF-8-Textes enthält. Ebenso akzeptieren die meisten anderen Befehle der Zeichenkettenbibliothek einen zusätzlichen Parameter `encoding`, mit dem Sie die Zeichencodierung der Zeichenkette, einstellen können. Wenn Ihr Skript die Zeichenkettenbibliotheks-Befehle nicht verwendet, um mit rohen Binärdaten zu arbeiten, müssen Sie sich keine Sorgen machen und Ihr Skript sollte im Unicode-Modus einwandfrei funktionieren.
- Die Standardtextcodierung wird auch automatisch von Hollywood 7.0 auf `#ENCODING_UTF8` gesetzt. Dies bedeutet, dass Befehle wie `TextOut()` und `Print()` jetzt UTF-8 codierten Text erwarten. Dies ist kein Problem, wenn Sie Ihr Skript nur in UTF-8 konvertieren, aber es könnte zu Problemen führen, wenn der auszugebende aus einer Datei (oder einer anderen externen Quelle) gelesen wird, die keine UTF-8-Codierung verwendet.
- `ReadChr()` und `WriteChr()` können jetzt bis zu 4 Bytes lesen und schreiben statt nur ein einziges Byte, wenn Hollywood im Unicode-Modus ist. Das ist, weil sie jetzt wirklich mit Zeichen umgehen und in UTF-8 kann ein Zeichen bis zu 4 Bytes für die Speicherung benötigen. Wenn Sie einzelne Bytes lesen und schreiben möchten, müssen Sie jetzt die neuen Befehle `ReadByte()` und `WriteByte()` verwenden.
- `ReadString()` und `WriteString()` können nicht mehr für binäre I/O verwendet werden, da sie Zeichenketten lesen und schreiben, d.h. wie ihre Namen implizieren eine Anzahl

von Zeichen (nicht Bytes!). Wenn Sie rohe Binärdaten lesen und schreiben müssen, verwenden Sie jetzt die neuen Befehle `ReadString()` und `WriteString()`.

- `ReadChr()`, `WriteChr()`, `ReadString()` und `WriteString()` lesen und schreiben jetzt standardmäßig UTF-8-Zeichen. Wenn Sie sie verwenden, um Daten aus Nicht-UTF-8-Textdateien zu lesen, kann es Probleme mit Nicht-ASCII-Zeichen geben. In diesem Fall müssen Sie diesen Befehlen mitteilen, dass sie Zeichen im Format ISO 8859-1 lesen, indem Sie `#ENCODING_ISO8859_1` im optionalen Argument `encoding` übergeben.
- Sie sollten die beiden Ereignis-Handler `OnKeyDown` und `OnKeyUp` für nicht-englische Zeichen nicht verwenden. Nicht-englische Zeichen sollten nur mit dem neuen Ereignis-Handler `VanillaKey` behandelt werden, der stattdessen eine vollständige Unicode-Unterstützung hat. `OnKeyDown` und `OnKeyUp` werden weiterhin wie bisher arbeiten, aber mit nicht-englischen Zeichen sind sie in der Regel unsicher. Sie könnten mit Ihrem System arbeiten, aber nicht auf Systemen mit einer anderen Länder-/Spracheinstellung. Verwenden Sie nur `VanillaKey`, um nicht-englische Zeichen zu behandeln.
- Die Befehle `IsKeyDown()` und `WaitKeyDown()` unterstützen nicht mehr englische Tasten. Wenn Sie den Zustand einer nicht-englischen Taste erhalten müssen, verwenden Sie stattdessen den Ereignis-Handler `VanillaKey`.
- Das plattformneutrale Format, das von `OpenCatalog()` unterstützt wird, muss nun in UTF-8 Zeichencodierung mit oder ohne BOM sein. ISO 8859-1 Dateien werden nicht mehr unterstützt.
- Multi-Byte-Zeichenkonstanten wie 'ABCD' werden nicht mehr unterstützt, da sie mit dem von Hollywood 7.0 unterstützten UTF-8-Zeichenkonstanten in Konflikt stehen. Wenn Ihr Skript Multi-Byte-Zeichenkonstanten verwendet, müssen Sie Ihr Skript umschreiben, um den direkten numerischen Wert der Zeichenkonstante zu verwenden.
- Hollywood 7.0 benutzt neu die Anweisung `FallThrough`, die es ermöglicht, in den nächsten `Case`-Bereich einer `Switch-Case`-Anweisung durchzufallen. Dies bedeutet allerdings, dass es nicht mehr erlaubt ist, Variablen oder Funktionen mit der Bezeichnung `FallThrough` zu verwenden. Dies wird einen Fehler auslösen, weil `FallThrough` nun eine reservierte Bezeichnung ist.

## Hollywood 6.0 API Änderungen

Es wurden einige kleine API-Änderungen in Hollywood 6.0 durchgeführt. Höchstwahrscheinlich werden Sie Ihre Skripte nicht anpassen müssen, um mit 6.0 zu arbeiten. Überprüfen Sie einfach die folgenden Hinweise, ob Ihr Skript eine Anpassung erfordert.

- Seit Hollywood 6.0 mit integrierter Unterstützung für vektorbasierte Zeichnungen daher kommt, verwendet die Vektorenbibliothek nicht mehr automatisch die ersten Bilder, um das richtige Plugin zu finden. Stattdessen wird es nun Hollywoods eingebauten Vektorenmodul als Standard verwenden. Wenn Sie dies nicht wollen, müssen Sie den Befehl `SetVectorEngine()` verwenden, um die Vektorenbibliothek anzuweisen, das Plugin beim Bilder zeichnen zu verwenden.
- Der `ChangeDisplayMode()` Befehl schaltet nicht mehr alle, sondern nur das aktive Display in den Vollbildmodus. Diese Änderung war notwendig, weil Hollywood 6.0 Multi-

Monitor-Unterstützung zur Verfügung stellt, die es ermöglicht, mehrere Displays im Vollbild-Modus auf separaten Monitoren darzustellen.

- Der Hollywood Display-Handler wurde neu geschrieben und unterstützt den Anzeigemodus **OwnScreen** nicht mehr. **OwnScreen** war ein spezieller Modus, der auf AmigaOS und kompatiblen Geräte verwendet wurde, um Hollywood im Vollbild-Modus zu öffnen, aber den Bereich um das Display nicht abgedeckt wurde. Das heißt, Hollywood öffnete kein Schutzfenster, welches die Amiga Bildschirm Dekorationen bedeckte. Wenn Sie das Aussehen von **OwnScreen** erreichen wollen, müssen Sie die Konsolenargumente **-nobackfill** und **-nostyleoverride** zusammen mit **-fullscreen** verwenden. Dann sollte es genau gleich aussehen, wie der alte **OwnScreen** Anzeigemodus.
- Vor Hollywood 6.0 überschrieben die angegebenen Attribute in der **@DISPLAY** Präprozessor-Anweisung automatisch die Displayattribute, die mit der Kommandozeile angegeben wurden. D.h. wenn das **Borderless**-Attribut in **@DISPLAY** auf **False** gesetzt und das Skript mit dem Argument **-borderless** gestartet wurde, dann würde das Skript trotzdem noch mit einem umrandeten Fenster angezeigt werden, da die Angaben in **@DISPLAY** höhere Priorität als die Angaben in der Befehlszeilen hatten. Beginnend mit Hollywood 6.0 wurde dieses Verhalten umgedreht: Displayattribute, welche mit der Kommandozeile gesetzt werden, setzen nun die Displayattribute der Präprozessor-Anweisung außer Kraft. Wenn Sie dieses Verhalten nicht wünschen, kompilieren Sie Ihr Skript mit dem **-locksettings** Argument. Dann sind die Kommandozeilenargumenten nicht mehr in der Lage, Ihre Präprozessor-Displayeinstellungen außer Kraft zu setzen.
- Vor Hollywood 6.0 wurden die Befehlszeilenargumente in den Präprozessor-Anweisungen des Skripts, die die Art der Anzeige definierten, auf alle Displays angewendet. Zum Beispiel wenn Sie ein Skript mit vier Displays mit dem Argument **-borderless** starteten, wurden alle vier Displays randlos geöffnet. In Hollywood 6.0 werden Befehlszeilenargumente, die den Anzeigestil ändern, standardmäßig nur auf Display Nummer 1 angewendet. Wenn Sie alle Displays beeinflussen wollen, müssen Sie das neue Argument **-alldisplays** verwenden.
- Einige Befehlszeilenargumente wurden aus rein kosmetischen Gründen umbenannt: **-audiodev** heißt jetzt **-audiodevice** und **-depth** ist nun **-scrdepth**. Ihre Funktionalität hat sich nicht geändert.
- Die Unterstützung für **mpega.library** auf AmigaOS und kompatible Geräte wurde fallen gelassen. **mpega.library** verursachte einige Schwierigkeiten. Weil es in der Regel jedes Datei-Format als MPEG-Strom erkannte, führte dies zu mehreren unerwünschten Wirkungen. Wenn Sie MP3s abspielen möchten, können Sie stattdessen ein Plugin wie **avcodec.hwp** verwenden.

## Hollywood 5.0 API Änderungen

Es wurden in Hollywood 5.0 einige kleine API-Änderungen vorgenommen. Höchstwahrscheinlich werden Sie Ihre Skripte nicht anpassen müssen, um mit 5.0 zu arbeiten. Überprüfen Sie einfach die folgenden Hinweise, um zu sehen, ob Ihr Skript eine Anpassung erfordert.

- Die Plugin-Schnittstelle wurde komplett neu geschrieben und ist nicht mehr mit alten Plugins kompatibel. Auf Amiga-Systeme müssen Plugins immer in **LIBS:Hollywood**



installiert werden. Das alte Plugin Verzeichnis `Hollywood:Plugins` wird nicht mehr von Hollywood 5.0 unterstützt. Alternativ können Plugins in das Verzeichnis des Programms installiert werden (besonders nützlich, wenn Sie Plugins mit kompilierten ausführbaren Dateien von Hollywood verteilen wollen).

- Schatten und Randeffekte auf Ebenen werden im Vergleich zu früheren Versionen anders (besser!) aussehen, weil Hollywood jetzt echte Alpha-Kanal-Mischung für diese verwendet. Der Nachteil ist, dass die neuen Schatten und Randeffekte langsamer als in früheren Versionen sind.
- Schatten und Ränder sind nun für jede Ebene globale Einstellungen. Das bedeutet, dass Sie nicht mehr Ebenen verwenden können, die mehrere Schatten- oder Rahmenarten haben (zum Beispiel eine Textebene, wo nur ein Teil des Textes einen Schatten oder einen Rand hat). Das wird nicht mehr unterstützt. Entweder hat die Ebene einen Schatten/Rand oder nicht. Aber es ist nicht mehr möglich, einen Schatten/Rand nur für einen Teil der Ebene zu haben.
- Ebenen mit einem Rand werden jetzt anders dargestellt, wenn sie mit einem Übergangseffekt angezeigt oder ausgeblendet werden. Da die Ränder in Hollywood 5.0 nicht mehr Teil der Hauptebene sind, kann der Übergangseffekt für den Rahmen nicht mehr mit der Hauptebene kombiniert werden. So wird jetzt Hollywood einfach den Rand verblassen, während der Übergang des Hauptebeneneffekts angezeigt wird. Wenn Sie dieses Verhalten nicht wünschen, können Sie den neuen Tag `NoBorderFade` verwenden.
- Aus Gründen der Kohärenz unterstützt Hollywood dicke Ebenen nicht mehr, wenn bei Füllungsstil `#FILLNONE` gesetzt ist. Der Grund ist, dass das Dicke-Ebenenkonzept sich oft nicht mit dem neuen Ebenenränderkonzept verträgt. Anstelle einer Dickeneinstellung darf man den Rändern der Ebene eine 'Dickegröße' zuweisen, um sie dicker zu machen. Die Hauptebene wird jedoch immer eine Dicke von 1 haben. Wenn Sie diese Dicke erhöhen wollen, schalten Sie die Ränder ein und setzen die gewünschte Dicke im Tag `BorderSize`.
- Hollywood 5.0 bietet Unterstützung für Vektorgrafiken. `CreateGradientBGPic()` und `CreateTexturedBGPic()` werden nun ein Vektor-BGPic für Sie erstellen. Das bedeutet, dass Sie nicht mehr die Grafiken dieser BGPics mit dem `SelectBGPic()` Befehl modifizieren können. Ein weiterer Unterschied ist, dass wenn eine Textur-BGPic skaliert (zum Beispiel, wenn der Benutzer die Größe eines Fensters ändert), Hollywood dann nicht das texturierte BGPic skaliert, sondern es wird auf die neue Auflösung gezeichnet. Vor 5.0 wurden die texturierten BGPic skaliert. Beginnend mit 5.0, werden die BGPic komplett neu gezeichnet.
- Vor 5.0 haben die Befehle `CopyFile()` und `DeleteFile()` Platzhalter im Dateinamen als Argument akzeptiert. Dies wird nicht mehr unterstützt. Wenn Sie selektiv Dateien kopieren/löschen möchten, müssen Sie jetzt ein optionales Argument verwenden. Für weitere Informationen lesen Sie die Dokumentation dieser beiden Befehle.
- Vor 5.0, verwendete Hollywood reguläre Dateifilter von AmigaOS in den Befehlen `MatchPattern()`, `CopyFile()` und `DeleteFile()`. Beginnend mit der Version 5.0 verwendet Hollywood plattformunabhängige Dateifilter, die sich von denen im AmigaOS in einigen Fällen unterscheiden. Siehe [Abschnitt 20.42 \[MatchPattern\]](#), [Seite 280](#), für Details.
- Lange Zeichenketten [...] verhalten sich in Hollywood 5.0 anders, wenn Ihr Skript mit

Wagenrücklauf und Zeilenvorschubcodierung gespeichert wurde (CR + LF-Codierung ist der Texteditor Standard auf Windows, Amiga und Unix-Systeme verwenden nur LF-Zeichen für Zeilenumbrüche). Zuvor wurden Wagenrücklauf-Returnzeichen ('\r') immer in die lange Zeichenkette integriert, dies ist nicht mehr der Fall. Alle Zeilenumbrüche werden nun in einzelne Zeilenvorschübe umgewandelt ('\n'). Wenn ein Wagenrücklaufzeichen vorhanden ist, wird es verworfen. Diese Änderung wurde vorgenommen, um zu verhindern, dass Skripte sich anders verhalten, wenn sie auf Windows und auf AmigaOS/Unix/Mac gespeichert wurden.

## Hollywood 4.5 API Änderungen

Es wurden in Hollywood 4.5 einige kleine API-Änderungen vorgenommen. Höchstwahrscheinlich werden Sie Ihre Skripte nicht anpassen müssen, um mit 4.5 zu arbeiten. Überprüfen Sie einfach die folgenden Hinweise, um zu sehen, ob Ihr Skript eine Anpassung erfordert.

- **RotateLayer()** verhält sich anders in 4.5, als es in 4.0 tat. Dieser Umbruch war notwendig, weil Hollywood 4.5 Ankerpunkte für die Ebenen einführt. In Hollywood 4.0, **RotateLayer()** dreht die Ebene um seinen Mittelpunkt, so dass ein Ankerpunkt 0.5/0.5 angenommen wurde. In 4.5 haben jedoch alle Ebenen einen Standardankerpunkt von 0.0/0.0. Wenn Sie also das 4.0-Verhalten replizieren möchten, müssen Sie den Ankerpunkt der Ebene auf 0.5/0.5 mit Aufruf von **SetLayerAnchor()** festlegen.
- Ab jetzt werden Datendateien für Programme, die für macOS kompiliert wurden, nur noch im Ordner "Ressourcen" vom Programmpaket gesucht. Diese Änderung wurde vorgenommen, um die macOS UI-Richtlinien zu erfüllen. Alle Datendateien müssen in seinem App-Packet platziert werden.
- Wenn Sie mit **CreateSprite()** einen Sprite-Link (das heißt Typ **#SPRITE**) erstellen, konnten Sie vorher auch Links von Sprite-Links erstellen. Dies ist nicht mehr möglich. Wenn Sie Sprite-Links erstellen, müssen Sie immer ein Sprite angeben, welches real existiert.
- Wenn sie **#VANILLACOPY** mit **SetAlphaIntensity()** verwendeten, zeichnete Hollywood vorher manchmal etwas nicht. Z.B. wenn man versuchte, einen Pinsel mit Maske in einen Alphakanal mit **SetAlphaIntensity()** und **#VANILLACOPY** zu zeichnen. Dieses Verhalten hat sich nun geändert: Hollywood zeichnet jetzt die sichtbaren Maskenpixel als 255 Alphaintensität und die unsichtbaren Maskenpixel als 0.
- **SelectBGPic()** hatte ein geheimes Feature, das nie irgendwo dokumentiert wurde (und somit nicht offiziell war): Wenn Sie **SelectBGPic()** auf dem aktuellen BGPic aktivierten Ebenen verwendeten, wurden alle Ebenen vor **EndSelect()** als versteckte Ebenen eingesetzt. Dieses Verhalten ist jetzt weg. **SelectBGPic()** wird nun normale Ebenen einfügen und zeichnen, wenn **EndSelect()** aufgerufen wird. Wenn Sie das bisherige Verhalten haben wollen, erstellen sie ihre Ebenen mit dem setzen von **Hidden** auf **True**.

## Hollywood 4.0 API Änderungen

Es wurden einige kleine API-Änderungen in Hollywood 4.0 durchgeführt. Wahrscheinlich müssen Ihre Skripte nicht anpassen, um mit 4.0 zu arbeiten. Überprüfen Sie einfach die folgenden Hinweise, um zu sehen, ob Ihr Skript eine Anpassung erfordert.

- **SetPointer()** Syntax hat sich völlig verändert. Es ist nicht mehr ein Dateinamen akzeptiert, sondern erfordert, dass Sie zuerst **CreatePointer()** aufrufen.



- Alle Übergangs-Effekte sowie die Befehle `PlayAnim()`, die `MoveXXX()` & `DisplayBGPicPart()` verwenden jetzt eine neue Syntax. Allerdings wird die alte Syntax noch aus Kompatibilitätsgründen unterstützt.

## Hollywood 3.1 API Änderungen

Es wurden einige kleine API-Änderungen in Hollywood 3.1 durchgeführt. Wahrscheinlich müssen Ihre Skripte nicht anpassen, um mit 3.1 zu arbeiten. Überprüfen Sie einfach die folgenden Hinweise, um zu sehen, ob Ihr Skript eine Anpassung erfordert.

- Doppelpunkt wird nicht mehr als Befehlstrenner unterstützt. In Hollywood 1.x wurde der Doppelpunkt verwendet, um mehrere Befehle auf der gleichen Zeile zu trennen, z.B.

```
; Hollywood 1.x Code - wird nicht länger unterstützt
x=100:y=200:width=50:height=50:Box(x, y, width, height, #RED)
```

Der 1.x Emulator innerhalb von Hollywood emuliert dieses Verhalten bis zu Hollywood 3.0. In Hollywood 3.1 wird es nun nicht mehr unterstützt, da der Doppelpunkt für die objektorientierte Programmierung erforderlich ist. So müssen Sie Ihre Skripte aktualisieren, wenn Sie noch Doppelpunkte verwenden, um mehrere Befehle in einer einzigen Zeile zu trennen. Seit Hollywood 2.0 können Sie so viele Befehle in einer einzigen Zeile setzen, wie Sie möchten, so dass der obigen Code nun wie folgt geschrieben wird:

```
x=100 y=200 width=50 height=50 Box(x, y, width, height, #RED)
```

Das sieht nicht sehr schön aus, so dass Sie wahrscheinlich vom Aufrufen mehrere Befehle auf der gleichen Zeile gänzlich verzichten werden.

- Der `#TYPEWRITER` Übergangseffekt ist jetzt weg. Dies war ein besonderer Effekt, der nur auf Textobjekte verwendet werden konnte. Allerdings machte es die Schriftschnittstelle unnötig komplex, so dass er wegfallen musste. Sie können nur mit einer Reihe von `Print()` aufrufen das Verhalten von `#TYPEWRITER` emulieren.

## Hollywood 3.0 API Änderungen

Es wurden einige kleine API-Änderungen in Hollywood 3.0 durchgeführt. Wahrscheinlich müssen Ihre Skripte nicht anpassen, um mit 3.0 zu arbeiten. Überprüfen Sie einfach die folgenden Hinweise, um zu sehen, ob Ihr Skript eine Anpassung erfordert.

- Wenn Hollywood 3 ohne Argumente gestartet wird, wird es im Fenstermodus geöffnet. Alle vorherigen Versionen öffneten Vollbildmodus. Aber ich denke, es ist viel klüger, Hollywood öffnet sich im Fenstermodus, weil der Vollbildmodus möglicherweise nicht auf jedem System läuft.
- Kommandozeilenargumente werden nun anders gehandhabt. Sie müssen sie mit einem Bindestrich als Präfix (-) verwenden. In früheren Versionen haben Sie Hollywood wie folgt aufgerufen:

```
Hollywood script.hws WINDOW BORDERLESS
```

Dies wird nicht mehr funktionieren! Sie müssen jetzt Bindestriche verwenden. Der richtige Aufruf von Hollywood ist nun:

```
Hollywood script.hws -window -borderless
```

Diese Änderung war notwendig, weil der neue `GetCommandLine()` Befehl mit seinen eigenen Argumenten arbeitet.

- Das zweite Argument von `FileRequest()` hat sich geändert. Bisher war es ein Muster im AmigaDOS Format. Nun ist es ein Zeichenkettenfilter der angibt, welche Dateien angezeigt werden sollen. Diese Änderung war wegen der neuen plattformübergreifende Natur von Hollywood notwendig. Betriebssysteme wie Windows und macOS haben einfach nicht so aufwendige Filtermuster wie es das AmigaOS bietet.
- In früheren Version fiel beim Befehl `OpenFile()` das optionale dritte Argument auf `#MODE_READWRITE` zurück, wenn es nicht angegeben wurde. Dies wurde geändert. Nun ist der Standardmodus `#MODE_READ`. Ich denke, es macht viel mehr Sinn, den schreibgeschützten Modus zum Öffnen von Dateien als Voreinstellung zu verwenden.

## Hollywood 2.5 API Änderungen

Es wurden einige kleine API-Änderungen in Hollywood 2.5 durchgeführt. Wahrscheinlich müssen Ihre Skripte nicht anpassen, um mit 2.5 zu arbeiten. Überprüfen Sie einfach die folgenden Hinweise, um zu sehen, ob Ihr Skript eine Anpassung erfordert.

- Die Unterstützung für `ttengine.library` wurde entfernt. Natürlich wird dieser Schriftartentyp in Hollywood immer noch unterstützt. Das einzige, was Sie nicht mehr tun können, ist mit `SetFont() *.ttf` Dateien direkt zu verwenden.

`SetFont("dh1:arial.ttf")` ; das ist Hollywood 2.0 Code!

Dies gilt nicht mehr. In Hollywood 2.5 können Sie nur TrueType-Schriftarten verwenden, die in Ihrem System mit dem FTManager oder einem ähnlichem Werkzeug installiert wurden. Sie öffnen diese, als wären sie normale Schriften:

`SetFont("Arial Narrow.font")` ; OKAY in 2.5!

Hollywood 2.5 lädt alle TrueTypes, die kompatibel mit der `bullet.library ft2` (OS4) oder `freetype2` (MorphOS, AROS, AmigaOS3) Schnittstellen sind.

- Der Befehl `CheckEvent()` wurde entfernt. Er passte nicht mehr in das Konzept. Bitte verwenden Sie jetzt stattdessen immer den Befehl `WaitEvent()`.
- Der Befehl `Plot()` funktioniert jetzt nur mit ausgeschalteten Ebenen. Die Ebenen des Typs `#PLOT` sind nicht mehr möglich. Es macht einfach keinen Sinn, 1x1 große Ebenen zu haben. Wenn Sie das wirklich benötigen, können Sie den Befehl `Box()` verwenden, um ein Pixel zu zeichnen.
- Aufgrund des neuen Textdarstellungmoduls ist es nun zwingend, zwei eckigen Klammern in den Zeichenketten von den Befehlen `Print()`, `TextOut()` und `CreateTextObject()` zu verwenden, wenn Sie eine einzelne eckige Klammer ausgeben möchten. Zum Beispiel kann der folgende Code

`Print("[Hello World]")` ; das ist Hollywood 2.0 Code!

einen Syntaxfehler in Hollywood 2.5 erzeugen, da das neue Textmodul einen Formatierungsbefehl nach der eckigen Klammer erwartet. Ab jetzt müssen Sie den Code so wie folgt schreiben:

`Print("[[Hello World]]")` ; OKAY in Hollywood 2.5!

Dann wird es funktionieren, wie Sie es erwarten.

- Wenn der Füllstil `#FILLTEXTURE` oder `#FILLGRADIENT` eingestellt ist und Sie einen ARGB-Wert verwenden, werden jetzt diese Stile auch den Alphawert beachten. Dies war in Hollywood 2.0 nicht der Fall.

- Wenn Ebenen aktiviert sind und Sie einen Befehl aus der Zeichenbibliothek aufrufen (z.B. `Ellipse()`) und eine ARGB-Farbe angeben (d.h. Sie möchten mit Transparenz zeichnen), würde Hollywood 2.0 für Sie eine transparente Ebene erstellen, als ob Sie den Befehl `SetLayerTransparency()` mit dem A-Byte des ARGB-Wert als Transparenzeinstellung angegeben hätten. Dies wird nicht mehr länger auf diese Weise geschehen. Wenn Sie mit einer ARGB-Farbe zeichnen, wird Hollywood 2.5 der Ebene keine Transparenzeinstellung geben, obwohl die Ebene jetzt eine Transparenz hat. Aber mit Hollywood 2.5 wird die Transparenz bereits in den Grafikdaten wiedergegeben (d.h. im Alpha-Kanal) und ist nicht dynamisch wie im Fall von `SetLayerTransparency()` gehalten.
- Bis zu Hollywood 2.0 hat `RotateBrush()` immer ein Pinsel der maximalen Größe zurückgegeben, die eine Rotation mit dem Originalpinsel einnehmen könnte, das heißt  $\text{maxs} = \sqrt{(\text{Breite} * \text{Breite} + \text{Höhe} * \text{Höhe})}$ . Der neue Pinsel von Hollywood wäre dann die Breite und Höhe 'maxs' zugewiesen. Dies ist nun nicht mehr der Fall. Der Pinsel ist genau so groß wie alle Grafiken sind.
- In Hollywood 2.0 verwenden `WriteMem()` und `ReadMem()` immer ungepuffertes IO, während alle anderen DOS-Befehle IO gepuffertes einsetzen. Jetzt sind alle Befehle einheitlich und sie verwenden nun alle standardmäßig gepuffertes IO. Darüber hinaus wurde in Hollywood 2.0 bei `WriteMem()` immer automatisch vor dem Start des Schreibvorgangs der Puffer geleert. Dies wird in 2.5 nicht mehr durchgeführt. Also wenn Sie `WriteMem()` oder `ReadMem()` in Ihren Skripten verwenden und Sie müssen ungepufferte IO wie in 2.0 haben, rufen Sie zunächst `SetIOMode()` auf, um den IO-Modus auf ungepuffert zu ändern. Dann wird es funktionieren, wie Sie es gewohnt sind. Aber denken Sie daran, dass es nicht den Puffer wie in Hollywood 2.0 leert. Und denken Sie daran, sobald Sie `SetIOMode()` aufrufen, alle anderen DOS-Befehle auch den hier eingestellte IO-Modus verwenden werden! Wenn Sie nur für `WriteMem()` oder `ReadMem()` ungepuffertes IO wollen, müssen Sie `SetIOMode()` wieder nach Ihrem Aufruf verwenden. Sie müssen auch `FlushFile()` manuell aufrufen, wenn Sie auf die gleiche Datei von ungepuffertem IO auf gepuffertes wechseln. Das alles klingt ein wenig kompliziert, aber es ist wirklich einfach. In der Tat gibt es Ihnen die volle Kontrolle über die DOS-Befehle. Bitte beachten Sie die Dokumentation von `SetIOMode()` für weitere Informationen.
- Bis zu Hollywood 2.0 richtete der Befehl `TextOut()` automatisch den Text aus, wenn eine spezielle Koordinatenkonstante wie `#CENTER` oder `#RIGHT` als `x` angegeben wurde. Dies wird nicht mehr auf diese Weise durchgeführt. Es gibt ein neues Argument, das Sie für die gewünschte Ausrichtung angeben können.

## Hollywood 2.0 API Änderungen

Obwohl Hollywood 2.0 ein gigantisches Update ist, waren nur wenige API-Änderungen erforderlich. Hier ist eine Liste der Dinge, die Sie in Ihrem Skript ändern müssen:

- Wenn Sie Befehle aufrufen, die keine Argumente akzeptieren, aber einen Wert zurückgeben, müssen Sie Klammern verwenden. Zum Beispiel arbeitete der folgende Code in 1.9, aber funktioniert nicht mehr in 2.0:

```
; falsch!
x = MouseX
y = MouseY
```

Sie müssen das nun folgendermaßen schreiben:

```
; richtig!
x = MouseX()
y = MouseY()
```

Die falsche Version wird durch die Art und Weise nicht einen Compiler-Fehler auslösen. Es ist richtiger Hollywood Code, tut aber etwas ganz anderes: Er weist den Befehl `MouseX()` der Variablen `x` zu. Das ist in diesem Fall aber nicht das, was Sie wollen.

- `GetTimer()` liefert jetzt immer den Wert in Millisekunden. In Hollywood 1.x war die Standardeinheit Sekunden. Natürlich hätte ich die alte Implementierung beibehalten können, aber es gibt ehrlich gesagt niemand, der einen Rückgabewert in Sekunden will, weil er einfach zu ungenau ist. So entschied ich den Programmierer einen Gefallen zu tun und Millisekunden als Standard zu setzen, so dass Sie nicht jedes Mal den langen `GetTimer(1, #MILLISECONDS)` eingeben müssen, sondern nur noch `GetTimer(1)`.
- Die Befehle `MoveBrush()`, `MoveTextObject()`, `MoveAnim()` ... können nicht mehr alte Objekte "greifen". Zum Beispiel arbeitet der folgende Code in 2.0 nicht korrekt:

```
MoveBrush(1, #LEFTOUT, #CENTER, #CENTER, #CENTER)
Wait(100)
MoveBrush(1, #CENTER, #CENTER, #RIGHTOUT, #CENTER)
```

In Hollywood 1.x bewegt dieser Code den Pinsel 1 von links außen in die Mitte, wartete 100 Millisekunden, und bewegt den Pinsel nach rechts außen. In Hollywood 2.0 wird es das Gleiche tun, aber die eine Kopie vom Pinsel wird in der Mitte des Displays bleiben. Dies ist auf große Veränderungen in dem Auffrischungssystem zurückzuführen. Wenn Sie das 1.x Verhalten imitieren wollen, verwenden Sie `MoveSprite()` anstelle von `MoveBrush()`.

- `DisplayTransitionFX()` kann nicht mehr für transparente Hintergrundbilder verwendet werden; Umschalten auf transparente BGPics kann jetzt nur noch ohne Wirkung erfolgen. Dies liegt daran, dass Hollywood 2.0 jetzt echte transparente Fenster auf MorphOS, OS4 und AROS verwendet. Diese Fenster haben eine Ebene, wo keine Grafiken erstellt werden können.
- In Hollywood 1.x `MixBrush()` skalierte die beiden Pinsel gleich groß, wenn sie unterschiedliche Abmessungen hatten. Dies ist nicht mehr der Fall. `MixBrush()` mischt nur noch die Teile, die passen und verwirft entsprechend den Rest.
- `RotateBrush()` wird nun eine Maske für den Pinsel erstellen, wenn es noch keine hat. Sie müssen dies nicht mehr auf eigene Faust tun.
- Wenn Ebenen aktiviert sind und Sie `InKeyStr()` benutzen, wird nur eine Ebene des Typs `#PRINT` installiert werden. In Hollywood 1.x erzeugte `InKeyStr()` für jedes Zeichen eine `#PRINT` Ebene.

## Hollywood 1.9 API Änderungen

Es gab in Hollywood 1.9 einige kleinere API-Änderungen, die hier aufgeführt sind:

- Die Befehle `EnableEventHandler()` und `DisableEventHandler()` wurden entfernt. Sie könnten viel Ärger verursachen, denn wenn man sie verwendet, müssen Sie nicht wissen, wann Ihre Ereignisfunktionen aufgerufen werden. Bitte benutzen Sie nun den neuen Befehl `CheckEvent()`.

- `EnablePrecalculation()` und `DisablePrecalculation()` wurden entfernt, weil Effektivkalkulation von Hollywood nicht mehr unterstützt wird. Das Argument/Tooltype `PRECALCULATION` fällt darum auch weg.
- `WhileMouseOn()` hatte einige Änderungen erfahren, die Sie wahrscheinlich unter bestimmten Umständen feststellen könnten: In früheren Versionen sprang Hollywood sofort zu Ihrer `WaitEvent()`-Schleife zurück, nachdem ein `OnButtonClick` Ereignis aufgetreten ist. Dieses Verhalten war falsch! Jetzt wird es zurück zu Ihrem Befehl `WhileMouseOn()` springen, weil die Maus nach `ONBUTTONCLICK` immer noch über dem Knopf ist. Wenn Sie möchten, dass Hollywood 1.9 sich wie Hollywood 1.0 und 1.5 verhält, können Sie den neuen Befehl `BreakWhileMouseOn()` verwenden.

## Hollywood 1.5 API Änderungen

Leider hatte ich in dem Update von Version 1.5 einige API-Änderungen an der Hollywood Sprache vornehmen müssen. Wenn Ihr Skript nicht richtig unter Hollywood 1.5 funktioniert, aber arbeitete unter 1.0 tadellos, so lesen Sie bitte die folgenden Informationen durch und passen Sie Ihr Skript an.

- Die Syntax der Konstante hat sich geändert. In Hollywood 1.0 gaben Sie nur durch ihren Namen Konstanten an, aber jetzt werden Sie auch ein '#' als Präfix angeben müssen (zum Beispiel `#CENTER` statt `CENTER` und `#BOLD` anstelle von `BOLD`). Es tut mir leid, aber diese Änderung war absolut notwendig.
- `Undo()` wird nicht funktionieren, bis Sie `EnableLayers()` aufgerufen haben. Wenn Sie `Undo()` in Ihrem Skript verwenden, stellen Sie also sicher, dass Sie `EnableLayers()` am Anfang benutzt haben.
- Syntax von `PlaySample()` hat sich geändert. Sie können nicht mehr einen Kanal für die Wiedergabe festlegen. Hollywood wird alles für Sie tun. Geben Sie einfach die Nummer des Samples an und ob es als Schleife (loop) abgespielt werden soll oder nicht.
- Syntax von `PlayAnim()` hat sich geändert. Es läuft jetzt synchron. Diese Änderung war notwendig, weil die alte `PlayAnim()` Umsetzung nicht mehr in das Konzept passt. Wenn Sie Anims asynchron abspielen wollen, verwenden Sie Pinsel-Link-Frames und zeigen sie mit `DisplayBrush()` an. Da `PlayAnim()` nun synchron ist, sind die Befehle `IsAnimPlaying()` und `WaitAnimEnd()` nicht mehr erforderlich und wurden entfernt.
- `ClearScreen()` wurde entfernt, weil es nicht mehr ins Konzept passte.
- `LoadModule()` lädt nicht mehr THX, P61 oder MED-Modul. Modulunterstützung konzentriert sich nun auf das Protracker-Format. Andere Modulformate können nicht sauber durch AHI abgespielt werden.
- `Print()` unterstützt Antialiasing für TrueType-Schriftarten nicht mehr. Diese Änderung war notwendig, um zurzeit mit Ebenen kompatibel zu bleiben. Antialiasing wird für alle Objekte in Hollywood 2.0 wieder eingeführt.

## 6.3 Zukunft

Hier sind einige Ideen, die auf meiner Liste stehen:

- Geschwindigkeit aller Übergangs-FX-Befehle sollten in Millisekunden statt eines benutzerdefinierten Typs übergeben werden

- Textübergangseffekte
- API für die Erstellung von Video-Streams mit Hollywood
- Wenn möglich schnelleres Zeichnen mit Polygon Clipping
- Mehr Befehle

Bitte schreiben Sie mir eine **Mail**, wenn Sie ein paar nette Ideen haben, was in Hollywood umgesetzt werden sollte.

## 7 Aufbau der Sprache

### 7.1 Ihr erstes Hollywood Programm

Hollywoods Skriptsprache ist leicht zu benutzen, aber sehr mächtig! Die Syntax basiert hauptsächlich auf BASIC. Aber Hollywood ist viel mächtiger, weil es eine dynamische Sprache ist! Wir werden das später erläutern, was das fürs Programmieren bedeutet. Hollywood integriert die besten Elemente von (Blitz-) BASIC, C, AmigaE, Pascal und Lua in eine mächtige, flexible Sprache, die es Ihnen erlaubt, fast alles mit wenig Anstrengung zu erstellen.

Hollywood-Skripte sind einfach nur Textdateien in UTF-8-Codierung. Deshalb starten Sie Ihren Lieblingstexteditor und wir fangen mit den ersten Schritten an!

So sieht das Berühmte 'Hello World' Programm mit Hollywood aus:

```
Print("Hello World!")
WaitLeftMouse()
End()
```

Das obige kleine Programm wird ein Display mit 640x480 Bildpunkten öffnen. Falls Sie andere Abmessungen wünschen, müssen Sie die Präprozessor-Anweisung `@DISPLAY` oder `@BGPIC` angeben. 640x480 ist die voreingestellte Größe, die Hollywood benutzt, wenn Sie nichts anderes angeben. Die Displaygröße ist nicht dieselbe wie die Bildschirmgröße. Es ist nur die Größe Ihres Displays (Ihres Arbeitsbereichs!). Die Bildschirmgröße kann alles Mögliche sein, sofern es nur groß genug für Ihr Display ist. Ihr Display wird auf dem Bildschirm zentriert. Das Fenster, das geöffnet wird und Ihr Display enthält, ist größer als das Display, falls es einen Rahmen hat. Wenn Sie das Argument `-borderless` angeben, sind Display und Fenster gleich groß.

Wenn Sie anstelle einer schwarzen Fläche lieber ein kunstvolles Hintergrundbild haben wollen, setzen Sie an den Anfang Ihres Skriptes die Präprozessor-Anweisung `@BGPIC`:

```
@BGPIC 1, "FancyBackground.jpg"
Print("Hello World!")
WaitLeftMouse()
End()
```

Sie können auch mehrere Befehle in eine Zeile schreiben, so könnte in dem Fall das obere Skript so aussehen:

```
Print("Hello World!") WaitLeftMouse() End()
```

Aber damit der Code besser lesbar ist, wird geraten, für jeden Befehl mindestens eine Zeile zu benutzen. Außerdem verbessern auch Kommentare die Lesbarkeit. Hollywood bietet dafür zwei Möglichkeiten: Der Kommentar beginnt mit `/*` und hört mit `*/` auf (kann auch mehrere Zeilen später sein) oder er startet mit `;` und gilt bis ans Zeilenende.

```
/* this is a comment */
Print("Hello World!")    ; this one too
WaitLeftMouse()         ; Wait for left mouse
End()                   ; Exit

/* Das ist ein Kommentar */
Print("Hello World!")    ; Das ist auch einer
```



```
WaitLeftMouse() ; Wartet auf die linke Maustaste
End() ; Beendet das Programm
```

Wenn ein Hollywood-Befehl keine Argumente akzeptiert oder zurückgeben wird, können die Klammern weggelassen werden. Hingegen wollen Sie Argumente dem Befehl übergeben oder erwarten einen Rückgabewert, dann müssen Sie Klammern setzen. In unserem Beispiel können wir die Klammern für die Befehle `WaitLeftMouse()` und `End()` weglassen, da sie keine Argumente verwenden:

```
Print("Hello World!")
WaitLeftMouse
End
```

Natürlich ist es auch möglich, Variablen statt Zahlen oder Zeichenketten zu benutzen. Sie müssen die Variable vorher nicht deklarieren. Sie werden mit 0 bzw. einer leeren Zeichenkette initialisiert, wenn Sie sie das erste Mal benutzen. Variablen müssen mit einem Buchstaben von A/a bis Z/z oder mit einem Unterstrich zu beginnen. Nach dem ersten Zeichen dienen die folgenden speziellen Zeichen zur Unterscheidung: Dollarzeichen "\$" (für Zeichenketten), Ausrufezeichen "!" (für Fließkommazahlen) und außerdem können sie Ziffern 0 bis 9 enthalten. Die Länge der Variablennamen darf 64 Zeichen nicht übersteigen. So sieht unser Beispiel mit einer Variablen aus:

```
mystring$ = "Hello World!"
Print(mystring$)
WaitLeftMouse
End
```

In Hollywood sind neben normalen Befehlen auch Präprozessor-Anweisung vorhanden. Diese Anweisungen müssen mit einem @-Zeichen (at) beginnen und müssen deklariert sein, bevor Ihr Skript startet. Eines dieser Präprozessor-Anweisung ist `@VERSION`. Sie erlaubt Ihnen festzulegen, welche Hollywoodversion als Minimum vorausgesetzt wird. Zum Beispiel wird das folgende Skript nur mit Hollywood 2.0 und höher arbeiten:

```
@VERSION 2,0
Print("Hello World!")
WaitLeftMouse
End
```

Sie sollten immer diese Präprozessor-Anweisung als erstes benutzen, damit die Version geprüft wird, bevor andere Befehle ausgeführt werden.

Wenn Sie den Code in Ihrem Texteditor getippt und als `MyScript.hws` abgespeichert haben, öffnen Sie eine Konsole und tippen:

```
Hollywood MyScript.hws [ARGUMENTS]
```

[ARGUMENTS] kann irgendeine Kombination von Hollywoods Konsolenargumenten sein. Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), Seite 33, für mehr Informationen über die unterstützten Argumente.

Wenn Sie Ihr Skript mit der GUI ausführen wollen, starten Sie die GUI, drücken den Knopf "Skript starten" und wählen Ihr Programm aus.

Gratulation, Sie haben gerade Ihr erstes Hollywood-Skript erstellt und ausgeführt!



## 7.2 Reservierte Bezeichnungen

Die folgenden Bezeichnungen sind für Hollywood reserviert und können nicht als Namen für Variablen oder Funktionen benutzt werden:

And  
Block  
Break  
Case  
Const  
Continue  
Default  
Dim  
DimStr  
Do  
Else  
ElseIf  
EndBlock  
EndFunction  
EndIf  
EndSwitch  
FallThrough  
False  
For  
Forever  
Function  
Global  
Gosub  
Goto  
If  
In  
Label  
Local  
Next  
Nil  
Not  
Or  
Repeat  
Return  
Step  
Switch  
Then  
To  
True  
Until  
Wend  
While  
Xor

Gebrauchen Sie trotzdem eines dieser Bezeichnungen (als Namen für Variablen oder Funktionen) wird sich Hollywood mit einer Fehlermeldung an Sie wenden.

## 7.3 Präprozessor-Anweisungen

Eine Präprozessor-Anweisung ist eine Anweisung, welche Hollywood vor dem Ablauf Ihres Skripts ausführt. In Hollywood werden sie hauptsächlich zum Vorladen von Dateien benutzt. Zum Beispiel wenn Ihr Skript irgendwann die Dateien `mainmenu.png`, `gamescreen.png` und `music.mod` verlangt, könnten Sie sie mit folgendem Code einfach vorladen:

```
@BGPIC 1, "mainmenu.png"
@BGPIC 2, "gamescreen.png"
@MUSIC 1, "music.mod"
```

Hollywood lädt dann alle diese Dateien, bevor Ihr Skript ausgeführt wird. Somit sind alle über Präprozessor-Anweisungen geladenen Dateien sofort zur Benutzung bereit, wenn Ihr Skript beginnt. Die meisten `LoadXXX()`-Befehle haben eine Präprozessor-Anweisungen. Zum Beispiel der Präprozessor von `LoadBrush()` ist `@BRUSH`, von `LoadBGPic()` ist `@BGPIC` und so weiter.

Präprozessor-Anweisungen fangen immer mit dem Textzeichen(@) an. Sie sollten diese auch in Großbuchstaben schreiben, damit sie besser von den normalen Befehlen unterschieden werden können. Die Präprozessor-Anweisungen können irgendwo im Skript plazierte werden. Es wird aber wegen der besseren Lesbarkeit empfohlen, sie an den Anfang des Skriptes zu schreiben.

Eine elementare Präprozessor-Anweisung ist `@VERSION`. Sie sollten diese als erstes in Ihrem Skript benutzen! `@VERSION` überprüft, ob die Hollywood-Version geeignet ist, um das Skript auszuführen. Ansonsten wird Hollywood abbrechen.

Die meisten Präprozessor-Anweisungen haben mehrere Argumente, die wie bei den normalen Befehlen durch Kommas getrennt werden. Sie können auch Ausdrücke darin benutzen. Zum Beispiel wäre die folgende Deklaration ungewöhnlich, aber vollkommen gültig:

```
@BRUSH 5+5, "MyBrush.png"
```

Dies würde `MyBrush.png` als Pinsel Nr. 10 laden. Sie können aber keine Variablen in Ihren Ausdrücken benutzen. Wenn Hollywood Ihr Skript auf Präprozessor-Anweisungen analysiert, weiß es noch nichts über die Inhalte der Variablen, da Ihr Skript noch nicht ausgeführt wird. Das heißt, dass alle Ausdrücke, die Sie benutzen, konstant sein müssen.

Ein weiterer Vorteil der Präprozessor-Anweisungen ist, dass alle angegebenen/vorgeladenen Dateien automatisch mit dem ausführbaren Programm oder Applet verbunden/gelinkt werden. Dieses Verhalten kann mit dem Tag `Link` gewechselt werden und ist bei allen Präprozessor-Anweisungen vorhanden, welche mit Dateien arbeiten. Voreingestellt ist `TRUE` (Wahr) und somit werden die Dateien verlinkt. Wenn Sie keine bestimmten Dateien verlinken wollen, weil sie z.B. zu groß sind, müssen Sie das ausdrücklich mit der Präprozessor-Anweisung angeben.

Die folgenden Präprozessor-Anweisungen sind verfügbar:

<code>@ANIM</code>	Lädt eine Animation vor
<code>@APPAUTHOR</code>	Deklariert den Programmautor
<code>@APPCOPYRIGHT</code>	Deklariert das Copyright des Programms
<code>@APPDESCRIPTION</code>	Deklariert die Beschreibung des Programms

<code>@APPENTRY</code>	Deklariert das Projekteingangsskript
<code>@APPICON</code>	Deklariert das Piktogramm für Ihr Programm
<code>@APPIDENTIFIER</code>	Deklariert den Programmidentifikator
<code>@APTITLE</code>	Deklariert den Programmtitel
<code>@APPVERSION</code>	Deklariert die Programmversion
<code>@BACKFILL</code>	Konfiguriert die Hintergrundfüllung des Skripts
<code>@BGPIC</code>	Lädt ein Hintergrundbild vor
<code>@BRUSH</code>	Lädt ein Pinsel vor
<code>@CATALOG</code>	Lädt einen Sprachkatalog vor
<code>@DIRECTORY</code>	Bindet ein gesamtes Verzeichnis im Applet/Programm ein
<code>@DISPLAY</code>	Erstellt ein neues Display
<code>@ELSE</code>	Wird betreten, wenn alle Bedingungen fehlschlagen
<code>@ELSEIF</code>	Testet eine andere Bedingung
<code>@ENDIF</code>	Definiert das Ende des bedingten Blocks
<code>@ERROR</code>	Kompilation mit einer Fehlermeldung abbrechen
<code>@FILE</code>	Öffnet eine Datei
<code>@FONT</code>	Lädt eine Schriftart vor
<code>@ICON</code>	Lädt ein Piktogramm vor
<code>@IF</code>	Prüft die Bedingung
<code>@INCLUDE</code>	Fügt Code von einer anderen Datei ein
<code>@LINKER</code>	übergibt Optionen an den Linker
<code>@MENU</code>	Erstellt eine Menüleiste
<code>@MUSIC</code>	Lädt eine Musikdatei vor
<code>@OPTIONS</code>	Konfiguriert verschiedene Optionen
<code>@PALETTE</code>	Lädt eine Palette vor
<code>@REQUIRE</code>	Deklariert ein Plugin
<code>@SAMPLE</code>	Lädt ein Sample vor
<code>@SCREEN</code>	Konfiguriert den Bildschirmmodus für ihr Skript
<code>@SPRITE</code>	Lädt ein Sprite vor
<code>@VERSION</code>	Definiert, welche Hollywoodversion Ihr Skript benötigt
<code>@VIDEO</code>	Lädt ein Video vor
<code>@WARNING</code>	Sendet eine Warnmeldung an das Debug-Gerät

## 7.4 Zeichenketten- und Zahlenumwandlung

Hollywood unterstützt automatische Zeichenkettenumwandlung, um Nummern und Zahlen umzuwandeln. Das bedeutet, wenn ein/e Befehl/Funktion in einem Argument eine Zeichenkette erwartet und Sie dann eine Zahl übergeben, wird Hollywood diese Zahl automatisch in eine Zeichenkette konvertieren und dem/r Befehl/Funktion weitergeben.

Zum Beispiel: `StrLen()` gibt die Länge der angegebenen Zeichenkette zurück. Wenn wir jetzt

```
a = StrLen(256)
```

aufrufen, wird Hollywood die Zahl 256 automatisch zur Zeichenkette "256" konvertieren. Die Variable `a` erhält den Wert 3, weil die Zeichenkette "256" aus drei Buchstaben besteht.

Dies funktioniert auch umgekehrt. Wenn Sie einem/r Befehl/Funktion eine Zeichenkette übergeben, welche eine Zahl erwartet, wird Hollywood versuchen, diese Zeichenkette in eine Zahl zu konvertieren. Der Unterschied ist aber, dass die Konvertierung scheitern könnte.

Z.B. kann Hollywood "Hello" nicht in eine Zahl umwandeln. Die Zeichenkette muss dezimal oder hexadezimal sein. Variablen gemischt mit Buchstaben und Zahlenzeichenketten können nicht konvertiert werden, auch wenn die Ziffern vor den Buchstaben sind. Hexadezimalzahlen müssen mit einem Dollar Zeichen oder 0x beginnen. Ein Beispiel:

```
LoadBrush("1", "Brush.iff")
```

**LoadBrush()** erwartet eine Zahl als Argument. So wird die Zeichenkette "1" automatisch in eine Zahl konvertiert. Die nächsten beiden Codezeilen können nicht funktionieren:

```
LoadBrush("Test", "Brush.iff")
LoadBrush("1Test", "Brush.iff")
```

Die Zeichenkette "Test" oder "1Test" können nicht in eine Zahl umgewandelt werden.

Sie können auch alle mathematischen Ausdrücke mit Zahlen und Zeichenketten benutzen, außer Bedienungs-/Vergleichsoperatoren. Sie können nur zwei Werte der gleichen Art vergleichen. Zum Beispiel:

```
a = "5" * 10 + 100 / "10" + ("100" - 60) ; a ist 100
```

Aber dieser Code wird eine Fehlermeldung generieren, weil Sie mit Zeichenketten und Zahlen Vergleichsoperatoren miteinander vergleichen:

```
If "10" < 20      ---> Fehler!
```

Wenn Sie trotzdem zwei verschiedene Arten vergleichen wollen, benützen Sie **Val()** für eine Konvertierung in eine Zahl oder **StrStr()** für die Umwandlung in eine Zeichenkette.

```
If Val("10") < 20 ---> Funktioniert!
```

## 7.5 Kommentare

Hollywood unterstützt zwei Arten von Anmerkungen: Ein Zeilen- und einen Blockkommentar. Wenn Hollywood in Ihrem Code einen Strichpunkt (;) entdeckt, wird alles dahinter bis Zeilenende ignoriert. Zum Beispiel:

```
DebugPrint("Hello") ; Ein Kommentar bis Zeilenende!
```

Der Blockkommentar muss vom Benutzer beendet werden. Er beginnt mit der Zeichenfolge /\* und endet mit \*/. Weil dieser Kommentar vom Benutzer beendet werden muss, kann er auch mehrere Zeilen lang sein. Aber er kann auch am Anfang oder in der Mitte einer Zeile stehen.

```
/*
Alles dazwischen wird von Hollywood ignoriert!
*/
DebugPrint("Hello") /* Ich bin ein Kommentar */ DebugPrint("World")
```

Bitte kommentieren Sie Ihren Code! Sie müssen nicht jede kleine lokale Variable erklären, aber eine kurze Beschreibung der Funktion tut nicht weh und macht es für andere Personen einfacher, Ihr Programm zu verstehen.

## 7.6 Code einbinden/Includes

Zum importieren von Codes aus einer anderen Datei in das aktuelle Skript können Sie **Includes** benutzen. Hollywood-Skripts (**.hws** Dateien) sowie **Applets** (**.hwa**) sind dazu geeignet. Beim Ausführen oder Kompilieren des Skripts werden diese Dateien eingebunden. Code zu

importieren ist besonders bei größeren Projekten nützlich, um den Überblick zu behalten, als wenn alles in nur einer Quelldatei mit viel Code geschrieben wurde.

Die Idee vom Einbinden ist, Ihr Programm in mehrere Teile zu splitten. Zum Beispiel könnte ein Jump'n'run-Spiel in folgende Stücke geteilt werden: Intro, Menü, Kartenmodul, Level und Spiel. Jetzt erstellen Sie für jedes dieser Teile eine Quellcodedatei, z.B. **Intro.hws**, **Menu.hws**, **Kartenmodul.hws**, **Level.hws** und **Spiel.hws**. Eine der Quellcodedateien muss der Hauptquellcode sein. Das ist der, den Sie mit Hollywood beginnen.

Eine weitere Verwendung von Includes ist, Bibliotheken in Form von Hollywood **Applets** zu erstellen. Diese Applets veröffentlichen Sie dann, damit andere Programmierer von Ihnen profitieren können. Sie importieren dann das Applet in ihr eigenes Projekt. Der Vorteil von der Veröffentlichung als Hollywood-Applet ist, dass Sie Ihren Quellcode nicht freilegen müssen. Hollywood-Applets enthalten kompilierten Bytecode, der für uns nicht lesbar ist. Wenn Sie also Ihren Code schützen und trotzdem mit anderen Benutzern teilen möchten, veröffentlichen Sie ihn einfach als Hollywood-Applet.

Lassen Sie uns zum Jump'n'run-Spiel zurückkehren. Wir nehmen an, dass das **Intro.hws** unser Hauptquellcode sein wird, da der Endbenutzer das Intro als Erstes sehen wird. Unser **Intro.hws** wird grob etwa so aussehen:

```
@INCLUDE "Menu.hws"
@INCLUDE "Kartenmodul.hws"
@INCLUDE "Level.hws"
@INCLUDE "Spiel.hws"

ZeigeIntro()
ZeigeMenu()      ; Funktion ZeigeMenu() deklariert in Menu.hws
SpielStart()     ; SpielStart() deklariert in Spiel.hws
ZeichneKarten()  ; ZeichneKarten() deklariert in Kartenmodul.hws
NaechstesLevel() ; NaechstesLevel() deklariert in Level.hws
```

Wie Sie sehen, benutzen wir die Präprozessor-Anweisung **@INCLUDE**, um die vier anderen Quellcodes in unser **Intro.hws** einzubinden. Das erlaubt es uns nun, alle Funktionen, welche in diesen vier Dateien deklariert wurden, im Hauptquellcode von **Intro.hws** aufzurufen.

Include-Dateien enthalten in den meisten Fällen nur Funktionen, Variablen und Konstanten. Wenn Sie direkte Befehle aufrufen, welche nicht in einer Funktion integriert wurden wie z.B. **DebugPrint("Hello")**, wird dieser Befehl vor dem Hauptquellcode ausgeführt. Hollywood arbeitet die Befehle in der Reihenfolge ab, wie sie eingebunden wurden. In unserem Beispiel oben würde zuerst **Menu.hws** geöffnet und sein Code importiert, dann **Kartenmodul.hws**, **Level.hws** und schließlich **Spiel.hws**. Also, was Hollywood kompiliert würde wie folgt aussehen:

```
@INCLUDE "Menu.hws"
@INCLUDE "Kartenmodul.hws"
@INCLUDE "Level.hws"
@INCLUDE "Spiel.hws"

<...Inhalt der Datei Menu.hws...>
<...Inhalt der Datei Kartenmodul.hws...>
<...Inhalt der Datei Level.hws...>
```

```
<...Inhalt der Datei Spiel.hws...>
ZeigeIntro()
ZeigeMenu()
...
```

Sie sehen, dass alle Include-Dateien vor Ihrem Haupt Quellcode eingebunden werden. Deshalb werden alle direkten Befehle, die nicht in einer Funktion deklariert wurden, vor dem Haupt Quellcode ausgeführt.

Wenn Sie Applets einbinden möchten, geschieht dies auch mit der Präprozessor-Anweisung **@INCLUDE**:

```
@INCLUDE "Test.hwa" ; Importiert Funktionen von Test.hwa

LibFunc() ; Ruft LibFunc() auf, definiert in Test.hwa
```

## 7.7 Fehlerbehandlung

Es gibt verschiedene Möglichkeiten, mit Fehlern in Hollywood umzugehen. Am einfachsten ist es, wenn Hollywood das alles für Sie erledigt und ist auch so Voreingestellt. Standardmäßig wird Hollywood Ihr Skript immer beenden, wenn ein Fehler in einem Hollywood-Befehl auftritt. Betrachten Sie den folgenden Code:

```
LoadBrush (1, "xyz")
```

Wenn die Datei `xyz` nicht existiert, beendet Hollywood Ihr Skript und meldet folgenden Fehler: "Datei xyz kann nicht gelesen werden!"

Wenn Ihnen dieses Verhalten nicht gefällt, können Sie Hollywood auch anweisen, eine von Ihnen erstellte Funktion aufzurufen, wenn ein Fehler auftritt. Dies ist durch den Aufruf des Befehls **RaiseOnError()** möglich, der die von Ihnen definierte Callback-Funktion aufruft, wenn ein Fehler auftritt. Hier ist ein Beispiel, wie Sie Hollywoods Standardfehlerbehandlung mit einem benutzerdefinierten Fehlerhandler ersetzen können:

```
Function p_ErrorFunc(code, msg$, cmd$, line)
    DebugPrint(code, msg$, cmd$, line)
EndFunction
RaiseOnError(p_ErrorFunc)
LoadBrush(1, "xyz")
```

Wenn Sie den obigen Code verwenden, lädt **LoadBrush()** den Pinsel `xyz`. Falls diese Datei nicht existiert, wird nicht Hollywoods-Standardfehlerhandler ausgelöst, sondern stattdessen die Benutzerfunktion **p\_ErrorFunc()** aufgerufen und gibt so weitere Informationen über den aufgetretenen Fehler aus. Siehe [Abschnitt 27.7 \[RaiseOnError\]](#), Seite 564, für Details.

Manchmal kann es jedoch nützlich sein zu wissen, ob ein einzelner Befehl erfolgreich war oder nicht. Dies kann erreicht werden, indem der Hollywood-Fehlerhandler vorübergehend deaktiviert wird, um den Fehlercode vom letzten Befehl zu erhalten:

```
ExitOnError(False)          ; Deaktiviert den Standardfehlerhandler
LoadBrush(1, "xyz")
error = GetLastError()
ExitOnError(True)           ; Aktiviert den Standardfehlerhandler
```

Der obige Code deaktiviert vorübergehend Hollywoods Standardfehlerbehandlung nur für die Dauer des Befehls **LoadBrush()**. Direkt nach dem **LoadBrush()** benutzen Sie den Be-

fehl `GetLastError()`, um herauszufinden, ob der Befehl `LoadBrush()` erfolgreich war oder nicht. Es ist wichtig, `GetLastError()` unmittelbar nach `LoadBrush()` zu benutzen, da beim nächsten Befehl der interne Fehler-Flag zurückgesetzt wird. Wenn Sie also einen anderen Befehl nach `LoadBrush()` aufrufen, wird `GetLastError()` den Fehlerstatus dieses Befehls anstelle von `LoadBrush()` zurück geben.

Da der obige Code viel Schreibaufwand für eine ziemlich einfache Aufgabe erfordert, gibt es auch eine syntaktische Vereinfachung, die das gleiche wie oben bewirkt und so die Menge der erforderlichen Eingaben reduziert. Anstatt wie oben gezeigt `ExitOnError()` und `GetLastError()` manuell aufzurufen, können Sie auch Hollywood alles automatisch erledigen lassen, indem Sie einfach dem Befehl ein Fragezeichen voranstellen. So könnte der obige Code auch so geschrieben werden:

```
error = ?LoadBrush(1, "xyz")
```

Wenn ein Befehl Werte zurückgibt und Sie ein Fragezeichen verwenden, um einen Fehlercode aus einem Befehlsaufruf zu erhalten, werden alle anderen Rückgabewerte einfach nach rechts verschoben. Der Fehlercode ist immer der erste Rückgabewert. Wenn wir zum Beispiel die automatische ID-Zuweisung mit dem Befehl `LoadBrush()` verwenden und dies mit dem Fragezeichensyntax kombinieren wollen, müssen wir den Code so schreiben:

```
error, id = ?LoadBrush(1, "xyz")
```

Normalerweise wäre `id` der erste Rückgabewert, aber da wir die Fragezeichensyntax verwenden, um einen Fehlercode vom Befehl `LoadBrush()` zu erhalten, wird der erste Rückgabewert nach rechts verschoben und wird jetzt zum zweiten Rückgabewert. Der Fehlercode ist immer im ersten Rückgabewert enthalten.

Um schließlich zu prüfen, ob ein Fehler aufgetreten ist, müssen Sie lediglich den Fehlercode `error` mit `#ERR_NONE` vergleichen, der zur Vereinfachung als 0 definiert ist. D.h. wenn `error` nicht 0 ist, wissen Sie, dass etwas schief gelaufen ist. Sie könnten dann `GetErrorName()` verwenden, um den Fehlercode in eine für Menschen lesbare Zeichenfolge zu konvertieren oder eine benutzerdefinierte Fehlerbehandlung in Abhängigkeit vom eingestellten Fehlercode implementieren. Siehe [Abschnitt 27.4 \[Fehlercodes\]](#), [Seite 524](#), für eine Liste aller Fehlercodes.

Bitte beachten Sie, dass einige Fehler nicht erfasst werden können. Wenn Sie beispielsweise eine falsche Anzahl von Argumenten oder falsche Variablentypen an einen Befehl übergeben, wird Hollywood immer sofort mit einem schwerwiegenden Fehler beendet, und Ihrem Skript wird keine Chance zum Auffinden solcher Fehler gegeben. Obwohl sie zur Laufzeit auftreten, wird Hollywood bei solchen Syntaxfehler sofort beendet. Hier ist ein Beispiel, wo wir eine Zeichenfolge im ersten Argument vom Befehl `LoadBrush()` übergeben, was nicht erlaubt ist, weil `LoadBrush()` eine Zahl erwartet:

```
ExitOnError(False)
LoadBrush("Hello", "xyz")
ExitOnError(True)
```

Obwohl wir den Fehlerhandler von Hollywood deaktiviert haben, indem wir `False` an `ExitOnError()` übergeben, wird Hollywood die Ausführung des Skripts sofort abbrechen, weil "Hello" an `LoadBrush()` übergeben wurde, was einfach falsch ist. Hollywood wird dies als einen fatalen Fehler betrachten und Ihrem Skript nicht erlauben, diesen Fehler in irgendeiner Weise abzufangen.



## 7.8 Automatische ID-Zuweisung

Bei allen Befehlen, die von Ihnen eine ID (Identität/Identifikator/Kennung) für das neue Hollywood-Objekt verlangen, können Sie **Nil** angeben. In diesem Fall wird Hollywood automatisch eine ID auswählen und an Sie zurückgeben. Dies ist besonders für größere Projekte nützlich. Wenn Ihr Projekt klein ist, ist es bequemer, die IDs selbst zu wählen, z.B:

```
LoadBrush(1, "brush1.iff")
LoadBrush(2, "brush2.iff")
LoadSample(1, "sample.wav")
OpenFile(1, "file.txt")
```

Wenn Ihr Projekt jedoch größer wird, kann das ID-Management sehr aufwendig werden und niemand will sich mit einer Vielzahl von unterschiedlichen IDs und Hollywood-Objekten beschäftigen. So können Sie einfach **Nil** anstelle einer ID angeben und Hollywood wird eine ID für das neue Objekt zurückgeben, die garantiert eindeutig ist, weil es den speziellen Variablentyp **#LIGHTUSERDATA** verwendet. Auf diese Weise wird sichergestellt, dass keine ID-Konflikte entstehen. Außerdem wird Hollywood nicht eine ID aus dem ID-Pool aussuchen (das heißt ganze Zahlen von 1 bis n). Somit werden alle normalen IDs immer noch zur Verfügung stehen, z.B:

```
brush1 = LoadBrush(Nil, "brush1.iff")
brush2 = LoadBrush(Nil, "brush2.iff")
sample1 = LoadSample(Nil, "sample.wav")
file1 = OpenFile(Nil, "file.txt")
```

Die Variablen **brush1**, **brush2**, **sample1** und **file1** werden keine für den Benutzer lesbare IDs generieren, aber spezielle IDs vom Typ **#LIGHTUSERDATA** erhalten. Somit werden alle vom Benutzer lesbaren IDs ab 1 bis n immer noch verfügbar sein. Daher müssen Sie sich nicht um irgendwelche ID-Konflikte sorgen, wenn Sie **Nil** zum Erstellen von Objekten benutzen, weil Hollywood zwei getrennte ID-Dimensionen verwendet: Eine für Benutzer lesbare, die nur zur Anwendung kommt, wenn Sie eine ID bei der Objekterstellung angeben und einen internen ID-Mechanismus, der verwendet wird, wenn Sie anstelle einer ID **Nil** bei der Objekterstellung eintippen.

## 7.9 Lade- und Adaptermodule

Viele Hollywood-Befehle unterstützen Lade- und Adaptermodule. Der Unterschied zwischen einem Lade- und einem Adaptermodul ist folgender: Ein Lademodul fügt Unterstützung für zusätzliche Bild-, Ton-, Video-, Piktogramm-, Schriftart- oder Animationsformate hinzu, während ein Adaptermodul bestimmte Teile von Hollywood durch eine eigene Implementierung ersetzt. Zum Beispiel gibt es Display-Adapter, die verwendet werden können, um Hollywoods integrierter Display-Handler durch einen benutzerdefinierten zu ersetzen (z.B. von SDL oder OpenGL verwalteten Displays), es gibt Netzwerkadapter, die es Plugins ermöglichen, Hollywoods integrierte Netzwerkimplementierung zu überschreiben, und natürlich gibt es Datei-Adapter.

Dateiadapter können beispielsweise verwendet werden, um Unterstützung für neue Containerformate hinzuzufügen. Sie können so an Lademodule gebunden werden, dass der Adapter die Rohdaten bereitstellt, die dann später im Prozess von einem Lademodul interpretiert werden. Beispielsweise könnte ein Adaptermodul das Lesen von gzip-komprimierten Dateien unterstützen. Die so von einem Adaptermodul extrahierten Daten könnten dann von



einem Lademodul verarbeitet werden. Beispielsweise könnte ein BMP-Bild in einer gzip-komprimierten Datei enthalten sein: Hollywood würde dann zuerst das Adaptermodul verwenden, um die unkomprimierten Daten der gzip-Datei bereitzustellen, und dann das Lademodul benutzen, um das eigentliche BMP-Bild zu laden. Ein Datei-Adaptermodul kann auch Daten aus einer zufälligen Quelle implementieren, z.B. von HTTP-Server oder anderen Quellen.

Beginnend mit Hollywood 6.0 können Sie bei fast allen Befehlen, die mit Dateien umgehen können, ein Lade- und/oder ein Adaptermodul in der optionalen Argumenttabelle angeben. Die Idee ist, das Laden von externen Daten zu beschleunigen. Wenn Sie nicht ein Lade- oder ein Adaptermodul angeben, wird Hollywood mit allen derzeit installierten Modulen prüfen, ob diese Datei geöffnet werden kann. Je nachdem, wie viele Plugins Sie installiert haben und wie viele Dateien geladen werden müssen, kann diese Überprüfung den ganzen Prozess erheblich verlangsamen. Wenn Sie das Lade- oder Adaptermodul kennen, mit dem Ihre externen Daten geladen werden sollen, können Sie den Namen in dem Ladebefehl angeben und so den Ladevorgang beschleunigen.

Die Zeichenfolge, die Sie zu den Tags **Loader** oder **Adapter** durch das optionale Tabellenargument übergeben, muss mindestens ein Einzellademodul- oder Adaptermodulnamen beinhalten. Oder ein reserviertes Schlüsselwort beschreibt ein spezielles Lade- oder Adaptermodul. Mehrere Namen und Schlüsselwörter müssen durch einen senkrechten Strich (|) getrennt werden. Die folgenden Schlüsselwörter sind zur Zeit verfügbar:

**Default:** Dies ist der Standardbetriebsmodus. Er kann nicht mit anderen Schlüsselwörtern oder Lade- sowie Adaptermodulnamen kombiniert werden. Er muss immer unabhängig von den anderen verwendet werden. Im Standard-Betriebsmodus wird Hollywood zuerst alle Lade- und Adaptermodule prüfen, ob sie eine Datei behandeln können. Wenn kein Plugin das handhaben kann, kommen die in Hollywood eingebauten Module zum Zuge. Wenn keine integrierten Module die Datei erkennen, werden native Ladeprogramme des Host-Betriebssystems benutzt, um die Datei zu laden.

**Inbuilt:** Wenn dieses Schlüsselwort angegeben ist, werden die in Hollywood eingebauten Lademodule überprüft, ob sie die Datei laden können. Hollywoods eingebaute Module unterstützen folgende Dateiformate:

**Eingebautes Bildlademodul:**

IFF ILBM, JPEG, PNG, GIF und BMP.

**Eingebautes Animlademodul:**

IFF ANIM, GIF ANIM und AVI MJPEG.

**Eingebautes Soundlademodul:**

IFF 8SVX, IFF 16SV, RIFF WAVE und Protracker.

**Eingebautes Videolademodul:**

CDXL video.

**Native:** Wenn dieses Schlüsselwort angegeben ist, wird Hollywood im Host-Betriebssystem nach Lade- und Adaptermodulen suchen, um die Datei zu laden. Dies wird nur von bestimmten Typen und Betriebssysteme unterstützt. Hier eine Übersicht:

- AmigaOS:** Als native Lademodule werden Datatypes verwendet, um Bilder, Animationen und Sounds zu laden. Es gibt kein natives Videolademodul.
- Windows:** Es gibt native Sound- und Videolader, die auf DirectShow und Media Foundation basieren, und native Bild- und Animationslader, die auf der Windows-Imaging-Komponente basieren.
- macOS:** Es hat native Sound-, Video- und Bildlademodule basierend auf macOS-Technologien. Es gibt kein natives Animlademodul.
- Linux:** Es gibt keine nativen Lademodule.
- iOS:** Es gibt native Sound-, Video- und Bildlademodule, die auf iOS-Technologien basieren. Es gibt kein natives Animlademodul.
- Android:** Es gibt keine nativen Lademodule.
- Plugin:** Wenn dieses Schlüsselwort angegeben ist, wird Hollywood alle Plugins prüfen, ob sie die Datei bearbeiten können. Plugins werden in der Reihenfolge geprüft, in der sie von Hollywood geladen wurden. Dies ist eher zufällig, da dies von der Reihenfolge abhängt, in der sie vom Dateisystem an Hollywood zurückgegeben werden. Wenn Sie sicherstellen möchten, dass ein bestimmtes Plugin aufgefordert wird, eine Datei vor einem anderen Plugin zu bearbeiten, müssen Sie den Namen dieses Plugins explizit in die Zeichenkette aufnehmen, anstatt das generische Schlüsselwort **Plugin** zu verwenden.

Wenn Sie ein allgemeines Schlüsselwort wie **Plugin** verwenden und Sie wollen herausfinden, welche Module für das Laden der Datei verwendet wurde, können Sie die Attribute **#ATTRLOADER** oder **#ATTRADAPTER** im Befehl **GetAttribute()** verwenden. Lade- und Adaptermodule können auch einen Formatnamen für die geladene Datei bereitstellen. Sie erhalten dies, indem Sie das Attribut **#ATTRFORMAT** ermitteln.

Hier sind einige Beispiele:

```
LoadBrush(1, "test.png", {Loader = "inbuilt"})
```

Der obige Code lädt die angegebene Datei mit dem eingebauten PNG-Bildlademodul. Weder externe Bild, Plugins oder Host-OS-Lademodule werden jemals berücksichtigt.

```
LoadBrush(1, "test.png", {Loader = "myplugin"})
```

Der obige Code versucht mit **myplugin.hwp** die Datei **test.png** zu laden. Wenn **myplugin.hwp** fehlschlägt, wird **LoadBrush()** auch scheitern. Die eingebauten Bildlademodule werden nicht berücksichtigt. Wenn Sie wollen, dass **LoadBrush()** zurück zu dem eingebauten Bildlademodul geht, müssen Sie das Schlüsselwort **Inbuilt** mit einem senkrechten Strich (**|**) getrennt hinzuzufügen:

```
LoadBrush(1, "test.png", {Loader = "myplugin|inbuilt"})
```

In diesem Fall wird **LoadBrush()** das eingebaute Bildlademodul bereitstellen, falls das Lademodul von **myplugin.hwp** fehlschlägt. Der folgende Code wird auf AmigaOS, Windows und macOS arbeiten, aber auf allen anderen Plattformen scheitern, da sie nicht über ein natives Bildlademodul verfügen (siehe oben):

```
LoadBrush(1, "test.png", {Loader = "native"})
```

Auf AmigaOS wird `test.png` über die Datatypes geladen, unter Windows mit der Windows-Imaging-Komponente, während dies auf dem macOS über die CGImage API geschieht. Unter Linux und Android wird das jedoch scheitern, weil Hollywood keine nativen Bildlademodule auf diesen Plattformen hat.

Sie können auch die Tags `Adapter` und `Loader` zum Beispiel wie folgt zusammen verwenden:

```
LoadBrush(1, "test.bmp.gz", {Adapter = "gzip", Loader = "inbuilt"})
```

Der obige Code wird zunächst die Datei `test.png.gz` mit `gzip.hwp` entpacken und dann das entpackten BMP-Bild mit dem eingebauten Bildlademodul laden. Natürlich können Sie auch den folgenden Code so schreiben und es würde auch funktionieren:

```
LoadBrush(1, "test.bmp.gz")
```

Allerdings gibt es hier einige Zusatzarbeiten, weil Hollywood zunächst alle Dateiadaptermodule und danach alle Bildlademodul-Plugins prüft, ob sie die Datei `test.bmp.gz` bearbeiten können oder nicht. Je nachdem, wie viele Plugins Sie installiert haben, kann dies einige Zeit in Anspruch nehmen. Wenn Sie also wissen, welche Adapter- und Lademodule Sie verwenden können, wird es die Ladegeschwindigkeit erhöhen, wenn Sie direkt den Namen des Lade- oder Adaptermoduls angeben.

Beachten Sie auch, dass die Reihenfolge der einzelnen Lader und Adapter von Bedeutung ist, wenn Sie mehrere Lader oder Adapter an einen Hollywood-Befehl übergeben und sie durch einen senkrechten Strich (`|`) trennen. Der folgende Code benutzt beispielsweise zuerst das Plugin `digibooster.hwp`, um die Datei zu öffnen, und dann das Plugin `xmp.hwp`:

```
OpenMusic(1, "shades.dbm", {Loader = "digibooster|xmp"})
```

Auf diese Weise können Sie auch bestimmte generische Lader wie `Native`, `Inbuilt` und `Plugin` priorisieren. Wie oben beschrieben, benutzt Hollywood standardmäßig zuerst Plugin-Lader, dann die integrierten und nachher die nativen Lader, um eine Datei zu öffnen. Wenn Sie möchten, dass native und integrierte Lademodule vor Plugin-Ladern benutzt werden, können Sie die Zeichenkette `native|inbuilt|plugin` übergeben, um dies zu erreichen. Mit einer solchen Zeichenkette werden native Lader zuerst und Plugin-Lader zuletzt benutzt. Sie können die Standardreihenfolge auch global ändern, indem Sie den Befehl `SetDefaultLoader()` bzw. `SetDefaultAdapter()` verwenden. Siehe [Abschnitt 50.27 \[SetDefaultLoader\]](#), [Seite 1111](#), für Details.

Beachten Sie, dass die Reihenfolge, in der die verschiedenen Lademodule überprüft werden, zur Zeit so festgelegt ist: Hollywood wird immer zuerst Plugins, dann die eingebauten und schließlich die einheimischen Lademodule des Gastgeber-OS durchgehen. Es gibt derzeit keine Möglichkeit, diese Reihenfolge zu ändern. Aber Sie können einfach durch die Angabe von einem Lade- oder Adaptermodul diese Einschränkung umgehen.

Außerdem werden Plugins in der Reihenfolge angefragt, wie sie in Hollywood geladen wurden. Die Reihenfolge in der Zeichenfolge im Tag `Loader` spielt keine Rolle.

Ein weiterer Vorteil vom direkt angegebenen Lade- oder Adaptermodul ist, dass Sie auf Lade- und Adaptermodule zugreifen können, die vor dem allgemeinen Gebrauch versteckt sind. Plugin-Autoren können entscheiden, dass Lade- oder Adaptermodul-Plugins nicht automatisch zur Verfügung stehen, sobald Hollywood das Plugin geladen hat. Es kann nur verwendet werden, wenn es entweder explizit mit dem Befehl `@Require` aktiviert oder direkt mit dem Namen des Plugins im Tag `Loader` oder `Adapter` angegeben wurde. Diese beiden Tags können also auch versteckte Plugins direkt verwenden.

Ab Version 8.0 unterstützt Hollywood jetzt auch Netzwerkadapter. Diese Adapter folgen demselben Prinzip wie die Datei- und Verzeichnisadapter, d.h. sie können verwendet werden, um die Befehle der Hollywood-Netzbibliothek durch benutzerdefinierte Handler zu leiten, die als Hollywood-Plugins implementiert sind. Netzwerkadapter können beispielsweise die Funktionalität von `DownloadFile()` und `UploadFile()` erweitern, um auch TLS/SSL-Verbindungen zu unterstützen. Sie können auch verwendet werden, um die Unterstützung für völlig unterschiedliche Netzwerktypen und -protokolle zu implementieren, da die Netzwerkadapter-Schnittstelle von Hollywood vollständig von jeder Art spezifischer Netzwerk-API abstrahiert wird. Dadurch ist die Anpassung an neue Umgebungen sehr flexibel.

Ab Version 10.0 unterstützt Hollywood auch Dateisystemadapter. Dateisystemadapter können verwendet werden, um zentrale Dateisystemfunktionen wie das Umbenennen von Dateien und Verzeichnissen, das Erstellen von Verzeichnissen, das Verschieben von Dateien und Verzeichnissen usw. durch benutzerdefinierte Funktionen zu ersetzen. Mehrere Hollywood-Befehle unterstützen Dateisystemadapter, z.B. `CopyFile()`, `MakeDirectory()` oder `DeleteFile()`.

## 7.10 Benutzer-Tags

Benutzer-Tags sind eine Möglichkeit, zusätzliche Informationen von Hollywood-Skripten an Plugins zu übergeben. Sie können verwendet werden, um eine unbegrenzte Anzahl von zusätzlichen Daten direkt aus dem Hollywood-Skript an Plugins zu übergeben. Die meisten Hollywood-Befehle, die Plugins unterstützen, erlauben Ihnen auch, Benutzer-Tags zu übergeben, die an Plugins weitergeleitet werden sollen. Dies macht natürlich nur dann Sinn, wenn die Benutzerdaten tatsächlich vom jeweiligen Plugin erkannt werden.

Nehmen wir zum Beispiel an, es gibt ein Plugin, das PDF-Seiten als Bilder laden kann. Dadurch ist es möglich, Hollywoods Befehl `LoadBrush()` zu verwenden, um einen Pinsel aus einer PDF-Seite zu erstellen. Befehle wie `LoadBrush()` unterstützen jedoch nicht die Angabe einer Seitenzahl oder eines Passworts, da sie nicht dafür ausgelegt sind, Seiten aus PDF-Dokumenten zu laden. Ein Plugin könnte diese Einschränkung umgehen, indem es einfach zwei neue Benutzer-Tags definiert, z.B. `Page` und `Password`, und dann könnten Skripte diese beiden Tags verwenden, um die Informationen an das Plugin zu übergeben.

Aus Sicht des Hollywood-Skripts werden Benutzer-Tags einfach in einer optionalen `UserTags-Tabelle` übergeben, die von vielen Hollywood-Befehlen akzeptiert wird, z.B. `LoadBrush()`. Die `UserTags-Tabelle` kann eine unbegrenzte Anzahl von Schlüssel-Wert-Paaren enthalten, die einzelne Benutzer-Tags definieren. Beachten Sie, dass der Schlüssel immer ein benannter Tabellenindex wie `Page` oder `Password` sein muss. Es ist nicht möglich, numerische Tabellenindizes als Benutzer-Tags zu verwenden. Der Wert kann eine Zeichenkette oder ein numerischer Wert sein. Wenn es sich um eine Zeichenkette handelt, kann er auch binäre Daten enthalten.

Um auf unser Beispiel von oben zurückzukommen: Um eine Seite und ein Passwort über Benutzer-Tags an ein Bild-Plugin zu übergeben, könnte ein Hollywood-Skript einfach folgendes tun:

```
; load page 5 of test.pdf as a brush, passing "mypwd" as the password
LoadBrush(1, "test.pdf", {UserTags = {Page = 5, Password = "mypwd"}})
```

Ein Plugin könnte dann nach den Tags **Page** und **Password** suchen, um die zu ladende Seitennummer und das Passwort für die PDF-Datei (falls vorhanden) herauszufinden. Dadurch ist es möglich, allerlei Zusatzinformationen an Hollywood-Plugins zu übergeben.

Darüber hinaus werden Benutzer-Tags auch von vielen Präprozessor-Anweisungen unterstützt, sodass Sie auch Folgendes tun könnten, um Seite 5 von `test.pdf` als Pinsel zu laden:

```
; load page 5 of test.pdf as a brush, passing "mypwd" as the password
@BRUSH 1, "test.pdf", {UserTags = {Page = 5, Password = "mypwd"}}
```

Beachten Sie, dass Benutzer-Tags von allen Arten von Plugins unterstützt werden: Sie werden von Bild-, Animations-, Ton-, Video-, Piktogramm-, Schriftartenladern, Datei-, Verzeichnis-, Display-, Netzwerkadaptern und Serialisierern unterstützt. Lader leiten die Benutzer-Tags normalerweise auch an Adapter weiter, sodass Dateiadapter in der Lage sind, Benutzer-Tags zu überwachen, die durch Lader geleitet werden.

## 7.11 Gestaltungsrichtlinien (Styleguide)

Hier sind einige Vorschläge, um Ihren Code lesbar zu halten. Wie Sie vorher schon erfahren haben, unterscheidet Hollywood nicht zwischen Groß- und Kleinschreibung. Aber um Ihren Code lesbar zu halten, schlagen wir die folgenden Richtlinien vor:

- Schreiben Sie Befehle immer so wie sie hier in der Dokumentation erscheinen und beginnen Sie sie vor allem mit einem Großbuchstaben.
- Schreiben Sie Konstanten in Großbuchstaben, um sie von Variablen zu unterscheiden.
- Schreiben Sie alle Präprozessor-Anweisungen in Großbuchstaben, um sie als Deklarationen innerhalb Ihres Skripts herauszuheben.
- Ein Befehl pro Zeile ist genug!
- Benutzen Sie das "\$"-Zeichen nur in Zeichenketten, um Verwirrungen zu vermeiden.
- Benutzen Sie das "!"-Zeichen nur in Fließkommazahlen, um Verwirrungen zu vermeiden.
- Ein paar Kommentare tun auch nicht weh.
- Wenn Sie Schleifen, If()- sowie andere Blöcke benutzen, sollten Sie mit Hilfe des Tabulators (oder mehreren Leerzeichen) die verschiedenen Ebenen absetzen.
- Sie sollten Ihre eigenen Funktionen mit einem "p\_" beginnen, um sie von Hollywood Befehlen unterscheiden zu können. Dies vor allem, wenn sie englische Namen benutzen. Es könnte sein, dass in Zukunft in Hollywood ein Befehl integriert wird, der denselben Namen wie Ihre Funktion hat. Das kann dazu führen, dass unerwartete Ergebnisse entstehen.



## 8 Datentypen

### 8.1 Datentypen Überblick

Dieses Kapitel deckt alle Datentypen ab, die in Hollywood verfügbar sind. Die folgenden fünf werden von Hollywood angeboten:

<b>Zahl</b>	Zahlenwerte wie 1, 2, \$FF, 3.5, 1.7e8, True, False
<b>Zeichenkette</b>	Reihen von Zeichen; normalerweise für Text wie "Hallo"
<b>Tabelle</b>	Sammlung von Datenelementen
<b>Funktion</b>	Vom Programmierer erstellte Funktionen
<b>Nil</b>	Eine Variable ohne Wert/Inhalt

Den Datentyp einer Variable können Sie herausfinden, indem Sie den Befehl `GetType()` benutzen. Bitte beachten Sie, dass Hollywood englische und nicht deutsche Konstanten zurückgibt. Zum Beispiel:

```

GetType(1)           ---> gibt #NUMBER zurück   (Zahl)
GetType(2.5)         ---> gibt #NUMBER zurück   (Zahl)
GetType(True)        ---> gibt #NUMBER zurück   (Zahl)
GetType('x')         ---> gibt #NUMBER zurück   (Zahl)
GetType(#STRING)     ---> gibt #NUMBER zurück   (Zahl)
GetType("What am I?") ---> gibt #STRING zurück   (Zeichenkette)
GetType({1, 2, 3})   ---> gibt #TABLE zurück    (Tabelle)
GetType(DebugPrint)  ---> gibt #FUNCTION zurück  (Funktion)
GetType(Nil)         ---> gibt #NIL zurück      (Nichts)

```

### 8.2 Zahlen

Der Zahlentyp (Number) kann zum Speichern von ganzen (Integer) und reellen Zahlen (Real) verwendet werden. Intern werden alle Zahlen als 64-Bit-Fließkommawerte gespeichert, was bedeutet, dass sie sehr große ganze und sehr genaue reelle Zahlen darstellen können. Der Zahlentyp kann Zahlen im Bereich von  $1,7 \cdot 10^{-308}$  bis  $1,7 \cdot 10^{308}$  speichern. Der Ganzzahlbereich reicht von -9007199254740992 bis 9007199254740992.

Sie können auch Hexadezimalzahlen definieren, indem Sie das Präfix \$ oder 0x benutzen:

```
a = $FF      ; a = 255
```

Auch exponentielle Notation ist möglich:

```
a = 2.5e5    ; a = 2.5 * 10^5 => a = 250000
```

Bei Fließkommazahlen zwischen -1 und 1 ist das Setzen der 0 (Null) freiwillig. So würde der folgende Code auch funktionieren:

```
a = .25 * 2 ; a = 0.5
```

Obwohl Hollywood nicht verschiedene Datentypen für ganze Zahlen und Fließkommazahlen hat, sollten nach den **Styleguide-Vorschlägen** Fließkommavariablen im Namen am Schluss ein Ausrufezeichen (!) enthalten:

```
a! = 3.14159265
```

Dies macht es leichter, Ihren Code zu lesen, weil Sie genau wissen, welche Variablen ganze Zahlen und welche Fließkommawerte beinhalten. Natürlich können Sie bei Variablen mit

Fließkommazahlen auf das Ausrufezeichen (!) verzichten. Aber die Benutzung wird empfohlen.

`True` und `False` sind Konstanten, welche auch zu den Zahlen gehören. Für weitere Details siehe `True/False`.

### 8.3 Zeichenketten

Zeichenketten (Zeichenfolgen, Strings) können dafür benutzt werden, um eine Abfolge von Zeichen oder Binärdaten in einer Variablen abzulegen. Standardmäßig wird Text in Zeichenketten mit der UTF-8-Zeichencodierung gespeichert, was bedeutet, dass bis zu 4 Bytes notwendig sein können, um ein Unicode-Zeichen zu speichern. Zeichenketten werden angegeben, indem man sie mit Anführungs- und Schlusszeichen begrenzt ("). Aufgrund der leichteren Lesbarkeit Ihres Codes sollten Sie im Variablennamen am Schluss ein Dollarzeichen \$ setzen. So kann man sie leichter von den Zahlenvariablen unterscheiden. Beispielsweise:

```
a$ = "Hello World!"
```

Natürlich kann das Dollarzeichen auch weggelassen werden:

```
a = "Hello World!"
```

Aber mit dem Dollarzeichen am Ende ist der Code lesbarer, weil wir wissen, dass `a` eine Zeichenkette ist.

Sie können Zeichenketten mit dem Operator `..` verketteten. Der obige Code könnte auch so geschrieben werden:

```
a$ = "Hello" .. " " .. "World!"
```

Dadurch werden drei Zeichenketten zu einer Zeichenketten zusammengefügt und in `a$` abgelegt. Siehe [Abschnitt 9.6 \[Zeichenketten verketteten\]](#), [Seite 115](#), für Details.

Wenn Ihre Zeichenkette ein Anführungs-/Schlusszeichen enthalten muss, können Sie diesen Escape Code benutzen: `\`"

```
; Das wird Hello, "Mr. John Doe" ausgegeben
DebugPrint("Hello, \"Mr. John Doe\"!")
```

Escape-Code wird immer nach einem umgekehrten Schrägstrich angegeben (`\`). Wenn Sie einen umgekehrten Schrägstrich in eine Zeichenkette schreiben müssen, benutzen Sie den umgekehrten Schrägstrich als Escape-Code: (`\\`). Die folgende Escape-Codes werden von Hollywood unterstützt:

```
\a    lässt den Systempiep/-ton hören
\b    Zeichen löschen links
\f    Seitenvorschub
\n    Neue Zeile
\r    Wagenrücklauf
\t    Horizontaler TAB
\v    Vertikaler TAB
\\    Umgekehrter Schrägstrich/Backslash
\"    Anführungs- oder Schlusszeichen
\'    Apostroph
\?    Fragezeichen
\[    Öffnende Eckige Klammer
```



```
\]    Schliessende Eckige Klammer
\xxx  Codepunkt
```

Die letzte Escape-Sequenz ermöglicht es Ihnen, Zeichen direkt einzufügen, indem Sie einfach ihren Codepunkt nach dem Backslash angeben. Der Codepunkt muss in der Dezimal-Notation angegeben werden und kann bis zu drei Ziffern belegen. Nur Latin 1 Codepunkte im Bereich von 0 bis 255 sind hier erlaubt. Jeder Wert größer als 255 wird nicht akzeptiert. Mit dieser Escape-Sequenz können Sie ein Nullzeichen in eine Zeichenkette einfügen:

```
a$ = "Hello\0World"
```

In vielen Programmiersprachen definiert ein Null-Zeichen das Ende einer Zeichenkette. Hollywood aber erlaubt Ihnen, so viele Null-Zeichen in eine Zeichenkette einzufügen, wie Sie wollen. Alle Befehle der Zeichenkettenbibliothek funktionieren so. Zum Beispiel würde dieser Code 11 zurückgeben:

```
DebugPrint(StrLen("Hello\0World"))
```

Aber das gilt nicht für Befehle, die Text ausgeben. Das folgende Beispiel wird wegen des Null-Zeichens "Hello" ausgeben:

```
; Dies wird "Hello" ausgegeben, weil ein Null-Zeichen das Ende der
; Zeichenkette definiert
DebugPrint("Hello\0World")
```

Wenn ein Neue-Zeile-Zeichen nach einem umgekehrten Schrägstrich folgt, wird Hollywood auch ein Neue-Zeile-Zeichen in die Zeichenkette einfügen und analysiert die Zeichenkette auf der nächsten Zeile weiter. Z.B. entstehen mit den beiden nächsten Variablendefinitionen die gleiche Zeichenkette:

```
a$ = "Hello\nWorld!"
a$ = "Hello\
World!"
```

Wenn Sie diese Eigenschaft benutzen wollen, vergewissern Sie sich, dass das Neue-Zeile-Zeichen nach dem umgekehrten Schrägstrich folgt. Es muss kein Leerzeichen/TAB zwischen dem umgekehrten Schrägstrich und dem Neue-Zeile-Zeichen haben.

Wenn Sie mehrere Zeilen Text haben, der in einer Zeichenkette zugewiesen wird, können Sie doppelte Eckige Klammern ([[]] und []) benutzen:

```
a$ = [[
<HTML>
<HEAD>
<TITLE>My HTML Page</TITLE>
</HEAD>
<BODY>
<A HREF="http://www.airsoftsoftwair.de/" TARGET="_NEW">
http://www.airsoftsoftwair.de/</A>
</BODY>
</HTML>
]]
```

Damit wird dasselbe erreicht wie mit diesem Code:

```
a$ = "<HTML>\n<HEAD>\n<TITLE>My HTML Page</TITLE>\n</HEAD>\n" ..
```

```
"<BODY>\n<A HREF=\"http://www.airsoftsoftwair.de/\" ..
" TARGET=\"_NEW\">http://www.airsoftsoftwair.de/</A>\n" ..
"</BODY>\n</HTML>\n"
```

Wie sie sehen können, ist die erste Version weitaus besser lesbar. Deshalb wird bei mehrfachen Zeilen Text empfohlen, die Version mit `[[...]]` zu benutzen. Folgt nach `[[` ein Neue-Zeile-Zeichen, wird dies ignoriert. Das Wagenrücklaufzeichen (`'\r'`) wird nicht in eine solch lange Zeichenkette eingefügt. Jeder Zeilenumbruch wird nur mit dem Neue-Zeile-Zeichen (`'\n'`) dargestellt. Ein weiterer Vorteil ist, dass Sie Anführungs-/Schlusszeichen direkt und ohne Escape-Code zwischen `[[...]]` verwenden können.

Sie können auch rohe Binärdaten in Zeichenketten speichern. Zum Beispiel kann der Befehl `DownloadFile()` verwendet werden, um eine Datei direkt in eine Zeichenkette herunterzuladen. Bei der Verwendung von binären Daten innerhalb von Zeichenketten müssen Sie beim Aufrufen von Befehlen der Zeichenkettenbibliothek vorsichtig sein. Befehle dieser Bibliothek erwarten normalerweise gültige UTF-8-Daten innerhalb der Zeichenkette, die an sie übergeben werden. Dies ist nicht der Fall, wenn Sie Zeichenketten als Container für rohe Binärdaten verwenden. Um Zeichenketten mit rohen Binärdaten zu erstellen, die auch mit den Befehlen der Zeichenkettenbibliothek arbeiten, müssen Sie diesen Befehlen explizit mitteilen, die Zeichenkettendaten nicht als UTF-8 zu interpretieren. Dies geschieht durch die Übergabe der Konstante `#ENCODING_RAW` im optionalen Codierungsparameter, die die meisten Befehle der Zeichenkettenbibliothek akzeptieren. Dann können diese Befehle auch mit Zeichenketten verwendet werden, die rohe Binärdaten enthalten. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Schließlich gibt es keine Begrenzung der Zeichenkettenlänge. Zeichenketten können so groß sein, wie es der Systemspeicher zulässt. Wenn Sie jedoch große Datenmengen in einer Zeichenkette speichern, sollten Sie etwas Vorsicht walten lassen und die Zeichenketten auf `Nil` setzen, wenn Sie sie nicht mehr benötigen. Somit weiss der `Speicherbereiniger`, dass diese Zeichenkette aus dem Speicher gelöscht werden kann. Betrachten Sie das folgende Beispiel:

```
data$ = DownloadFile("http://www.airsoftsoftwair.de/images/" ..
                    "products/hollywood/47_shot1.jpg")
...machen Sie etwas mit data$...
data$ = Nil
```

Dadurch wird die Datei unter der angegebenen URL heruntergeladen und als binäre Daten in `data$` gespeichert. Nachdem die Binärdaten in `data$` verarbeitet wurden, wird `data$` auf `Nil` gesetzt, um dem `Speicherbereiniger` mitzuteilen, dass er die Daten in `data$` aus dem Speicher löschen kann. Dies ist sehr wichtig, da ansonsten Ihr Skript ständig mehr Speicherplatz benötigt.

## 8.4 Tabellen

Eine Tabelle (Table) ist eine Sammlung von vielen Datenelementen, die von jeder Art sein können. Die Tabelle ist die universale Datenstruktur in Hollywood, welche in vielen Formen, z.B. als Array, Liste oder Record benutzt werden kann. Tabellen werden deklariert, indem man den Konstruktor verwendet. Folgender Code erstellt eine leere Tabelle:

```
a = {}
```

Eine leere Tabelle kann nicht verwendet werden, weil es keine Daten darin hat. Hollywood verlangt, dass alle Elemente einer Tabelle initialisiert werden müssen, bevor sie verwendet werden können. Wenn sie jetzt versuchen, auf ein leeres Element in der Tabelle zuzugreifen

```
b = a[0]
```

würde Ihnen eine Fehlermeldung mitteilen, dass Element 0 noch nicht initialisiert worden ist. Sie haben nur Zugang zu Elementen, die Sie vorher initialisiert haben. Die korrekte Version wäre deshalb:

```
a[0] = 5      ; weist 5 a[0] zu
b = a[0]      ; weist 5 b zu
```

Oder Sie könnten auch den Konstruktor benutzen, um die Tabelle zu initialisieren:

```
a = {5}      ; Kreiert eine Tabelle und weist die Zahl 5 dem Element 0 zu
b = a[0]      ; weist 5 b zu
```

Den Konstruktor können Sie auch verwenden, um eine Tabelle mit mehreren Elementen zu erzeugen. Er setzt die angegebenen Werte der Tabelle fest, der bei Element/Index 0 beginnt:

```
a = {1, 2, 4, 8, 16, 32, 64} ; erstellt eine Tabelle mit 7 Elementen
b = a[5]      ; weist b 32 zu (32 ist in Element 5 der Tabelle a abgelegt)
```

Um eine Tabelle mit einer spezifischen Größe zu definieren und initialisieren, können Sie die Anweisungen **Dim** und **DimStr** benutzen.

Es ist wichtig zu wissen, dass wenn Sie einer Variablen eine Tabelle zuweisen, die Variable nur eine Referenz/Assign zur Tabelle erhält. Es wurde keine unabhängige Kopie der Tabelle erzeugt:

```
a = {1, 2, 3, 4, 5}      ; erzeugt eine Tabelle mit 5 Elementen
b = a                    ; b erhält eine Referenz auf a
b[0] = 2                  ; legt in Element 0 den Wert 2 ab
DebugPrint(a[0], b[0])   ; wird "2 2" ausgegeben
```

Möchten Sie eine unabhängige Kopie einer Tabelle, können Sie den Befehl **CopyTable()** verwenden.

In Hollywood können Elemente nicht nur positive ganze Zahlen, sondern auch negative ganze Zahlen, Fließkommazahlen und Zeichenketten sein:

```
a      = {}      ; erstellt eine leere Tabelle
a[-5]   = 3      ; legt 3 in Element -5 ab
a[1.5]  = 2      ; legt 2 in Element 1.5 ab
```

Wenn Sie diese spezielle Initialisierung mit dem Konstruktor machen wollen, müssen Sie eckige Klammern benützen. Die drei Zeilen oben können auch so geschrieben werden:

```
a = {[-5] = 3, [1.5] = 2} ; erzeugt eine Tabelle mit zwei Elementen
```

Wollen Sie Zeichenketten als Index benutzen, können Sie die folgenden Anweisungen benutzen:

```
a      = {}      ; erstellt eine leere Tabelle
a["name"] = "John Doe" ; legt "John Doe" in Element "name" ab
a["age"]  = 20     ; legt "20" in Element "age" ab
a["sex"]  = "male"  ; legt "male" in Element "sex" ab
```

Ein leichter Weg, Zeichenketten als Index zu benutzen, wäre der Ausdruck mit Punkt (.):

```
a      = {}      ; erstellt eine leere Tabelle
```

```

a.name    = "John Doe"   ; legt "John Doe" in Element "name" ab
a.age     = 20           ; legt "20" in Element "age" ab
a.sex     = "male"       ; legt "male" in Element "sex" ab

```

Mit dem Konstruktor erreichen Sie dasselbe wie die beiden Beispiele vorher:

```
a = [{"name"} = "John Doe", [{"age"} = 20, [{"sex"} = "male"]}
```

Oder der leichtere Weg:

```
a = {name = "John Doe", age = 20, sex = "male"}
```

Auf die Elemente können Sie mit folgenden zwei Möglichkeiten zugreifen:

```

b = a["name"]
b = a.name

```

Beide Zeilen legen den gleichen Wert in **b** ab. Der übliche Weg ist die Methode mit dem Punkt. Hollywood unterscheidet nicht zwischen Groß- und Kleinschreibung. Deshalb würde auch `a.NAME` oder sogar `a.NaMe` funktionieren.

Es gibt jedoch eine Ausnahme: Wenn Sie Tabellenfelder mit eckigen Klammern initialisieren oder darauf zugreifen, unterscheidet Hollywood zwischen Groß- und Kleinbuchstaben. Weitere Details zu diesem Thema finden Sie in der Dokumentation des Befehls `RawGet()`. Siehe [Abschnitt 51.17 \[RawGet\]](#), [Seite 1130](#), für Details.

Einer Tabelle können sie Elemente hinzufügen, indem Sie ihnen einfach einen Wert zuweisen. Wollen Sie Elemente entfernen, dann weisen Sie ihm `Nil` zu. Das ist ein weiterer Vorteil von Hollywood gegenüber anderen Programmiersprachen. Tabellen werden nicht auf eine bestimmte Größe beschränkt.

In Tabellen können Sie die bisher genannten Elemente vermischen:

```
a = {x = 1, y = 2, 10, 11, 12, 13, z = 3, [6] = 16, 14, 15, obj="Cube"}
```

Dies erstellt eine neue Tabelle und initialisiert die Elemente 0 bis 6 mit den Zahlen 10 bis 16. Zusätzlich werden vier Elementen mit den Namen `x`, `y`, `z` und `obj` erstellt und mit 1, 2, 3 und "Cube" initialisiert.

Nun werfen wir einen Blick auf einige komplizierte Tabellenkonstruktionen für fortgeschrittene Programmierer. Als Anfänger wollen Sie vielleicht den folgenden Teil überspringen.

```

buts = { {x1 = 0, y1 = 0, x2 = 100, y2 = 50},
         {x1 = 100, y1 = 0, x2 = 80, y2 = 50},
         {x1 = 180, y1 = 0, x2 = 100, y2 = 50} }

```

Dieser Code erzeugt eine Tabelle mit dem Namen `buts` und initialisiert die ersten drei Elemente mit Start- und Endpunkt eines Knopfes (Button). Nun können wir folgenden Code benutzen, um jene drei Knöpfe zu erstellen:

```

For k = 0 To 2
    CreateButton(k + 1, buts[k].x1, buts[k].y1, buts[k].x2, buts[k].y2)
Next

```

Mehrdimensionale Tabellen sind auch kein Problem. Der folgende Code erstellt eine Matriz von der Größe 50x100 und initialisiert alle Elemente mit Null:

```

N = 50
M = 100
mtx = {} ; erstellt eine leere Tabelle
For i = 0 To N - 1

```

```

    mtx[i] = {}          ; erstellt eine neue Reihe in der Tabelle
    For j = 0 To M - 1
        mtx[i][j] = 0    ; initialisiert die Elemente mit 0
    Next
Next

```

Sie können auch die Anweisungen **Dim** und **DimStr** benutzen, um mehrdimensionale Tabellen zu erstellen.

Sie müssen keine Konstanten verwenden, wenn Sie eine Tabelle mit dem Konstruktor erstellen und initialisieren. Zum Beispiel:

```

s$ = "test"
i = 5
a = {[s$] = "An element", [i * 5 + 1] = "Another element"}

```

Dieser Code erstellt eine Tabelle mit dem Namen **a** und den beiden Elementen "test" (**a.test**) und 26 (**a[26]**). **a.test** wird mit "An element" und **a[26]** mit "Another element" initialisiert.

Seien Sie nicht verwirrt, wenn Sie so etwas sehen:

```

x = 5
y = 4
a = {x = x, y = y}      ; legt 5 in "x" und 4 in "y" ab

```

Diese Tabellendeklaration ist ein Unsinn. Es erstellt eine Tabelle mit den beiden Elementen **x** und **y**. Das Element **x** bekommt den Wert der Variable **x** und Element **y** den von der Variable **y**. Eine andere Möglichkeit, den oben erwähnten Code zu schreiben, wäre:

```

x = 5
y = 4
a = {}          ; erstellt eine leere Tabelle
a.x = x         ; legt 5 in a.x ab
a.y = y         ; legt 4 in a.y ab

```

Beide Codeschnipsel erstellen die gleiche Tabelle.

Schlussendlich können Sie auch Funktionen in einer Tabelle ablegen. Hier ist ein Beispiel:

```

a = {Add = Function(v1, v2) Return(v1 + v2) EndFunction,
     ShowBrush = DisplayBrush}
a.ShowBrush(1, #CENTER, #CENTER) ; ruft DisplayBrush() auf
b = a.Add(15, 16)                ; weist 31 b zu

```

Dieser Code erstellt eine Tabelle mit zwei Funktionen. Die erste Funktion fügt zwei Werte hinzu, die zweite bezieht sich einfach auf den Hollywood-Befehl **DisplayBrush()**. Sie könnten diesen Code auch auf folgende Weise schreiben:

```

a = [{"Add"} = Function(v1, v2) Return(v1 + v2) EndFunction,
     ["ShowBrush"] = DisplayBrush}
a["ShowBrush"](1, #CENTER, #CENTER) ; führt DisplayBrush() aus
b = a["Add"](15, 16)                ; legt 31 in b ab (15+16=31)

```

## 8.5 Funktionen

Ja, das ist richtig: Funktionen sind auch Teil unseres Datentypen-Kapitels. In Hollywood ist jede Funktion eine Variable. Das bedeutet, dass Sie sie einfach initialisieren können, Funktionen als Parameter an andere Funktionen weitergeben und sie können auch ein Rückgabewert

anderer Funktionen sein. Beachten Sie, dass die Befehle von Hollywood auch Funktionen sind. Hier ein Beispiel:

```
p_Print = Function(s) DebugPrint(s) EndFunction
```

ist nur eine weitere Möglichkeit für:

```
Function p_Print(s)
    DebugPrint(s)
EndFunction
```

Weil Funktionen Variablen sind, können Sie ihnen zum Beispiel auch neue Werte übertragen:

```
DebugPrint = Print
```

Jetzt werden alle Aufrufe des Befehls `DebugPrint()` wie der `Print()`-Befehl ausgeführt.

Es gibt eine Menge, die Sie über den Funktionsdatentypen wissen sollten. Deshalb hat es ein eigenes Kapitel in diesem Handbuch: Siehe [Abschnitt 12.1 \[Funktionen\]](#), [Seite 145](#), für Details.

## 8.6 Nil

Wenn Sie eine Variable benutzen, ohne ihr einen Wert zu übertragen, ist sie vom Typ `Nil` (Nichts). Konkret bedeutet das, dass sie nicht existiert. Hollywood behält nur Variablen, die einen Wert haben. Wenn Sie einer Funktion oder einem Befehl eine nicht initialisierte Variable übergeben oder mit einem Operator benutzen, wird sie automatisch zu Null konvertiert. Erwartet die Funktion oder der Befehl eine Zeichenkette, wird sie zu einer leeren Zeichenkette ("" ) konvertiert.

Wenn Sie eine Variable nicht mehr länger brauchen, können Sie sie auf `Nil` setzen und sie wird im nächsten Zyklus vom [Speicherbereiniger](#) gelöscht werden. So können sie auch eine ganze Tabelle löschen.

Seien Sie aber vorsichtig mit dem Vergleichen von Variablen mit `Nil`, weil `0=Nil` in Hollywood wahr (`True`) ist. Daher sind `IsNil()` und `GetType()` die einzige zuverlässigen Möglichkeiten, um herauszufinden, ob eine Variable wirklich `Nil` ist. Das einfache Überprüfen gegen `Nil` führt auch zu `True`, wenn die Variable 0 ist.

## 9 Ausdrücke und Operatoren

### 9.1 Übersicht

Ein Ausdruck ist eine Kombination von Operanden und Operatoren. Wenn mindestens ein Operand und ein Operator vorhanden ist, sprechen wir von einem Ausdruck. Hollywood muss Ausdrücke bewerten, bevor das Ergebnis an eine Funktion/einem Befehl übergeben wird. Ein Ausdruck kann konstant oder variabel sein, abhängig davon, ob es Variablen enthält oder nicht. Beispielsweise  $5 + 3$  ist ein Ausdruck. Die Operanden sind 5 sowie 3 und der Operator ist  $+$ .  $-1$  ist auch ein Ausdruck, weil wir einen Operanden und einen Operator haben. Normalerweise sind die Operatoren binär, was bedeutet, dass sie zwei Operanden benötigen. Aber es gibt Ausnahmen: Zum Beispiel ist der Negation-Operator ( $-$ ) einstellig und erfordert daher nur einen Operanden.

In Ausdrücken können Sie Klammern verwenden, um Hollywood mitzuteilen, was zuerst ausgewertet werden soll. In der folgenden Zeile

```
a = (3 + 4) * 5
```

wird Hollywood zunächst 3 und 4 addieren und dann das Ergebnis mit 5 multiplizieren. Wenn Sie die Klammern im obigen Code nicht setzen würden, dann multipliziert Hollywood zuerst 4 und 5 und wird erst dann 3 zum Ergebnis addieren, da der Multiplikationsoperator ( $*$ ) eine höhere Priorität besitzt als der Additionsoperator ( $+$ ). Siehe [Abschnitt 9.7 \[Prioritäten der Operatoren\]](#), Seite 116, für Details.

### 9.2 Arithmetische Operatoren / Rechenzeichen

Hollywood unterstützt die folgenden arithmetischen Operatoren:

BINÄR		
Operator	Beschreibung	Beispiel
$+$	Addition	$a + b$
$-$	Subtraktion	$a - b$
$*$	Multiplikation	$a * b$
$/$	Division	$a / b$
$\backslash$	Ganzzahldivision	$a \backslash b$
$\%$	Division mit Rest	$a \% b$
$^$	Potenz	$a ^ b$
UNÄR		
Operator	Beschreibung	Beispiel
$-$	Negierung	$-a$

Es sollte ziemlich selbsterklärend sein, wie diese Operatoren zu verwenden sind. So ist hier nur eine kurze Beschreibung der einzelnen Operatoren:

Addition:  $a + b$

Addiert a und b, z.B.  $5 + 3 = 8$

Subtraktion:  $a - b$

Subtrahiert  $b$  von  $a$ , z.B.  $10 - 5 = 5$

Multiplikation:  $a * b$

Multipliziert  $a$  mit  $b$ , z.B.  $10 * 8 = 80$

Division:  $a / b$

Ergibt eine genaue Aufteilung. Ergebnis könnte ein Fließkommawert sein, z.B.  $5 / 2 = 2.5$ .  $b$  darf nicht 0 sein

Ganzzahlen Division:  $a \setminus b$

Dividiert  $a$  durch  $b$ . Dezimalstellen werden gelöscht, z.B.  $5 \setminus 2 = 2$ .  $b$  darf nicht 0 sein

Ganzzahliger Rest der Division:  $a \% b$

Gibt den ganzzahligen Rest der Division zurück  $a \% b$ , z.B.  $5 \% 2 = 1$ .  $b$  darf nicht 0 sein

Potenzieren:  $a ^ b$

Potenziert  $a$  mit der Hochzahl  $b$ , z.B.  $2 ^ 8 = 256$

Negation:  $-a$

Negativiert  $a$ , z.B.  $-5 = 5$

## 9.3 Vergleichsoperatoren

Hollywood unterstützt folgende relationale Operatoren:

BINÄR		
Operator	Beschreibung	Beispiel
<code>=</code>	gleich	<code>a = b</code>
<code>&lt;&gt;</code>	ungleich	<code>a &lt;&gt; b</code>
<code>&lt;</code>	kleiner als	<code>a &lt; b</code>
<code>&gt;</code>	größer als	<code>a &gt; b</code>
<code>&lt;=</code>	kleiner gleich	<code>a &lt;= b</code>
<code>&gt;=</code>	größer gleich	<code>a &gt;= b</code>

Jeder der Operatoren vergleicht die Operanden  $a$  und  $b$  und gibt True zurück, wenn die Bedingung erfüllt ist und sonst False. Bitte beachten Sie, dass Sie nur die Werte des gleichen Typs vergleichen. Die automatische Nummer-/Zeichenkettenkonvertierung gilt hier nicht!

Das Gleichheitszeichen kann mit allen Arten verwendet werden, das heißt Sie können auch mit ihm Funktionen und Tabellen vergleichen. Die anderen Operatoren (`<>` `<=` `=>`) arbeiten nur mit Zahlen und Zeichenketten. Wenn Sie in Hollywood Zeichenketten miteinander vergleichen, wird dies alphabetisch erfolgen. Beispielsweise:

```
r = ("Hello" < "World")      -> True, weil H ist vor W im Alphabet
r = ("Commodore" < "Amiga") -> False, da C nach A im Alphabet folgt
```

Beachten Sie, dass aus Kompatibilitätsgründen das Vergleichen von Zeichenketten mit den relationalen Operatoren nur für ASCII-Zeichen unterstützt wird. Um Zeichenketten mit voller Unicode-Sortierung zu vergleichen, verwenden Sie stattdessen den Befehl `CompareStr()` Siehe [Abschnitt 57.15 \[CompareStr\]](#), [Seite 1262](#), für Details.



## 9.4 Logische Operatoren

Hollywood unterstützt folgende logische Operatoren:

BINÄR		
Operator	Beschreibung	Beispiel
And	Logisches UND	a And b
Or	Logisches ODER	a Or b

UNÄR		
Operator	Beschreibung	Beispiel
Not	Logisches NICHT	Not a

Die binären logischen Operatoren erlauben Ihnen, Entscheidungen zu treffen, die auf mehreren Bedingungen basieren. Jeder binäre logische Operator benötigt zwei Operanden, die verwendet werden, um das Ergebnis der logischen Bedingung auszuwerten. Alle Werte, die nicht 0, **Nil** oder eine leere Zeichenkette ("" ) sind, werden als True betrachtet.

Der Operator **Or** wird benutzt, um festzustellen, ob mindestens eine von zwei Bedingungen wahr ist. Der Operator **And** überprüft, ob beide Bedingungen wahr sind und der Operator **Not** ist mit einem logischen Minus-Vorzeichen vergleichbar. Er wandelt True in False und False in True um.

Die Operatoren **And** und **Or** verwenden Short-Cut-Auswertung. Das bedeutet, dass wenn der erste Operand bereits das Ergebnis definiert, wird der zweite Operand überhaupt nicht ausgewertet. Zum Beispiel wenn der erste Operand von einem **And**-Ausdruck False (Null) ist, dann wird der zweite Operand nicht gebraucht, weil der ganze Ausdruck sowieso nicht mehr True sein kann. Das gleiche gilt für einen **Or**-Ausdruck, wenn der erste Operand True (ungleich Null) ist. Dann wird der gesamte Ausdruck immer True sein - egal, welchen Wert der zweite Operand hat.

Bitte beachten Sie: **And** sowie **Or** geben die Konstante, nicht aber True (1) zurück, wenn sie wahr sind. **And** gibt den zweiten Operanden, wenn der Ausdruck wahr ist, **Or** gibt den ersten Operanden zurück, wenn es wahr ist. Beispielsweise:

```
a = 5 And 4      ; a = 4
a = 5 And 0      ; a = 0
a = 0 And 4      ; a = 0
b = 5 Or 4       ; b = 5
b = 5 Or 0       ; b = 5
b = 0 Or 4       ; b = 4
```

Der unäre **Not** Operator wird seine Operanden negieren. Das Ergebnis wird immer True (1) oder False (0) sein. Wenn **NOT** mit einer Zeichenkette verwendet wird, gibt der Ausdruck bei einer leeren Zeichenfolge ("" ) True zurück. Beispielsweise:

```
a = Not True     ; a = 0 (False)
a = Not False    ; a = 1 (True)
a = Not 5        ; a = 0 (False)
a = Not Not True ; a = 1 (True)
a = Not "Hello"  ; a = 0 (False)
```

```
a = Not ""      ; a = 1 (True)
```

Bitte beachten Sie: Der Not Operator hat eine hohe Priorität. Sie werden in den meisten Fällen Klammern benötigen. Zum Beispiel funktioniert das nicht:

```
If Not a = -1      ; falsch!
```

Hollywood wird es zu

```
If (Not a) = -1
```

übersetzen, weil der Not-Operator eine höhere Priorität als das Gleichheitszeichen hat! Aber offensichtlich macht diese Übersetzung keinen Sinn, weil das Ergebnis des Ausdrucks in Klammern (Not a) immer 0 oder 1 sein wird, aber nie -1. Deshalb müssen Sie Klammern um den Ausdruck mit der niedrigeren Priorität verwenden:

```
If Not (a = -1)      ; korrekt!
```

Siehe [Abschnitt 9.7 \[Prioritäten der Operatoren\]](#), [Seite 116](#), für Details.

## 9.5 Bit-Operatoren

Hollywood unterstützt die folgenden Bit-Operatoren:

BINÄR		
Operator	Beschreibung	Beispiel
<<	Linksverschiebung (logisch)	a << b
>>	Rechtsverschiebung (logisch)	a >> b
&	Bitweises UND	a & b
~	Bitweises exklusives ODER	a ~ b
	Bitweises ODER	a   b

UNÄR		
Operator	Beschreibung	Beispiel
~	Bitweise Negierung	~a

Mit den Bit-Operatoren können Sie mit Ausdrücken auf Bit Ebene arbeiten. Diese Vorgänge sind alle beschränkt auf 32-Bit Werte. Hier ist eine Beschreibung der Bit Operatoren:

Der Linksverschiebungs-Operator (<<) verschiebt alle Bits des a-Operanden b mal nach links. Die durch diesen Vorgang erzeugten Bitlöcher auf der rechten Seite der Zahl werden mit Nullen aufgefüllt. b darf nicht negativ sein. Eine Zahl a x-mal nach links verschieben ist dasselbe, wie diese Zahl mit  $2^x$  zu multiplizieren. Aber natürlich ist die Verschiebung viel schneller als die Multiplikation. Beispiele:

```
7 << 1 = %111 << 1 = %1110 = 14 (7 * 2^1 = 14)
```

```
256 << 4 = %100000000 << 4 = %1000000000000 = 4096 (256*2^4=4096)
```

Der Rechtsverschiebungs-Operator (>>) verschiebt alle Bits des a-Operanden b mal nach rechts. Diesmal werden die Bitlöcher auf der linken Seite der Zahl mit Nullen aufgefüllt. Wenn Sie eine arithmetische Verschiebung benötigen (Bitlöcher werden mit dem höchstwertigen Bit aufgefüllt), nutzen Sie bitte stattdessen den Befehl **Sar()**. b darf auch hier nicht negativ sein. Eine Zahl x-mal nach rechts verschieben ist gleich, wie diese Zahl

$2^x$  zu dividieren. Aber natürlich ist die Verschiebung viel schneller als dividieren, wenn eine Ganzzahlergebnis präzise genug für Ihre Zwecke ist. Hier sind einige Beispiele:

```
65 >> 1 = %1000001 >> 1 = %100000 = 32 (65\2^1=32)
256 >> 4 = %100000000 >> 4 = %10000 = 16 (256\2^4=16)
```

Die bitweise Operatoren **And**, **Xor** und **Or** sind im Grunde dieselben wie die logischen **And**- / **Xor**- / **Or**-Operatoren mit dem Unterschied, dass **&**, **~** und **|** auf Bit Ebene arbeiten. Das heißt, sie vergleichen alle 32-Bit-Operanden **a** und **b** und setzen die Bits in der Ergebnismenge nach diesem Vergleich. Der Operator **And** wird das Bit zurückgeben, wenn beide Bits in den Operanden **a** und **b** auf dieser Position festgelegt sind. Der **Xor**-Operator wird das Bit im Rückgabewert setzen, wenn eines der beiden Bits 1 ist, aber nicht, wenn beide 1 haben. Der **Or**-Operator setzt dann ein Bit im Rückgabewert, wenn eine oder beide der Operanden das Bit auf derselben Position gesetzt ist. Eine Tabelle:

Bit 1	Bit 2	Bit1 & Bit2	Bit1 ~ Bit2	Bit1   Bit2
1	1	1	0	1
1	0	0	1	1
0	1	0	1	1
0	0	0	0	0

Beispiele:

```
%10011001 & %11101000 = %10001000 ; Bitweise And
%10011001 ~ %11101000 = %01110001 ; Bitweise Xor
%10011001 | %11101000 = %11111001 ; Bitweise Or
```

Der unäre Negation-Operator (**~**) wird eine bitweise Inversion auf die Zahl verwenden. Alle Bits werden invertiert. Bitte beachten Sie, dass der Wert immer vor der Inversion in ein 32-Bit Ganzzahl umgewandelt wird. So könnte man eine Menge von führenden Einsen bekommen. Beispielsweise:

```
~%00000000 = %11111111111111111111111111111111
~%10111001 = %111111111111111111111111111101000110
```

Um das richtige Resultat mit diesen nicht eingeplanten 1er zu erhalten, verwenden Sie einfach zusätzlich den bitweise **And**-Operator (**&**) auf den resultierenden Wert. Zum Beispiel, wenn Sie nur 8 Bits wie im obigen Beispiel umgekehrt haben wollen, verwenden Sie das bitweise **And** mit 255:

```
~%00000000 & %11111111 = %11111111
~%10111001 & %11111111 = %01000110
```

## 9.6 Zeichenketten verketteten

BINÄR		
Operator	Beschreibung	Beispiel
..	Verkettung	a .. b

Der Zeichenketten-Verkettungsoperator kann verwendet werden, um zwei Zeichenketten in eine neue zu verketteten. Weil Hollywood automatische **Zeichen- und Zahlenumwandlung**

bietet, können Sie sogar zwei Nummern verketteten. Das Ergebnis wird aber immer eine Zeichenkette sein. Beispiele:

```
DebugPrint("Hello" .. " World")    ; gibt "Hello World" aus
DebugPrint(5 .. " + " .. 5 .. " = " .. 10)    ; gibt "5 + 5 = 10" aus
```

Dieser Operator ist auch nützlich, wenn Sie eine Zeichenkette über mehrere Zeilen verteilen. Beispielsweise:

```
DebugPrint("My Program v1.0\n" ..
           "(c) by me 2005\n" ..
           "Press RETURN to start!")
```

## 9.7 Prioritäten der Operatoren

Dies ist die komplette Liste aller verfügbaren Operatoren und ihren Prioritäten. Sie müssen dies nicht auswendig wissen. Im Zweifelsfall verwenden Sie Klammern. Es tut nicht weh und macht Ihr Programm besser lesbar, weil nicht jeder weiß, dass der Linksverschiebungs-Operator eine höhere Priorität als der Bit-Or-Operator hat.

Priorität	Operator	Beschreibung
12	$\wedge$	Potenz
11	$-$	Negierung
11	$\sim$	Bitweise Negierung
11	Not	Logisches NICHT
10	$*$	Multiplikation
10	$/$	Division
10	$\backslash$	Ganzzahldivision
10	$\%$	Division mit Rest
9	$+$	Addition
9	$-$	Subtraktion
8	$\ll$	Linksverschiebung
8	$\gg$	Rechtsverschiebung
7	$\&$	Bitweises UND
6	$\sim$	Bitweises exklusives ODER
5	$ $	Bitweises ODER
4	$..$	Verkettung
3	$=$	Gleichheit
3	$<>$	Ungleichheit
3	$<$	Kleiner als
3	$>$	Größer als
3	$<=$	Kleiner gleich
3	$>=$	Größer gleich
2	And	Logisches UND
1	Or	Logisches ODER

## 9.8 Metamethoden

Metamethoden können verwendet werden, um zu definieren, wie Hollywoods Operatoren sich mit Tabellen verhalten soll. Normalerweise können Sie Hollywoods Operatoren nicht mit Tabellen verwenden. Beispielsweise ist die folgendes nicht möglich:

```
table_A = {1, 2, 3, 4, 5}
table_B = {5, 4, 3, 2, 1}
result = table_A + table_B    ; erzeugt Compilerfehler!
```

Der obige Code versucht `table_A` mit `table_B` zu addieren, aber das funktioniert nicht, weil Tabellen Zufallsdaten enthalten können (Funktionen, Untertabellen, Zeichenketten, etc.). So gibt es keine Möglichkeit zu sagen, wie der Add-Operator sich bei einer Tabelle verhalten soll. Dies ist der Moment, wo Metamethoden ins Spiel kommen. Mit Metamethoden können Sie festlegen, wie ein Operator sich zu verhalten hat, wenn er eine Tabelle als Operanden enthält. Mit anderen Worten, mit Metamethoden können Sie eine Funktion definieren,

die ausgeführt wird, wenn ein Operator mit einer Tabelle verwendet wird. Diese Funktion berechnet dann das Ergebnis.

Metamethoden sind keine globalen Einstellungen, sie sind für jede Tabelle privat. Wenn Sie eine Tabelle erstellen, wird sie keine Metamethoden haben. Versuchen Sie einen Operator auf dieser Tabelle zu verwenden, wird er fehlschlagen. Für das Zuweisen einer Metamethode an einer Tabelle müssen Sie den Befehl `SetMetaTable()` verwenden. Ein Metatable ist eine Tabelle, die einen Satz von Metamethoden enthält. `SetMetaTable()` akzeptiert zwei Argumente: Das erste Argument ist der Name der Tabelle, der Sie eine Metamethode zuweisen wollen, und das zweite Argument ist die eigentliche Metatable, d.h. die Tabelle, die die Metamethoden enthält.

Lassen Sie uns nun ein Beispiel anschauen. Wir werden den Code von oben so mit Metamethoden neu schreiben, dass wir die beiden Tabellen addieren können.

```
mt = {} ; erstellt unsere Metatable

Function mt.__add(a, b)
    Local sizeA = ListItems(a) ; Anzahl Elemente in Tabelle A
    Local sizeB = ListItems(b) ; Anzahl Elemente in Tabelle B
    Local result = {}          ; erstellt resultierende Tabelle

    For Local k = 0 To Min(sizeA, sizeB) - 1
        result[k] = a[k] + b[k] ; zufügen der addierten Elemente
    Next

    Return(result) ; resultierende Tabelle zurückgeben
EndFunction

table_A = {1, 2, 3, 4, 5}
table_B = {5, 4, 3, 2, 1}

SetMetaTable(table_A, mt) ; weist "mt" der Tabelle table_A...
                           ; ...als Metatable zu

result = table_A + table_B
```

Die resultierende Tabelle wird fünf Elemente aufweisen, die alle auf 6 gesetzt sind. Nun schauen wir, was der Code oben gemacht hat. Zuerst erstellen wir eine leere Tabelle, die für uns als Metatable dient. Dann fügen wir eine Funktion namens `__add` (zwei Unterstriche verwenden) an diese Tabelle. Diese Funktion wird die Metamethode für den `+` Operator. Beachten Sie, dass wir den Namen `__add` für diese Funktion verwenden müssen, weil Hollywood anhand des Funktionsnamen den Operator erkennt, die die Metamethode verkörpert. Mit `__add` als Name wird eine Metamethode für den Add-Operator (`+`) definiert. Der Code in unserer Metamethode berechnet die Länge der beiden Tabellen, addiert zwei Tabellenelemente und speichert sie in einer Ergebnistabelle, die sie zurückgibt.

Beachten Sie, dass die Umsetzung unserer `__add` Metamethode oben erfordert, dass beide Argumente Tabellen sind. Und die Tabellen dürfen nur Zahlen enthalten (oder Zeichenketten, die in Zahlen umgewandelt werden können). Z.B. die folgenden Ausdrücke würden mit der oben definierten Metamethode nicht funktionieren:

```
result = table_A + 10      ; --> Fehler, weil "10" ist keine Tabelle
result = table_A + "Hello" ; --> gleicher Fehler
```

Natürlich ist es möglich, Metamethoden zu schreiben, die diese Situationen umgehen kann. Sie müssen nur die Typen der Parameter überprüfen, die auf Ihre Metamethode übergeben werden. Dann können Sie abhängig vom Variablentyp benutzerdefinierte Aktionen ausführen.

Jetzt haben wir die Metamethode für den Add-Operator (+) kennengelernt. Natürlich können Sie eine Metamethode für jeden anderen Hollywood-Operator anwenden. Sie können auch Metamethoden für alle relationalen Operatoren erstellen (= <> < <=> =), so dass Sie Tabellen direkt untereinander vergleichen können. Alles was Sie wissen müssen, ist der richtige Name für den Operator der Metamethode, so dass Sie ihn installieren können. Hier ist eine Liste aller verfügbaren Metamethoden und die entsprechenden Operatoren:

Metamethode	Operator	Beschreibung
__pow	^	Potenz
__unm	-	Negierung
__not	~	Bitweise Negierung
__mul	*	Multiplikation
__div	/	Division
__divint	\	Ganzzahldivision
__mod	%	Division mit Rest
__add	+	Addition
__sub	-	Subtraktion
__lsh	<<	Linksverschiebung
__rsh	>>	Rechtsverschiebung
__and	&	Bitweises UND
__xor	~	Bitweises exklusives ODER
__or		Bitweises ODER
__concat	..	Verkettung
__eq	=	Gleichheit
__lt	<	Kleiner als
__le	<=	Kleiner gleich
__index	[]	Wert aus Table lesen
__newindex	[]=	Wert in Table schreiben
__call	()	Table aufrufen

Wie Sie sehen können, gibt es keine Metamethoden für die >, >= und <> Operatoren. Dies liegt daran, weil Hollywood durch neu formulierte Bedingungen der bekannten Operatoren diese Resultate auf folgendem Weg liefern kann:

```
a <> b      ist das gleiche wie      Not (a = b)
a > b       ist das gleiche wie      b < a
a >= b      ist das gleiche wie      b <= a
```

Wenn Sie zwei Tabellen miteinander vergleichen möchten, die beide mit Metamethoden verknüpft sind, ohne die `__eq` Metamethode aufrufen, müssen Sie den Befehl `RawEqual()` verwenden. Dieser Befehl wird beide Tabellen nur als Referenz verglichen, ohne die Metamethoden aufzurufen.

### 9.8.1 Abweichende Metatabellen mit Binäroperatoren

Wie Sie sehen können, hat jede Tabelle seine eigene private Metatabelleeinstellung. Werden binäre Operatoren verwendet, kann es vorkommen, dass die beiden Operanden nicht die gleichen Metatabelle verwenden. Wie wählt jetzt Hollywood die Metatabelle für die Operanden aus? Dies hängt von verschiedenen Bedingungen ab:

- Wenn der Operator ein Vergleichsoperator (`=` `<>` ist `<=>` `=`), wird die Metamethode nur dann aufgerufen, wenn die beiden genannten Tabellen die gleiche Metatabelle verwenden. Wenn sie unterschiedlich Metatabellen haben, wird der Vergleich fehlschlagen.
- Wenn der Operator eine arithmetischer Operator, ein bitweise Operator oder der Zeichenfolgeverkettungsoperator (`..`) ist, wird Hollywood zunächst in Operand A nach einer Metatabelle sehen. Wenn Operand A eine Metatabelle hat, dann wird diese Metatabelle verwendet. Wenn Operand A keine Metatabelle hat, wird Hollywood in Operand B nachschauen. Wenn Operand B eine Metatabelle besitzt, wird sie verwendet. Wenn keine der Operanden eine Metatabelle vorweisen kann, wird eine Fehlermeldung ausgegeben.

### 9.8.2 Einschränkungen der relationalen Metamethoden

Sie haben oben bereits gelesen, dass die relationale Metamethoden nur dann funktionieren, wenn die beiden Operanden die gleiche Metatabelle verwenden. Jedoch gibt es eine weitere Einschränkung bei der Verwendung von relationalen Metamethoden: Sie werden nur dann aufgerufen, wenn die zwei Operanden Tabellen sind. Es ist nicht möglich, eine Tabelle mit einer Nummer oder einer Zeichenfolge mit einer Tabelle zu vergleichen etc. Die arithmetischen und bitweise Metamethoden können mit jedem Variablentyp arbeiten, aber die relationale Metamethoden sind auf das Vergleichen von Tabellen beschränkt.

### 9.8.3 Erweiterte Metamethoden

Bisher haben wir nur die relationalen, arithmetischen, bitweise und verkettete Metamethoden angeschaut. Es gibt jedoch ein paar mehr Metamethoden, die Sie verwenden können, nämlich `__index`, `__newindex`, `__call` und `__tostring`. Hier ist eine detaillierte Beschreibung dieser Metamethoden:

**\_\_index:** Diese Metamethode wird aufgerufen, wenn Sie versuchen, einen Tabellenindex zu lesen, der nicht existiert. Diese Metamethode könnte verwendet werden, um einen Standardwert für alle nicht initialisierten Tabellenfelder zu erstellen. Normalerweise wird Hollywood fehlschlagen, wenn Sie von nicht initialisierten Feldern lesen. Dieses Verhalten könnte geändert werden, wenn Sie diese Metamethode verwenden. Hier ist ein Codeschnipsel, der den Standardwert auf 0 setzt:

```
mt = {}
Function mt.__index(t, idx)
    Return(0)
```



```
EndFunction
```

```
t = {x = 10, y = 20}
SetMetaTable(t, mt)
NPrint(t.x, t.y, t.z) ; --> gibt 10 20 0 aus
```

Ohne unsere Metatabelle würde der Aufruf von `NPrint()` scheitern, weil `z` wurde nicht initialisiert. Durch die Verwendung der Metatabelle jedoch wird `z` wieder automatisch auf 0 gesetzt, weil es nicht existiert.

Manchmal kann es notwendig werden, aus einer Tabelle ohne Metamethoden zu lesen. Sie können dies mit dem `RawGet()` Befehl tun. `RawGet()` wird nie eine Metamethode aufrufen. Wenn ein Index nicht vorhanden ist, wird `Nil` zurückgegeben.

#### `__newindex:`

Diese Metamethode wird aufgerufen, wenn Sie versuchen, einen Wert auf einen Tabellenindex zu schreiben, der noch nicht in der Tabelle vorhanden ist. Sie könnten diese Metamethode zum Beispiel verwenden, um nur lesbare Tabellen zu erstellen. Der folgende Code wird ein Fehler ausgegeben, wenn Sie versuchen, in eine nur lesbare Tabelle zu schreiben:

```
mt = {}
Function mt.__newindex(t, idx, val)
    NPrint("Blocked writing", val, "at index", idx)
EndFunction

t = {x = 10, y = 20}
SetMetaTable(t, mt)
t.z = 45 ; --> Blockiert Schreib 45 bei Index z
```

Der obige Code setzt die Tabelle `t` unter Schreibschutz. Sie können keine Änderungen an der Tabelle vornehmen.

Manchmal kann es notwendig werden, in eine Tabelle ohne Metamethoden zu schreiben. Sie können dies mit dem `RawGet()` Befehl tun. `RawGet()` wird nie eine Metamethode aufrufen. Man könnte sogar mit dem `RawSet()` Befehl in schreibgeschützte Tabellen schreiben.

`__call:` Diese Metamethode wird aufgerufen, wenn Sie versuchen, eine Tabelle aufzurufen. Normalerweise schlägt der Versuch fehl, weil Tabellen offensichtlich nur Typen von Datenspeicher und keine Funktionen sind. Es gibt jedoch Fälle, in denen es nützlich sein kann, wenn Sie auch eine Tabelle aufrufen könnten. Das folgende Beispiel zeigt eine Metamethode, die den Durchschnitt aller Tabellenwerte berechnet:

```
mt = {}
Function mt.__call(t)
    Local c = ListItems(t)
    Local sum = 0

    For Local k = 0 To c - 1 Do sum = sum + t[k]
```

```

        Return(sum / c)
    EndFunction

    t = {10, 23, 45, 5, 107, 45, 18, 46}
    SetMetaTable(t, mt)
    NPrint(t())    ; --> 37.375

```

Der obige Code gibt 37,375 zurück, welcher der Mittelwert der acht gespeicherten Werte in der Tabelle `t` ist.

#### `__toString:`

Diese Metamethode wird durch Befehle wie `Print()` oder `DebugPrint()` verwendet. Normalerweise, wenn Sie eine Tabelle an `Print()` übergeben, werden Sie so etwas wie "Tabelle: 1acd432f" erhalten. Hollywood handhabt dies intern so, um auf die Tabelle zu verweisen und ist offensichtlich für Sie nicht sinnvoll. Jedoch mit der Metamethode `__toString` können Sie dieses Verhalten leicht ändern. Hier ist eine Metamethode, die eine Zeichenkettendarstellung einer Tabelle erstellt:

```

    mt = {}
    Function mt.__toString(t)
        Local r$
        For Local k=0 To ListItems(t)-1 Do r$=r$..t[k].." "
        Return(r$)
    EndFunction

    t = {"Jeff", "Andy", "Mike", "Dave"}
    SetMetaTable(t, mt)
    NPrint(t)    ; --> Jeff Andy Mike Dave

```

Der obige Code gibt "Jeff Andy Mike Dave" aus, weil unsere `__toString` Metamethode einfach alle Elemente der Tabelle verkettet hat.

## 10 Variablen und Konstanten

### 10.1 Variablen und Konstanten

Eine Variable kann verwendet werden, um einen Teil der Daten unter einem gegebenen Namen zu speichern. In Hollywood müssen Variablen nicht deklariert werden. Das bedeutet, dass Sie einfach einen Wert einer Variablen zuweisen, ohne zunächst den Typ für die Variable zu definieren. Hollywood wird dies automatisch tun. Der Name der Variable muss mit einem Zeichen aus dem englischen Alphabet (a-z) oder einem Unterstrich (\_) beginnen. Nach dem ersten Zeichen können Sie auch Zahlen, das Ausrufezeichen (!) und das Dollarzeichen (\$) verwenden. Es wird vorgeschlagen, dass Sie das Dollarzeichen nur in Variablen vom Typ Zeichenkette anwenden. Hollywood achtet nicht auf Groß- und Kleinschreibung bei den Variablennamen. Dies bedeutet, dass beispielsweise die Namen "MYVAR" und "myvar" auf die gleiche Variable verweisen. Hollywood ist eine dynamisch typisierte Sprache, was bedeutet, dass auch Variablen dynamisch sind. Zum Beispiel führt folgender Codeschnipsel zu keiner Fehlermeldung:

```
myvar = 1.5
myvar = "Hello"
myvar = {1, 2, 3}
myvar = Function(s) DebugPrint(s) EndFunction
```

Sie können den Typ einer Variablen jederzeit ändern. Aber das ist kein guter Programmierstil!

Konstanten sind festgelegte Werte, die überall in Ihrem Skript global verfügbar sind. Sie sind mit einem vorangestellten Hash-Zeichen (#) im Name versehen, so dass Sie sie von Variablen und Funktionen unterscheiden können.

### 10.2 Globale Variablen

Wenn Sie zum ersten Mal einen Wert einer Variablen zuweisen, dann wird diese Variable automatisch global, falls Sie Hollywood nicht explizit sagen, dass sie unter Verwendung der Anweisung **Local** lokal sein soll. Auf globale Variablen kann von jedem beliebigen Ort in Ihrem Skript zugegriffen werden. Sie sind auch für alle Funktionen global verfügbar. Wenn jedoch eine lokale Variable mit dem gleichen Namen wie eine globale Variable hat, dann wird Hollywood immer diese lokale Variable zuerst berücksichtigen.

Globale sind langsamer als lokale Variablen und sie können nur mit dem **Speicherbereiniger** gelöscht werden, wenn Sie die Variable ausdrücklich auf **Nil** setzen. Daher sollten Sie nur Global verwenden, wenn es wirklich notwendig ist. In Funktionen sollten Sie versuchen, nur mit lokalen Variablen zu arbeiten.

Hier ist ein Beispiel:

```
; schlechter Code!
Function p_Add(a, b)
    tmp = a + b
    Return(tmp)
EndFunction
```

Die Variable `tmp` wird als globale Variable erstellt. Das macht hier nicht viel Sinn, weil Sie die Variable `tmp` nur in dieser Funktion benötigen. So sollten Sie besser diese Variable in der Funktion lokal deklarieren, z.B:

```
; guter Code!
Function p_Add(a, b)
    Local tmp = a + b
    Return(tmp)
EndFunction
```

Um die Lesbarkeit des Programms zu verbessern, können Sie die **Global**-Anweisung verwenden, um bestimmte Variablen eindeutig als Global zu markieren. Das ist natürlich optional, da alle nicht explizit als lokal deklarierten Variablen automatisch global werden. Aber bei Verwendung von **Global** wird Ihr Programm besser lesbar.

Siehe [Abschnitt 10.4 \[Lokale Variablen\]](#), Seite 124, für Details.

## 10.3 Global-Anweisung

```
Global <var1> [, <var2>, ...] [= <expr1> [, <expr2>, ...]]
```

Die **Global**-Anweisung wird verwendet, um Hollywood mitzuteilen, dass die angegebene Variable global sein soll. Darüber hinaus kann es auch Ihre Variable initialisieren. Diese Anweisung ist nur enthalten, um die Lesbarkeit des Programms zu verbessern. Man könnte sie auch weglassen und der Code würde auf die gleiche Weise arbeiten. Die **Global**-Anweisung funktioniert genau wie die **Local**-Anweisung.

Wenn Sie die **Global**-Anweisung verwenden, ist es ratsam, dass Sie alle Anweisungen am Anfang des Codes schreiben. So kann jeder deutlich sehen, welche Variablen global zur Verfügung stehen. Globale Variablen an anderer Stelle in Ihrem Code zu benutzen, ist nicht ratsam und kann beim Lesen ziemlich verwirrend sein.

Siehe [Abschnitt 10.2 \[Globale Variablen\]](#), Seite 123, für Details.

## 10.4 Lokale Variablen

Sie sollten wann immer und wo immer möglich lokale Variablen verwenden. Sie verbessern die Speicherverwaltung des Programms, da der **Speicherbereiniger** automatisch weiß, wann er sie löschen kann. Darüber hinaus ist der Zugriff auf lokale Variablen viel schneller als auf globale, weil Hollywood nicht den gesamten globalen Code durchsuchen muss und schließlich erhöht es die Lesbarkeit des Programms.

Lokale Variablen haben eine begrenzte Lebensdauer. Sie werden nur in dem Block zur Verfügung stehen, wo man sie deklariert hat. Ein Block wird in der Regel eine Funktion oder Steuerungsstruktur sein. Sie können sogar Blöcke definieren, indem Sie **Block** und **EndBlock** verwenden.

Wichtig: Lokale Variablen müssen explizit deklariert werden. Wenn Sie dies nicht tun, wird die Variable automatisch global sein. Zum Beispiel, wenn Sie

```
Function p_Add(a, b)
    tmp = a + b
    Return(tmp)
EndFunction
```

schreiben, wird `tmp` automatisch als globale Variable erstellt werden. Aber das ist nur eine Verschwendung von Ressourcen, weil Sie die Variable nur innerhalb der Funktion benötigen. So sollten Sie besser schreiben:

```
Function p_Add(a, b)
    Local tmp = a + b
    Return(tmp)
EndFunction
```

Nun ist `tmp` explizit als lokale Variable deklariert und wird gelöscht, wenn die Funktion `p_Add()` beendet ist.

Wie Sie bereits gesehen haben, werden lokale Variablen mit der **Local**-Anweisung deklariert. Um eine lokale Variable zu deklarieren, setzen Sie einfach die Anweisung **Local** vor die Deklaration:

```
a = 10          ; Globale Variable
Local b = 10    ; Lokale Variable
```

Wenn Sie mehrere Variablen mit einer **Local**-Anweisung initialisieren wollen, verwenden Sie einfach Kommas, wie Sie es bei einer normalen Zuordnung tun würden:

```
a, b = 10, 5      ; Globale Variablen! a erhält 10, b bekommt 5
Local x, y = 10, 5 ; Lokale Variablen! x bekommt 10, y erhält 5
```

Sobald Sie eine Variable als lokal deklariert haben, müssen Sie die **Local**-Anweisung auf diese Variable nicht mehr verwenden oder wiederholen:

```
; if x > 10, multipliziere mit 2, sonst dividiere mit 2
If x > 10
    Local tmp = x ; deklariert die lokale Variable "tmp"
    tmp = tmp * 2 ; multipliziert die lokale variable "tmp" mit 2
    x = tmp       ; Wert der lokalen Variable "tmp" wird "x" zugewiesen
Else
    Local tmp = x ; deklariert die lokale Variable "tmp"
    tmp = tmp / 2 ; dividiert die lokale Variable "tmp" durch 2
    x = tmp       ; Wert der lokalen Variable "tmp" wird "x" zugewiesen
EndIf
```

```
Print(tmp)          ; hier wird 0 ausgegeben, weil "tmp" ist weg
```

Der obige Code erzeugt die lokale Variable `tmp` in den beiden Blöcken der **If**-Anweisung. Danach wird sie multipliziert mit oder dividiert durch 2. Die Anweisung **Local** ist da nicht mehr erforderlich, da Hollywood bereits an dieser Stelle weiß, dass `tmp` eine lokale Variable ist. `tmp` wird am Ende des **If**-Blocks gelöscht, so dass sie in der Zeile `Print(tmp)` nicht mehr da ist. `tmp` wird am Ende des **If**-Blocks zu **Nil**.

Wenn Sie einer lokalen Variablen keinen Wert zuweisen, wird es den Wert **Nil** erhalten, aber Hollywood wird wissen, dass es sich um eine lokale Variable handelt, z.B.:

```
If True          ; Block wird immer betreten
    Local x       ; deklarieren der lokalen Variable x
    Print(x)      ; gibt 0 aus, da x Nil ist
    x = 5         ; weist lokalem x 5 zu
    Print(x)      ; gibt jetzt 5 aus
```

```
EndIf          ; x wird hier gelöscht
```

```
Print(x)       ; gibt 0 aus, weil x wieder Nil ist
```

Sie können auch den Namen einer globalen Variablen (oder einer lokalen Variable des oberen Blocks) für eine neue lokale Variable verwenden. Zum Beispiel:

```
a = 50          ; deklariert a als global und weist ihr 50 zu
Block           ; begrenzt die nächsten beiden Zeilen als Block
  Local a = 40   ; deklariert die lokale Variable "a", weist ihr 40 zu
  NPrint(a)      ; gibt 40 aus
EndBlock        ; lokale Variable "a" wird hier gelöscht
```

```
NPrint(a)       ; gibt 50 aus
```

Ein komplexeres Beispiel, die viele Variablen mit dem gleichen Namen verwendet:

```
x = 10          ; global x (x1 = 10)
Block           ; neuer Block
  Local x = x + 1 ; weist 11 der lokalen Variable x zu (x2 = x1 + 1)
  Block         ; neuer Block
    Local x = x + 1 ; weist 12 der lokalen Variable x zu (x3 = x2 + 1)
    Block       ; neuer Block
      Local x = x + 1 ; weist 13 der lokalen Variable x zu (x4 = x3 + 1)
      NPrint(x)      ; gibt 13 aus (= x4)
    EndBlock       ; der Bereich von x4 endet hier
  NPrint(x)        ; gibt 12 aus (= x3)
EndBlock          ; der Bereich von x3 endet hier
NPrint(x)         ; gibt 11 aus (= x2)
EndBlock          ; der Bereich von x2 endet hier
NPrint(x)         ; gibt 10 aus (= x1)
```

Dieser Code könnte ein wenig verwirrend aussehen, aber er macht Sinn. In jedem neuen Block wird die neue lokale Variable `x` ausgehend vom aktuellen Rahmen durch die Variable `x` des vorhergehenden Block mit `+1` initialisiert.

Es ist zu beachten, dass Sie lokale Variablen nicht zusammen mit `Gosub()` verwenden können, weil Hollywood bei einer `Gosub()`-Anweisung aus dem Block springen wird. Bei der Rückkehr haben die Variablen völlig verschiedene Daten. So arbeitet der folgende Code nicht:

```
;Ungültiger Code
Block
  Local a = 50      ; erstellt lokale Variable "a"
  Gosub(SUBROUTINE) ; Sprung aus dem Block; "a" wird weg geworfen
  Print(a)          ; lokal "a" hat jetzt irgendein zufälliger Wert
EndBlock
```

Dies sollte kein großes Problem darstellen, da `Gosub()` ohnehin seit Hollywood 2.0 veraltet ist und in Ihrem Code nicht mehr verwendet werden sollte. Benutzen Sie nun die Anweisung `Function.`)

Sie können auch lokale Funktionen nutzen. Sie arbeiten fast in der gleichen Weise. Sie sollten lokale den globalen Funktionen vorziehen, obwohl Sie sie höchstwahrscheinlich nicht

so übermäßig verwendet werden, wie Sie lokale Variablen gebrauchen. Aber sie können von Zeit zu Zeit sehr nützlich sein. Siehe [Abschnitt 12.8 \[Lokale Funktionen\]](#), [Seite 153](#), für Details.

## 10.5 Local-Anweisung

```
Local <var1> [, <var2>, ...] [= <expr1> [, <expr2>, ...]]
```

Die `Local`-Anweisung wird verwendet, um Hollywood mitzuteilen, dass die angegebene Variable lokal sein soll. Darüber hinaus kann es auch Ihre Variable initialisieren.

```
Local myvar          ; deklarieren die lokale Variable "myvar"
r = GetType(myvar)   ; legt in "r" #NIL ab
DebugPrint(myvar)    ; gibt 0 aus
<other code>
myvar = 5            ; nun ist "myvar" initialisiert
```

Der obige Code sagt Hollywood, dass `myvar` lokal sein soll, wenn ein Wert zugewiesen wird. Die Anweisung `"Local myvar"` initialisiert die Variable nicht. Sie wird nach wie vor vom Typ `Nil` sein, sie existiert also gar nicht. `"myvar"` wird erstellt, wenn Sie sie auf einen bestimmten Wert festlegen. Normalerweise werden Sie die Initialisierung in der `Local`-Anweisung erledigen, z.B.

```
Local myvar = 5      ; erstellt die lokale Variable "myvar" und
                    ; initialisiert sie mit 5
```

Sie können beliebig so viele Variablen erstellen und initialisieren, wie Sie möchten. Verwenden Sie einfach ein Komma auf jeder Seite des Gleichheitszeichens. Beispiel:

```
Local myvar, myvar2, myvar3 = 5, 4, 3
```

Der obige Code erstellt drei neue lokale Variablen und weist den Wert 5 `myvar`, 4 `myvar2` und 3 `myvar3` zu.

Bitte beachten Sie, dass die `Local`-Anweisung nicht am Anfang einer Funktion oder eines Blockes platziert werden muss, wie es der Fall mit Variablendeklarationen in anderen Programmiersprachen ist. Sie können sie platzieren, wo Sie wollen und es ist kein schlechter Programmierstil, `Local` in der Mitte einer Funktion zu verwenden. Zum Beispiel ist dieser Code in Ordnung:

```
Block
  DebugPrint("Now calling TestFunc()")
  Local r = TestFunc()
  DebugPrint("Now calling TestFunc2()")
  Local r2 = TestFunc2()
  DebugPrint("Results:", r, r2)
EndBlock
```

Dieser Code verwendet `Local` in der Mitte eines neuen Blocks. Das ist kein Problem mit Hollywood.

Siehe [Abschnitt 10.4 \[Lokale Variablen\]](#), [Seite 124](#), für Details.

## 10.6 Speicherbereiniger/Garbage Collector

Hollywood wird seinen Speicherbereiniger von Zeit zu Zeit aufrufen, während das Skript ausgeführt wird. Der Speicherbereiniger verwaltet alle von Ihrem Skript zugewiesenen Ressourcen und entlastet den gesamten Speicher, der nicht mehr benötigt wird. Beispielsweise:

```
Print("Hello World")
```

Nachdem Hollywood den `Print()` Befehl aufgerufen hat, um den Speicher für die Zeichenfolge "Hello World" zu zuweisen, wird der Speicherbereiniger diesen Speicher wieder freigegeben, da er nicht mehr benötigt wird. Sie können den Speicherbereiniger unterstützen, indem sie Variablen `Nil` zuweisen, wenn Sie sie nicht mehr brauchen. Dies ist besonders für lange Zeichenketten oder umfangreiche Tabellen nützlich, z.B.

```
a = {}
For k = 1 To 1000
    a[k] = {e1 = x, e2 = y}
    x = x + 5
    y = y + 5
Next
```

Dieser Code erstellt eine ziemlich umfangreiche Tabelle, die einiges an Speicher Ihres Systems einnimmt. Wenn Sie diese Tabelle nicht mehr benötigt, weisen Sie ihr einfach `Nil` zu, z.B.

```
a = Nil
```

Der Speicherbereiniger wird dann die von dieser Tabelle belegten Speicher freigeben.

Es wird auch dringend empfohlen, dass Sie lokale Variablen verwenden, wann immer und wo immer es möglich ist. Der Speicherbereiniger kann automatisch lokale Variablen freigeben, wenn deren Bereich endet (zum Beispiel am Ende einer Funktion). Siehe [Abschnitt 10.4 \[Lokale Variablen\]](#), [Seite 124](#), für Details.

## 10.7 Konstanten

Konstanten sind Werte, die nach der ersten Initialisierung nicht mehr verändert werden können. Sie können durch einen vom Benutzer angegebenen Namen angesprochen werden. Wie alle Sprachelemente unterscheidet Hollywood nicht zwischen Groß- und Kleinschreibung bei Konstanten, aber Sie sollten sie aus Styleguide-Gründen in Großbuchstaben schreiben. Konstanten müssen auch ein Hash-Zeichen (`#`) als Präfix haben, um sie von Variablen zu unterscheiden. Die Konstanten `True` und `False` sind Ausnahmen. Sie müssen kein Hash-Zeichen als Präfix haben, weil sie elementare Teile der Hollywood-Skriptsprache sind. Alle anderen Konstanten sind nur Ergänzungen für Befehle und müssen daher mit `#` vorangestellt werden.

Konstanten müssen entweder Zahlen oder Zeichenketten beinhalten. Sie können Ihre eigenen Konstanten unter Verwendung der `Const`-Anweisung erzeugen, zum Beispiel:

```
Const #MYCONSTANT = 5 * 5
```

Wenn Sie diese Anweisung verwenden, sollten Sie sie immer am Anfang des Skripts verwenden. Da es sich um eine globale Erklärung handelt, die nicht während dem Ausführung des Skripts verändert werden kann.



## 10.8 Const-Anweisung

```
Const #<name> = <expr>
```

Mit der Anweisung **Const** können Sie eine neue Konstante deklarieren. Der Name, den Sie in **name** angeben, muss mit einem Hash-Zeichen (#) vorangestellt werden. **expr** muss ein Ausdruck sein, das heißt Sie verwenden hier keine Variable. Beispielsweise:

```
Const #MYCONSTANT = (5 * 10) / 2      ; #MYCONSTANT = 25
Const #MYCONSTANT2 = #MYCONSTANT * 10 ; #MYCONSTANT2 = 250
Const #MYCONSTANT3 = b * 5            ; funktioniert nicht!
```

Das letzte Beispiel funktioniert nicht, da eine Variable verwendet wird und der Ausdruck konstant sein muss.

Alternativ kann **expr** auch eine konstante Zeichenkette sein. Z.B.

```
Const #PRGVERSTRING = "$VER: MyProgram 1.0 (13.04.2005)"
```

Konstanten können auch mit dem Konsolenargumente **-setconstants** deklariert werden. Dies ist besonders in Verbindung mit der Präprozessor-Anweisung **@IF** nützlich. Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), [Seite 33](#), für Details.

## 10.9 Integrierte Konstanten

Die integrierten Konstanten werden von vielen Befehlen für eine spezielle Aktion verwendet, damit unterscheidet sich ihre Funktion von Konstante zu Konstante. Wenn ein Befehl eine spezielle Konstante als Argument erfordert (z.B. **SetFontStyle()** die Konstanten **#BOLD**, **#ITALIC**, **#NORMAL** und **#UNDERLINED**), dann werden diese Konstanten für diesen Befehl in der Dokumentation beschrieben.

Zusätzlich gibt es einige Konstanten, die Sie jedes Mal benutzen können, wenn ein Hollywood-Befehl nach den x- oder y-Koordinaten fragt. Diese Konstanten sind die sogenannten "Positionskonstanten". Sie ermöglichen es, einfach sehr oft benutzte Positionen anzugeben.

Die folgenden Konstanten können als x-Koordinate verwendet werden:

```
#CENTER:  Gibt den Mittelpunkt Ihres Displays an (=(Displaybreite-Objektbreite)/2)
#LEFT:    Gibt den linken Rand des Displays an (=0)
#LEFTOUT:
           Gibt die äussere linke Seite des Displays an (=0-Objektbreite)
#RIGHT:   Gibt den rechten Rand des Displays an (=(Displaybreite-Objektbreite)
#RIGHTOUT:
           Gibt die äussere rechte Seite des Displays an (=(Displaybreite+Objektbreite)
#USELAYERPOSITION:
           Gibt die aktuelle x-Position der Ebene an.
```

Die nun folgenden Konstanten können als y-Koordinate verwendet werden:

```
#CENTER:  Gibt den Mittelpunkt Ihres Displays an (=(Displayhöhe-Objekthöhe)/2)
#TOP:     Gibt die Oberkante des Displays an (=0)
#TOPOUT:  Gibt den oberen Rand Ihres Displays an (=0-Objekthöhe)
```

**#BOTTOM:** Gibt den unteren Rand des Displays an (=Displayhöhe-Objekthöhe)

**#BOTTOMOUT:**

Gibt die äussere Unterseite des Display an (=Displayhöhe+Objekthöhe)

**#USELAYERPOSITION:**

Gibt die aktuelle x-Position der Ebene an.

Diese Konstanten machen es sehr einfach, Ihre Objekte zu positionieren. Zum Beispiel, wenn Sie Pinsel 1 in der Mitte des Displays angezeigt werden soll, dann benutzen Sie **DisplayBrush()** und setzen für alle Argumente, **#CENTER**, **#CENTER** et voilà!

Sie können sogar eine Feinabstimmung der Positionen durch Subtrahieren sowie Addieren der Werte dieser Konstanten erhalten! Z.B. **DisplayBrush(1, #CENTER, #CENTER + 25)** zeigt den Pinsel 1 25 Pixel unterhalb der vertikalen Mitte des Displays.

Es gibt auch einige Konstanten, die Ihnen einfachen Zugang einiger Grundfarben ermöglicht: **#BLACK**, **#MAROON**, **#GREEN**, **#OLIVE**, **#NAVY**, **#PURPLE**, **#TEAL**, **#GRAY**, **#SILVER**, **#RED**, **#LIME**, **#YELLOW**, **#BLUE**, **#FUCHSIA**, **#AQUA**, **#WHITE**.

#BLACK	\$000000	
#MAROON	\$800000	
#GREEN	\$008000	
#OLIVE	\$808000	
#NAVY	\$000080	
#PURPLE	\$800080	
#TEAL	\$008080	
#GRAY	\$808080	
#SILVER	\$C0C0C0	
#RED	\$FF0000	
#LIME	\$00FF00	
#YELLOW	\$FFFF00	
#BLUE	\$0000FF	
#FUCHSIA	\$FF00FF	
#AQUA	\$00FFFF	
#WHITE	\$FFFFFF	

Schließlich definiert Hollywood einige plattformspezifische Konstanten. Diese sind abhängig von der Plattform, auf der Hollywood derzeit läuft oder kompiliert. Sie können die Präprozessor-Anweisung **@IF** verwenden, um diese Konstanten zu überprüfen und die gewünschte Aktion auszuführen. Diese plattformspezifischen Konstanten finden Sie bei der Präprozessor-Anweisung **@IF** in dieser Dokumentation. Siehe **Abschnitt 50.17 [IF]**, **Seite 1099**, für Details.

## 10.10 Zeichenkonstanten

Zeichenkonstanten werden in der Regel verwendet, um den Codepunktwert eines Zeichens in einer einfachen Art und Weise zu erhalten. Wenn Sie ein Zeichen in einfache Anführungszeichen ( ' ') umschließen, dann wird Hollywood diese Angabe mit dem

Codepunktwert des Zeichens ersetzen. Somit sind Zeichenkonstanten immer vom Datentyp **Zahl**. Hier ist ein Beispiel:

```
DebugPrint('A')    ; gibt 65 aus
```

Sie können auch Escape-Sequenzen in eine Zeichenkonstante setzen. Beispielsweise:

```
DebugPrint('\n')   ; Gibt 10 aus
```

Siehe **Abschnitt 8.3 [Zeichenketten]**, **Seite 104**, für weitere Informationen zu den Escape-Sequenzen, die Hollywood unterstützt.



## 11 Kontrollstrukturen

### 11.1 Skriptablaufkontrolle

Dieses Kapitel beschreibt alle Anweisungen von Hollywood, die verwendet werden, um den Programmablauf zu steuern. Es ist sehr wichtig, dass Sie diese Kontrollstrukturen kennen, weil so Ihr Programm viel besser lesbar wird. Wir können die Kontrollstrukturen in zwei Gruppen kategorisieren:

1) Bedingte Blöcke: Sie werden verwendet, um zu überprüfen, ob ein bestimmter Ausdruck wahr (ungleich Null) oder falsch (Null) ist. Dies ist sehr wichtig, weil Ihr Programm die ganze Zeit Entscheidungen fällt. Die folgenden Arten von bedingten Blöcke stehen zur Verfügung:

**If-Else-ElseIf-EndIf**  
**Switch-Case-Default-EndSwitch**

2) Schleifen (Loops): Sie werden verwendet, bestimmte Teile des Codes zu wiederholen. Stellen Sie sich vor, Sie wollen die Zahlen von 1 bis 100 ausgeben. Sie könnten den `Print()` Befehl hundertmal dafür in Ihr Skript schreiben. Aber man könnte auch eine einfache For-Schleife verwenden, die den Befehl `Print()` hundertmal ausführt. Die folgenden Schleifenstrukturen stehen zur Verfügung:

**While-Wend**  
**For-Next**  
**Repeat-Until**

### 11.2 If-EndIf-Anweisung

Es gibt zwei Versionen der **If**-Anweisung: Eine lange und eine kurze.

#### 1) Lange Version der If-Anweisung:

```
If <expr> <block> [ElseIf <expr> <block> ...] [Else <block>] EndIf
```

Die **If**-Anweisung prüft, ob der angegebene Ausdruck wahr (ungleich Null) ist. Ist dies der Fall, werden die Befehle nach der Anweisung ausgeführt. Wenn der angegebene Ausdruck falsch (Null) ist, springt **If** auf die nächste **ElseIf**-Anweisung (sofern vorhanden) und prüft, ob der gegebene Ausdruck wahr ist. Dies wird wiederholt, bis die **Else**-Anweisung erreicht ist. Wenn keine der Ausdrücke vorher wahr war, wird der Code **Else** ausgeführt.

Die Anweisungen **If** und **EndIf** sind obligatorisch. **ElseIf** und **Else**-Anweisungen sind optional. Sie können beliebig viele **ElseIf**'s verwenden, aber es darf nur eine **Else**-Anweisung vorhanden sein. Darüber hinaus muss die **Else**-Anweisung die letzte Bedingung vor dem **EndIf** sein. Hier ist ein Beispiel:

```
If a > 5                                ; überprüft, ob a größer ist als 5
    DebugPrint("a > 5")
ElseIf a < 5                             ; überprüft, ob a kleiner ist als 5
    DebugPrint("a < 5")
Else                                     ; sonst ist a gleich 5
    DebugPrint("a = 5")
EndIf
```

Sie können auch komplexere Ausdrücke als Bedingung verwenden:

```
If country$ = "USA" And age < 21
```

```

    DebugPrint("No alcohol permitted under 21 in the USA!")
EndIf

```

## 2) Kurze Version der If-Anweisung:

```

If <expr> Then <true-stat> [ElseIf <expr> <true-stat> ...] [Else <stat>]

```

Die kurze Version der **If**-Anweisung funktioniert auf die gleiche Weise wie die lange, hat aber den Vorteil, dass Sie **EndIf** nicht gebrauchen müssen. Die kurze **If**-Anweisung hat die Einschränkung, dass alle seine Teile in einer einzigen Zeile platziert werden müssen. Eine weitere Einschränkung ist, dass nur eine Anweisung platziert werden darf, der dann die Anweisungen **Then** / **ElseIf** / **Else** folgen müssen. Wenn Sie mehrere Anweisungen ausführen möchten, müssen Sie die lange Version benutzen.

Das obige Beispiel haben wir mit der kurzen **If**-Anweisung folgender Weise geschrieben:

```

If a>5 Then Print("a>5") ElseIf a<5 Print("a<5") Else Print("a=5")

```

Sie können sehen, dass das Ergebnis nicht besonders gut lesbar ist. Für das obige Beispiels ist es nicht empfehlenswert, die kurze Version zu verwenden. Die kurze **If**-Anweisung passt besser, wenn Sie nur eine Bedingung haben, z.B.

```

If a = True Then b = 5

```

Das ist besser lesbar als

```

If a = True
    b = 5
EndIf

```

Eine andere Version von **If** ist die so genannte Sofort-If-Anweisung **IIf()**. Diese Version wird als Befehl in Hollywood durchgeführt und ist Teil der Systembibliothek. Siehe [Abschnitt 50.18 \[IIf\]](#), [Seite 1101](#), für Details.

## 11.3 While-Wend-Anweisung

Es gibt zwei Versionen der **While**-Anweisung: Eine lange und eine kurze.

### 1) Lange Version der While-Anweisung:

```

While <expr> <loop-block> Wend

```

Die **While**-Anweisung tritt in die Schleife, wenn der angegebene Ausdruck wahr (nicht Null) ist. Wenn der Ausdruck falsch (Null) ist, wird überhaupt nicht in die Schleife eingetreten und die Ausführung wird nach der **Wend**-Anweisung fortgesetzt. Nach dem Eintreten in die Schleife wird sie so lange wiederholt, bis der angegebene Ausdruck wahr ist.

```

i = 0
While i < 100
    i = i + 1
Wend
DebugPrint(i) ; gibt 100 aus

```

Die Schleife oben wird wiederholt, bis der Ausdruck  $i < 100$  falsch wird. Dies ist der Fall, wenn  $i$  gleich oder größer 100 ist.  $i$  startet mit 0 und es wird pro Schleifendurchgang 1 dazu addiert, bis  $i$  den Wert von 100 erreicht.

Werfen Sie auch einen Blick auf die beiden **Break** und **Continue**-Anweisungen. Diese können verwendet werden, von einer Schleife zu verlassen oder an deren Ende zu springen.

**2) Kurze Version der While-Anweisung:**

```
While <expr> Do <stat>
```

Die kurze Version verhält sich genau wie die lange Version, aber Sie muss nicht die **Wend**-Anweisung enthalten. Die kurze Anweisung hat die Einschränkung, dass der Schleifen-Block (Loop-Block) nur aus einer Anweisung besteht. Wenn Sie mehrere Anweisungen in dem Schleifenblock ausführen wollen, müssen Sie die lange **While**-Version verwenden. Die Anweisung **Do** signalisiert Hollywood, dass Sie die kurze Version verwenden möchten.

Das Beispiel von oben könnte als kurze Anweisung auf folgende Weise geschrieben werden:

```
While < 100 Do i = i + 1
```

**11.4 For-Next-Anweisung**

Es gibt drei Versionen der **For**-Anweisung: Eine lange, eine kurze und eine generische. Die generische Version der Anweisung kann auch in einer langen und kurzen Version verwendet werden, so dass es insgesamt vier verschiedene Versionen der **For**-Anweisung sind. Jedoch verbleiben wir aus Gründen der Übersichtlichkeit nur bei drei verschiedenen Versionen in dieser Dokumentation.

**1) Lange Version der For-Anweisung:**

```
For [Local] <var> = <expr1> To <expr2> [Step <expr3>] <loop-block> Next
```

Als erstes weist die **For**-Anweisung der Variable **var** den Wert im Ausdruck **expr1** zu. Nun wird dieser Wert der **Step**-Anweisung an **expr3** übergeben. Dieser Schritt-Wert ist optional. Wenn Sie nichts angeben, ist bei der **Step**-Anweisung der Schritt-Wert in **expr3** standardmäßig auf 1 gesetzt.

Wenn **expr3** positiv ist, wird die **For**-Anweisung überprüfen, ob der Wert der Variable **var** kleiner oder gleich **expr2** ist. Wenn dies der Fall ist, wird in die Schleife eingetreten und wiederholt, bis **var** größer ist als **expr2**. Am Ende jeder Schleife wird der Wert **expr3** zur Variablen addiert.

Wenn **expr3** negativ ist, prüft die **For**-Anweisung, ob der Wert der Variable **var** größer oder gleich **expr2** ist. Wenn dies der Fall ist, wird in die Schleife eingetreten und so lange wiederholt, bis der Wert in **var** kleiner ist als der von **expr2**. Am Ende jeder Schleife wird der Wert **expr3** der Variable **var** abgezogen.

Ist **expr3** Null, wird die Schleife für immer wiederholt. Zu beachten ist außerdem, dass die Ausdrücke **expr2** und **expr3** nur einmal am Anfang der Schleife ausgewertet werden. Somit ist die Schleifengrenze und der Schritt konstant und kann nicht geändert werden, während die Schleife aktiv ist.

Ein Beispiel:

```
For i = 1 To 100
  DebugPrint(i)
Next
```

Dieser Code gibt die Zahlen von 1 bis 100 aus. **DebugPrint()** wird hundertmal ausgeführt. Wenn die Schleife beendet wird, hat die Variable **i** den Wert 101. Sie sehen, dass wir nicht eine **Step**-Anweisung angegeben haben. Dies bedeutet, dass bei jeder Wiederholung der Schleife 1 zur Variable addiert wird. Wenn wir mit dem Faktor 2 arbeiten möchten, könnten wir den folgenden Code verwenden:

```
For i = 1 To 100 Step 2
```

```

    DebugPrint(i)
Next

```

Das wird "1 3 5 7 9... 95 97 99" ausgegeben. Wenn die Schleife verlassen wird, hat die Variable `i` den Wert 101.

Wenn wir rückwärts von 100-0 zählen wollen, verwenden wir einen negativen Stufenwert wie in dem folgenden Beispiel:

```

For i = 100 To 0 Step -1
    DebugPrint(i)
Next

```

Dies ruft `DebugPrint()` hundert und ein Mal auf. Nachdem die Schleife beendet ist, hat die Variable `i` den Wert -1.

Wenn Sie `Local` vor der Initialisierung der Variable der `For`-Anweisung schreiben, wird die Variable lokal in den Schleifen-Block erstellt. Das bedeutet, dass nicht von außerhalb des Schleifenblock auf sie zugegriffen werden kann. Ein Beispiel:

```

For Local i = 1 To 50
    DebugPrint(i)      ; gibt 1, 2, 3 ... 49, 50 aus
Next
DebugPrint(i)         ; gibt 0 aus (i ist nur innerhalb ...
                      ; ... der Schleife existent)

```

Der Vorteil von einer lokalen Zählervariable in `For`-Schleifen ist, dass sie schneller laufen als Schleifen, die eine globale Variable verwenden. Wenn Sie auf die Variable von außerhalb der `For`-Anweisung nicht zugreifen müssen, sollten Sie immer `Local` verwenden. Eine Einschränkung für Schleifen mit `Local` ist, dass Sie keinen neuen Wert auf den lokale Zählerwert zuordnen können. Wenn Sie die Schleife beenden müssen, verwenden Sie `Break`. Ändern der Zählervariable während der Durchführung der Schleife funktioniert nur ohne die Anweisung `Local`.

Werfen Sie auch einen Blick auf die `Break` und die `Continue`-Anweisungen. Diese können verwendet werden, um eine Schleife zu verlassen oder an deren Ende zu springen.

## 2) Kurze Version der For-Anweisung:

```

For [Local] <var> = <expr1> To <expr2> [Step <expr3>] Do <stat>

```

Die kurze Version verhält sich genau wie die lange, aber Sie muss nicht die `Next`-Anweisung enthalten. Die kurze `For`-Anweisung hat die Einschränkung, dass der Schleifenblock nur aus einer Anweisung bestehen darf. Wenn Sie mehrere Anweisungen im Schleifenblock ausführen wollen, müssen Sie die lange Version verwenden. Die Anweisung `Do` signalisiert Hollywood, dass Sie die kurze Version verwenden möchten.

Das erste Beispiel von oben könnte auf folgende Weise geschrieben werden, um eine kurze `For`-Anweisung zu erhalten:

```

For i = 1 To 100 Do DebugPrint(i)

```

## 3) Generische Version der For-Anweisung:

```

For <var1> [, <var2>, ...] In <expr> [Do <stat>] or [<loop-block> Next]

```

Die generische Version der `For`-Anweisung unterscheidet sich von den beiden anderen Varianten durch die Tatsache, dass sie eine benutzerdefinierte Funktion aufruft, um die Werte für jede Zählervariablen zu erhalten. Diese Tatsache macht die generische `For`-Anweisung



für eine Vielzahl von Zwecken nützlich. Sie können Ihre eigenen Zählerfunktionen schreiben, aber für die meisten Fälle werden Sie wahrscheinlich die eingebauten Zählerfunktionen verwenden, die von Befehlen wie `Pairs()`, `IPairs()` oder `PatternFindStr()` zur Verfügung gestellt werden.

Der in `expr` angegebener Ausdruck wird nur einmal ausgewertet. Er gibt drei Werte zurück: Eine Zählerfunktion, einen Zustands- und einen Anfangswert für `var1`. Die Zählerfunktion und der Zustandswert sind private Werte und sie sind während der Laufzeit der `For`-Schleife als Variablen weder sichtbar noch zugänglich. Sobald die generische `For`-Schleife diese drei Werte abgerufen hat, startet sie die Zählerfunktion mit dem Zustands- und dem aktuellen Wert in `var1`. Die Schleife wird beendet werden, sobald der aktuelle Wert in `var1` `Nil` ist.

Die meisten Zählerfunktionen geben mehrere Werte für den Zähler zurück. Deshalb können Sie auch mehrere Variablen in der allgemeinen `For`-Anweisung angeben. Die Fähigkeit, mehrere Variablen auf verschiedene Zählerzustände initialisiert zu haben, macht die allgemeine `For`-Anweisung sehr flexibel.

Lassen Sie uns nun ein Beispiel anschauen und betrachten Sie die folgende Tabelle:

```
months = {"January", "February", "March", "April", "May", "June",
          "July", "August", "September", "October", "November", "December"}
```

Wir wollen nun herauszufinden, welchen Index (1 bis 12) jeder Monat hat. Dafür verwenden wir die Namen als Referenz. Natürlich könnten wir die Tabelle durchlaufen und den Namen mit dem entsprechenden Index zu vergleichen. Aber wenn es sich um größere Datenmengen handelt, ist es häufig schneller, eine Umkehr-Tabelle zu erstellen, um die gewünschten Informationen zu erhalten. In unserem Fall wollen wir eine Tabelle, die den Namen der Monate als Indizes verwendet, so dass die Tabelle ["Januar"] 1, Tabelle ["Februar"] liefert 2, und so weiter. Wir können diese Umkehr-Tabelle mit der generischen Zählerfunktion `IPairs()` und der `For`-Schleife zusammen erstellen. Die `IPairs()`-Zählerfunktion gibt zwei Werte zurück: Den Index- sowie den Schlüsselwert für jedes Tabellenelement, welche an sie übergeben werden. Wir können diese `IPairs()`-Zählerfunktion verwenden, um die Tabelle sehr leicht durchzugehen:

```
revmonths = {}
For i,v In IPairs(months)
    revmonths[v] = i + 1
Next
```

Alternativ könnten wir auch für diesen Code die kurze Version der generischen `For`-Schleife verwenden, da sich nur eine Anweisung in der `For`-Schleife befindet. Mit Hilfe der Kurzversion würde der Code wie folgt aussehen:

```
revmonths = {}
For i,v In IPairs(months) Do revmonths[v] = i + 1
```

Die `IPairs()`-Zählerfunktion zählt nur alle Ganzzahlen-Indizes in einer Tabelle. Wenn Sie alle Felder einer Tabelle durchgehen wollen, können Sie stattdessen die Zählerfunktion `Pairs()` verwenden.

Ein weiterer Befehl, der oft in Verbindung mit der generischen `For`-Schleife verwendet wird, ist `PatternFindStr()`. Er wird eine Zählerfunktion zurückgeben, die verwendet werden kann, um eine Zeichenfolge zu analysieren. Zum Beispiel wird der folgende Code alle Wörter in einer Zeichenkette durchgehen:

```
s$ = "Hello World This is a test"
```

```
For w$ In PatternFindStr(s, "%a+") Do Print(w$)
```

Natürlich ist es auch möglich, eigene Zählerfunktionen zu schreiben. Dies kann jedoch ziemlich kompliziert werden. Deshalb wird das hier nicht erläutert. Bitte konsultieren Sie das Buch "Programming in Lua (zweite Auflage)" von Roberto Ierusalimsky für weitere Informationen, wie eigene Zählerfunktionen (Iterator-Function) zu schreiben sind.

## 11.5 Repeat-Until-Anweisung

Es existieren zwei Versionen der **Repeat**-Anweisung: Eine bedingte und eine endlose.

### 1) Bedingte Version der Repeat-Anweisung:

```
Repeat <loop-block> Until <expr>
```

Die bedingte **Repeat**-Anweisung wiederholt den angegebenen Schleifenblock, bis der gegebene Ausdruck wahr wird (ungleich Null). Mit anderen Worten: Der Block wird durchschritten, während **expr** falsch (Null) ist. Dies ist umgekehrt, wie sich die **While**-Anweisung verhält: **While**-Schleifen werden durchlaufen, während der Ausdruck wahr ist, hingegen Repeat-Schleifen während der Ausdruck falsch ist.

Hier ist ein Beispiel:

```
i = 1
Repeat
    i = i + 1
Until i = 100
```

Dieser Code zählt von 1 bis 100. Wenn die Schleife verlassen wird, hat die Variable **i** den Wert 100.

Schauen Sie sich in der Dokumentation auch die beiden Anweisungen **Break** und **Continue** an. Diese können verwendet werden, um eine Schleife zu verlassen oder an ihr Ende zu springen.

### 2) Endlose Version der Repeat-Anweisung:

```
Repeat <loop-block> Forever
```

Die endlose Version kann einen bestimmten Teil des Codes immer wiederholen. Sie können immer noch mit der **Break**-Anweisung aus der Schleife springen. Die endlose Version wird meist in der Hauptschleife eines Skripts verwendet, die **WaitEvent()** aufruft, z.B.

```
Repeat
    WaitEvent
Forever
```

## 11.6 Switch-Case-Anweisung

```
Switch <ex1> Case <ex2>[:] <blk> [...] [Default[:] <blk>] EndSwitch
```

Die **Switch**-Anweisung kann verwendet werden, um den Ausdruck **ex1** mit allen anderen Ausdrücken nach **Case** zu vergleichen. Sie können so viele **Case**-Anweisungen verwenden, wie Sie wollen, aber es muss mindestens eine in Ihrer **Switch**-Anweisung vorhanden sein. Wenn der Ausdruck **ex1** mit einem **ex2** übereinstimmt, wird der Code nach dieser **Case**-Anweisung ausgeführt. Nach der Ausführung setzt Hollywood Ihr Programm nach der **EndSwitch**-Anweisung fort. Wenn keiner der **ex2** Ausdrücke mit **ex1** übereinstimmt, wird der Code nach der **Default**-Anweisung ausgeführt. Die Verwendung von der **Default**-Anweisung ist

optional. Wenn Sie sie nicht brauchen, müssen Sie sie auch nicht benutzen. Aber wenn Sie sie benutzen, muss sie immer die letzte Anweisung des **Switch**-Blockes sein. Nach der **Default**-Anweisung dürfen keine weiteren **Case**-Anweisungen folgen. Der Doppelpunkt nach **Case** oder **Default** ist auch optional.

Bitte beachten Sie, dass der Ausdruck **ex2** und folgende immer konstant sein müssen. Sie können hier nicht Variablen oder Rückgabewerte von Funktionen/Befehlen verwenden. Auch dürfen Sie nicht Variablentypen in dieser Anweisung mischen: Wenn **ex1** eine Zeichenkette ist, müssen auch alle **ex2** und folgende Ausdrücke Zeichenketten sein. Wenn eine Zahl verwendet wird, müssen alle anderen Ausdrücke auch Zahlen sein.

Ein Beispiel:

```
Switch x
Case 1:
    DebugPrint("x = 1")
Case 2:
    DebugPrint("x = 2")
Default:
    DebugPrint("x <> 1 And x <> 2")
EndSwitch
```

Der obige Code schaut auf die Variable **x** und tritt in die erste **Case**-Anweisung ein, wenn **x** eins ist. In die zweite **Case**-Anweisung wird eingetreten, wenn **x** 2 enthält. Andernfalls wird die **Default**-Anweisung ausgeführt.

Jede Anweisung **Switch-Case** kann auch als normale **If**-Anweisung geschrieben werden. Das Beispiel von oben würde dann wie folgt aussehen:

```
If x = 1
    DebugPrint("x = 1")
ElseIf x = 2
    DebugPrint("x = 2")
Else
    DebugPrint("x <> 1 And x <> 2")
EndIf
```

C und Java-Programmierer sollten beachten, dass die **Switch**-Anweisung in Hollywood nach dem Ende des vorherigen **Case**-Blocks nicht automatisch in den nächsten **Case**-Block durchfällt. Stattdessen wird Hollywood automatisch zum Ende der Anweisung springen, nachdem ein **Case**-Block ausgeführt wurde. Somit müssen Sie am Ende eines **Case**-Blocks auch nicht die **Break**-Anweisung verwenden. Aber Sie können sie früher verwenden, um die **Switch**-Anweisung zu beenden.

Es ist jedoch möglich, manuell ein Durchfallen mit der Anweisung **FallThrough** zu erzwingen. Wenn Hollywood diese Anweisung in einem **Case**-Block begegnet, wird bis zum nächsten **Case**- (oder dem **Default**-Block) fallen, d.h. es wird direkt in diesen Block gesprungen. Daher kann **FallThrough** nur verwendet werden, wenn ein **Case**- oder **Default**-Block folgt. Andernfalls wird ein Fehler erzeugt. Da die **Default**-Anweisung immer der letzte Block einer **Switch**-Anweisung sein muss, ist es nicht erlaubt, **FallThrough** im **Default**-Block zu verwenden.

Hier ein Beispiel:

```
Switch msg.action
```

```

Case "OnKeyDown":
    FallThrough
Case "OnKeyUp":
    DebugPrint("Key event:", msg.key)
Default:
    DebugPrint("Other event")
EndSwitch

```

Der obige Code gibt die Taste aus, welche gedrückt wurde, falls `msg.action OnKeyDown` oder `OnKeyUp` beinhaltet. Wenn `msg.action OnKeyDown` ist, wird die Anweisung `FallThrough` verwendet, um in den `OnKeyUp`-Block zu springen.

Alternativ kann die Anweisung `FallThrough` auch direkt nach der Anweisung `Switch` eingefügt werden. In diesem Fall wird jeder `Case`-Block automatisch zum nächsten Block weitergeleitet, es sei denn, es gibt eine `Break`-Anweisung, die das Durchfallen verbietet.

Hier ist ein Beispiel:

```

Switch msg.action FallThrough
Case "OnKeyDown":
Case "OnKeyUp":
    DebugPrint("Key event:", msg.key)
    Break
Case "OnMouseDown":
Case "OnMouseUp":
    DebugPrint("Left mouse event")
    Break
Case "OnRightMouseDown":
Case "OnRightMouseUp":
    DebugPrint("Right mouse event")
    Break
Default:
    DebugPrint("Other event")
EndSwitch

```

Der obige Code verwendet `FallThrough` global, um denselben Code für `OnKeyDown`, `OnKeyUp`, `OnMouseDown`, `OnMouseUp`, `OnRightMouseDown` und `OnRightMouseUp` zu verwenden. Die `Break`-Anweisungen sind notwendig, da sonst Hollywood bis zur letzten Codezeile im `Default`-Block durchgehen würde.

## 11.7 Break-Anweisung

```
Break [( <level> )]
```

Wenn Sie `Break` innerhalb einer Schleife oder einer `Switch`-Anweisung verwenden, dann wird Hollywood diese Kontrollstruktur verlassen. Ein Beispiel:

```

For k = 1 To 100
    DebugPrint(k)
    If IsKeyDown("ESC") = True Then Break
Next

```

Die obige Schleife zählt von 1 bis 100 und kann durch Drücken der Escape-Taste jederzeit abgebrochen werden.

Mit dem optionalen Argument können Sie auch höhere Schleifen beenden. Wenn Sie keine Ebene in `level` angeben, ist in Hollywood 1 voreingestellt. Dies bedeutet, dass nur diese Schleife beendet wird. Wenn Sie höhere Werte verwenden, wird Hollywood auch die übergeordneten Schleifen verlassen. Ein Beispiel:

```
For x = 1 To 100
  For y = 1 To 100
    DebugPrint(x, y)
    If IsKeyDown("ESC") Then Break(2)
  Next
Next
```

Dieser Code verwendet zwei verschachtelte **For**-Schleifen. Um beide Schleifen zu beenden, müssen wir jetzt **Break(2)** verwenden, da die normale **Break**-Anweisung nur die innere Schleife beenden würde. Darauf hin würde sie wieder gestartet werden, weil noch die äußere Schleife durchlaufen wird.

Bitte beachten Sie: Wenn Sie das optionale Argument `level` angeben, sind die Klammern obligatorisch.

## 11.8 Continue-Anweisung

```
Continue [( <level> )]
```

Die **Continue**-Anweisung wird dazu benutzt, um ans Ende einer Schleife zu springen. Ein Beispiel:

```
While i < 100
  i = i + 1
  If i > 50 Then Continue
  j = j + 1
Wend
```

Der obige Code zählt `i` von 0 bis 100. Die Variable `j` wird nur addiert, solange `i` kleiner oder gleich 50 ist. Sobald `i` größer als 50 ist, wird `j` nicht mehr erhöht. Am Ende der Schleife hat `i` den Wert von 100 und `j` den Wert von 50.

Mit dem optionalen `level`-Argument können Sie auch bis zum Ende einer höheren Schleife springen. Wenn Sie für `level` nichts angeben, ist in Hollywood 1 voreingestellt. Das bedeutet, dass bis zum Ende der nächsten Schleife gesprungen wird. Wenn Sie höhere Werte verwenden, wird Hollywood auch bei den höheren Schleifen an deren Enden springen.

Bitte beachten Sie: Wenn Sie das optionale Argument `level` angeben, ist das Setzen von Klammern um `level` obligatorisch.

## 11.9 Return-Anweisung

```
Return [( <retval1>, <retval2>, ... )]
```

Die **Return**-Anweisung wird zum Verlassen einer benutzerdefinierten Funktion verwendet. Die Programmsteuerung kehrt an die Position im Skript zurück, von wo aus die Funktion aufgerufen wurde.

Optional können über **Return** so viele Werte zurückgegeben werden, wie Sie wollen. Dabei ist es obligatorisch, diese Werte in Klammern zu setzen. Beispielsweise:

```
; falsch!
```

```

Function p_Min(a, b)
    If a < b Then Return a
    Return b
EndFunction

; richtig!
Function p_Min(a, b)
    If a < b Then Return(a)
    Return(b)
EndFunction

```

Siehe [Abschnitt 12.1 \[Funktionen\]](#), [Seite 145](#), für Details.

Hinweis zur Kompatibilität mit Hollywood 1.x Skripts: **Return** kann auch nach einer **Label**-Anweisung verwendet werden, um zur **Gosub**-Position zurück zu springen. Diese Methode ist nur für die Kompatibilität mit Hollywood 1.x Skripts enthalten. Bitte verwenden Sie sie nicht mehr.

## 11.10 Block-EndBlock-Anweisung

```
Block <block-code> EndBlock
```

Die **Block**-Anweisung führt den folgenden Code in einem separaten Rahmen aus. Diese Anweisung werden Sie selten gebrauchen. Normalerweise werden Sie dies nicht benötigen. Hier ist ein Beispiel:

```

For k = 1 To 100
    Block
        Local k
        For k = 1 To 2
            DebugPrint(k)
        Next
    EndBlock    ; lokal "k" ist nun gelöscht
Next

```

Der obige Code verwendet zwei Variablen in zwei verschachtelten Schleifen mit dem Namen **k**. Dies ist nur möglich, weil wir die innere Schleife in einem eigenen **Block** gesetzt und eine neue lokale Variable **k** in diesem **Block** erstellt haben. Diese lokale Variable ist nur in diesem **Block** zugänglich. Nach der **EndBlock**-Anweisung wird die lokale Variable **k** gelöscht und die globale **k** wieder verwendet.

## 11.11 Dim- und DimStr-Anweisung

```

Dim <varname>[<dim1-size>] ([<dim2-size>], ...), ...
DimStr <varname>[<dim1-size>] ([<dim2-size>], ...), ...

```

Die **Dim**- und **DimStr**-Anweisungen können verwendet werden, um eine n-dimensionale Tabelle mit den angegebenen Größen zu erstellen und alle Elemente auf 0 ( **Dim** ) oder "" ( **DimStr** ) zu initialisieren.

Wie Sie bereits aus der Dokumentation über [Tabellen](#) wissen, müssen Tabellenfelder initialisiert werden, bevor sie verwendet werden können. Selbst wenn ein Feld nur eine Null oder eine leere Zeichenfolge tragen soll, müssen Sie es mit diesem Wert initialisieren, bevor Sie

darauf zugreifen können. Die **Dim**-Anweisung kann Ihnen dabei helfen. Sie wird die Tabelle **varname** in der angegebenen Größe **size** erstellen. Der Parameter **size** muss ein konstanter Wert sein, also keine Variable. Bitte beachten Sie, dass **size** die wirkliche Größe der Tabelle angibt und nicht das letzte Element, das initialisiert werden soll. Wenn Sie also 50 als **size** verwenden, wird Hollywood Tabellenfelder von 0 bis 49 initialisieren.

Hier ein Beispiel:

```
Dim mytable[100]
```

Diese Deklaration würde als **For**-Anweisung so aussehen:

```
Local mytable = {}
For k = 0 To 99 Do mytable[k] = 0
```

Die **Dim**-Anweisung ist wirklich praktisch, wenn Sie mehrdimensionale Tabellen erstellen möchten. Sie können so viele der eckigen Klammern nach der **varname** Deklaration verwenden, wie Sie möchten. Jede neue eckige Klammer wird eine neue Tabellendimension in der angegebenen Größe erstellen. Beispielsweise:

```
Dim vector[10][10][10]
```

Die obige Deklaration erzeugt eine dreidimensionale Vektortabelle und initialisiert alle Elemente mit Nullen. Diese Deklaration als **For**-Anweisung übersetzt, sieht ziemlich viel komplexer aus:

```
Local vector = {}
For i = 0 To 9
  vector[i] = {}
  For j = 0 To 9
    vector[i][j] = {}
    For k = 0 To 9
      vector[i][j][k] = 0
    Next
  Next
Next
```

Sie können auch mehr als eine Tabelle mit dieser Anweisung erstellen und initialisieren. Verwenden Sie einfach ein Komma nach der letzten Dimensionsdeklaration und Sie können den ganzen Vorgang so oft wiederholen, wie Sie wollen. Beispiel:

```
Dim table1[50], table2[50], table3[50]
```

Die **DimStr**-Anweisung funktioniert auf die gleiche Weise wie **Dim**, aber initialisiert alle Felder mit leeren Zeichenfolgen ("").

Bitte beachten Sie, dass **Dim**/**DimStr** immer lokale Tabellen erstellen, wenn Sie sich nicht im Hauptblock des Skripts befinden. Wenn Sie eine globale Tabelle wollen, sollten Sie die **Dim**/**DimStr**-Anweisung im Hauptblock des Skripts benutzen.

Denken Sie auch daran, dass **Dim**/**DimStr** die Tabelle nicht auf die angegebene Größe beschränkt. Die Tabelle kann immer noch wachsen, weil Hollywood eine dynamisch typisierte Sprache ist! Um eine Tabelle zu vergrößern, initialisieren Sie einfach die Felder, die Sie brauchen. Hollywood wird sie dann automatisch wachsen lassen. Beispielsweise:

```
Dim table[50]
...
For k = 50 To 59 Do table[k] = 0 ; Tabelle wächst um 10 Felder
```

Der obige Code erstellt eine Tabelle mit 50 Feldern und wächst anschließend auf 60. Wenn Sie eine Tabelle verkleinern möchten, weisen Sie den entsprechenden Feldern **Nil** zu. Zum Beispiel:

```
Dim table[50]
```

```
...
```

```
For k = 40 To 49 Do table[k] = Nil ; Tabelle schrumpft um 10 Felder
```

Der obige Code schrumpft die Tabelle von 50 initialisierten Feldern auf 40. Hollywood ist eine dynamisch typisierte Sprache, in der Tabellen keine feste Größe haben. Sie lassen die Tabelle einfach wachsen und schrumpfen, wie Sie sie gerade brauchen.



## 12 Funktionen

### 12.1 Übersicht

Funktionen können verwendet werden, um Ihr Programm in mehrere kleinere Codeabschnitte zu teilen, womit die Lesbarkeit und Struktur des Codes erhöht wird. Eine Funktion kann als kleines eigenes Programm angesehen werden. Es kann Variablen verwenden, die lokal sind. Das bedeutet, dass sie nur innerhalb der Funktion zur Verfügung stehen und von außen nicht auf sie zugegriffen werden. Natürlich können Sie auf globale Variablen aus einer Funktion zugreifen. Synonym für den Begriff "Funktion" sind die Begriffe wie "Prozedur", "(Unter)Routine" oder "Anweisung". Funktionen können nichts oder eine beliebige Anzahl von Werten beliebigen Typs zurückgeben. Beachten Sie, dass die Befehle von Hollywood auch Funktionen sind.

Sie können Ihre eigenen Funktionen unter Verwendung der Anweisungen `Function` und `EndFunction` erstellen:

```
Function p_Add(a, b)
    Return(a + b)
EndFunction
c = p_Add(5, 2)           ; c erhält den Wert 7
```

Sie sollten immer das Präfix `p_` in den eigenen Funktionsnamen verwenden, vor allem dann, wenn Sie englische Namen verwenden. Dies hilft, zwischen den eigenen Funktionen und die in Hollywood integrierten Befehlen zu unterscheiden. Auch in zukünftigen Versionen von Hollywood könnte es neue Befehle geben, die den gleichen Namen wie ihre Funktionen in Ihrem Code haben. Dies könnte zu unerwarteten Ergebnissen führen. So sollten Sie immer `p_` in Ihren Funktionsnamen verwenden, so dass keine Verwirrungen entstehen. Das `p_` steht für "Private Funktion".

Funktionen werden deklariert, bevor sie aufgerufen werden, so dass der folgende Code Ihnen ein Fehler melden wird:

```
c = p_Add(5, 2)
Function p_Add(a, b)
    Return(a + b)
EndFunction
```

Hollywood wird versuchen, die Funktion `p_Add()` aufzurufen, wird sie aber nicht finden, weil sie noch nicht deklariert wurde. Die beiden Variablen `a` und `b` werden in der Funktion lokal sein, was bedeutet, dass Sie nur in der `p_Add()` Funktion auf sie zugreifen können. Wenn Sie mehr Argumente an die Funktion übergeben, als sie es erwartet,

```
c = p_Add(5, 2, 4)       ; c erhält den Wert 7
```

dann werden alle überflüssigen Argumente verworfen. In diesem Fall wird das Argument Nummer 3 weggeworfen. Wenn Sie weniger Argumente an die Funktion übergeben, als es erwartet,

```
c = p_Add(5)             ; c erhält den Wert 5, weil 5 + Nil = 5
```

dann wird Hollywood den besonderen Wert `Nil` für alle Argumente verwenden, die keinen Wert erhalten haben.

Funktionen können Werte mit der **Return()**-Anweisung zurückgeben. Es ist unbedingt erforderlich, dass die Rückgabewerte in Klammern eingeschlossen sind. Wenn Sie mehrere Werte zurückgeben müssen, trennen Sie diese einfach durch Kommata. Beispielsweise:

```
Function p_SomeValues()
    Return(5, 6, 7, 8, 9, 10)
EndFunction
```

Wenn Sie Funktionen mit mehreren Rückgabeargumenten aufrufen und Sie nicht alle abfragen/zuweisen, werden diese unzugewiesenen Werte verworfen:

```
a, b, c = p_SomeValues()
```

Die Zeile weist 5 **a**, 6 **b** und 7 **c** zu. Die Rückgabewerte 8, 9 und 10 werden verworfen. Wenn Sie mehr Variablen angeben, als die Funktion zurückgeben kann, werden die überflüssigen Variablen den besonderen Wert **Nil** erhalten:

```
a, b, c, d, e, f, g, h = p_SomeValues() ; "g" und "h" sind Nil
```

Diese Zeile hat zwei überflüssige Variablen **g** und **h**. Sie werden den Wert **Nil** erhalten, weil die Funktion **p\_SomeValues()** nur sechs Werte zurückgibt.

Natürlich können Sie auch Funktionen definieren, die keinen Wert zurückgeben. Beispielsweise:

```
Function p_WaitSecs(s)
    Wait(s, #SECONDS)
EndFunction
```

Wenn Sie versuchen, hier einen Rückgabewert zu erhalten, bekommen Sie nur **Nil** zurück:

```
a = p_WaitSecs(5)
```

Die Variable **a** wird **Nil** beinhalten, weil **p\_WaitSecs()** keinen Wert zurückgibt.

## 12.2 Funktionen sind Variablen

In Hollywood sind Funktionen nur Variablen vom Typ Funktion. Daher können Sie leicht anderen Variablen zuweisen, z.B.:

```
myfunc = DisplayBrush          ; weist "myfunc" DisplayBrush zu
myfunc(1, #CENTER, #CENTER) ; ruft DisplayBrush(1, #CENTER,...
                           ; ... #CENTER) auf
```

Sie können sogar die Definition einer Funktion als Auftrag schreiben:

```
p_Add = Function(a, b) Return(a + b) EndFunction
c = p_Add(5, 2)    ; c erhält 7
```

Die Definition von **p\_Add()** in der ersten Zeile ist die gleiche wie wenn Sie schreiben:

```
Function p_Add(a, b)
    Return(a + b)
EndFunction
```

Sie könnten auch Hollywood-Befehle mit Ihrem eigenen Funktionen ersetzen. Wenn Sie z.B. wollen, dass alle **Print()**-Aufrufe stattdessen **DebugPrint()** verwenden, könnte der folgende Code dies zu tun:

```
Function p_Print(...)
    DebugPrint(Unpack(arg))    ; leitet die Argumente an DebugPrint() um
```

```
EndFunction
Print = p_Print          ; alle Print() rufen nun p_Print() auf
Print("Hello World!")    ; Print() leitet nun nach p_Print() um
```

Was die in Klammer eingeschlossenen drei Punkte ... bedeuten, können sie unter **Variable Anzahl von Argumenten (...)** nachlesen.

Oder eine noch einfachere Lösung:

```
Print = DebugPrint      ; leitet alle Aufrufe von Print()...
                        ; ... nach DebugPrint() um
Print("Hello World!") ; ruft direkt DebugPrint() auf
```

## 12.3 Callback-Funktionen

Bei mehreren Hollywood-Befehlen können Sie Callback-Funktionen angeben. Callback-Funktionen sind normale Funktionen mit dem Unterschied, dass sie nicht durch das Skript, sondern von Hollywood-Befehlen aufgerufen werden. Sie sind ein integraler Bestandteil von Hollywood und machen Ihr Programm viel flexibler. Das ganze Tasten-, Knopf- und Ereignis-Handler-System in Hollywood stützt sich stark auf Callback-Funktionen. Zum Beispiel wenn der Benutzer eine Taste drückt, dann wird Hollywood die Callback-Funktion, aufrufen, welche Sie für dieses spezielle Ereignis vorgesehen haben. Callback-Funktionen werden einfach als normale Argumente den entsprechenden Hollywood-Befehlen übergeben. Ein Beispiel für einen Hollywood-Befehl, der Callback-Funktionen verwendet, ist **MakeButton()**. Dieser Befehl erwartet ab dem siebten Parameter eine Tabelle, die Callback-Funktionen für die verschiedenen Ereignisse definiert. Mögliche Ereignisse für eine Schaltfläche sind **OnMouseOver**, **OnMouseOut**, **OnMouseDown**, **OnMouseUp**, **OnRightMouseDown** und **OnRightMouseUp**. Wenn Sie bei einem Knopf auf einen linken Mausklick reagieren möchten, verwenden Sie das **OnMouseUp**-Ereignis. Hier ist ein Beispiel:

```
MakeButton(1, #SIMPLEBUTTON, 0, 0, 100, 100, {OnMouseUp = p_MyFunc})
```

Dieser Befehl erstellt eine neue Schaltfläche mit der ID 1 und legt fest, dass die Funktion **p\_MyFunc()** aufgerufen wird, wenn der Benutzer diesen Knopf drückt. In diesem Fall ist **p\_MyFunc()** eine Callback-Funktion. Sie wird nicht von Ihnen, sondern von Hollywood aufgerufen, wenn der Benutzer den Knopf drückt. Dies funktioniert automatisch. Genauer gesagt werden die Callback-Funktionen tatsächlich durch den Hollywood Befehl **WaitEvent()** aufgerufen, den Sie in jedem Skript verwenden sollten. Die Callback-Funktion selbst könnte nun wie folgt aussehen:

```
Function p_MyFunc()
    DebugPrint("Button 1 pressed!")
EndFunction
```

Sie könnten diese Funktion auch direkt in der Liste der Argumente von **MakeButton()** platzieren. Dies würde dann so aussehen:

```
MakeButton(1, #SIMPLEBUTTON, 0, 0, 100, 100, {OnMouseUp =
    Function() DebugPrint("Button 1 pressed!") EndFunction})
```

Sie sehen, dass Hollywood sehr flexibel ist. Denken Sie daran, dass wenn Sie Funktionen innerhalb einer Argumentenliste definieren, Sie den Namen einer Funktion nicht angeben dürfen, weil diese Funktionen anonym sind. So ist der folgende Code ungültig:

```
; ungültiger Code!
```

```
MakeButton(1, #SIMPLEBUTTON, 0, 0, 100, 100, {OnMouseUp =
    Function p_MyFunc() DebugPrint("Button 1 pressed!") EndFunction})
```

Callback-Funktionen erhalten in der Regel eine Nachrichtentabelle in Parameter 1. Im obigen Beispiel mussten wir diese Nachricht nicht abfragen, weil wir die Funktion ohne Argumente definiert haben. Für den obigen Code ist dies in Ordnung, aber im folgenden Beispiel sieht es anders aus:

```
MakeButton(1, #SIMPLEBUTTON, 0, 0, 100, 100, {OnMouseUp = p_MyFunc})
MakeButton(2, #SIMPLEBUTTON, 200, 0, 100, 100, {OnMouseUp = p_MyFunc})
```

Jetzt haben wir zwei Tasten erstellt, die beide die gleiche Funktion aufrufen, wenn der Benutzer sie drückt. Die Funktion `p_MyFunc()` muss nun wissen, welche Taste gedrückt wurde, wenn sie aufgerufen wird. `p_MyFunc()` kann dies, indem man die Nachricht in Argument 1 auswertet:

```
Function p_MyFunc(msg)
    If msg.id = 1
        DebugPrint("Button 1 pressed!")
    ElseIf msg.id = 2
        DebugPrint("Button 2 pressed!")
    EndIf
EndFunction
```

Sie sehen, `p_MyFunc()` prüft das `id`-Feld der Nachricht aus dem Argument 1 und so kann es zwischen Taste 1 und 2 unterscheiden. Natürlich könnten Sie das Ganze um eine beliebige Anzahl von Tasten erweitern. Aber da gibt es noch mehr zu erfahren. Betrachten Sie die folgende Situation:

```
evtable = {OnMouseUp = p_MyFunc, OnRightMouseUp = p_MyFunc}
MakeButton(1, #SIMPLEBUTTON, 0, 0, 100, 100, evtable)
MakeButton(2, #SIMPLEBUTTON, 200, 0, 100, 100, evtable)
MakeButton(3, #SIMPLEBUTTON, 400, 0, 100, 100, evtable)
```

Jetzt haben wir drei Tasten erstellt und sie alle nutzen die gleiche Ereignistabelle. So wird Hollywood die gleiche Funktion für alle Tasten aufrufen. Darüber hinaus reagieren diese Tasten auf ein anderes Ereignis, nämlich `OnRightMouseUp`. Jetzt muss `p_MyFunc()` in der Lage sein, nicht nur zwischen mehreren Tasten, sondern auch zwischen den verschiedenen Ereignissen zu unterscheiden. Aber das ist alles kein Problem, weil die übergebene Nachricht an `p_MyFunc()` noch ein anderes Feld hat, von dem Sie das Ereignis lesen können. Unsere Funktion `p_MyFunc()` sieht nun wie folgt aus:

```
Function p_MyFunc(msg)
    Switch msg.action
    Case "OnMouseUp":
        DebugPrint("Left mouse button pressed:", msg.id)
    Case "OnRightMouseUp":
        DebugPrint("Right mouse button pressed:", msg.id)
    EndSwitch
EndFunction
```

Sie sehen also, dass es kein Problem ist, mehrere Schaltflächen und Ereignisse mit der gleichen Callback-Funktion zu behandeln. Dies erhöht die Lesbarkeit des Programms um

ein vielfaches. Es gibt noch viel mehr zu entdecken. Darum lesen Sie die Dokumentation über `MakeButton()` durch.

Wenn Sie benachrichtigt werden möchten, falls der Benutzer das Fenster schließt oder verschiebt, können Sie eine Callback-Funktion mit dem `InstallEventHandler()` installieren. Die Funktion, die Sie diesem Hollywood-Befehl übergeben, wird dann jedes Mal aufgerufen werden, wenn der Benutzer den Schließknopf (Close-Gadget) drückt oder das Fenster verschiebt. Aber `InstallEventHandler()` unterstützt mehrere Ereignistypen: Sie können auch eine Callback-Funktion installieren, die aufgerufen wird, wenn der Benutzer eine Taste auf der Tastatur drückt oder loslässt und vieles mehr.

Der Hollywood-Befehl `SetInterval()` verwendet auch Callback-Funktionen. Der Befehl `SetInterval()` wird immer wieder im angegebenen Intervall aufgerufen werden. Dies ist nützlich, wenn Sie sicherstellen möchten, dass Ihr Skript auf jedem System mit der gleichen Geschwindigkeit abläuft. Um dies zu realisieren, verwenden Sie einfach `SetInterval()`, um Hollywood mitzuteilen, dass Ihre Callback-Funktion 25-mal pro Sekunde ausgeführt werden soll. So können Sie sicherstellen, dass es nicht rascher auf schnellere Maschinen läuft. Siehe `SetInterval()` für einen guten Überblick über die Intervall-Technik.

`SetTimeout()` ist ein weiteres Beispiel für einen Hollywood-Befehl, der mit Callback-Funktion arbeitet. Sie übergeben eine Funktion sowie einen Timeout-Wert auf `SetTimeout()`. Ihre Callback-Funktion wird genau nach Ablauf der vorgegebenen Zeit aufgerufen. Dies ist sehr nützlich für das korrekte Timing des Skripts, z.B. Timing des Skripts mit der Musik.

Der Hollywood Befehl `CopyFile()` übernimmt eine Funktion im vierten Parameter. Diese Funktion wird von Zeit zu Zeit aufgerufen, während `CopyFile()` Dateien kopiert. Dies ist ein Unterschied zu den Callback-Funktionen von `MakeButton()`, `SetInterval()`, `SetTimeout()` usw. Diese werden immer durch `WaitEvent()` und nicht durch den Hollywood-Befehl selbst aufgerufen. `CopyFile()` wird jedoch die angegebene Funktion aufrufen, während es ausgeführt wird.

So könnten Sie Ihre Callback-Funktion nach erfolgreichem `CopyFile()` löschen (Sie können Funktion mit `Nil` löschen). Dies ist mit `MakeButton()` oder `SetInterval()` nicht möglich, da diese Befehle nur die Callback-Funktionen installieren, aber Sie sie nicht aufrufen. Diese Aufgabe wird von `WaitEvent()` übernommen. Die Callback-Funktion von `CopyFile()` wird in der Regel verwendet, um einen Fortschrittsbalken zu aktualisieren oder den Kopiervorgang jederzeit abbrechen zu können. Siehe [Abschnitt 20.6 \[CopyFile\(\)\], Seite 240](#), für Details.

## 12.4 Rückgabewerte (Return)

Wenn Ihre Funktion einen oder mehrere Werte zurückgibt, ist es erforderlich, diese Werte in Klammern zu setzen, weil sonst der Parser sie als separate Anweisungen behandeln würde. Betrachten Sie den folgenden Code:

```
; falscher Code
Function p_Max(a, b)
    If a > b Then Return a
    Return b
EndFunction
```

Der Hollywood-Parser würde diesen Code wie folgt interpretieren:

```
Function p_Max(a, b)
```

```

    If a > b Then Return    ; Wenn a > b, dann gib keinen Wert zurück
    a                      ; führt Funktion a() aus !!
    Return                 ; gibt keinen Wert zurück !!
    b                      ; führt Funktion b() aus !!
EndFunction

```

Sie sehen, dass dies nicht viel Sinn macht. Die Klammern sind obligatorisch, weil in Hollywood können Sie beliebig viele Befehle in einer Zeile ohne Trennzeichen eingeben. Und Hollywood ermöglicht es Ihnen, Funktionen aufzurufen, die Argumente ohne Klammern nicht akzeptieren. Daher wird eine Anweisung wie `Return a` in zwei Anweisungen umgewandelt, nämlich `Return` und ein `a()`. Wenn Sie die Variable `a` zurückgeben wollen, müssen Sie `Return(a)` schreiben. Die richtige Version unserer Funktion `p_max()` hat also wie folgt auszusehen:

```

Function p_Max(a, b)
    If a > b Then Return(a)
    Return(b)
EndFunction

```

Durch die Verwendung der Klammern signalisieren Sie Hollywood, dass die Variablen `a` und `b` zum Befehl `Return()` gehören und nicht zwei getrennte Funktionen sind.

Wenn eine Funktion mehr als einen Wert zurückgibt, aber Sie wollen nur den ersten Rückgabewert haben, müssen Sie rund um den Funktionsaufruf ein Paar runde Klammern setzen. Dadurch wird das Ergebnis dieser Funktion auf einen einzigen Wert reduziert. Zum Beispiel gibt die folgende Funktion drei Werte zurück:

```

Function p_ThreeVals()
    Return(1, 2, 3)
EndFunction

```

Wenn Sie diese Funktion nun als Argument an einen Befehl anhängen, die eine mehrfache Anzahl von Argumenten akzeptiert, z.B. `DebugPrint()`, dann werden alle drei Rückgabewerte an `DebugPrint()` weitergegeben:

```

DebugPrint(p_ThreeVals()) ; gibt "1 2 3" aus

```

Wenn Sie wollen, dass `DebugPrint()` nur den ersten Rückgabewert von `p_ThreeVals()` erhält, müssen Sie Klammern um die Funktion `p_ThreeVals()` setzen, so dass es wie folgt aussieht:

```

DebugPrint((p_ThreeVals())) ; gibt "1" aus

```

Funktionen können nicht nur Zahlen zurückgeben, sondern auch Zeichenketten, Tabellen und sogar andere Funktionen. Zum Beispiel ist der folgende Code völlig legal:

```

Function p_Return_a_Table()
    Return({1, 2, 3, 4, 5})
EndFunction
a = p_Return_a_Table()
DebugPrint(a[3])          ; gibt 4 aus

```

In der Praxis werden Sie wahrscheinlich diese Funktion nicht sehr viel verwenden. Aber Sie sollten wissen, dass es zumindest möglich ist, Funktionen zu haben, die Tabellen oder andere Funktionen zurückgeben können. Ein anderes Beispiel:

```

Function p_Return_Func()

```

```

    Return(Function(s) DebugPrint(s) EndFunction)
EndFunction
myfunc = p_Return_Func()
myfunc("Hello World!") ; führt DebugPrint("Hello World!") aus

```

## 12.5 Rekursive Funktionen

Hollywood unterstützt rekursive Funktionen. Das heißt Sie können Funktionen schreiben, die sich selbst aufrufen. Zum Beispiel ist hier eine Funktion, die die Fakultät von *n* berechnet:

```

Function p_Fac(n)
    If n = 0 Then Return(1) ; 0! = 1
    Return(n * p_Fac(n - 1)) ; multipliziert n mit n - 1 bis n = 0
EndFunction

```

Wie Sie oben sehen können, ruft die Funktion `p_Fac()` sich selbst wieder und wieder auf, bis der Zähler *n* gleich Null ist. Dies ist, was wir eine rekursive Funktion nennen.

## 12.6 Variable Anzahl von Argumenten (...)

Sie können auch Funktionen schreiben, die eine beliebige Anzahl von Argumenten akzeptieren. Um dies zu tun, müssen Sie den `...`-Bezeichner als letzten Parameter verwenden. Ihre Funktion wird dann eine lokale Tabelle `arg` erhalten, die alle Parameter enthält, die für Ihre Funktion einschließlich eines Elements *n* übergeben wurden. *n* trägt die Anzahl von Parametern, die an die Funktion weitergeleitet wurde. Bitte beachten Sie auch, dass die Argumente in der Tabelle `arg` bei Index 0 beginnend gespeichert werden. Als Beispiel ist hier eine Funktion, die den Durchschnitt aller Parameter berechnet, die an sie übergeben werden:

```

Function p_Average(...)

    Local pars = arg.n ; wie viele Parameter wurden übergeben
    Local avg, k ; temporäre lokale Variablen

    For k = 1 To pars
        avg = avg + arg[k-1] ; summiert alle Parameter
    Next

    Return(avg / pars) ; und teile die Summe durch ihre Menge

EndFunction

a = p_Average(10, 20, 30, 40, 50) ; (10 + 20 + 30 + 40 + 50) / 5 = 30
b = p_Average(34, 16, 27, 39) ; (34 + 16 + 27 + 39) / 4 = 29
c = p_Average(10, 10) ; (10 + 10) / 2 = 10
Print(a, b, c) ; gibt "30 29 10" aus

```

Es ist wichtig zu beachten, dass der `...`-Bezeichner als letzten Eintrag in Ihrer Parameterliste angegeben werden muss. Sie können nicht Dinge tun, wie:

```

; falscher Code

```

```
Function p_Test(a, b, ..., c)
...
EndFunction
```

Dies funktioniert natürlich nicht, weil Hollywood nie wissen kann, welcher Parameter zu c gehört. Verwendung von Parametern, bevor der ...-Bezeichner steht, ist jedoch in Ordnung:

```
Function p_MinMax(ismin, ...)

    Local pars = arg.n          ; wie viele Parameter wurden übergeben
    Local k

    If ismin = True              ; kleinstes Element herausfinden
        Local min = arg[0]      ; hier das kleinste Element speichern
        For k = 2 To pars       ; Durchlaufen aller Elemente
            If arg[k-1] < min Then min = arg[k-1] ; kleiner?
        Next
        Return(min)             ; und gibt das kleinste zurück
    Else
        Local max = arg[0]      ; hier das grösste Element speichern
        For k = 2 To pars       ; Durchlaufen aller Elemente
            If arg[k-1] > max Then max = arg[k-1] ; größer ?
        Next
        Return(max)             ; und gibt das grösste Element zurück
    EndIf

EndFunction
```

```
a = p_MinMax(True, 4, 8, 2, 3, 10, 1, 7, 9, 5, 6) ; gibt 1 zurück
b = p_MinMax(False, 4, 8, 2, 3, 10, 1, 7, 9, 5, 6) ; gibt 10 zurück
```

Diese Funktion gibt die kleinste Anzahl der angegebenen Parameter zurück, wenn das erste Argument TRUE oder die größte Zahl, wenn das erste Argument FALSE ist.

Wenn Sie alle Argumente einer anderen Funktion übergeben müssen, kann der Befehl **Unpack()** praktisch sein. Damit werden alle Elemente einer Tabelle zurückgegeben. Wenn Sie zum Beispiel Ihre eigene **Print()** Funktion schreiben möchten:

```
Function p_Print(...)
    Print(Unpack(arg))
EndFunction
```

Alle übergebenen Argumente von **p\_PRINT()** werden an **Print()** mit Hilfe des Befehls **Unpack()** weiter gereicht.

## 12.7 Funktionen in Tabellenelemente

Wie wir bereits gelernt haben, sind Funktionen in Hollywood nur Variablen vom Typ "Function". Daher können Sie sie überall wie die Variablen verwenden. Dazu gehören auch Tabellen. Sie können Funktionen wie normale Zeichenketten oder Werte in einer Tabelle speichern und von dort aufrufen. Lassen Sie uns ein Beispiel anschauen:

```
mathlib = {} ; erstellt eine leere Tabelle
```



```
Function mathlib.add(a, b)
    Return(a + b)
EndFunction
```

```
Function mathlib.sub(a, b)
    Return(a - b)
EndFunction
```

```
Function mathlib.mul(a, b)
    Return(a * b)
EndFunction
```

```
Function mathlib.div(a, b)
    Return(a / b)
EndFunction
```

```
a = mathlib.mul(5, 10) ; a erhält den Wert 50
```

Die Tabelle `mathlib` enthält vier Funktionen, von wo aus sie aufgerufen werden. Natürlich könnte man auch die Funktionen während der Initialisierung der Tabelle erstellen. Dies würde wie folgt aussehen:

```
mathlib = {add = Function(a, b) Return(a + b) EndFunction,
           sub = Function(a, b) Return(a - b) EndFunction,
           mul = Function(a, b) Return(a * b) EndFunction,
           div = Function(a, b) Return(a / b) EndFunction}
a = mathlib.mul(5, 10) ; a erhält den Wert 50
```

Dieser Code funktioniert gleich wie der Code oben, ist aber kompakter. Funktionen innerhalb einer Tabelle werden auch oft als "Methoden" bezeichnet. Dies ist ein Begriff aus der Welt der objektorientierten Programmierung.

## 12.8 Lokale Funktionen

Da Funktionen in Hollywood nur Variablen vom Typ "Function" sind, können Sie auch lokale Funktionen nutzen, die eine begrenzte Lebensdauer haben. Sie arbeiten so ziemlich auf die gleiche Art und Weise wie **lokale Variablen** und haben auch die gleichen Vorteile. Hier ist ein Beispiel für eine lokale Funktion:

```
Block
    Local p_Add = Function(a, b) Return(a + b) EndFunction
    Print(p_Add(5, 6)) ; gibt 11 aus
EndBlock
```

In dem obigen Code wird die Funktion `p_Add()` im Block als lokal erstellt. Daher können alle anderen Versuche, `p_Add()` nach der **EndBlock**-Anweisung aufzurufen zu einem Fehler führen.

Sie könnten auch den obigen Code so schreiben:

```
Block
    Local Function p_Add(a, b) Return(a + b) EndFunction
```

```
Print(p_Add(5, 6))      ; gibt 11 aus
EndBlock
```

Lokale Funktionen können auch hilfreich sein, wenn Sie vorübergehend einen Hollywood-Befehl ersetzen wollen. Beispielsweise ersetzt der folgende Code den Befehl `DebugPrint()` mit dem Befehl `Print()` nur für die Lebensdauer des Blocks:

```
If error = True
    Local Function DebugPrint(...) Print(Unpack(arg)) EndFunction
    DebugPrint("An error occurred!") ; Umleitung zu Print()
EndIf
DebugPrint("Hello")                ; verweist wieder auf DebugPrint()
```

Die Zeichenkette "An error occurred!" wird oben im Code auf Ihrem Display wiedergegeben werden, da wir eine lokale Funktion definiert haben `DebugPrint()`, die der Hollywood-Befehl `Print()` aufruft. Diese lokale `DebugPrint()` wird gelöscht, wenn der Block verlassen wird. Der folgende Aufruf von `DebugPrint()` wird den realen Hollywood-Befehl `DebugPrint()` aufrufen.

## 12.9 Methoden

Es ist auch möglich, Hollywood für objektorientierte Programmierung zu nutzen. Hollywood hat nicht das Konzept der Klassen, aber Sie können das Verhalten mit Hilfe von Tabellen und Metatabellen leicht emulieren. Eine für die objektorientierte Programmierung wichtig Sache ist, dass Objektfunktionen üblicherweise einen Handler auf sich selbst als den ersten Parameter erhalten. Dieser Parameter wird in der Regel `self` (selbst) oder `this` (diese) genannt. Natürlich könnten Sie diesen Parameter in Ihrer Funktion emulieren, indem Sie einfach einen Parameter mit dem Namen `self` oder `this` deklarieren. Aber Sie können auch eine spezielle Syntax für die objektorientierte Programmierung gebrauchen, die Hollywood zur Verfügung stellt.

Wenn Sie Ihre Funktionen mit dem Operator Doppelpunkt erstellen, wird Hollywood automatisch einen versteckten `self`-Parameter für Sie initialisieren. Sie müssen ihn nicht ausdrücklich erstellen. Wenn Sie die Doppelpunkt-Syntax verwenden, ist er immer da. Funktionen, die mit der Doppelpunkt-Syntax deklariert werden, werden Methoden genannt, weil sie von einem Wurzelobjekt abhängig sind.

Hier ist ein einfaches Beispiel für ein Verfahren in Hollywood:

```
cart = {items = {}, numitems = 0}

Function cart:AddItem(n$, p)
    self.items[self.numitems] = {name = n$, price = p}
    self.numitems = self.numitems + 1
EndFunction

Function cart:RemoveItem(n$)
    For Local k = 0 To self.numitems - 1
        If self.items[k].name = n$
            RemoveItem(self.items, k)
            self.numitems = self.numitems - 1
        Return
    EndIf
```

```

    Next
EndFunction

Function cart:Checkout()
    Local total = 0

    For Local k = 0 To self.numitems - 1
        NPrint(self.items[k].name, self.items[k].price)
        total = total + self.items[k].price
    Next

    NPrint("Your total is", total)
EndFunction

cart:AddItem("DVD", 10)
cart:AddItem("Blizzard PPC", 1000)
cart:AddItem("AAA Chipset", 100000)
cart:AddItem("68070", 500)
cart:Checkout()
cart:RemoveItem("Blizzard PPC")
cart:Checkout()

```

Der obige Code erstellt eine einfache Klasse, die einen Einkaufswagen darstellt. Die Klasse hat drei Methoden: Artikel hinzufügen, Artikel entfernen und Check-Out. Darüber hinaus hat sie zwei Eigenschaften: Eine Tabelle enthält eine Liste aller Elemente im Einkaufswagen und ein Zählwert, der die Anzahl Elemente im Wagen zählt. Sie können sehen, dass jeder der drei Methoden mit einer **self** Variable arbeitet, die nicht deklariert wurde. Dies liegt daran, dass alle Methoden mit dem Doppelpunkt-Operator erstellt wurden und damit Hollywood automatisch den **self**-Parameter hinzugefügt hat.

Natürlich gibt es viel mehr in der objektorientierten Programmierung, als in diesem kurzen Ausflug abgedeckt wurde. Sie in die Tiefen der OOP (Vererbung, Mehrfachvererbung, Datenschutz, etc.) einzuführen, würde den Rahmen von diesem Leitfaden sprengen. Aber Sie wissen nun, dass das alles mit Hollywood- und Metatabellen möglich ist. Wenn Ihnen dieses Thema interessiert und Sie mehr lernen wollen, sollten Sie ein Buch über die Lua-Programmierung konsultieren, weil Hollywood einen Lua-Kernel verwendet. Zum Beispiel hat das Buch "Programming in Lua (zweite Auflage)" von Roberto Ierusalimsky ein umfangreiches Kapitel über OOP in Lua, die Sie in der Regel direkt an den Hollywood-Code anpassen können.



## 13 Unicode-Unterstützung

### 13.1 Übersicht

Hollywood 7.0 führt endlich die volle Unicode-Unterstützung ein. Vor Hollywood 7.0 war das Programm unter Windows, Linux und macOS auf ISO 8859-1 beschränkt sowie auf den Standard-Zeichensatz des Systems auf AmigaOS und kompatiblen Systemen. Hollywood 7.0 kommt jetzt mit voller Unicode-Unterstützung, die mit der UTF-8 Zeichencodierung implementiert wird. Also, ab Hollywood 7.0 sollten alle Ihre Skripte in der UTF-8 Zeichencodierung gespeichert werden, entweder mit oder ohne BOM (Byte Order Mark/Bytereihenfolge-Markierung)).

Alle in Zeichenketten gespeicherten Texte werden nun als UTF-8 gespeichert und alle Befehle in den Zeichenketten- und Textbibliotheken werden nun standardmäßig UTF-8 formatierten Text erwarten. Es ist möglich, Hollywood in Legacy-Modus zu setzen, d.h. Hollywood zu zwingen, ISO 8859-1 oder den Standard-Zeichensatz des Systems auf AmigaOS zu verwenden, indem Sie das Konsolenargument `-encoding` oder sein Gegenstück bei der Präprozessor-Anweisung angeben. Aber das wird nicht empfohlen, da im Legacy-Modus Ihr Skript nicht garantiert einwandfrei mit verschiedenen Sprachen/Lokalisierungen laufen wird.

Alle Zeichenketten- und Textbibliothekbefehle, die mit Zeichen in Zeichenfolgen arbeiten müssen, akzeptieren einen optionalen Codierungsparameter, mit dem Sie die Zeichencodierung festlegen können. Normalerweise ist es nicht notwendig, diesen optionalen Codierungsparameter zu verwenden, da dringend empfohlen wird, immer UTF-8 zu verwenden. In manchen Fällen ist es jedoch schwierig Zeichenketten- und Textoperationen mit UTF-8 zu handhaben. Dies gilt insbesondere dann, wenn Sie mit rohen Binärdaten in einer Zeichenkette arbeiten müssen. In diesem Fall können Sie einfach die Konstante `#ENCODING_RAW` an den jeweiligen Befehl übergeben. Somit wird eingestellt, dass Sie innerhalb der Zeichenfolge mit rohen Binärdaten statt in UTF-8-Codierung gespeicherten Zeichen arbeiten möchten. Die Zeichenketten-Befehle führen auf der übergebenen Zeichenfolge keine Integritätsprüfungen durch und werden nur mit den in der Zeichenfolge gespeicherten rohen Binärdaten arbeiten.

Um die Standard-Zeichencodierungen für die Zeichenkette- und Textbibliothek zu ändern, können Sie den Befehl `SetDefaultEncoding()` verwenden. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details. Allerdings ist dies in der Regel nicht erforderlich und Sie sollten nur `#ENCODING_UTF8` als Standard-Codierung verwenden.

Wenn Sie nicht-englische Tastaturzeichen abfragen wollen, müssen Sie den neuen Ereignis-Handler `VanillaKey` mit `InstallEventHandler()` verwenden. `VanillaKey` unterstützt den kompletten Unicode-Bereich von Zeichen, während `OnKeyDown` und `OnKeyUp` nur Steuerzeichen und Standard-Englisch-Tasten unterstützen. Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), [Seite 483](#), für Details.

Im Zuge des Übergangs zu Unicode in Hollywood 7.0 könnte es einige Kompatibilitätsprobleme mit älteren Skripten geben. Alle potenziellen Fragen werden im [Hinweise zur Kompatibilität/API Änderungen](#) besprochen. Siehe [Abschnitt 6.2 \[API Änderungen\]](#), [Seite 73](#), für Details.

Beachten Sie, dass die Textwiedergabe von Hollywood derzeit nur traditionellen Text unterstützt, der von links nach rechts auf horizontalen Linien angelegt wird. Es wird derzeit kein Text unterstützt, der von rechts nach links oder vertikal geht.

## 13.2 Zeichencodierungen

Die meisten Zeichen- und Textbibliotheksbefehle akzeptieren einen optionalen Parameter, der die zu verwendende Zeichencodierung angibt. Dieser Parameter teilt dem Befehl mit, wie die von Ihnen übergebenen Zeichenfolgen intern formatiert sind, d.h. welche Zeichencodierung sie verwenden.

Normalerweise sollten Sie diesen Parameter überhaupt nicht verwenden, denn ab Hollywood 7.0 sollte jeder Text als UTF-8 gespeichert werden. Unter bestimmten Umständen kann es jedoch notwendig sein, den optionalen Zeichencodierungsparameter zu verwenden. Zum Beispiel können Zeichenketten von Hollywood auch rohe Binärdaten enthalten. Diese Daten sind natürlich im UTF-8-Format nicht gültig und die Zeichenketten-Befehle werden sie ablehnen. Darum muss dann diesen Befehlen mitgeteilt werden, dass es sich nicht um UTF-8, sondern um rohe Folge von Bytes handelt. Dies kann durch die Übergabe der Konstante `#ENCODING_RAW` im Zeichencodierungs-Parameter erfolgen.

Hier ist ein Überblick über die verschiedenen Kodierungen in Hollywood:

### `#ENCODING_UTF8:`

Dies ist die Standard-Kodierung ab Hollywood 7.0 und sollte verwendet werden, wann immer Sie mit Texten arbeiten.

### `#ENCODING_ISO8859_1:`

Dies war die Standard-Kodierung vor Hollywood 7.0. Sie wird aus Kompatibilitätsgründen noch unterstützt. Aber es wird nicht empfohlen, sie zu benutzen.

### `#ENCODING_RAW:`

Dies ist ein Synonym für `#ENCODING_ISO8859_1`. Es kann verwendet werden, um den Textbibliotheksbefehlen mitzuteilen, dass sie die Zeichenfolge als rohe Binärdaten anstelle von Text behandeln sollen.

### `#ENCODING_AMIGA:`

Dies legt den Standardzeichensatz des Systems auf AmigaOS und kompatiblen Systemen fest. Diese Konstante wird nur von `ConvertStr()` und nur auf AmigaOS und kompatiblen Systemen unterstützt. `#ENCODING_AMIGA` ermöglicht es Ihnen, zwischen AmigaOS' Standard-Zeichensatz und UTF-8 (oder umgekehrt) zu konvertieren.

Sie können den Befehl `SetDefaultEncoding()` verwenden, um die Standard-Zeichencodierungen für die Zeichenkette- und Textbibliotheken zu ändern. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details.

## 14 Fehlerbehebung

### 14.1 Fehlerbehebung

Dieser Abschnitt umfasst einige häufig auftretende Probleme und ihre Lösungen.

1. **Tabellen initialisieren:** Versuchen Sie nicht eine Tabelle mit einer nie gebrauchten Variable zu initialisieren, denn diese ist **Nil**. Wenn Sie es trotzdem tun, wird das Tabellenfeld nicht angelegt werden. Zum Beispiel funktioniert folgender Code nicht:

```
t = {}      ; Erstellt eine Tabelle
t.x = y     ; weist 'y' dem Feld x zu; Beachte y ist Nil
DebugPrint(t.x) ; ---> Fehler! Feld 'x' ist nicht initialisiert!
```

Die Lösung ist, y zuerst zu initialisieren:

```
t = {}      ; Erstellt eine Tabelle
y = 0       ; setzt y auf 0
t.x = y     ; weist 'y' dem Feld x zu
DebugPrint(t.x) ; Funktioniert! Gibt '0' aus
```

2. **überprüfen, ob eine Variable Nil ist:** Seien Sie vorsichtig, wenn Sie eine Variable kontrollieren, ob sie **Nil** ist. **GetType()** ist die einzige zuverlässige Möglichkeit, um das herauszufinden. Eine direkte Überprüfung mit **Nil** ist keine gute Idee, denn das kann auch True liefern, wenn die Variable Null ist. Beispiel:

```
a = 0
b = Nil
DebugPrint(GetType(a) = #NIL, a = Nil) ; Gibt "0 1" aus
DebugPrint(GetType(b) = #NIL, b = Nil) ; Gibt "1 1" aus
```

Sie sehen, dass "a = Nil" True liefert, obwohl eine Null in a abgelegt ist. Das ist, weil **Nil** immer als Null angesehen wird, wenn es in Ausdrücken verwendet wird. Darum benutzen Sie **GetType()** um herauszufinden, ob eine Variable wirklich **Nil** ist. Ab Hollywood 6.0 können Sie auch den dedizierten Befehl **IsNil()** verwenden, um eine Variable gegen Nil zu überprüfen. Siehe **Abschnitt 50.20 [IsNil]**, **Seite 1103**, für Details.

3. **Falsche Initialisierung von Variablen:** Mehrere Variablen in Hollywood zu initialisieren ist ein bisschen anders als in den meisten anderen Sprachen, weil Hollywood nur ein einziges Gleichheitszeichen erwartet. Zum Beispiel könnte dieser Code richtig aussehen, ist aber falsch:

```
; schlechter Code!
Local a = 5, b = 6, c = 7
```

Leider würde dieser schlechte Code nicht einmal einen Fehler auslösen, aber er würde falsch interpretiert werden. Hollywood würde mit dem obigen Code der Variablen a 5 zuweisen und den Rest einfach ignorieren, weil es nur eine Variable auf der linken Seite des Gleichheitszeichens hat. Dementsprechend wäre die richtige Version der folgende Code:

```
; guter Code!
Local a, b, c = 5, 6, 7
```

Dieser Code würde 5 an a, 6 an b und 7 an c zuweisen.

4. **Werte zurückgeben:** Seien Sie vorsichtig mit Funktionen, die etwas zurückgeben. Der Rückgabewert muss in Klammern eingeschlossen werden. Code wie der folgende ist falsch und löst keinen Fehler aus:

```
Function p_Add(a, b)
    Local r = a + b
    Return r    ; HUCH!!!
EndFunction
```

Dieser Code würde so interpretiert werden, dass **Return** ausgeführt und dann die Funktion **r()** aufgerufen wird. Natürlich wird die Funktion **r()** nicht erreicht werden, aber die oben beschriebene Funktion wird **Nil** zurückgeben, d.H. in jedem Fall nichts. Dieser Code ist die richtige Version:

```
Function p_Add(a, b)
    Local r = a + b
    Return(r)
EndFunction
```

## 14.2 Häufig gestellte Fragen (FAQ)

In diesem Abschnitt stehen einige häufig gestellte Fragen. Bitte lesen Sie diese zuerst durch, bevor Sie in einem Forum nachfragen. Ihr Problem könnte hier schon behandelt worden sein.

**F: Ist es möglich, \*.APK Android-Pakete direkt mit Hollywood zu kompilieren, damit ich sie auf Google Play veröffentlichen kann?**

A: Ja, das ist mit dem Hollywood-APK-Compiler möglich, der als Hollywood-Add-on zur Verfügung steht. Bitte besuchen Sie für weitere Informationen das offizielle Hollywood-Portal unter <https://www.hollywood-mal.de/>. Alternativ können Sie auch den frei verfügbaren Hollywood-Player für Android verwenden, wenn Ihre Hollywood-Projekte auf Android ausgeführt werden soll. Siehe [Abschnitt 2.4 \[Mobile Plattformen\]](#), [Seite 23](#), für Details.

**F: Ist es möglich, GUIs in Hollywood zu erstellen, die die nativen Bedienelemente (Widgets) des Betriebssystems verwenden?**

A: Ja, das ist jetzt mit dem Plugin RapaGUI möglich. Mit RapaGUI können Sie native GUIs für AmigaOS und kompatible, Windows, macOS und Linux erstellen. GUI-Layouts können bequem in XML definiert werden. RapaGUI ist kostenlos erhältlich und kann unter <https://www.hollywood-mal.de/> heruntergeladen werden. Wenn Sie Ihr Programm nur für AmigaOS und kompatible benutzen wollen, können Sie auch das Plugin MUI Royale verwenden. Damit können Sie fast die komplette MUI API von Hollywood nutzen und GUIs mit MUI Royale erstellen. Auch diese GUI wird in XML definiert.

**F: Ich habe mein Projekt für macOS kompiliert, aber macOS weigert sich, es zu starten. Was kann ich tun?**

A: macOS ist sehr streng, wenn es um die Ausführung von Apps geht, die nicht von einem registrierten Apple-Entwickler signiert wurden. Normalerweise landen nicht signierte Apps



in der "Quarantäne", was bedeutet, dass Sie sie nicht öffnen können. Sie können jedoch das Quarantäne-Flag einer App löschen, indem Sie die folgende Zeile im Terminal ausführen:

```
xattr -dr com.apple.quarantine /path/to/your/App.app
```

Dann sollten Sie Ihre Anwendung ohne Probleme öffnen können.

**F: 2D-Zeichnen ist zu langsam. Was kann ich tun, um dies zu beschleunigen?**

A: Auf einigen Plattformen nutzt Hollywood die CPU für alle Zeichnungsvorgänge. Dies gewährleistet maximale Kompatibilität mit einer Vielzahl von Hardware. Mit hardwarebeschleunigtem Zeichnen erhöht sich das Risiko von Störungen mit den Grafiktreibern des Host-Betriebssystems. Trotzdem unterstützt Hollywood hardwarebeschleunigtes Zeichnen. Dafür müssen Sie einen Hardware-Doppelpuffer einrichten und Ihre Pinsel als Hardware-Pinsel erstellen. Dann wird die 2D-Zeichnung mit der GPU durchgeführt, womit die Zeichnungen schneller sind. Beachten Sie, dass die Windows-, macOS- und Linux-Versionen im Moment keine eingebaute Unterstützung für Hardware-Doppelpuffer und Hardware-Pinsel haben. Sie müssen ein Plugin wie GL Galore oder RebelSDL verwenden, um auf diesen Systemen Hardware-Doppelpuffer und Hardware-Pinsel zu verwenden. Siehe [Abschnitt 43.37 \[Hardware-Pinsel\]](#), [Seite 940](#), für Details.

**F: Ich benutze die Autoskalierung (oder den "FullScreenScale" Display-Modus), um mein Skript in einer völlig anderen Auflösung zu betreiben. Die Leistung ist sehr schlecht. Kann Hollywood nicht die GPU zum Skalieren verwenden?**

A: Hollywood unterstützt standardmäßig nicht auf allen Plattformen die hardwarebeschleunigte Skalierung. Unter Windows ist dies nur unter Windows 7 und höher verfügbar. Unter macOS kann sie erst ab 10.10 und höher benutzt werden. Wenn Sie Hollywood auf einem System betreiben, auf dem die Autoskalierung eine schlechte Leistung zeigt, können Sie möglicherweise massive Geschwindigkeitsverbesserungen erzielen, indem Sie ein Plugin verwenden, welches hardwarebeschleunigte Skalierung unterstützt, wie z.B. die GL Galore- oder RebelSDL-Plugins. Plugins, die hardwarebeschleunigte Skalierung unterstützen, können die Autoskalierung in kürzester Zeit anwenden. Wenn also die interne Autoskalierungs-Leistung von Hollywood für Ihre Anforderungen zu schlecht ist, sollten Sie ein Plugin verwenden, das hardwarebeschleunigte Skalierung unterstützt. Siehe [Abschnitt 5.4 \[Vorhandene Plugins\]](#), [Seite 68](#), für Details.

**F: Die Windows-Version von Hollywood hat eine schöne IDE. Warum gibt es keine solche IDE auf der Amiga Plattform?**

A: Es wäre zu viel Arbeit, eine solche IDE für Amiga und kompatible zu erstellen. Es gibt bereits mehrere andere Programme, die Sie auf dem Amiga verwenden können, um Skripte für Hollywood zu erstellen. Schauen Sie sich Cubic IDE von Dietmar Eilert (Link: <http://devplex.awardspace.biz>) oder Codebench von Simon Archer (Link: <http://codebench.co.uk>) an. Beide Programme unterstützen Hollywood durch Plugins.

**F: Wie kann ich Bilder, Sounds, Schriften usw. in meine kompilierte ausführbare Datei einbinden?**

A: Dies ist mit den Präprozessor-Anweisungen wie `@BRUSH`, `@BGPIC`, `@MUSIC`, `@FONT` etc. möglich. Die externen Daten werden dann automatisch in die ausführbare Datei eingebunden. Wenn Sie hingegen Ihre externe Daten mit `LoadBrush()`, `OpenMusic()`, `SetFont()` etc. manuell laden, dann müssen Sie das Konsole Argument `-linkfiles` angeben, damit die Dateien mit der ausführbaren Datei verknüpft werden.

**F: Wie wechsle ich zwischen Fenster- und Vollbildmodus?**

A: Es gibt ein Tastaturkürzel (Hotkey), mit dem Ihre Skripte zwischen Fenster- und Vollbildmodus wechseln können: Drücken Sie einfach `CMD+RETURN` auf AmigaOS und macOS oder `LALT+RETURN` auf Windows. Wenn Sie den Modus von Ihrem Skript wechseln wollen, verwenden Sie den Befehl `ChangeDisplayMode()`.

**F: Wie kann ich das Piktogramm von dem Program ändern?**

A: Verwenden Sie dafür die Präprozessor-Anweisung `@APPICON`.

**F: Wenn ich TrueType-Schriftarten verwende, bemerke ich, dass der Text zwischen AmigaOS und Windows oder Windows und macOS oder AmigaOS und macOS etwas anders aussieht. Wie kann ich das beheben?**

A: Wenn Sie TrueType-Text wollen, der auf jeder Plattform genau gleich aussieht, dann verwenden Sie das in Hollywood eingebaute Schriftartenmodul. Sie können dieses Modul aktivieren, indem sie den Parameter `#FONTENGINE_INBUILT` zusammen mit den Befehlen `SetFont()`, `OpenFont()` oder `@FONT` verwenden. Standardmäßig wird Hollywood das Schriftartenmodul der nativen Host-Betriebssystem für die Schrift verwenden (`#FONTENGINE_NATIVE`). Das führt auf jeder Plattform zu einem etwas anderem Aussehen. Wenn Sie das nicht möchten, verwenden Sie `#FONTENGINE_INBUILT`.

**F: Wie kann ich die Leistung von meinem Skript erhöhen?**

A: Vielleicht möchten Sie "LineHook" für kurze Zeit deaktivieren, um die Leistung der virtuellen Hollywood-Maschine zu erhöhen. Weitere Details finden Sie unter `DisableLineHook`.

**F: Ich habe mein Skript für macOS kompiliert. Aber wenn ich es unter macOS starte, erhalte ich eine Fehlermeldung, dass die Datendateien für mein Programm nicht gefunden werden konnte!**

A: Stellen Sie sicher, dass Sie alle Datendateien von Ihrem Programm in den `Ressourcen` Ordner des Programmpackets gestellt haben. Zum Beispiel, wenn Hollywood ein Paket namens `MyCoolProgram.app` kompiliert, dann müssen Sie alle Datendateien in den Paketordner kopieren: `MyCoolProgram.app/Contents/Resources` Dann wird es funktionieren.

**F: Ich habe mein Skript für macOS/Linux kompiliert, aber es startet nicht. Was ist da falsch?**

A: Stellen Sie sicher, dass beim Hauptprogramm innerhalb des Programmpackets das executable Flag gesetzt ist. Bei Cross-Kompilieren von Programmen für macOS sowie Linux

auf Windows bzw. AmigaOS, wird das executable/ausführbare Flag oft nicht richtig eingestellt, weil macOS und Linux ein anderes Dateisystem verwenden. So müssen Sie manchmal dieses Flag manuell setzen.

**F: Gibt es einen visuellen Designer für Hollywood Skripte oder muss ich jedes Skript mit einem Texteditor bearbeiten?**

A: Ja, es gibt ein Programm namens Hollywood Designer, das eine leistungsfähige WYSIWYG-Schnittstelle hat, um Ihre eigenen Hollywood-Projekte zu erstellen. Werfen Sie einen Blick auf <https://www.hollywood-mal.de/> für weitere Informationen zu diesem großen Programm. Bitte beachten Sie, dass Hollywood Designer derzeit nur für Amiga-kompatible Systeme verfügbar ist.

**F: Gibt es ein Hollywood-Forum, in dem ich in Kontakt mit anderen Benutzern bekomme?**

A: Ja, es existieren ein englisch- sowie deutschsprachiges Forum und eine Mailingliste. Siehe Kapitel [Forum](#) für mehr Informationen.

**F: Wenn ich mein Skript für Windows/macOS kompilieren, erhalte ich eine Fehlermeldung, dass Hollywood nicht die von mir benutzten Schriften öffnen kann! Was mache ich falsch?**

A: Siehe [Arbeiten mit Schriften](#) für eine detaillierte Erklärung, wie Sie mit Schriften in Multi-Plattform-Skripten umgehen müssen.

**F: Ich sehe, dass durch Hollywood kompilierte Programme viele verschiedene Konsolenargumente unterstützen. Aber ich starte/kompiliere meine Hollywood Programme nie von der Konsole aus! Kann ich die Konsolenargumente trotzdem verwenden?**

A: Ja, das ist möglich. Siehe [Konsolenargumente ohne Konsolen](#) für weitere Details.

**F: Beim Versuch eine Animation zu laden, erhalte ich immer ein "Out of Memory" Fehler, obwohl ich 512 MB RAM habe.**

A: Stellen Sie sicher, dass Sie die diskgepufferte Wiedergabe ermöglichen. Dies kann unter Verwendung der Angaben von `FromDisk` in `LoadAnim()` oder `@ANIM` erfolgen. Wenn Sie den Tag `FromDisks` nicht angeben, wird Hollywood die gesamte Animation im RAM Speicher ablegen und weil Hollywood immer 32-Bit-Grafik verwendet, werden 512 MB ziemlich schnell verbraucht.

**F: Wenn ich auf ein nicht vorhandenes Feld in meiner Tabelle zugreife, beendet Hollywood sofort das Programm mit einer Fehlermeldung! Kann ich irgendwie überprüfen, ob ein Tabellenfeld vor dem Zugriff existiert?**

A: Das ist mit dem Befehl `HaveItem()` möglich. Es wird `False` zurückgegeben, wenn das angegebene Tabellenfeld nicht existiert.

**F: Der Hollywood Sound wird unter AmigaOS verzerrt. Was ist da los?**

A: Überprüfen Sie Ihre AHI Einstellungen. Setzen Sie im AHI Register "Erweiterte Einstellungen" die Hauptlautstärke auf "Mit Begrenzung". Wenn das nicht hilft, versuchen Sie die

Hauptlautstärke in Hollywood mit Hilfe vom Argument `-mastervolume` zu reduzieren. Sie können auch die Hauptlautstärke in den GUI-Einstellungen verringern. Sie sollten auch die Echo und Surround-Modi ausschalten, wenn es irgendwelche Sound-Probleme sind. Stellen Sie außerdem sicher, dass Sie die Frequenz für Ihre Sound-Treiber richtig eingegeben haben. Sie sollten mindestens 22050 Hz betragen.

**F: Ich möchte Hollywood meine eigenen Befehle über ein Plugin hinzufügen. Steht ein SDK zur Verfügung?**

A: Ja, unter dem offiziellen Hollywood-Portal <https://www.hollywood-mal.de/> können Sie das Hollywood-SDK runterladen. Es beinhaltet viele Beispiele und eine umfangreiche Dokumentation.

**F: Wie kann ich Skripte in einem Fenster unterbrechen, die keinen Schließknopf besitzen?**

A: Drücken Sie einfach CTRL+C. Dies wird immer funktionieren, außer wenn Sie CTRL+C mit dem Befehl `CtrlCQuit()` deaktiviert haben.

**F: Wo kann ich um Hilfe bitten?**

A: Es existieren ein englisch- <https://forums.hollywood-mal.com> sowie deutschsprachiges <https://amiga-resistance.info> Forum. Außerdem können Sie sich auch für den Newsletter anmelden <http://www.hollywood-mal.com> Siehe Kapitel [Forum](#) für mehr Informationen.

**F: Ich habe einen Fehler gefunden.**

A: Bitte schreiben Sie darüber im Bereich "Bug reports" des englischsprachigen Forums.

## 15 Lernprogramme

### 15.1 Tutorial

Dieses kleine Tutorial zeigt Ihnen Schritt für Schritt, wie Sie Ihre eigene Slideshow erstellen können. Versuchen Sie, jeden der hier gemachten Schritte zu verstehen und Sie werden bald in der Lage sein, Ihre eigenen Skripts zu erstellen.

Die folgenden Dinge werden für dieses Tutorial benötigt:

1. Ein Hintergrundbild mit Namen **BG.png** mit zwei Pfeilen. Das Rechteck um Pfeil 1 ist X: 4, Y: 430, W: 35, H: 19. Das Rechteck um Pfeil 2 ist X: 591, Y: 430, W: 35, H: 19. Die Koordinaten, wo die Bilder angezeigt werden sind: X: 29, Y: 41.
2. 11 Bilder mit Namen **0.jpg**, **1.jpg**, **2.jpg** ... **10.jpg** der Größe 571x377 Pixel.
3. Protracker Modul mit Namen **MyMusic.mod**

Natürlich habe ich all das für dieses Tutorial vorbereitet. Sie finden die Dateien in Ihrem Hollywood-Verzeichnis unter **Help/Tutorial**. Bitte kopieren Sie die Dateien in das Verzeichnis, wo Sie Ihr Skript erstellen werden. Dann folgen Sie diesen Schritten:

1. Starten sie Ihren gewohnten Texteditor
2. Der Hintergrund für Ihre Slideshow ist ein Bild, das Sie bereits mit Ihrem bevorzugten Malprogramm erstellt haben. In unserem Beispiel habe ich den Hintergrund vorbereitet.
3. Jetzt müssen wir Hollywood sagen, dass es die Datei **BG.png** als erstes Hintergrundbild benutzen soll. Das wird mit Hilfe der Präprozessor-Anweisung **@BGPIC** zusammen mit **BG.png** erledigt. Also müssen Sie den folgenden Code in Ihr Skript schreiben:

```
@BGPIC 1, "BG.png"
```

Dieser Befehl teilt Hollywood mit, dass es **BG.png** als erstes Hintergrundbild benutzen soll. Das erste Hintergrundbild muss immer den Identifikator 1 haben. Falls es keines mit dem Identifikator 1 gibt, erzeugt Hollywood ein leeres Display.

4. Unsere Slideshow soll außerdem eine Hintergrundmusik enthalten. Diese Musik ist ein Protracker Modul mit Namen **MyMusic.mod**. Also fügen wir die folgende Zeile zu unserem Skript hinzu:

```
@MUSIC 1, "MyMusic.mod"
```

5. Nun müssen wir Gebiete in unserem Hintergrundbild festlegen, die als Knöpfe zur Verfügung stehen sollen. Wie Sie sehen gibt es zwei Pfeile im Hintergrundbild. Nachdem alle Knöpfe als Rechtecke definiert werden müssen, müssen wir die Koordinaten der linken oberen Ecke und der rechten unteren Ecke eines jeden Pfeiles herausfinden. Sie können ein Malprogramm wie PPaint benutzen, um die Koordinaten zu bestimmen. Für unser Hintergrundbild ist der linke Pfeil in einem Rechteck mit den Koordinaten 4:430 (obere linke Ecke) und eine Breite/Höhe von 35/19. Also können wir jetzt den linken Pfeil so als Knopf 1 in unser Skript einfügen:

```
MakeButton(1, #SIMPLEBUTTON, 4, 430, 35, 19,  
{OnMouseUp = p_Back})
```

Dasselbe können wir nun für den rechten Pfeil durchführen, der sich in einem Rechteck mit den Koordinaten 591:430 befindet und dieselbe Größe wie Pfeil 1 hat. Also schreiben wir die folgende Zeile in unser Skript:

```
MakeButton(2, #SIMPLEBUTTON, 591, 430, 35, 19,
```

```
{OnMouseUp = p_Forward})
```

Jetzt haben wir zwei definierte Knöpfe, die angeklickt werden können. Wird Knopf 1 gedrückt, springt Hollywood zur Funktion mit Namen `p_Back()`. Wird Knopf 2 gedrückt, springt Hollywood zur Funktion `p_Forward()`.

6. Jetzt können wir einen Haufen Befehle hinzufügen, die Hollywood sagen, was es zu tun hat. Zuerst wollen wir, dass die Musik zu spielen beginnt. Da wir `MyMusic.mod` als Modul mit der Nummer 1 deklariert haben, rufen wir jetzt `PlayModule()` mit 1 als Argument auf. Fügen Sie die folgende Zeile Ihrem Skript hinzu und Hollywood wird Ihr Protracker-Modul abspielen:

```
PlayMusic(1)
```

Wir müssen auch noch definieren, welches Bild das letzte sein soll. In unserem Beispiel haben wir 11 Bilder, die von `0.jpg` bis `10.jpg` reichen, also ist das letzte Bild das mit der Nummer 10. Deshalb geben wir diese Zeile in unser Skript ein:

```
lastpic = 10
```

Wir fügen auch die folgende Zeile ein, weil unser erstes Bild `0.jpg` ist:

```
pic = 0
```

7. Der nächste Befehl soll das nächste Bild laden und anzeigen. Nachdem es keinen Befehl gibt der das aufs Mal tut, brauchen wir ein kleines Unterprogramm. Sie werden in Schritt 10 sehen, wie Sie diese Funktion schreiben können. Lassen Sie uns für den Moment annehmen, dass es die bereits gibt und deshalb werden wir Hollywood auftragen, sie auszuführen:

```
p_NextPic()
```

8. Nun brauchen wir eine Hauptschleife in unserem Skript. Eine beliebige Form hierfür ist folgende:

```
Repeat
    WaitEvent
Forever
```

Der Befehl `WaitEvent()` hält die weitere Ausführung des Skripts an, bis etwas passiert wie z.B. ein Knopf wird gedrückt. Wenn ein Ereignis eintritt, springt `WaitEvent()` zur Funktion, die dieses Ereignis behandelt. Wird z.B. Knopf 1 gedrückt, springt `WaitEvent()` zu der Funktion `p_Back()`. Wenn die Funktion ihre Arbeit beendet hat, springt Hollywood zurück in die Hauptschleife und `WaitEvent()` wird wieder aufgerufen. Das wird wiederholt, bis der Benutzer das Fenster schließt.

9. Jetzt ist die Struktur unseres Programms komplett. Was wir noch tun müssen ist, die Funktionen `p_Back()` (welche das vorhergehende Bild anzeigen soll, welches bei `WaitEvent()` aufgerufen wird, falls Knopf 1 (rückwärts) gedrückt wurde) und `p_Forward()` (das von `WaitEvent()` für Knopf 2 (vorwärts) aufgerufen wird) hinzuzufügen. Es ist wichtig, dass Sie die Funktionen definieren, bevor Sie auf sie verweisen. So müssen Sie den folgenden Code vor dem Aufrufen von `MakeButton()` hinzufügen. Also lassen Sie uns nun diese beiden Funktionen für die Knöpfe hinzufügen:

```
Function p_Back()
    If pic = 0
        pic = lastpic
```

```

Else
    pic = pic - 1
EndIf
p_NextPic()
EndFunction

Function p_Forward()
    If pic = lastpic
        pic = 0
    Else
        pic = pic + 1
    EndIf
    p_NextPic()
EndFunction

```

Wie Sie im obigen Code sehen können, enthält die Variable `pic` die momentane Bildnummer. Wenn der Benutzer den Vorwärtssknopf anklickt, wird `pic` um eins erhöht, wird der Rückwärtssknopf gedrückt wird `pic` um eins verringert. Die Variable wird auch gegen 0 und `lastpic` geprüft, so dass sie immer im Bereich der Bilder bleiben.

10. Und nun müssen wir noch unsere Funktion `p_NextPic()` erstellen, welche das Bild mit der Nummer in der Variable `pic` lädt und anzeigt. Denken Sie daran, diesen Code einzufügen, bevor der Befehl `MakeButton()` aufgerufen wird.

```

Function p_NextPic()
    LoadBrush(1, pic .. ".jpg")
    DisplayBrush(1, 29, 41)
EndFunction

```

Was macht also die Funktion `p_NextPic()` ? Es hängt ein ".jpg" an `pic` an. Danach lädt es die Datei und stellt den Pinsel bei den Koordinaten 29:41 dar. Deshalb müssen die Bilder auf diese Weise benannt werden 0.jpg (erstes Bild), 1.jpg (zweites Bild), 2.jpg (drittes Bild) und so weiter.

Zusammengesetzt sieht unser Skript jetzt so aus:

```

@BGPIC 1, "BG.png"
@MUSIC 1, "MyMusic.mod"

Function p_NextPic()
    LoadBrush(1, pic .. ".jpg")
    DisplayBrush(1, 29, 41)
EndFunction

Function p_Back()
    If pic =0
        pic = lastpic
    Else
        pic = pic - 1
    EndIf

    p_NextPic()

```

```

EndFunction

Function p_Forward()
    If pic = lastpic
        pic = 0
    Else
        pic = pic + 1
    EndIf

    p_NextPic()
EndFunction

MakeButton(1, #SIMPLEBUTTON, 4, 430, 35, 19,
    {OnMouseUp = p_Back})
MakeButton(2, #SIMPLEBUTTON, 591, 430, 35, 19,
    {OnMouseUp = p_Forward})
PlayMusic(1)

lastpic = 10
pic = 0

p_NextPic()

Repeat
    WaitEvent
Forever

```

Nun können Sie Ihr Skript abspeichern und mit der Hollywood-GUI oder aus der Shell starten. Und das ist bereits Ihre Slideshow! War das nicht einfach? Nur 35 Codezeilen. Nun können sie loslegen und den Code erweitern, wenn Sie möchten. Falls Sie zum Beispiel Übergangseffekte zwischen den Bildern haben möchten, ersetzen Sie einfach die Zeile

```
DisplayBrush(1, 29, 41)
```

mit der Zeile

```
DisplayBrushFX(1, 29, 41, #RANDEFFECT)
```

und Ihr Bild wird mit einem schönen Übergangseffekt aus der breiten Palette von Hollywoods Übergangseffekten erscheinen.

Im englischsprachigen Forum <https://www.hollywood-mal.de/> finden Sie unter Tutorials weitere Lernprogramme.

## 15.2 Animationstechniken

Wenn es um Animationen geht, haben Sie drei Techniken zur Auswahl: Sprites, doppelte Pufferung und Ebenen. Dieser Abschnitt soll Ihnen einen Überblick über die drei Techniken geben, so dass die Entscheidung für Sie einfacher wird.

1. Sprites: Sprites sind besonders nützlich, wenn nicht viele Grafiken gezeichnet werden. Zum Beispiel wenn Sie nur um ein paar Blöcke oder Spieler und Feind Sprites bewegen



müssen. In diesem Fall ist es besser, Sprites zu verwenden, weil Hollywood das Display ziemlich schnell aktualisieren kann, da nicht viele Änderungen gezeichnet werden müssen. Siehe [Abschnitt 49.1 \[Einführung in Sprites\]](#), [Seite 1075](#), für Details.

2. Doppelpufferung: Ein Doppelpuffer in Hollywood muss immer das ganze Display aktualisieren. Obwohl hier wenn möglich die Hardwarebeschleunigung verwendet wird, kann das viele Ressourcen in Anspruch nehmen, wenn ein Display von 640x480 25 Mal pro Sekunde aufgefrischt werden muss. Somit wird Doppelpuffer nur empfohlen, wenn eine große Menge von benutzerdefinierten Grafiken gezeichnet werden müssen. Zum Beispiel, dass Hollywood einen Doppelpuffer bei einer echten Sinuskurve verwendet, weil eine Menge verschiedener Teile gezeichnet werden müssen. Solche Dinge würden mit Sprites nicht möglich sein, weil die Zeichnungsoperationen stark angepasst und jedes Einzelbild geändert werden muss. Siehe [Abschnitt 28.3 \[BeginDoubleBuffer\(\)\]](#), [Seite 568](#), für Details.
3. Ebenen: Hollywood hat ein leistungsfähiges Ebenen-System, so dass Sie auf jedes Grafikelement auf dem Display über seine eigene Ebene zugreifen und im laufenden Betrieb die Position sowie Größe ändern können. Das Ebenensystem ist äußerst flexibel und leistungsstark auf Kosten der Geschwindigkeit. Wenn Sie eine Menge Grafiken zu zeichnen haben, könnte es schneller sein, stattdessen die Doppelpufferung zu verwenden.

Hier ist eine Empfehlung von Animationstechniken, die für die häufigsten Arten von Anwendungen geeignet sind:

Brett-/Kartenspiel:

Sprites oder Ebenen, weil schnelle Grafiken nicht erforderlich sind.

Tetris: Sprites oder Ebenen, da gibt es nicht viele Aktionen und Bildschirm-Updates müssen nicht sehr schnell sein.

PacMan: Sprites oder Ebenen. Das einzige, was sich bewegt, sind die Feinde und der Spieler.

2D shooter:

Doppelpuffer, weil der Hintergrund scrollt. Daher muss der gesamte Bildschirm für jedes Bild aktualisiert werden.

Jump'n'Run:

Doppelpuffer, wenn der Hintergrund gescrollt wird. Wird das Spiel nicht gescrollt, dann Sprites oder Ebenen.

Scene demo:

Doppelpuffer mit allen Mitteln. Viele eigene Grafiken sind zu zeichnen. Dies ist ein klassischer Doppelpuffer-Fall.

Wenn Sie Sprites oder Ebenen verwenden, sollten Sie alle Zeichnungsbefehle eines Bildes zwischen [BeginRefresh\(\)](#) und [EndRefresh\(\)](#) kapseln. Dies ermöglicht Hollywood optimierte Zeichnungen auf Systeme zu verwenden, die nicht Teil-Bildschirmaktualisierung wie Android unterstützen. Als willkommener Nebeneffekt wird mit [BeginRefresh\(\)](#) und [EndRefresh\(\)](#) auch die Zeichnungsgeschwindigkeit verbessert, wenn die Autoskalierung aktiv ist. Siehe [Abschnitt 28.4 \[BeginRefresh\]](#), [Seite 569](#), für Details.

### 15.3 Zeitsteuerung des Skripts

Das richtige Timing ist ein entscheidender Punkt für jedes gute Skript, welches in der Lage sein sollte, auf vielen verschiedenen Systemen laufen zu können. Weil Hollywood für eine Vielzahl von Plattformen verfügbar ist, müssen Sie an die Skript Zeitsteuerung denken, wenn Sie Ihr Skript anderen weitergeben. Das grundlegende Problem ist, dass wenn Sie keinen Geschwindigkeitsbegrenzer Ihrem Skript hinzufügen, es so schnell wie möglich ausgeführt wird. Das könnte kein Problem mit dem 200MHz System sein, aber auf einer 1GHz Maschine ist es sicherlich ein Problem. Stellen Sie sich vor, Sie haben ein Spiel und verwenden die folgende Schleife:

```
/* Schlechter Code */
While quit = False
    dir = p_QueryInput()
    If dir = "Left" Then p_MoveSpriteLeft()
    If dir = "Right" Then p_MoveSpriteRight()
    ....
    p_RedrawDisplay()
Wend
```

Diese Schleife hat zwei schwerwiegende Probleme:

1. Es gibt keine Zeitsteuerung in dieser Schleife. Die Schleife wird immer so schnell ausgeführt, wie es die CPU des Host-Systems gestattet. Das ist wirklich schlecht. Es bedeutet, dass der Zeitpunkt nur auf Ihrem System korrekt sein wird und sonst nirgendwo anders.
2. Diese Schleife wird 100% der CPU-Leistung in Anspruch nehmen, weil es keine Grenzen gibt, die zum Beispiel sagt: "Führen Sie diese Schleife 25-mal pro Sekunde aus und das ist genug!". 100% der CPU-Leistung zu verbrauchen ist möglicherweise auf 68k basierten Systemen kein Problem, aber wenn Sie das Skript auf einem neueren System ausführen, wird das Betriebssystem merklich verlangsamt und der CPU-Lüfter könnte beginnen, einige störende Geräusche zu verursachen. Darüber hinaus denken Sie daran, dass Hollywood in einer Multitasking-Umgebung läuft. Somit werden durch Schleifen wie diese die anderen Aufgaben weniger CPU-Zeit erhalten.

Die Lösung für das Problem ist recht einfach: Wir müssen Hollywood mitteilen, diese Schleife nur mit einer bestimmten Anzahl von Durchläufen pro Sekunde auszuführen. Für die meisten Spiele ist es völlig ausreichend 25-mal pro Sekunde eine Eingabe abzufragen und Grafiken zu zeichnen. Es gibt zwei Methoden, wie Sie ein Drosseln implementieren können:

1. Mit `WaitTimer()`. Dieser Befehl übernimmt eine Uhr und einen Timeout-Wert. Sie pausiert das Skript bis die vorgegebene Zeit abgelaufen ist. Dann wird die Uhr zurückgesetzt und Sie können sie wieder verwenden. Unsere Schleife von oben würde mit `WaitTimer()` wie folgt aussehen:

```
/* Guter Code */
StartTimer(1)      ; Startet die Uhr #1

While quit = False
    dir = p_QueryInput()
    If dir = "Left" Then p_MoveSpriteLeft()
    If dir = "Right" Then p_MoveSpriteRight()
```

```

.....
p_RedrawDisplay()
WaitTimer(1, 40)      ; laufe nicht schneller ...
                      ; ... als 40 Millisekunden

Wend

```

Jetzt wird unsere Schleife nie schneller als 40 Millisekunden ausgeführt. So wird sie nie mehr als 25-mal pro Sekunde ausgeführt werden, weil  $25 * 40\text{ms} = 1000\text{ms}$ . Daher wird ein Spiel mit dieser Schleife immer mit der gleichen Geschwindigkeit auf jedem System laufen - unabhängig davon, ob die CPU 50MHz oder 1GHz hat.

2. Die zweite Methode ist `SetInterval()`. Mit diesem Befehl können Sie eine Intervallfunktion installieren, die Hollywood auf die von Ihnen angegebene Frequenz aufruft. So können Sie Hollywood mitteilen, die Schleife 25-mal pro Sekunde zu durchlaufen. So sieht dieser Code aus:

```

/* Guter Code */
Function p_MainLoop()
    ; dieser Code ist derselbe, wie die While-Wend Schleife oben
    dir = QueryInput()
    If dir = "Left" Then MoveSpriteLeft()
    If dir = "Right" Then MoveSpriteRight()
    ....
    RedrawDisplay()
EndFunction

; ruft MainLoop() 25 Mal in der Sekunde ...
; ... auf -> 40 * 25ms = 1000 Millisekunden
SetInterval(1, p_MainLoop, 40)

While quit = False
    WaitEvent
Wend

```

Dieser Code erledigt das gleiche, wie der Code oben mit `WaitTimer()`. Der einzige Unterschied ist, dass Sie anstelle von `WaitEvent()` `SetInterval()` verwenden müssen, weil Intervallfunktionen Hollywood Ereignisse auslösen.

Beide der oben diskutierten Verfahren sind einfach zu bedienen und effizient. Es ist an Ihnen zu entscheiden, welches Sie bevorzugen.



## 16 Amiga-Bibliothek

### 16.1 Informationen über AmiDock

Hollywood hat native Unterstützung für das AmiDock System in AmigaOS 4. Sie können Ihr Skript im AmiDock erscheinen lassen, indem Sie den Tag `RegisterApplication` bei der Präprozessor-Anweisung `@OPTIONS` auf `True` setzen. Siehe [Abschnitt 50.25 \[OPTIONS\]](#), [Seite 1107](#), für Details.

Standardmäßig wird Hollywood in AmiDock das Piktogramm `*.Info` aus dem Skript oder dem Programm zeigen. Wenn Sie ein benutzerdefiniertes Piktogramm im AmiDock zeigen wollen, können Sie dies tun, indem sie eine Reihe von Piktogrammen mit der Präprozessor-Anweisung `@APPICON` einbinden und dann müssen sie Hollywood mit dem Tag `DefaultIcon` mitteilen, welches Piktogramm angezeigt werden soll. Siehe [Abschnitt 18.5 \[APPICON\]](#), [Seite 219](#), für Details.. Alternativ können Sie auch den Tag `DockyBrush` mit der Präprozessor-Anweisung `@OPTIONS` verwenden.

Hollywood unterstützt zwei verschiedene Arten von Dockies:

1. Standard-Docky: Dies ist der Standard-Docky Typ. Ihr Programm wird als Piktogramm im AmiDock angezeigt werden, welches zwei verschiedene Zustände hat. Der zweite Zustand wird jedes Mal angezeigt, wenn der Benutzer auf das Piktogramm klickt. Standard-Dockies haben den Nachteil, dass sie nicht über ein Kontextmenü verfügen und es braucht auch viel Zeit, das Standard-Docky-Piktogramm zur Laufzeit mit dem Befehl `ChangeApplicationIcon()` zu ändern. Das Aktualisieren des Standard-Docky wird deutlich spürbar, wenn Sie das Piktogramm ändern. Wenn Sie kein Kontextmenü benötigen und Sie nie Ihr Docky-Piktogramm aktualisieren müssen, ist das Standard-Docky die beste Wahl.
2. App-Docky: App-Dockies sind flexibler als Standard-Dockies. Sie können ein Kontextmenü haben und es ist auch möglich, ihr Piktogramm sehr schnell mit dem Befehl `ChangeApplicationIcon()` zu ändern. Dies ermöglicht es, Animationen in AmiDock zu zeigen. Die Kehrseite der App-Dockies ist, dass App-Docky-Piktogramme nur einen einzigen Zustand haben können. Es ist nicht möglich, ein zweites Piktogramm zu verwenden, das angezeigt werden soll, wenn der Benutzer auf das App-Docky klickt.

Standardmäßig wird Hollywood ein Standard-Docky für Sie erstellen. App-Dockies werden nur erzeugt, wenn Sie durch die Angabe des Tags `DockyContextMenu` mit `@OPTIONS` ein Kontextmenü zuweisen oder beim Aufruf von `ChangeApplicationIcon()` nur ein statt zweier Bilder an den Befehl übergeben. Eine alternative Möglichkeit Ihre Anwendung als App-Docky zu integrieren ist, den Tag `DockyBrush` mit der Präprozessor-Anweisung `@OPTIONS` zu verwenden.

Wenn Sie Ihr Skript als OS4-Anwendung ohne ein Piktogramm im AmiDock wollen, setzen Sie bei der Präprozessor-Anweisung den Tag `NoDocky` auf `True`.

### 16.2 CloseAmigaGuide

#### BEZEICHNUNG

CloseAmigaGuide – schließt das aktuelle AmigaGuide-Fenster (V6.1)

**ÜBERSICHT**

CloseAmigaGuide()

**PLATTFORMEN**

Nur AmigaOS und kompatible

**BESCHREIBUNG**

Dieser Befehl schließt ein AmigaGuide-Fenster, das mit dem Befehl [OpenAmigaGuide\(\)](#) geöffnet wurde. Wenn Hollywood heruntergefahren wird, wird dieser Befehl automatisch aufgerufen.

**EINGABEN**

keine

## 16.3 CreateRexxPort

**BEZEICHNUNG**

CreateRexxPort – erstellt einen ARexx-Port für Ihr Skript (V2.5)

**ÜBERSICHT**

CreateRexxPort(name\$)

**PLATTFORMEN**

Nur AmigaOS und kompatible

**BESCHREIBUNG**

Mit diesem Befehl wird ein ARexx-Port für das Skript erstellt und ihm der angegebene Name in `name$` zugewiesen. Um ARexx-Nachrichten mit Ihrem Skript zu erhalten, muss es einen ARexx-Port haben. Andere Anwendungen können dann mit dem Skript kommunizieren, indem sie Nachrichten an diesen Port senden. Alle Nachrichten, die an Ihrem ARexx-Port ankommen, werden an die Callback-Funktion weitergeleitet, die Sie mit dem Befehl [InstallEventHandler\(\)](#) (benutzen Sie den ARexx-Ereignis-Handler) aufrufen können. Wenn Sie diesen Ereignis-Handler nicht installieren, erhalten Sie keine Benachrichtigungen über eingehende Nachrichten.

Bitte beachten Sie, dass der ARexx-Portname immer zwischen Groß- und Kleinschreibung unterscheidet. So bezeichnen "MYPORT" und "myport" zwei verschiedene ARexx-Porte. Aus Stilgründen wird empfohlen, dass Sie nur Großbuchstaben für Ihren Portnamen verwenden. Außerdem muss jeder ARexx-Port eindeutig im System sein. Wenn Sie einen Portnamen angeben, der bereits verwendet wird, schlägt dieser Befehl fehl. Stellen Sie daher sicher, dass Sie einen eindeutigen Namen verwenden.

Bitte beachten Sie, dass jedes Hollywood-Skript nur einen ARexx-Port haben kann. Daher kann dieser Befehl nur einmal in Ihrem Skript aufgerufen werden. Sie können den von diesem Befehl erstellten Port nicht löschen. Er wird automatisch geschlossen, wenn Hollywood beendet wird.

Nachfolgend finden Sie ein Beispiel, wie ARexx-Nachrichten erfasst werden können. Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), [Seite 483](#), für mehr Informationen darüber, wie die Callback-Funktion des Benutzers aufgerufen wird.

Siehe auch [RunRexxScript\(\)](#) und [SendRexxCommand\(\)](#).

**EINGABEN**

name\$      gewünschter Namen für Ihren REXX-Anschluss

**BEISPIEL**

```
Function p_EventFunc(msg)
  Switch msg.action
  Case "OnARexx"
    Switch msg.command
    Case "RealFunc"
      Return(100)
    Default
      Local t = SplitStr(msg.args, "\0")
      DebugPrint(msg.command, "called with", msg.argc, "argument(s)")
      For Local k = 1 To msg.argc
        DebugPrint("Argument", k .. ":", t[k - 1])
      Next
    EndSwitch
  EndSwitch
EndFunction
CreateRexxPort("MY_COOL_REXX_PORT_123")
InstallEventHandler({OnARexx = p_EventFunc})
Repeat
  WaitEvent
Forever
```

Speichern Sie den Code oben als ein Hollywood-Skript und führen Sie es mit Hollywood aus. Speichern Sie dann den folgenden Code als REXX-Skript und führen Sie es von einer Shell mit "RX test.rx": aus.

```
/* Die erste Zeile von jedem REXX-Skript muss ein Kommentar sein */
OPTIONS RESULTS

/* Der Port von unserem Hollywood-Skript ist nun der Host */
ADDRESS MY_COOL_REXX_PORT_123

/* Sendet Befehle von REXX nach Hollywood */
/* Schauen Sie auf die Debug-Ausgabe */
DummyFunc_1 "Dummy Arg 1"
DummyFunc_2 1 2 3
DummyFunc_3 "First arg" "Second arg" "Third arg"
DummyFunc_4 /* no args */
DummyFunc_5 "These will be handled as separate arguments"
DummyFunc_6 "This is a single argument (use double quotes!)"

'RealFunc'
SAY RESULT /* Dies gibt 100 aus; ist das Resultat von RealFunc */
```

## 16.4 GetApplicationList

### BEZEICHNUNG

GetApplicationList – erstellt eine Liste aller registrierten Anwendungen (V6.0)

### ÜBERSICHT

```
t = GetApplicationList()
```

### PLATTFORMEN

Nur AmigaOS 4

### BESCHREIBUNG

Dieser Befehl gibt eine Tabelle zurück, die eine Liste aller Anwendungen enthält, die über die application.library registriert wurden.

### EINGABEN

keine

### RÜCKGABEWERTE

t            eine Tabelle, die eine Liste von Zeichenfolgen enthält, die alle registrierten Anwendungen beschreibt

### BEISPIEL

```
t = GetApplicationList()  
For Local k = 0 To ListItems(t) - 1 Do DebugPrint(t[k])
```

Der obige Code gibt alle registrierten Anwendungen aus.

## 16.5 GetFrontScreen

### BEZEICHNUNG

GetFrontScreen – gibt den Namen des vordersten Bildschirms zurück (V8.0)

### ÜBERSICHT

```
n$ = GetFrontScreen()
```

### PLATTFORMEN

Nur AmigaOS und kompatible

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Namen des Bildschirms zu ermitteln, der sich gerade im Vordergrund befindet. Falls der Bildschirm kein öffentlicher Bildschirm ist, wird eine leere Zeichenkette zurückgegeben.

Beachten Sie, dass in einer Multitasking-Umgebung wie AmigaOS alle Bildschirme, die nicht Ihrer Anwendung zugewiesen sind, jederzeit verschwinden können und sich auch ihre Stapelreihenfolge ändern kann. Sie müssen also darauf vorbereitet sein, dass zum Zeitpunkt des Aufrufs von diesem Befehl der Bildschirm nicht mehr vorne oder gar nicht mehr vorhanden ist.

Weitere Bildschirmbefehle sind `GetPubScreens()`, `HideScreen()`, `ShowScreen()`, `SetDisplayAttributes()` und `SetScreenTitle()`.

### EINGABEN

keine



**RÜCKGABEWERTE**

**n\$**            der Bildschirm, der aktuell vorne ist (siehe die Warnung oben bezüglich der Zuverlässigkeit dieser Informationen)

**BEISPIEL**

```
ShowScreen("WORKBENCH")
Print(GetFrontScreen())
```

Es wird nicht garantiert, dass "WORKBENCH" zurückgegeben wird, obwohl dies normalerweise der Fall ist. Siehe die Warnung oben.

**16.6 GetPubScreens****BEZEICHNUNG**

GetPubScreens – gibt eine Liste aller verfügbaren öffentlichen Bildschirme zurück (V5.2)

**ÜBERSICHT**

```
t, info = GetPubScreens()
```

**PLATTFORMEN**

Nur AmigaOS und kompatible

**BESCHREIBUNG**

Mit diesem Befehl kann das System nach einer Liste aller verfügbaren öffentlichen Bildschirme abgefragt werden. Er gibt eine Tabelle **t** zurück, die ein Zeichenkettenelement für jeden zurzeit geöffneten öffentlichen Bildschirm enthält.

Ab Hollywood 5.3 gibt dieser Befehl die zweite Tabelle **info** mit Informationen über die Bildschirmdimensionen und der Farbtiefe zurück. Diese zweite Rückgabetabelle enthält so viele Elemente, wie die erste Rückgabetabelle und es gibt eine Untertabelle für jeden momentan geöffneten öffentlichen Bildschirm. Jede Untertabelle enthält die folgenden Tags:

**Width:**      Enthält die Breite des öffentlichen Bildschirm.

**Height:**     Enthält die Höhe des öffentlichen Bildschirm.

**Depth:**     Enthält die Farbtiefe des öffentlichen Bildschirm.

Sie können **ShowScreen()** verwenden, um zu einem öffentlichen Bildschirm zu wechseln. Wenn Sie das Display auf einen bestimmten öffentlichen Bildschirm verschieben möchten, verwenden Sie den Befehl **SetDisplayAttributes()**.

Beachten Sie, dass in einer Multitasking-Umgebung wie AmigaOS alle Bildschirme, die nicht Ihrer Anwendung zugewiesen sind, jederzeit verschwinden können und sich auch ihre Stapelreihenfolge ändern kann. Sie müssen also darauf vorbereitet sein, dass zum Zeitpunkt des Aufrufs von diesem Befehl der Bildschirm nicht mehr vorne oder gar nicht mehr vorhanden ist.

Weitere Bildschirmbefehle sind **GetFrontScreen()**, **HideScreen()**, **ShowScreen()**, **SetDisplayAttributes()** und **SetScreenTitle()**

**EINGABEN**

keine

**RÜCKGABEWERTE**

<b>t</b>	eine Tabelle, die eine Anzahl von Zeichenfolgen enthält, die alle offenen öffentlichen Bildschirme beschreibt
<b>info</b>	zusätzliche Tabelle mit Informationen über die Bildschirmdimensionen und -tiefe (V5.3)

**BEISPIEL**

```
t = GetPubScreens()
For Local k = 0 To ListItems(t) - 1 Do DebugPrint(t[k])
```

Dieser Code listet alle öffentlichen Bildschirme auf.

**16.7 HideScreen****BEZEICHNUNG**

HideScreen – schiebt einen öffentlichen Schirm in den Hintergrund (V7.1)

**ÜBERSICHT**

HideScreen([s\$])

**PLATTFORMEN**

AmigaOS und kompatible

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um den im Argument **s\$** angegebenen öffentlichen Bildschirm in den Hintergrund zu verschieben. Wenn das Argument **s\$** weggelassen wird, wird der momentan an vorderster Stelle befindliche Bildschirm nach hinten verschoben.

Beachten Sie, dass in einer Multitasking-Umgebung wie AmigaOS alle Bildschirme, die nicht Ihrer Anwendung zugewiesen sind, jederzeit verschwinden können. Sie müssen also darauf vorbereitet sein, dass dieser Befehl fehlschlägt, da der Bildschirm nicht mehr vorhanden ist.

Weitere Bildschirmbefehle sind `GetFrontScreen()`, `GetPubScreens()`, `ShowScreen()`, `SetDisplayAttributes()` und `SetScreenTitle()`.

**EINGABEN**

<b>s\$</b>	optional: Name des öffentlichen Bildschirms, der nach hinten verschoben werden soll (voreingestellt ist eine leere Zeichenkette. Damit wird der momentan vorderste Bildschirm nach hinten verschoben)
------------	---

**BEISPIEL**

```
HideScreen()
```

Dieser Code verschiebt den momentan aktiven Bildschirm in den Hintergrund.

**16.8 OpenAmigaGuide****BEZEICHNUNG**

OpenAmigaGuide – öffnet das AmigaGuide-Dokument in einem neuen Fenster (V6.1)

**ÜBERSICHT**

```
OpenAmigaGuide(file$[, node$])
```

**PLATTFORMEN**

Nur AmigaOS und kompatible

**BESCHREIBUNG**

Dieser Befehl öffnet die AmigaGuide-Datei, die in `file$` angegeben ist und zeigt sie in einem eigenen Fenster an. Wenn das optionale Argument `node$` angegeben ist, zeigt Hollywood diese bestimmte Seite der AmigaGuide-Datei an, ansonsten wird das Inhaltsverzeichnis präsentiert.

Sie können die AmigaGuide-Datei schließen, indem Sie den Befehl `CloseAmigaGuide()` aufrufen. Offene AmigaGuide-Dateien werden auch automatisch geschlossen, wenn Hollywood beendet wird.

Beachten Sie, dass es jeweils nur eine offene AmigaGuide-Datei geben kann. Wenn Sie diesen Befehl aufrufen und eine AmigaGuide-Datei bereits sichtbar ist, wird Hollywood diese alte AmigaGuide-Datei zuerst schließen.

**EINGABEN**

<code>file\$</code>	AmigaGuide-Datei
<code>node\$</code>	optional: Seite der AmigaGuide (standardmäßig auf "", das heißt das Inhaltsverzeichnis wird angezeigt)

**BEISPIEL**

```
OpenAmigaGuide("Hollywood:Help/Hollywood.guide", "OpenAmigaGuide")
```

Der obige Code zeigt diese Seite.

## 16.9 RunRexxScript

**BEZEICHNUNG**

RunRexxScript – führt ein ARexx-Skript aus einer Datei oder dem Arbeitsspeicher aus (V2.5)

**ÜBERSICHT**

```
res$ = RunRexxScript(script$[, nofile])
```

**PLATTFORMEN**

Nur AmigaOS und kompatible

**BESCHREIBUNG**

Mit diesem Befehl können Sie das im `skript$` angegebene ARexx-Skript ausführen. Darüber hinaus können Sie auch ein ARexx-Code direkt ausführen, indem Sie das optionale Argument `nofile` auf `True` setzen. In diesem Fall darf das `skript$` kein Pfad zu einem ARexx-Skript beinhalten, muss aber den auszuführenden ARexx-Code enthalten. Der Befehl gibt das Ergebnis aus dem ARexx-Skript zurück. Der Rückgabewert ist immer eine Zeichenfolge, auch wenn er nur eine Zahl enthält. Wenn ARexx nichts zurückgibt, erhalten Sie eine leere Zeichenfolge.

Vor der Verwendung dieses Befehls müssen Sie RexxMast starten. Es ist jedoch nicht notwendig, einen ARexx-Port zu erstellen, um diesen Befehl nutzen zu können. Dieser

Befehl funktioniert auch, wenn Ihr Skript keinen ARexx-Port hat. Das Skript wird immer mit "REXX" als Host-Port gestartet. Wenn Sie also einen anderen Port adressieren möchten, müssen Sie zuerst den Befehl "ADDRESS" von ARexx verwenden.

Wenn Sie diesen Befehl zum Starten externer ARexx-Skripts verwenden, stellen Sie sicher, dass die erste Zeile Ihres ARexx-Skripts ein Kommentar ist. Ansonsten erhalten Sie eine Fehlermeldung "Programm nicht gefunden". Als Syntax muss die erste Zeile aller ARexx-Skripts ein Kommentar sein.

Siehe auch `CreateRexxPort()` und `SendRexxCommand()`.

#### EINGABEN

`script$` Pfad zu einem externen direkten ARexx-Skript oder ARexx-Code; im letzteren Fall muss `nofile True` sein

`nofile` optional: `False`, um das in `script$` angegebene Arexx-Skript zu starten (enthält einen Pfad zu einem ARexx-Skript) und `True`, wenn `script$` ein ARexx-Code ist (standardmäßig ist `False`).

#### RÜCKGABEWERTE

`res$` Rückgabewert von ARexx; dies ist immer eine Zeichenfolge

#### BEISPIEL

```
RunRexxScript("dh0:MyScript.rx")
```

Der obige Code führt das Skript "dh0:MyScript.rx" aus.

```
r$ = RunRexxScript("SAY 'Hello'\nRETURN 5\n", True)
```

Der obige Code gibt "Hello" über die Konsole aus und gibt 5 an Hollywood zurück. Die Variable `r$` enthält also nach dem Aufruf "5".

## 16.10 SendApplicationMessage

#### BEZEICHNUNG

`SendApplicationMessage` – sendet eine Nachricht an eine andere Anwendung (V6.0)

#### ÜBERSICHT

```
SendApplicationMessage(app$, msg$)
```

#### PLATTFORMEN

Nur AmigaOS 4

#### BESCHREIBUNG

Mit diesem Befehl können Sie eine Nachricht an eine registrierte AmigaOS 4-Anwendung senden. Der Name der empfangenden Anwendung muss in `app$` und die Nachricht selbst wird an `msg$` übergeben.

Bitte beachten Sie, dass dieser Befehl nur verwendet werden kann, wenn Sie den Tag `RegisterApplication` in der Präprozessor-Anweisung `@OPTIONS` auf `True` gesetzt haben. Siehe [Abschnitt 50.25 \[OPTIONS\]](#), [Seite 1107](#), für Details.

#### EINGABEN

`app$` Name der Anwendung, die die Nachricht erhalten soll

`msg$`        die zu sendende Nachricht

## 16.11 SendRexxCommand

### BEZEICHNUNG

SendRexxCommand – sendet einen Befehl an den ARexx-Port (V2.5)

### ÜBERSICHT

```
res$ = SendRexxCommand(port$, cmd$)
```

### PLATTFORMEN

Nur AmigaOS und kompatible

### BESCHREIBUNG

Dieser Befehl sendet den in `cmd$` angegebenen Befehl an den in `port$` angegebenen ARexx-Port. Der Befehl gibt dann das Ergebnis aus dem Befehl zurück. Der Rückgabewert ist immer eine Zeichenfolge, auch wenn es nur eine Zahl enthält. Wenn der Befehl nichts zurückgibt, erhalten Sie eine leere Zeichenfolge. Sie können auch mehrere Befehle mit diesem Befehl senden. Trennen Sie einfach die Anweisungen mit einem Strichpunkt (";") oder Sie können auch Zeilenumbrüche ("\n") für die Trennung verwenden.

Wenn Sie keinen spezifischen ARexx-Port adressieren möchten, geben Sie einfach "REXX" in `port$` an. In diesem Fall ist der Standard-ARexx-Port des Systems der Host-Port. Bitte beachten Sie auch, dass Portnamen zwischen Groß- und Kleinschreibung unterschieden werden, d. h. "MYPORT" und "myport" bezeichnen zwei verschiedene ARexx-Ports. Aus Stilgründen sind Portnamen meist nur in Großbuchstaben geschrieben.

Vor der Verwendung von diesem Befehl müssen Sie RexxMast starten. Es ist jedoch nicht notwendig, einen ARexx-Port zu erstellen, um diesen Befehl nutzen zu können. Dieser Befehl funktioniert auch, wenn Ihr Skript keinen ARexx-Port hat.

Siehe auch [CreateRexxPort\(\)](#) und [RunRexxScript\(\)](#).

### EINGABEN

`port$`        Name des zu adressierenden Ports

`cmd$`        den/die Befehl(e) den/die Sie an diesen Port senden möchten

### RÜCKGABEWERTE

`res$`        Rückgabewert von ARexx; dies ist immer eine Zeichenfolge

### BEISPIEL

```
SendRexxCommand("WORKBENCH", "WINDOW 'Sys:' OPEN")
```

Der obige Code öffnet die SYS: Schublade auf Ihrer Workbench. Bitte beachten Sie, dass die ARexx-Schnittstelle der Workbench eine in OS3.5 eingeführte Funktionalität ist. So benötigen Sie OS3.5 oder besser. MorphOS unterstützt wahrscheinlich nicht die ARexx-Schnittstelle der Workbench, weil sie nur selten verwendet wird. Siehe in der OS3.9 NDK-Dokumentation die verfügbaren Befehle.

## 16.12 SetScreenTitle

### BEZEICHNUNG

SetScreenTitle – ändert den Bildschirmtitel des aktuellen Displays (V6.0)

### ÜBERSICHT

SetScreenTitle(title\$)

### PLATTFORMEN

Nur AmigaOS und kompatible

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Text zu ändern, der in der Titelleiste des Bildschirms angezeigt werden soll, wenn das aktuelle Display aktiv ist. Standardmäßig wird "Workbench-Screen" verwendet.

Möchten Sie den Fenster-/Displaytitel ändern, verwenden Sie `SetTitle()`.

Weitere Bildschirmbefehle sind `GetFrontScreen()`, `GetPubScreens()`, `HideScreen()`, `SetDisplayAttributes()` und `ShowScreen()`.

### EINGABEN

title\$      Neuer Bildschirmtitel

### BEISPIEL

```
SetScreenTitle("My cool program")
```

Der obige Code ändert den Bildschirmtitel zu "My cool program".

## 16.13 ShowRinghioMessage

### BEZEICHNUNG

ShowRinghioMessage – zeigt eine Ringhio-Benachrichtigung (V6.0)

### ÜBERSICHT

ShowRinghioMessage(title\$, text\$[, table])

### PLATTFORMEN

Nur AmigaOS 4

### BESCHREIBUNG

Mit diesem Befehl kann eine Benachrichtigung über das Ringhio-System des AmigaOS 4 angezeigt werden. Sie müssen einen Titel für die Benachrichtigung im ersten Argument `title$` und den eigentlichen Text im zweiten Argument `text$` übergeben.

Das optionale Tabellenargument `table` kann verwendet werden, um zusätzliche Parameter für die Behandlung der Ringhio-Benachrichtigung festzulegen. Folgende Tags werden hier akzeptiert:

#### PubScreen:

Mit diesem Tag können Sie den Namen des öffentlichen Bildschirms angeben, auf dem die Benachrichtigung erscheinen soll.

**ImageFile:**

Diesen Tag kann verwendet werden, um den Pfad zu einer Bilddatei anzugeben, die innerhalb der Ringhio-Benachrichtigung angezeigt werden soll. Für die besten Ergebnisse sollten Sie hier nur PNG-Bilder mit Alphakanal verwenden.

**DoubleClickClose:**

Wenn Sie diesen Tag auf **True** setzen, kann das Ringhio-Benachrichtigungsfenster durch Doppelklicken geschlossen werden. In diesem Fall wird eine Nachricht an Ihre Anwendung gesendet, die eine Zeichenfolge enthält, die Sie im Tag **DoubleClickMessage** angeben (siehe unten).

**DoubleClickMessage:**

Wenn der Benutzer das Ringhio-Benachrichtigungsfenster doppelklickt, um es zu schließen, wird die hier angegebene Zeichenfolge mit der Ereignisprozedur **OnApplicationMessage** an Ihre Anwendung gesendet. Diesen Tag wird nur unterstützt, wenn Sie auch den Tag **DoubleClickClose** auf **True** gesetzt haben (siehe oben). Wenn die hier angegebene Zeichenfolge das folgende Format "URL:http://www.example.com/" hat, sendet der Ringhio-Server keine Nachricht an Ihre Anwendung zurück, sondern zeigt automatisch die angegebene URL im Standardbrowser an, wenn der Benutzer das Benachrichtigungsfenster doppelt anklickt.

Beachten Sie, dass dieser Befehl nur verwendet werden kann, wenn Sie den Tag **RegisterApplication** bei der Präprozessor-Anweisung **@OPTIONS** auf **True** gesetzt haben. Siehe **Abschnitt 50.25 [OPTIONS]**, **Seite 1107**, für Details.

**EINGABEN**

<b>title\$</b>	Titel für die Ringhio-Benachrichtigung
<b>text\$</b>	anzuzeigender Text in der Ringhio-Benachrichtigung
<b>table</b>	optional: Tabelle mit weiteren Parametern (siehe oben)

**16.14 ShowScreen****BEZEICHNUNG**

ShowScreen – wechselt zum angegebenen öffentlichen Bildschirm (V5.2)

**ÜBERSICHT**

ShowScreen(s\$)

**PLATTFORMEN**

Nur AmigaOS und kompatibel

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um den in **s\$** angegebenen öffentlichen Bildschirm nach vorne zu bringen. Wenn Sie das Display auf einen bestimmten öffentlichen Bildschirm verschieben möchten, verwenden Sie den Befehl **SetDisplayAttributes()** mit dem Tag **PubScreen**.

Beachten Sie, dass in einer Multitasking-Umgebung wie AmigaOS alle Bildschirme, die nicht Ihrer Anwendung zugewiesen sind, jederzeit verschwinden können. Sie müssen also darauf vorbereitet sein, dass dieser Befehl fehlschlägt, da der Bildschirm nicht mehr vorhanden ist.

Weitere Bildschirmbefehle sind `GetFrontScreen()`, `GetPubScreens()`, `HideScreen()`, `SetDisplayAttributes()` und `SetScreenTitle()`.

#### **EINGABEN**

`s$`            Name des öffentlichen Bildschirms, der nach vorne gebracht werden soll

#### **BEISPIEL**

```
ShowScreen("WORKBENCH")
```

Dieser Code bringt den Workbench-Bildschirm zurück auf die Vorderseite.



## 17 Animationsbibliothek

### 17.1 Übersicht

Animationen sind Hollywood-Objekte, die mehrere Einzelbilder von Bilddaten enthalten. Sie können von der Festplatte abgespielt oder vollständig im Speicher gepuffert werden. Um eine Animation von der Festplatte abzuspielen, können Sie den Befehl `OpenAnim()` verwenden, um die Animationsdatei zu öffnen, oder alternativ den Tag `FromDisk` im Befehl `LoadAnim()` oder bei der Präprozessor-Anweisung `@ANIM` auf `True` setzen. Um eine Animation vollständig in den Speicher zu laden, verwenden Sie den Befehl `LoadAnim()`. Es wird jedoch empfohlen, größere Animationen immer von der Festplatte abzuspielen, da das Puffern aller Einzelbilder viel Speicher benötigt.

Hier können Sie eine Animation für das Streaming von der Festplatte mit der Präprozessor-Anweisung `@ANIM` öffnen (natürlich können Sie auch den Befehl `OpenAnim()` verwenden):

```
@ANIM 1, "test.anim", {FromDisk = True}
```

Standardmäßig kann Hollywood die Animationsformate IFF ANIM und GIF ANIM öffnen. Es gibt jedoch mehrere Plugins, die die Anzahl der Animationsformate erweitern, die Sie mit Hollywood öffnen können. Zum Beispiel können Sie Plugins für die APNG- und FLI/FLC-Formate vom offiziellen Hollywood-Portal herunterladen.

Animationen können mit dem Befehl `PlayAnim()` abgespielt werden. `PlayAnim()` blockiert die Skriptausführung, außer Sie setzen den Tag `Async` auf `True`, um ein asynchrones Zeichnungsobjekt von `PlayAnim()` zu erhalten. Dieses können Sie verwenden, um die Animation asynchron mit dem Befehl `AsyncDrawFrame()` abzuspielen. Siehe [Abschnitt 17.18 \[PlayAnim\]](#), [Seite 206](#), für Details.

Alternativ können Sie auch einzelne Animationsbilder zeichnen, indem Sie den Befehl `DisplayAnimFrame()` verwenden. Wenn Ebenen aktiviert sind, ist es sehr einfach, das nächste Einzelbild anzuzeigen, indem Sie einfach den Befehl `NextFrame()` verwenden, nachdem Sie eine Animationsebene mit dem Befehl `DisplayAnimFrame()` erstellt haben. Siehe [Abschnitt 17.7 \[DisplayAnimFrame\]](#), [Seite 193](#), für Details.

Normalerweise enthalten Animationen Rasterpixeldaten, aber Hollywood unterstützt auch spezielle Vektoranimationen, die stattdessen aus Vektorpfaddaten bestehen und somit frei transformiert werden können. Siehe [Abschnitt 17.25 \[Informationen zu Vektoranimation\]](#), [Seite 214](#), für Details.

### 17.2 ANIM

#### BEZEICHNUNG

ANIM – lädt eine Animation vor (V2.0)

#### ÜBERSICHT

```
@ANIM id, filename$, [table]
```

#### BESCHREIBUNG

Diese Präprozessor-Anweisung lädt die in `filename$` angegebene Animation und weist ihr den Identifikator `id` zu.

Anim-Formate, die auf allen Plattformen von Hollywood unterstützt werden, sind IFF ANIM, GIF ANIM, AVI (unkomprimiert oder mit Motion-JPEG-Komprimierung) und

Formate, für die Sie ein Plugin haben. Je nach Plattform, auf der Hollywood ausgeführt wird, können mehr Anim-Formate unterstützt werden. Zum Beispiel wird Hollywood auf Amigakompatiblen Systemen in der Lage sein, alle Anim-Formate zu öffnen, für die Sie Datatypes haben. Unter Windows kann @ANIM auch Animationsformate laden, die von der Windows-Imaging-Komponente unterstützt werden.

Ab Hollywood 4.5, kann @ANIM auch automatisch Animationen aus einer Bilddatei erstellen, wenn Sie das optionale Argument **Frames** angeben. Siehe unten für weitere Informationen.

Das dritte Argument ist optional. Es ist eine Tabelle, mit der Sie weitere Möglichkeiten für den Ladevorgang einstellen können. Folgende Tags stehen zur Verfügung:

**Transparency:**

Mit diesem Tag können Sie eine Farbe in **RGB-Notation** angeben, die in der Animation transparent werden soll.

**Link:** Setzen Sie diesen Tag auf **False**, wenn Sie diese Animation beim Kompilieren nicht in die ausführbare Datei oder ins Applet einbinden möchten. Diesen Tag ist standardmäßig auf **True** gesetzt, womit die Animation in Ihre Datei oder mit ihrem Applet im Kompilierungsmodus eingebunden ist.

**FromDisk:**

Wenn Sie diesen Tag auf **True** setzen, wird Hollywood nicht die gesamte Animation in den Speicher, sondern wenn nötig nur die einzelnen Bilder direkt von der Festplatte laden. Dies ist langsamer, aber erfordert viel weniger Speicher. Für die Befehle der Animbibliothek spielt es keine Rolle, ob die Animation vollständig im Speicher ist oder dynamisch von der Festplatte geladen wird. Sie können alle Anim-Befehle wie **ScaleAnim()** auch mit Anims verwenden, die von der Festplatte geladen werden. Animebenen werden auch mit Plattenanimationen korrekt behandelt. (V3.0)

**LoadAlpha:**

Setzen Sie diesen Tag auf **True**, wenn der Alphakanal der Animation auch geladen werden soll. Bitte beachten Sie, dass die meisten Animformate keine Alphakanäle unterstützen. Daher ist es ratsam, dass Sie die Animation manuell aus einem PNG-Bild mit **CreateAnim()** erstellen, wenn Sie einen Alphakanal in Ihrer Animation haben müssen. Dieser Tag ist standardmäßig auf **False** gesetzt. (V4.5)

**X, Y, Width, Height, Frames, FPR:**

Diese Gruppe von Tags müssen Sie nur verwenden, wenn Sie eine Bilddatei als Quelle angeben, damit @ANIM aus dem Bild eine Animation erstellen kann. **Width** und **Height** (Breite und Höhe) definieren die Dimensionen für die Animation und **Frames** gibt an, wie viele Einzelbilder @ANIM aus dem Quellbild lesen wird. Wenn die Einzelbilder in mehreren Reihen im Quellbild gespeichert sind, müssen Sie auch das Argument **FPR** (Abkürzung für Frames (Einzelbilder) pro Reihe) verwenden, um @ANIM zu mitzuteilen, wie viele Bilder es in jeder Reihe hat. Schließlich kann man mit den Tags **X** und **Y** angeben, wo in der Bilddatei @ANIM mit dem Scannen starten soll (beide standardmäßig auf 0 gesetzt). @ANIM wird dann an der Position X/Y die

in **Frames** definierte Anzahl Bilder mit den Abmessungen von **Width** und **Height** aus der Bilddatei **filename\$** lesen. Nachdem es die in **FPR** angegebene Anzahl Bilder gelesen hat, wird es in die nächste Zeile vorrücken. (V4.5)

**SkipLoopFrames:**

Wenn Sie diesen Tag auf **True** setzen, wird Hollywood automatisch die letzten beiden Einzelbilder der Animation überspringen. Dies ist nur für IFF-Anims erforderlich, die zwei Schleifeneinzelbilder (loop frames) am Ende der Animation haben. Die automatische Erkennung von Schleifeneinzelbilder ist nicht möglich, denn das würde von Hollywood erfordern, zunächst die gesamte Animation zu dekodieren. Deshalb müssen Sie Hollywood manuell mitteilen, ob die Animation Schleifeneinzelbilder hat oder nicht. (V5.3)

**Deinterlace:**

Hiermit können Sie festlegen, wie Hollywood interlaced Anims darstellen soll. Dies kann entweder **#DEINTERLACE\_DEFAULT** oder **#DEINTERLACE\_DOUBLE** eingestellt werden. Wenn Sie **#DEINTERLACE\_DEFAULT** benutzen (das ist, wie der Name auch impliziert, voreingestellt), wird Hollywood zwei Halbbilder in ein Vollbild kombinieren. Dies geschieht meist in bester Qualität, kann aber zu visuelle Artefakte führen, wenn eine Menge Bewegungen in der Animation vorkommen. Wenn Sie stattdessen **#DEINTERLACE\_DOUBLE** verwenden, wird Hollywood die Zeilen eines Halbbildes verdoppeln, um ein Vollbild zu erhalten. Dies führt zu einem gewissen Qualitätsverlust, kann aber die Animation zu einem glatteren Aussehen verhelfen. Welches der bessere Deinterlacing-Modus ist, hängt immer von der Anim ab. Beachten Sie, dass Sie sich normalerweise überhaupt nicht um diesen Tag kümmern müssen, weil Deinterlacing eigentlich nur für einige obskure IFF-Formate erforderlich ist, die ANIM16i und ANIM32i Interlaced-Einzelbilder speichern. (V5.3)

**Loader:** Mit diesem Tag können Sie ein oder mehrere Formatlademodule angeben, die die Anim laden soll. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Lademodulen beinhaltet. Standardmäßig wird der mit **SetDefaultLoader()** eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, der die angegebene Datei öffnet. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Adaptermodulen beinhaltet. Standardmäßig wird der mit **SetDefaultAdapter()** eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**LoadTransparency:**

Ist dieser Tag auf **True** gesetzt, wird die monochrome Transparenz der Animation geladen. Bitte beachten Sie, dass dieser Tag speziell für monochrome Transparenzkanäle ausgelegt ist, die einen transparenten Stift in einer palettenbasierten Animation haben. Wenn Sie den Alphakanal einer Ani-

mation laden möchten, stellen Sie den Tag **LoadAlpha** auf **True**. Dieser **LoadTransparency**-Tag ist standardmäßig auf **False** gesetzt. (V6.0)

**LoadPalette:**

Wenn dieser Tag auf **True** gesetzt ist, lädt Hollywood die Animation als Palettenanimation. Dies bedeutet, dass Sie die Palette der Animation abrufen und ändern können, was für bestimmte Effekte wie Farbwechsel nützlich ist. Sie können Stifte auch mit dem Tag **TransparentPen** (siehe unten) oder dem Tag **LoadTransparency** (siehe oben) transparent machen. Palettenanimationen haben auch den Vorteil, dass weniger Speicher benötigt wird, da 1 Pixel nur 1 Byte Speicher anstelle von 4 Byte für 32-Bit-Bilder benötigt. Dieser Tag ist standardmäßig auf **False** eingestellt. (V9.0)

**TransparentPen:**

Wenn der Tag **LoadPalette** auf **True** gesetzt wurde (siehe oben), kann mit dem Tag **TransparentPen** ein Stift definiert werden, der transparent gemacht werden soll. Stifte werden ab 0 gezählt. Alternativ können Sie den Tag **LoadTransparency** auch auf **True** setzen, um Hollywood zu zwingen, den in der Animationsdatei gespeicherten transparenten Stift zu verwenden (sofern das Animationsformat die Speicherung transparenter Stifte unterstützt). Dieser Tag ist standardmäßig auf **#NOPEN** gesetzt. (V9.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Bitte beachten Sie, dass die Tags **Transparency**, **LoadTransparency** und **LoadAlpha** sich gegenseitig ausschließen. Eine Animation kann nicht eine Maske und einen Alphakanal haben!

Ab Hollywood 9.0 kann diese Präprozessor-Anweisung auch Vektor-Animationsformate laden, wenn Sie ein entsprechendes Plugin installiert haben. Beachten Sie jedoch, dass beim Laden von Vektor-Animationsformaten mit **@ANIM** die Animation möglicherweise nicht alle Funktionen normaler Animationen unterstützt. Siehe [Abschnitt 17.25 \[Vektoranimationen\]](#), [Seite 214](#), für weitere Informationen zu Vektoranimationen.

Möchten Sie Animationen manuell laden, benutzen Sie den Befehl **LoadAnim()**.

## EINGABEN

<b>id</b>	ein Wert, der diese Animation später im Code identifiziert
<b>filename\$</b>	die Animationsdatei
<b>table</b>	optional: eine Tabelle, die eine Kombination der oben erwähnten Tags enthält

## BEISPIEL

```
@ANIM 1, "MyAnim.gif"
```

Lädt "MyAnim.gif" als Animation Nummer 1.

```
@ANIM 1, "MyAnim.gif", {Transparency = $FF0000}
```

Ist das gleiche wie oben, aber die Animation ist jetzt transparent (Transparenzfarbe ist Rot = \$FF0000).

```
@ANIM 1, "Huge_Animation.iff", {Link = False}
```

Der obige Code lädt die angegebene Animation und erklärt Hollywood, beim Kompilieren die Animation nicht mit dem ausführbaren Programm oder Applet zu verknüpfen, weil sonst die Datei zu groß wird.

## 17.3 BeginAnimStream

### BEZEICHNUNG

BeginAnimStream – erstellt eine sequentielle Animation (V4.5)

### ÜBERSICHT

```
[id] = BeginAnimStream(id, file$, width, height[, format, table])
```

### BESCHREIBUNG

Mit diesem Befehl können Sie ein leeres Animationsobjekt auf der Festplatte erstellen, dem Sie dann anschließend mit dem Befehl `WriteAnimFrame()` Einzelbilder (Frames) einfügen können. Der Vorteil von `BeginAnimStream()` gegenüber `SaveAnim()` ist, dass `SaveAnim()` ein Animationsobjekt als Quelle erfordert. Wenn Sie hingegen `BeginAnimStream()` verwenden, können Sie Bilder von einzelnen Pinselobjekten in Ihre Animation einfügen. Dies gibt Ihnen die größtmögliche Flexibilität. Wegen seinem sequentiellen Design kann `BeginAnimStream()` für neue Animationen verwendet werden, die nahezu unbegrenzte Größe und Länge haben. Sie könnten leicht eine zweistündiges AVI-Video mit diesem Befehl erstellen.

Das erste Argument für `BeginAnimStream()` muss eine ID für das neue Animationsobjekt sein. Alternativ können Sie auch `Nil` angeben und `BeginAnimStream()` wird eine ID zuweisen und Ihnen zurückgeben. Im zweiten Argument `file$` übergeben Sie den Dateinamen, der für diese Animation verwendet werden soll. In den Argumenten `width` (Breite) und `height` (Höhe) geben sie die gewünschten Abmessungen der Animation an. Beim fünften Argument tragen Sie das Format der Animation ein. Dies kann entweder eine der folgenden Animationstypen oder ein von einem Plugin bereitgestelltes Animationsformat sein:

#### #ANMFMT\_GIF:

GIF-Format. Da GIF-Animationen immer palettenbasiert sind, müssen RGB-Grafiken quantisiert werden, bevor sie als GIF exportiert werden können. Beim Aufruf vom Befehl `WriteAnimFrame()` können Sie die Tags `Colors` und `Dither` verwenden, um die Anzahl der Paletteneinträge festzulegen, die den Einzelbildern zugewiesen werden sollen und ob Dithering angewendet werden soll oder nicht. Bei Verwendung von `#ANMFMT_GIF` mit einem Paletteneinzelbild wird keine Quantisierung

durchgeführt. `#ANMFMT_GIF` unterstützt auch Palettenanimationen mit einem transparenten Stift. `#ANMFMT_GIF` ist das von `BeginAnimStream()` verwendete Standardformat.

#### `#ANMFMT_MJPEG:`

AVI mit Motion JPEG-Komprimierung. Dies ist ein verlustbehaftetes Animationsformat, so dass Sie das **Quality-Tag** (siehe unten) festlegen können, um die zu verwendende Komprimierungsstufe zu wählen.

#### `#ANMFMT_IFF:`

IFF-Animation. Hollywood verwendet für IFF-Animationen den Komprimierungsmodus 5 (den gebräuchlichsten Komprimierungsmodus). Da IFF-Animationen immer palettenbasiert sind, müssen RGB-Grafiken quantisiert werden, bevor sie als IFF exportiert werden können. Beim Aufruf von `WriteAnimFrame()` können Sie die Tags **Colors** und **Dither** verwenden, um die Anzahl der Paletteneinträge festzulegen, die dem Einzelbild zugewiesen werden sollen und ob Dithering angewendet werden soll oder nicht. Bei Verwendung von `#ANMFMT_IFF` mit einem Paletteneinzelbild wird keine Quantisierung durchgeführt. `#ANMFMT_IFF` unterstützt auch Palettenanimationen mit einem transparenten Stift. (V9.0)

Mit dem optionale Tabellenargument können Sie weitere Parameter konfigurieren:

**Quality:** Hier können Sie einen Wert zwischen 0 und 100 angeben, womit Sie die Komprimierungsqualität für verlustbehaftete Kompressionsformate wählen dürfen. Ein Wert von 100 bedeutet beste Qualität, 0 hingegen ist die schlechteste. Dies steht nur für Anim-Formate zur Verfügung, die verlustbehaftete Kompression unterstützen. Der Standardwert ist 90, womit eine ziemlich gute Qualität möglich ist.

**FPS:** Da Videoformate wie AVI keinen individuellen Verzögerungswert für jedes Einzelbild (Frame) unterstützen, erfordert es einen globalen Wert, der angibt, wie viele Bilder pro Sekunde angezeigt werden sollen. Mit **FPS** können Sie die Anzahl an Bildern festlegen. Dies wird nur für Videodateiformate berücksichtigt. Voreingestellt sind 25 Bildern pro Sekunde.

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, die überprüfen sollen, ob sie die angegebene Datei speichern können. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Zeichenkette setzen, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()` gesetzt wurde. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V10.0)

#### **UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an den Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Hier ist eine Tabelle, die einen Überblick zeigt, welche Tabellenelemente mit den verschiedenen Animationsformate verwendet werden können:

	<b>GIF</b>	<b>IFF</b>	<b>MJPEG AVI</b>
Quality	Nein	Nein	Ja
FPS	Nein	Nein	Ja

Wenn Sie erfolgreich ein neues Animationsobjekt erstellt haben, können Sie dann nacheinander Einzelbilder (Frames) mit dem Befehl **WriteAnimFrame()** anhängen. Wenn Sie mit dem Hinzufügen von Einzelbildern fertig sind, müssen Sie den Befehl **FinishAnimStream()** aufrufen, um die Animationsdatei auf der Festplatte abzuschließen. Somit wäre sie nun einsatzbereit.

#### EINGABEN

<b>id</b>	ID des Animationsobjekt oder <b>Nil</b> für die automatische ID-Zuweisung
<b>file\$</b>	Zieldatei
<b>width</b>	gewünschte Breite der Animation
<b>height</b>	gewünschte Höhe der Animation
<b>format</b>	optional: Animationsformat (voreingestellt ist <b>#ANMFMT_GIF</b> )
<b>table</b>	optional: weitere Argumente fürs Speichern; siehe oben

#### RÜCKGABEWERTE

<b>id</b>	optional: Identifikator der Animation; Wird nur zurückgegeben werden, wenn Sie <b>Nil</b> als Argument 1 angegeben haben (siehe oben)
-----------	---

#### BEISPIEL

```
CreateBrush(1, 320, 240)
SelectBrush(1)
SetFillStyle(#FILLCOLOR)
BeginAnimStream(1, "test.gif", 320, 240)
For Local k = 1 To 100
    Circle(#CENTER, #CENTER, k * 2, #RED)
    WriteAnimFrame(1, 1)
Next
FinishAnimStream(1)
EndSelect
```

Der obige Code erstellt eine neue GIF-Animation mit 100 Bildern. Die Animation wird einen roten Kreis zeigen und in den Bildschirm zoomen.

## 17.4 CloseAnim

#### BEZEICHNUNG

CloseAnim – schließt eine Animation (V9.0)

#### ÜBERSICHT

```
CloseAnim(id)
```



**BESCHREIBUNG**

Dieser Befehl schließt die durch `id` angegebene Animation. Die Animation muss zuvor mit `OpenAnim()` geöffnet worden sein. Siehe [Abschnitt 17.17 \[OpenAnim\]](#), Seite 203, für Details.

**EINGABEN**

`id`            Identifikator der Animation

## 17.5 CopyAnim

**BEZEICHNUNG**

CopyAnim – kopiert eine Animation (V2.0)

**ÜBERSICHT**

```
[id] = CopyAnim(src, dst)
```

**BESCHREIBUNG**

Dieser Befehl kopiert die Animation, welche in `src` angegeben wurde und weist ihr die ID von `dst` zu. Wenn Sie `Nil` als ID verwenden, wird `CopyAnim()` automatisch eine ID für Sie auswählen. Die neue Animation ist völlig unabhängig von der alten, so dass Sie die alte aus dem Speicher löschen können, nachdem sie kopiert wurde.

**EINGABEN**

`src`            ID der Animation, von der eine Kopie erstellt wird  
`dst`            ID der neuen Animation oder `Nil` für die [automatische ID-Zuweisung](#)

**RÜCKGABEWERTE**

`id`            optional: Identifikator der neuen Animation; Wird nur zurückgegeben werden, wenn Sie `Nil` als Argument 2 angegeben haben (siehe oben)

## 17.6 CreateAnim

**BEZEICHNUNG**

CreateAnim – erstellt eine Animation von einem Pinsel (V2.0)

**ÜBERSICHT**

```
[id] = CreateAnim(id, brush, width, height, frames, fpr[, sx, sy])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um eine neue Animation aus einem Pinsel zu erstellen. Wenn Sie `Nil` als `id` verwenden, wird `CreateAnim()` automatisch eine ID für Sie auswählen. Die Einzelbilder werden von dem in `brush` angegebenen Pinsel gelesen und in einer neuen Animation zusammengesetzt werden. Sie müssen die Breite `width` und Höhe `height` der Animation sowie die Anzahl der Einzelbilder `frames` von dem Pinsel angeben und falls vorhanden im Argument `fpr` wie viele Bilder in einer Reihe sind. Optional können Sie in `sx` und `sy` eine Position definieren, von wo aus in dem Pinsel die Umwandlung beginnen soll.



Wenn der Pinsel transparent ist, wird die neue Animation auch transparente Bereiche haben. Falls der Pinsel einen Alphakanal verwendet, wird auch die Animation einen Alphakanal erhalten.

#### EINGABEN

<code>id</code>	Identifikator der neuen Animation
<code>brush</code>	Pinsel, aus dem die Einzelbilder gelesen werden
<code>width</code>	Breite der Animation
<code>height</code>	Höhe der Animation
<code>frames</code>	Anzahl Einzelbilder, die aus dem Pinsel gelesen werden
<code>fpr</code>	wie viele Einzelbilder in einer Reihe sind
<code>sx</code>	optional: Position x im Pinsel (voreingestellt ist 0)
<code>sy</code>	optional: Position y im Pinsel (voreingestellt ist 0)

#### RÜCKGABEWERTE

<code>id</code>	optional: Identifikator der Animation; Wird nur zurückgegeben werden, wenn Sie <code>Nil</code> als Argument 1 angegeben haben (siehe oben)
-----------------	---

## 17.7 DisplayAnimFrame

#### BEZEICHNUNG

`DisplayAnimFrame` – zeigt ein Einzelbild der Animation (V4.0)

#### ÜBERSICHT

`DisplayAnimFrame(id, x, y, frame[, table])`

#### BESCHREIBUNG

Dieser Befehl zeigt an der Position `x` und `y` das Einzelbild `frame` einer Animation.

Wenn Ebenen aktiviert sind, wird dieser Befehl eine neue Ebene des Typs `#ANIM` dem Ebenenstapel hinzuzufügen.

`DisplayAnimFrame()` kennt auch ein optionales Tabellenargument `table`, wo Sie eine oder mehrere der [Standard-Tags zum Zeichnen](#) angeben können. Siehe [Abschnitt 29.17 \[Standard-Tags zum Zeichnen\], Seite 613](#), für weitere Informationen über die Standard-Tags, die fast alle Hollywood Zeichnungsbefehle unterstützen.

#### EINGABEN

<code>id</code>	ID der Animation
<code>x</code>	X-Koordinate, an der das Einzelbild gezeigt wird
<code>y</code>	Y-Koordinate, an der das Einzelbild gezeigt wird
<code>frame</code>	Einzelbild der Animation, welches angezeigt werden soll (1 = erstes Einzelbild)
<code>table</code>	optional: Tabelle für weitere Optionen

**BEISPIEL**

```
DisplayAnimFrame(1, #CENTER, #CENTER, 5)
```

Der obige Code zeigt das Einzelbild Nr. 5 der Animation 1 in der Mitte des Bildschirms.

## 17.8 FinishAnimStream

**BEZEICHNUNG**

FinishAnimStream – finalisiert das sequentielle Animationsobjekt (V4.5)

**ÜBERSICHT**

```
FinishAnimStream(id)
```

**BESCHREIBUNG**

Dieser Befehl muss verwendet werden, um ein sequentielles Animationsobjekt nach dem Einfügen von Einzelbildern (Frames) fertig zu stellen. Danach ist die neue Animation auf ihrer Festplatte bereit.

Siehe [Abschnitt 17.3 \[BeginAnimStream\]](#), [Seite 189](#), für weitere Informationen zu den sequentiellen Animationsobjekten.

**EINGABEN**

id	ID des Animationsobjektes, welches finalisiert werden soll; muss vorher mit <a href="#">BeginAnimStream()</a> erstellt worden sein
----	--

**BEISPIEL**

Siehe [Abschnitt 17.3 \[BeginAnimStream\]](#), [Seite 189](#).

## 17.9 FreeAnim

**BEZEICHNUNG**

FreeAnim – gibt den von einer Animation belegten Speicher frei

**ÜBERSICHT**

```
FreeAnim(id)
```

**BESCHREIBUNG**

Dieser Befehl löscht den Speicher, der von der durch `id` angegebene Animation belegt wird. Um den Speicherverbrauch zu reduzieren, sollten Sie Animationen aus dem Speicher löschen, wenn Sie sie nicht mehr benötigen.

Beachten Sie, dass dieser Befehl mit Animationen verwendet werden sollte, die von [LoadAnim\(\)](#), [CreateAnim\(\)](#) oder [@ANIM](#) zugewiesen wurden. Animationen, die von [OpenAnim\(\)](#) zugewiesen wurden, sollten mit [CloseAnim\(\)](#) geschlossen werden.

**EINGABEN**

id	Identifikator der Animation
----	-----------------------------

## 17.10 GetAnimFrame

### BEZEICHNUNG

GetAnimFrame – kopiert ein Animationseinzelbild als Pinsel (V3.0)

### ÜBERSICHT

```
GetAnimFrame(id, frame, animid)
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um ein einzelnes Bild (Frame) einer Animation als Pinsel zu kopieren. Die Animation muss mit dem Befehl `LoadAnim()` oder der Präprozessor-Anweisung `@ANIM` vorher geladen werden. Wenn Sie ein Einzelbild direkt aus einer Animationsdatei laden möchten, verwenden Sie stattdessen `LoadAnimFrame()`. `GetAnimFrame()` wird jedoch bevorzugt, weil dieser Befehl schneller ist. Im ersten Argument `id` geben Sie den Identifikator ein, welcher der neue Pinsel erhalten soll. Im zweiten Argument `frame` müssen Sie angeben, welches Bild der Animation geladen werden soll und das dritte Argument `animid` übergeben Sie schließlich die ID der Animation, welche als Quelle dienen wird.

### EINGABEN

<code>id</code>	ID des Pinsels, der mit diesem Befehl erstellt wird
<code>frame</code>	das zu ladende Einzelbild (Bereich geht von 1 bis Anzahl Einzelbilder); geben Sie -1 an, wenn Sie das letzte Einzelbild laden möchten
<code>animid</code>	ID der Animation

### BEISPIEL

```
LoadAnim(1, "TestAnim.anim")
```

```
GetAnimFrame(1, 15, 1)
```

Der obige Code kopiert das Einzelbild 15 der Animation 1 als Pinsel 1.

## 17.11 IsAnim

### BEZEICHNUNG

IsAnim – stellt fest, ob eine Animation in einem unterstützten Format vorliegt

### ÜBERSICHT

```
ret = IsAnim(file$, table)
```

### BESCHREIBUNG

Dieser Befehl prüft, ob die in `file$` angegebene Datei in einem unterstützten Animationsformat vorliegt. Wenn ja, wird dieser Befehl `True` zurückgeben, andernfalls `False`. Wenn der Befehl `True` zurückgibt, können Sie die Animation mit `LoadAnim()` laden.

Ab Hollywood 6.0 kann eine optionale Tabelle angegeben werden, mit welcher Sie einige erweiterte Optionen konfigurieren können. Die folgenden Tags werden derzeit unterstützt:

**Loader:** Mit diesem Tag können Sie ein oder mehrere Formatlademodule angeben, die diese Animation laden soll. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehreren Lademodulen enthält. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen soll. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehreren Adaptermodulen enthält. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Siehe [Abschnitt 17.13 \[LoadAnim\]](#), [Seite 197](#), für eine Liste der unterstützten Animationsformate.

## EINGABEN

`file$` zu überprüfende Datei  
`table` optional: Tabelle mit weiteren Argumenten

## RÜCKGABEWERTE

`ret` `True`, wenn die Animation in einem unterstützten Format vorliegt, andernfalls `False`.

## 17.12 IsAnimPlaying

### BEZEICHNUNG

`IsAnimPlaying` – stellt fest, ob eine Animation gerade abgespielt wird (nur V1.0)

### ÜBERSICHT

`playing = IsAnimPlaying(id)`

### BESCHREIBUNG

Achtung: Dieser Befehl wurde in Hollywood 1.5 entfernt.

Dieser Befehl überprüft, ob die in `id` angegebene Animation gerade abgespielt wird und liefert `True`, wenn es so ist, andernfalls `False`.

### EINGABEN

`id` ID der zu überprüfenden Animation

### RÜCKGABEWERTE

`playing` `True` wenn die in `id` angegebene Animation abgespielt wird, ansonsten `False`

### BEISPIEL

```
LoadAnim(1,"dh0:Gfx/Anims/CoolAnim.anm")
PlayAnim(1,#PLAYONCE)
playing=IsAnimPlaying(1)
While(playing=TRUE)
```

```

    playing=IsAnimPlaying(1)
Wend
FreeAnim(1)

```

Der obige Code lädt die Animation "dh0:Gfx/Anims/CoolAnim.anm", spielt sie ab und wartet dann, bis sie fertig ist. Danach wird der Speicher der Animation freigegeben.

Wenn Sie die Aufgabe wie oben erledigen wollen, ist es einfacher für Sie, wenn Sie den Befehl `WaitAnimEnd()` verwenden. Aber falls Sie einige Dinge während dem Abspielen der Animation tun möchten, müssen Sie es auf diese Weise mit `IsAnimPlaying()` und einer Schleife lösen.

## 17.13 LoadAnim

### BEZEICHNUNG

LoadAnim – lädt eine Animation

### ÜBERSICHT

```
[id] = LoadAnim(id, filename$[, table])
```

### BESCHREIBUNG

Dieser Befehl lädt die durch `filename$` angegebene Animation in den Speicher und weist ihr den Identifikator `id` zu. Wenn Sie `Nil` angeben, wird `LoadAnim()` automatisch eine ID für Sie auswählen.

Beachten Sie, dass dieser Befehl standardmäßig alle Animationseinzelbilder in den Speicher lädt, was eine Weile dauern und viel Speicherplatz in Anspruch nehmen kann. Wenn Sie eine Animation erstellen möchten, die Einzelbilder dynamisch nach Bedarf lädt und nur das aktuelle Einzelbild im Speicher behält, verwenden Sie stattdessen den Befehl `OpenAnim()` oder setzen Sie den Tag `FromDisk` auf `True` (siehe unten). Siehe [Abschnitt 17.17 \[OpenAnim\]](#), [Seite 203](#), für Details.

Animationsformate, die auf allen Plattformen unterstützt werden, sind IFF ANIM, GIF ANIM, AVI (unkomprimiert oder mit Motion-JPEG-Komprimierung) und Formate, für die Sie ein Plugin haben. Je nach Plattform, auf der Hollywood ausgeführt wird, können mehr Animationsformate unterstützt werden. Zum Beispiel auf den Amigakompatiblen Systemen wird Hollywood in der Lage sein, alle Animationsformate zu öffnen, für die Sie ein Datatype haben.

Ab Hollywood 4.5 kann `LoadAnim()` auch automatisch Animationen aus einer Bilddatei erstellen. Wenn Sie eine Bilddatei mit `LoadAnim()` laden möchten, müssen Sie das optionale Argument `Frames` angeben. Siehe unten für weitere Informationen.

Das dritte Argument `table` ist optional. Es ist eine Tabelle, die für weitere Möglichkeiten des Ladevorgangs verwendet werden kann. Die folgenden Tags stehen dafür zur Verfügung:

#### Transparency:

Mit diesem Tag können Sie eine Farbe in der **RGB-Notation** angeben, die in der Animation transparent werden soll.

#### FromDisk:

Wenn Sie dieser Tag auf `True` setzen, wird Hollywood nicht die gesamte Animation in den Speicher, sondern wenn nötig nur die einzelnen Bilder

direkt von der Festplatte laden. Dies ist langsamer, aber erfordert viel weniger Speicher. Für die Befehle der Animbibliothek spielt es keine Rolle, ob die Animation vollständig im Speicher ist oder dynamisch von der Festplatte geladen wird. Sie können alle Anim-Befehle wie `ScaleAnim()` auch mit Anims verwenden, die von der Festplatte geladen werden. Animationsebenen werden auch mit Plattenanims korrekt behandelt. (V3.0)

#### LoadAlpha:

Setzen Sie diesen Tag auf `True`, wenn der Alphakanal der Animation auch geladen werden soll. Bitte beachten Sie, dass die meisten Animformate keine Alphakanäle unterstützen. Daher ist es ratsam, dass Sie die Animation manuell aus einem PNG-Bild mit `CreateAnim()` erstellen, wenn Sie einen Alphakanal in Ihrer Animation haben müssen. Dieser Tag ist standardmäßig auf `False` gesetzt. (V4.5)

#### X, Y, Width, Height, Frames, FPR:

Diese Gruppe von Tags müssen Sie nur verwenden, wenn Sie eine Bilddatei als Quelle angeben, damit `LoadAnim()` aus dem Bild eine Animation erstellen kann. `Width` und `Height` (Breite und Höhe) definieren die Dimensionen für die Animation und `Frames` gibt an, wie viele Einzelbilder `LoadAnim()` aus dem Quellbild lesen wird. Wenn die Einzelbilder in mehreren Reihen im Quellbild gespeichert sind, müssen Sie auch das Argument `FPR` (Abkürzung für Frames (Einzelbilder) pro Reihe) verwenden müssen, um `LoadAnim()` zu sagen, wie viele Bilder es in jeder Reihe sind. Schließlich kann man `LoadAnim()` mitteilen, wo in der Bilddatei das Scannen starten soll, indem Sie die Tags `X` und `Y` angeben (beide standardmäßig auf 0 gesetzt). `LoadAnim()` wird dann an der Position `X/Y` die in `Frames` definierte Anzahl Bilder mit den Abmessungen von `Width` und `Height` aus der Bilddatei `filename$` lesen. Nachdem die in `FPR` angegebene Anzahl Bilder gelesen hat, wird in die nächste Zeile vorgerückt. (V4.5)

#### SkipLoopFrames:

Wenn Sie diesen Tag auf `True` setzen, wird Hollywood automatisch die letzten beiden Einzelbilder der Animation überspringen. Dies ist nur für IFF-Anims erforderlich, die zwei Schleifeneinzelbilder (loop frames) am Ende der Animation haben. Die automatische Erkennung von Schleifeneinzelbilder ist nicht möglich, denn das würde von Hollywood erfordern, zunächst die gesamte Animation zu dekodieren. Deshalb müssen Sie Hollywood manuell mitteilen, ob die Animation Schleifeneinzelbilder hat oder nicht. (V5.3)

#### Deinterlace:

Hiermit können Sie festlegen, wie Hollywood interlaced Anims darstellen soll. Dies kann entweder `#DEINTERLACE_DEFAULT` oder `#DEINTERLACE_DOUBLE` eingestellt werden. Wenn Sie `#DEINTERLACE_DEFAULT` benutzen (das ist, wie der Name auch impliziert, voreingestellt), wird Hollywood zwei Halbbilder in ein Vollbild kombinieren. Dies geschieht meist in bester Qualität, kann aber zu visuelle Artefakte führen, wenn eine Menge Bewegungen in der Animation vorkommen. Wenn Sie stattdessen `#DEINTERLACE_DOUBLE` verwenden, wird Hollywood die Zeilen eines

Halbbildes verdoppeln, um ein Vollbild zu erhalten. Dies führt zu einem gewissen Qualitätsverlust, kann aber die Animation zu einem glatteren Aussehen verhelfen. Welches der bessere Deinterlacing-Modus ist, hängt immer von der Anim ab. Beachten Sie, dass Sie sich eigentlich überhaupt nicht um diesen Tag kümmern müssen, weil Deinterlacing eigentlich nur für einige obskure IFF-Formate erforderlich ist, die ANIM16i und ANIM32i Interlaced-Einzelbilder speichern. (V5.3)

**Loader:** Mit diesem Tag können Sie ein oder mehrere Formatlademodule angeben, die die Anim laden sollen. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Lademodulen beinhaltet. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, der die angegebene Datei öffnet. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Adaptermodulen beinhaltet. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**LoadTransparency:**

Ist dieser Tag auf `True` gesetzt, wird die monochrome Transparenz der Animation geladen. Bitte beachten Sie, dass dieser Tag speziell für monochrome Transparenzkanäle ausgelegt ist, die einen transparenten Stift in einer palettenbasierten Animation haben. Wenn Sie den Alphakanal einer Animation laden möchten, stellen Sie den Tag `LoadAlpha` auf `True`. Dieser `LoadTransparency`-Tag ist standardmäßig auf `False` gesetzt. (V6.0)

**LoadPalette:**

Wenn dieser Tag auf `True` gesetzt ist, lädt Hollywood die Animation als Palettenanimation. Dies bedeutet, dass Sie die Palette der Animation abrufen und ändern können, was für bestimmte Effekte wie Farbwechsel nützlich ist. Sie können Stifte auch mit dem Tag `TransparentPen` (siehe unten) oder dem Tag `LoadTransparency` (siehe oben) transparent machen. Palettenanimationen haben auch den Vorteil, dass weniger Speicher benötigt wird, da 1 Pixel nur 1 Byte Speicher anstelle von 4 Byte für 32-Bit-Bilder benötigt. Dieser Tag ist standardmäßig auf `False` eingestellt. (V9.0)

**TransparentPen:**

Wenn der Tag `LoadPalette` auf `True` gesetzt wurde (siehe oben), kann mit dem Tag `TransparentPen` ein Stift definiert werden, der transparent gemacht werden soll. Stifte werden ab 0 gezählt. Alternativ können Sie den Tag `LoadTransparency` auch auf `True` setzen, um Hollywood zu zwingen, den in der Animationsdatei gespeicherten transparenten Stift zu verwenden (sofern das Animationsformat die Speicherung transparenter Stifte unterstützt). Dieser Tag ist standardmäßig auf `#NOPEN` gesetzt. (V9.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden

den, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Bitte beachten Sie, dass die Tags **Transparency**, **LoadTransparency** und **LoadAlpha** sich gegenseitig ausschließen. Eine Animation kann nicht eine Maske und einen Alphakanal haben!

Ab Hollywood 9.0 kann dieser Befehl auch Vektor-Animationsformate laden, wenn Sie ein entsprechendes Plugin installiert haben. Beachten Sie jedoch, dass beim Laden von Vektoranimformaten mit **LoadAnim()** die Animation möglicherweise nicht alle Funktionen normaler Animationen unterstützt. Siehe [Abschnitt 17.25 \[Vektoranimationen\]](#), [Seite 214](#), für weitere Informationen zu Vektoranimationen.

Dieser Befehl ist auch als Präprozessor vorhanden: Benutzen Sie **@ANIM**, um Animationen vorzuladen.

#### EINGABEN

<b>id</b>	ein Wert, der diese Animation später im Code identifiziert oder <b>Nil</b> für die automatische ID-Zuweisung
<b>filename\$</b>	die Animationsdatei
<b>table</b>	optional: eine Tabelle, die eine Kombination der oben erwähnten Tags enthält (2.5)

#### RÜCKGABEWERTE

<b>id</b>	optional: Identifikator der Animation; Wird nur zurückgegeben werden, wenn Sie <b>Nil</b> als Argument 1 angegeben haben (siehe oben)
-----------	---

#### BEISPIEL

```
LoadAnim(2, "MyAnim.gif", {Transparency = #RED})
```

Dieser Code lädt die Datei "MyAnim.gif" als Animation 2 mit Rot als Transparenzfarbe.

## 17.14 LoadAnimFrame

#### BEZEICHNUNG

**LoadAnimFrame** – lädt ein Animationseinzelbild als Pinsel (V1.5)

#### ÜBERSICHT

```
LoadAnimFrame(id, frame, anim$[, table])
```

#### BESCHREIBUNG

Dieser Befehl lädt ein einziges Animationseinzelbild (Frame) als Pinsel und weist ihm den Identifikator **id** zu. Die Animationsdatei wird durch die Zeichenfolge **anim\$** angegeben. Das Argument **frame** legt fest, welches Einzelbild geladen wird. Wenn Sie das letzte Bild laden möchten, setzen Sie **frame** auf -1.

Das vierte Argument **table** ist optional. Es ist eine Tabelle, die weitere Möglichkeiten für den Ladevorgang bereit hält. Die folgenden Tags können verwendet werden:



**Transparency:**

Mit diesem Tag können Sie eine Farbe in der **RGB-Notation** angeben, die in der Animation transparent werden soll.

**LoadAlpha:**

Setzen Sie diesen Tag auf **True**, wenn der Alphakanal der Animation auch geladen werden soll. Bitte beachten Sie, dass die meisten Animformate keine Alphakanäle unterstützen. Daher ist es ratsam, dass Sie die Animation manuell aus einem PNG-Bild mit **CreateAnim()** erstellen, wenn Sie einen Alphakanal in Ihrer Animation haben müssen. Dieser Tag ist standardmäßig auf **False** gesetzt. (V4.5)

**Loader:** Mit diesem Tag können Sie ein oder mehrere Formatlademodule angeben, die diese Animation laden sollen. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehrerer Lademodulen enthält. Standardmäßig wird der mit **SetDefaultLoader()** eingestellte Lader verwendet. Siehe **Abschnitt 7.9 [Lade- und Adaptermodule]**, Seite 96, für Details. (V6.0)

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen sollen. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen enthält. Standardmäßig wird der mit **SetDefaultAdapter()** eingestellte Adapter verwendet. Siehe **Abschnitt 7.9 [Lade- und Adaptermodule]**, Seite 96, für Details. (V6.0)

**LoadTransparency:**

Ist dieser Tag auf **True** gesetzt, wird die monochrome Transparenz der Animation geladen. Bitte beachten Sie, dass dieser Tag speziell für monochrome Transparenzkanäle ausgelegt ist, die einen transparenten Stift in einer palettenbasierten Animation haben. Wenn Sie den Alphakanal einer Animation laden möchten, stellen Sie den Tag **LoadAlpha** auf **True**. Dieser **LoadTransparency**-Tag ist standardmäßig auf **False** gesetzt. (V6.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe **Abschnitt 7.10 [Benutzer-Tags]**, Seite 100, für Details. (V10.0)

Bitte beachten Sie, dass die Tags **Transparency**, **LoadTransparency** und **LoadAlpha** sich gegenseitig ausschließen. Eine Animation kann nicht eine Maske und einen Alphakanal haben!

Siehe **Abschnitt 17.13 [LoadAnim]**, Seite 197, für eine Liste der unterstützten Animationsformate.

**EINGABEN**

<b>id</b>	ID des Pinsels, der mit diesem Befehl erstellt wird
<b>frame</b>	das zu ladende Einzelbild (Bereich geht von 1 bis Anzahl Einzelbilder); geben Sie -1 an, wenn Sie das letzte Einzelbild laden möchten

**animid**     ID der Animation

**table**     optional: weitere Optionen (siehe oben) (V5.1)

**BEISPIEL**

```
LoadAnimFrame(1, 5, "Animations/HugeAnim.gif")
DisplayBrushFX(1, #CENTER, #CENTER, #CROSSFADE)
```

Der obige Code lädt das Einzelbild 5 der Animation "Animationen/HugeAnim.gif" als Pinsel 1 und zeigt es mit einem Effekt auf dem Display.

## 17.15 ModifyAnimFrames

**BEZEICHNUNG**

ModifyAnimFrames – fügt ein Einzelbild hinzu oder entfernt es (V4.5)

**ÜBERSICHT**

```
ModifyAnimFrames(id, frames[, pos])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um eine vorhandene Animation zu erweitern oder zu schrumpfen. Wenn Sie einen positiven Wert in **frames** übergeben, dann wird die Animation durch diese Anzahl von Einzelbildern (Frames) verlängert. Wenn Sie einen negativen Wert eintragen, gibt das die Anzahl der Einzelbilder an, welche aus der Animation entfernt werden.

Das optionale Argument **pos** kann verwendet werden, um festzulegen, wo oder von wo die Einzelbilder eingefügt bzw. entfernt werden sollen. Wenn Sie das optionale Argument **pos** nicht angeben oder auf 0 setzen, werden die Einzelbilder am Ende der Animation angehängt oder vom Ende der Animation entfernt werden.

Dieser Befehl funktioniert nur mit Animationen, welche vollständig im Speicher gepuffert sind. Sie können ihn nicht für Animationen verwenden, die direkt von der Festplatte abgespielt werden.

**EINGABEN**

**id**             ID der Animation, welche modifiziert wird

**frames**        Anzahl einzufügender Einzelbilder (wenn **frames** positiv ist) oder Anzahl der zu entfernenden Einzelbilder (wenn **frames** negativ ist)

**pos**            optional: wo die Einzelbilder eingefügt oder entfernt werden (voreingestellt ist 0, was einfügen oder entfernen am Ende bedeutet)

**BEISPIEL**

```
ModifyAnimFrames(1, -5, 1)
```

Dieser Code entfernt die ersten fünf Bilder aus der Animation Nummer 1.

## 17.16 MoveAnim

### BEZEICHNUNG

MoveAnim – bewegt eine Animation von a nach b

### ÜBERSICHT

```
MoveAnim(id, xa, ya, xb, yb[, table])
```

### BESCHREIBUNG

Dieser Befehl bewegt (scrollt) die durch `id` angegebene Animation von der durch `xa` und `ya` angegebenen Stelle zu der Stelle `xb/yb`.

Weitere Optionen sind mit dem optionalen Argument `table` möglich. Sie können die Bewegungsgeschwindigkeit sowie Spezialeffekte angeben und ob die Bewegung asynchron ausgeführt wird. Siehe [Abschnitt 43.46 \[MoveBrush\]](#), Seite 948, für weitere Informationen über das optionale Tabellenargument.

Neben den Tabellenelementen, welche beim Befehl `MoveBrush()` erklärt werden, akzeptiert `MoveAnim()` das zusätzliche Tabellenargument `AnimSpeed`. Dieser Geschwindigkeitswert definiert, nach wie vielen Bewegungsschritten das nächste Einzelbild angezeigt werden soll. Eine höhere Zahl bedeutet also eine geringere Wiedergabegeschwindigkeit der Animation.

Ab Hollywood 4.5 können Sie die neue Konstante `#DEFAULTSPEED` im Tabellenargument `Speed` angeben. (Siehe `MoveBrush()`). Wenn Sie `#DEFAULTSPEED` verwenden, wird Hollywood die Abspielgeschwindigkeit verwenden, wie sie in der Animationsdatei definiert ist. Beachten Sie, dass nicht alle Animationen eine solche Geschwindigkeit definieren, womit die Wiedergabe falsch aussehen kann.

### EINGABEN

<code>id</code>	ID der Animation, die als Quelle benutzt werden soll
<code>xa</code>	Anfangs-X-Position
<code>ya</code>	Anfangs-Y-Position
<code>xb</code>	End-X-Position
<code>yb</code>	End-Y-Position
<code>table</code>	optional: weitere Konfigurationen für die Bewegung

### BEISPIEL

```
MoveAnim(1, 100, 50, 0, 50, {Speed = 5, AnimSpeed = 4})
```

Verschiebt die Animation 1 von 100:50 nach 0:50 mit der Geschwindigkeit 5 und der Abspielgeschwindigkeit 4.

## 17.17 OpenAnim

### BEZEICHNUNG

OpenAnim – öffnet eine Animation (V9.0)

### ÜBERSICHT

```
[id] = OpenAnim(id, filename$[, table])
```

## BESCHREIBUNG

Dieser Befehl öffnet die in `filename$` angegebene Animation und weist ihr den Identifikator `id` zu. Wenn Sie in `id` `Nil` übergeben, wählt `OpenAnim()` automatisch eine ID aus und gibt sie zurück.

Im Gegensatz zum Befehl `LoadAnim()` lädt `OpenAnim()` keine Einzelbilder in den Speicher. Somit wird die Kontrolle schnell wieder an das Skript zurückgegeben und es wird nicht viel Speicher verbraucht. Die Verwendung von `OpenAnim()` entspricht im Wesentlichen dem Aufruf von `LoadAnim()`, mit dem Tabellenargument `FromDisk`, das auf `True` gesetzt ist.

Animationsformate, die auf allen Plattformen unterstützt werden, sind IFF ANIM, GIF ANIM, AVI (unkomprimiert oder mit Motion JPEG-Komprimierung) und Formate, für die Sie ein Plugin haben. Abhängig von der Plattform, auf der Hollywood ausgeführt wird, werden möglicherweise mehr Animationsformate unterstützt. Auf Amiga-kompatiblen Systemen kann Hollywood beispielsweise auch alle Animationsformate öffnen, für die Sie Datentypen haben. Unter Windows kann `OpenAnim()` auch Animationsformate laden, die von der Windows-Imaging-Komponente unterstützt werden.

Das dritte Argument ist optional. Es ist eine Tabelle, mit der weitere Optionen für den Ladevorgang eingestellt werden können. Die folgenden Felder der Tabelle können verwendet werden:

### Transparency:

In diesem Feld können Sie eine Farbe im **RGB-Format** angeben, die in der Animation transparent erscheinen soll.

### LoadAlpha:

Setzen Sie dieses Feld auf `True`, wenn auch der Alphakanal der Animation geladen werden soll. Die Voreinstellung für dieses Feld ist `False`.

### SkipLoopFrames:

Wenn Sie diesen Tag auf `True` setzen, wird Hollywood automatisch die letzten beiden Einzelbilder der Animation überspringen. Dies ist nur für IFF-Anims erforderlich, die zwei Schleifeneinzelbilder (loop frames) am Ende der Animation haben. Die automatische Erkennung von Schleifeneinzelbildern ist nicht möglich, denn das würde von Hollywood erfordern, zunächst die gesamte Animation zu dekodieren. Deshalb müssen Sie Hollywood manuell mitteilen, ob die Animation Schleifeneinzelbilder hat oder nicht.

### Deinterlace:

Hiermit können Sie festlegen, wie Hollywood Interlaced-Anims darstellen soll. Dies kann entweder `#DEINTERLACE_DEFAULT` oder `#DEINTERLACE_DOUBLE` eingestellt werden. Wenn Sie `#DEINTERLACE_DEFAULT` benutzen (das ist, wie der Name auch impliziert, voreingestellt), wird Hollywood zwei Halbbilder in ein Vollbild kombinieren. Dies geschieht meist in bester Qualität, kann aber zu visuelle Artefakte führen, wenn eine Menge Bewegungen in der Animation vorkommen. Wenn Sie stattdessen `#DEINTERLACE_DOUBLE` verwenden, wird Hollywood die Zeilen eines Halbbildes verdoppeln, um ein Vollbild zu erhalten. Dies führt zu einem gewissen Qualitätsverlust, kann aber die Animation zu einem glatteren

Aussehen verhelfen. Welches der bessere Deinterlacing-Modus ist, hängt immer von der Animation ab. Beachten Sie, dass Sie sich normalerweise überhaupt nicht um diesen Tag kümmern müssen, weil Deinterlacing eigentlich nur für einige obskure IFF-Formate erforderlich ist, die ANIM16i und ANIM32i Interlaced-Einzelbilder speichern.

**Loader:** Mit diesem Tag können Sie einen oder mehrere Formatlademodule angeben, die die Animation laden soll. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Lademodulen beinhaltet. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details.

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, der die angegebene Datei öffnet. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Adaptermodulen beinhaltet. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details.

**LoadTransparency:**

Ist dieser Tag auf `True` gesetzt, wird die monochrome Transparenz der Animation geladen. Bitte beachten Sie, dass dieser Tag speziell für monochrome Transparenzkanäle ausgelegt ist, die einen transparenten Stift in einer palettenbasierten Animation haben. Wenn Sie den Alphakanal einer Animation laden möchten, stellen Sie den Tag `LoadAlpha` auf `True`. Dieser `LoadTransparency`-Tag ist standardmäßig auf `False` gesetzt.

**LoadPalette:**

Wenn dieser Tag auf `True` gesetzt ist, lädt Hollywood die Animation als Palettenanimation. Dies bedeutet, dass Sie die Palette der Animation abrufen und ändern können, was für bestimmte Effekte wie Farbwechsel nützlich ist. Sie können Stifte auch mit dem Tag `TransparentPen` (siehe unten) oder dem Tag `LoadTransparency` (siehe oben) transparent machen. Palettenanimationen haben auch den Vorteil, dass weniger Speicher benötigt wird, da 1 Pixel nur 1 Byte Speicher anstelle von 4 Byte für 32-Bit-Bilder benötigt. Dieser Tag ist standardmäßig auf `False` eingestellt. (V9.0)

**TransparentPen:**

Wenn der Tag `LoadPalette` auf `True` gesetzt wurde (siehe oben), kann mit dem Tag `TransparentPen` ein Stift definiert werden, der transparent gemacht werden soll. Stifte werden ab 0 gezählt. Alternativ können Sie den Tag `LoadTransparency` auch auf `True` setzen, um Hollywood zu zwingen, den in der Animationsdatei gespeicherten transparenten Stift zu verwenden (sofern das Animationsformat die Speicherung transparenter Stifte unterstützt). Dieser Tag ist standardmäßig auf `#NOPEN` gesetzt. (V9.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die

die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Bitte beachten Sie, dass die Tags `Transparency`, `LoadTransparency` und `LoadAlpha` sich gegenseitig ausschließen. Eine Animation kann nicht eine Maske und einen Alphakanal haben!

Dieser Befehl ist auch als Präprozessor-Anweisung verfügbar: Verwenden Sie `@ANIM`, um Animationen vom Präprozessor vorzuladen. Beachten Sie jedoch, dass Sie in diesem Fall `FromDisk` auf `True` setzen müssen, um das gleiche Verhalten wie `OpenAnim()` zu erzielen. Wenn Sie `FromDisk` nicht auf `True` setzen, lädt `@ANIM` die gesamte Animation in den Speicher!

Ab Hollywood 9.0 kann dieser Befehl auch Vektoranimationsformate öffnen, wenn Sie ein entsprechendes Plugin installiert haben. Beachten Sie jedoch, dass beim Öffnen von Vektoranimationsformaten mit `OpenAnim()` möglicherweise nicht alle Funktionen normaler Animationen unterstützt werden. Siehe [Abschnitt 17.25 \[Vektoranimationen\]](#), [Seite 214](#), für weitere Informationen zu Vektoranimationen.

Verwenden Sie den Befehl `CloseAnim()`, um eine von `OpenAnim()` zugewiesene Animation zu schließen und aus dem Speicher zu löschen. Siehe [Abschnitt 17.4 \[CloseAnim\]](#), [Seite 191](#), für Details.

#### EINGABEN

`id` ID der Animation oder `Nil` für die automatische ID-Zuweisung

`filename$` Datei zum Laden

`table` optional: weitere Optionen (siehe oben)

#### RÜCKGABEWERTE

`id` optional: Identifikator der Animation; wird nur zurückgegeben, wenn Sie `Nil` als Argument 1 angegeben haben (siehe oben)

#### BEISPIEL

```
OpenAnim(2, "MyAnim.gif", {Transparency = #RED})
```

Dies öffnet "MyAnim.gif" als Animation 2, wobei die Farbe Rot transparent ist.

## 17.18 PlayAnim

#### BEZEICHNUNG

`PlayAnim` – spielt eine Animation ab

#### ÜBERSICHT

```
[handle] = PlayAnim(id[, x, y, table])
```

#### BESCHREIBUNG

Dieser Befehl beginnt, eine zuvor geladene und durch `id` angegebene Animation abzuspielen. Optional können Sie die X- und Y-Koordinaten der Ausgabe angeben.

Wenn Ebenen aktiviert sind, wird dieser Befehl eine neue Ebene des Typs `#ANIM` dem Ebenenstapel hinzuzufügen.

Ab Hollywood 4.0 akzeptiert `PlayAnim()` ein zusätzliches optionales Tabellenargument, das verwendet werden kann, um mehrere weitere Wiedergabeoptionen zu konfigurieren:

- Speed:** Legt die Wiedergabegeschwindigkeit für die Animation fest. Je höher die Zahl, desto langsamer ist die Wiedergabegeschwindigkeit. Sie können auch eine Konstante für die Geschwindigkeit (`#SLOWSPEED`, `#NORMALSPEED` oder `#FASTSPEED`) angeben. Neu in Hollywood 4.5: Sie können hier die neue Konstante `#DEFAULTSPEED` angeben. Wenn Sie `#DEFAULTSPEED` verwenden, wird Hollywood die Geschwindigkeit verwenden, wie sie in der Animationsdatei definiert ist. Beachten Sie, dass nicht alle Animationen eine solche Geschwindigkeit definieren. Daher kann die Wiedergabe falsch aussehen.
- Times:** Gibt an, wie oft die Animation gespielt wird (Standard: 1, spielt die Animation nur einmal). Wenn Sie die Animation als Endlosschleife wollen, geben Sie hier 0 ein.
- Async:** Sie können diesen Tag verwenden, um ein asynchrones Zeichnungsobjekt für diese Wiedergabe zu erstellen. Wenn Sie hier `True` angeben, wird `PlayAnim()` sofort verlassen und es wird ein Handler für das asynchrone Zeichnungsobjekt kreiert, den Sie dann mit dem Befehl `AsyncDrawFrame()` verwenden können. Siehe [Abschnitt 19.1 \[AsyncDrawFrame\]](#), [Seite 231](#), für weitere Informationen über asynchrone Zeichnungsobjekte.

## EINGABEN

- id** ID der abzuspielenden Animation
- x** optional: X-Position bei der Wiedergabe (voreingestellt: 0)
- y** optional: Y-Position bei der Wiedergabe (voreingestellt: 0)
- table** optional: weitere Konfigurationen für das Abspielen der Anim

## RÜCKGABEWERTE

- handle** optional: Handler auf ein asynchrones Zeichnungsobjekt; wird nur zurückgegeben, wenn `Async` auf `True` gesetzt wurde (siehe oben)

## BEISPIEL

```
PlayAnim(1)
```

Spielt die Animation 1 ab.

# 17.19 PlayAnimDisk

## BEZEICHNUNG

`PlayAnimDisk` – spielt eine Animation direkt von einem Datenträger (V1.5)

## ÜBERSICHT

```
PlayAnimDisk(anim$, x, y, frameskip, speed, transcolor, times)
```

## BESCHREIBUNG

Dieser Befehl spielt die in `anim$` angegebene Animation direkt von einem Datenträger. Das ist sinnvoll, wenn Sie große Animationen darstellen wollen, die nicht in den Speicher

passen. Natürlich ist diese Methode um ein vielfaches langsamer als `PlayAnim()`, welches eine Animation aus dem Speicher abspielt.

Neu in Hollywood 1.9 ist das optionale `times` Argument, welches angibt, wie viele Male die Animation gespielt werden soll. Voreingestellt ist hier 1; die Animation wird also nur einmal gespielt. Soll die Animation unendlich wiederholt werden, geben Sie bitte 0 an.

Wichtiger Hinweis: Ab Hollywood 2.5 ist es besser, `LoadAnim()` zu verwenden und das Argument `FromDisk` auf `True` zu setzen. Dies gibt Ihnen mehr Flexibilität, da Sie auch die anderen Befehle von der Animationsbibliothek (wie `ScaleAnim()` etc.) verwenden können und Sie die Möglichkeit haben, auch auf die Animationsebenen mit `NextFrame()` zuzugreifen. All das ist mit `PlayAnimDisk()` nicht möglich.

Siehe [Abschnitt 17.13 \[LoadAnim\]](#), Seite 197, für eine Liste der unterstützten Animationsformate.

## EINGABEN

<code>anim\$</code>	Animationsdatei, die abgespielt werden soll
<code>x</code>	optional: X-Position der Animation auf dem Display (voreingestellt auf 0)
<code>y</code>	optional: Y-Position der Animation auf dem Display (voreingestellt auf 0)
<code>frameskip</code>	optional: Anzahl zu überspringender Bilder (voreingestellt auf 0)
<code>speed</code>	optional: Wartezeit nach jedem Bild in Ticks (voreingestellt auf 0)
<code>transcolor</code>	optional: transparente Farbe in der Animation (voreingestellt auf <code>#NOTRANSARENCY</code> )
<code>times</code>	optional: Anzahl der Male, die die Animation gespielt wird (voreingestellt ist 1) (V1.9)

## BEISPIEL

```
PlayAnimDisk("Animations/LargeAnim.gif", 0, 0, 3)
```

Obenstehender Code spielt die Animation "Animations/LargeAnim.gif" ab und überspringt 3 Bilder pro Durchlauf.

## 17.20 SaveAnim

### BEZEICHNUNG

SaveAnim – speichert die Animation auf der Festplatte (V4.5)

### ÜBERSICHT

```
SaveAnim(id, file$[, format, table])
```

### BESCHREIBUNG

Dieser Befehl speichert die durch `id` angegebene Animation in der durch `file$` angegebenen Datei im durch `format` angegebenen Animationsformat. Dies kann entweder eine



der folgenden Konstanten oder ein von einem Plugin bereitgestelltes Animationsformat sein:

**#ANMFMT\_GIF:**

GIF-Format. Da GIF-Anims immer palettenbasiert sind, müssen RGB-Grafiken quantisiert werden, bevor sie als GIF exportiert werden können. Sie können die Tags **Colors** und **Dither** (siehe unten) verwenden, um die Anzahl der Paletteneinträge festzulegen, die der Animation zugewiesen werden sollen und ob Dithering angewendet werden soll oder nicht. Wenn **#ANMFMT\_GIF** mit einer Palettenanimation verwendet wird, wird keine Quantisierung durchgeführt. **#ANMFMT\_GIF** unterstützt auch Palettenanimation mit einem transparenten Stift. **#ANMFMT\_GIF** ist das von **SaveAnim()** verwendete Standardformat.

**#ANMFMT\_MJPEG:**

AVI mit Motion JPEG-Komprimierung. Dies ist ein verlustbehaftetes Animationsformat, so dass Sie den Tag **Quality** (siehe unten) festlegen können, um die zu verwendende Komprimierungsstufe zu wählen.

**#ANMFMT\_IFF:**

IFF-Animation. Hollywood verwendet für IFF-Animationen den Komprimierungsmodus 5 (den gebräuchlichsten Komprimierungsmodus) verwenden. Da IFF-Animationen immer palettenbasiert sind, müssen RGB-Grafiken quantisiert werden, bevor sie als IFF exportiert werden können. Sie können die Tags **Colors** und **Dither** (siehe unten) verwenden, um die Anzahl der Paletteneinträge festzulegen, die der Animation zugewiesen werden sollen und ob Dithering angewendet werden soll oder nicht. Wenn **#ANMFMT\_IFF** mit einer Palettenanimation verwendet wird, wird keine Quantisierung durchgeführt. **#ANMFMT\_IFF** unterstützt auch Palettenanimation mit einem transparenten Stift. (V9.0)

Mit dem optionalen Argument **table** können Sie weiterer Parameter festlegen:

- Dither:** Setzen Sie **Dither** auf **True**, um Dithering zu ermöglichen. Dieser Tag wird nur verarbeitet, wenn das Zielformat palettenbasiert ist und die Quelldaten RGB sind. GIF- und IFF-Animationen verwenden immer eine Farbpalette. Der Standardwert ist **False**, was kein Dithering bedeutet.
- Depth:** Gibt die gewünschte Farbtiefe der Animation an. Dies wird nur berücksichtigt, wenn das Format palettenbasiert ist und die Quelldaten im RGB-Format vorliegen. Gültige Werte liegen zwischen 1 (= 2 Farben) und 8 (= 256 Farben). Die Voreinstellung ist 8. (V9.0)
- Colors:** Dies ist eine Alternative zum Tag **Depth**. Anstelle einer Bittiefe können Sie hier angeben, wie viele Farben die Animation verwenden soll. Auch dies wird nur berücksichtigt, wenn das Format palettenbasiert ist und die Quelldaten im RGB-Format vorliegen. Gültige Werte liegen zwischen 1 und 256. Die Voreinstellung ist 256.

- Optimize:** Gibt an, ob Hollywood versuchen soll, die Animation zu optimieren. Optimiertes Speichern ist langsamer, aber in der Regel führt es zu kleineren Animationen. Der Standardwert ist **True**.
- Quality:** Hier können Sie einen Wert zwischen 0 und 100 angeben, womit Sie die Komprimierungsqualität für verlustbehaftete Kompressionsformate wählen dürfen. Ein Wert von 100 bedeutet beste Qualität, 0 hingegen ist die schlechteste. Dies steht nur für Anim-Formate zur Verfügung, die verlustbehaftete Kompression unterstützen. Der Standardwert ist 90, womit eine ziemlich gute Qualität möglich ist.
- FPS:** Da Videoformate wie AVI keinen individuellen Verzögerungswert für jedes Einzelbild (Frame) unterstützen, erfordert es einen globalen Wert, der angibt, wie viele Bilder pro Sekunde angezeigt werden sollen. Mit **FPS** können Sie diese Anzahl an Bildern festlegen. Dies wird nur für Videodateiformate berücksichtigt. Voreingestellt sind 25 Bildern pro Sekunde.
- FillColor:** Wenn Sie eine RGB-Animation mit transparenten Pixeln speichern, können Sie hier eine RGB-Farbe angeben, die in alle transparenten Pixel geschrieben werden soll. Dies ist wahrscheinlich nicht von großem praktischen Nutzen. Die Voreinstellung ist **#NOCOLOR**, was bedeutet, dass transparente Pixel so belassen werden, wie sie sind. (V9.0)
- Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, die überprüft werden, ob sie die angegebene Datei speichern können. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Zeichenkette setzen, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der mit **SetDefaultAdapter()** eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V10.0)
- UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an den Dateiadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Hier ist eine Tabelle, die einen Überblick zeigt, welche Tabellenelemente mit den verschiedenen Animationsformate verwendet werden können:

	<b>GIF</b>	<b>IFF</b>	<b>MJPEG AVI</b>
Dither	Ja	Ja	Nein
Colors	Ja	Ja	Nein
Optimize	Ja	Nein	Nein
Depth	Ja	Ja	Nein
FillColor	Ja	Ja	Ja
Quality	Nein	Nein	Ja
FPS	Nein	Nein	Ja

`SaveAnim()` kann sowohl mit diskbasierten Animationen oder vollständig im internen Speicher gepufferten Animationen verwendet werden.

Wenn Sie eine Animation aus mehreren Einzelbildern (z.B. eine Reihe von Pinseln) speichern möchten, können Sie dies mit `BeginAnimStream()`, `WriteAnimFrame()` und `FinishAnimStream()` erledigen.

#### EINGABEN

`id` Animation, die gespeichert werden soll  
`file$` Name der Datei  
`format` optional: Animationsformat (voreingestellt ist `#ANMFMT_GIF`)  
`table` optional: Weitere Parameter fürs Speichern; siehe oben

#### BEISPIEL

```
SaveAnim(1, "my_anim.gif", #ANMFMT_GIF, {Colors = 64, Dither = True})
```

Der obige Code speichert Anim 1 als "my\_anim.gif" in 64 Farben mit aktiviertem Dithering ab.

## 17.21 ScaleAnim

#### BEZEICHNUNG

`ScaleAnim` – skaliert eine Animation

#### ÜBERSICHT

```
ScaleAnim(id, width, height[, smooth])
```

#### BESCHREIBUNG

Dieser Befehl skaliert die durch `id` angegebene Animation auf die gewünschte Breite `width` und Höhe `height`. Bitte beachten Sie, dass das Skalieren einer Animation mit einem 68k-Prozessor eine ganze Weile dauern kann. Optional können Sie `smooth` auf `True` setzen, damit die Animation unter Verwendung von Antialiasing skaliert wird.

Neu in V2.0: Sie können `#KEEPASPRAT` entweder bei der Breite oder Höhe angeben. Hollywood wird dann die Größe automatisch berechnen, indem das Seitenverhältnis der Animation berücksichtigt wird.

Ab Version 2.0 kann bei `width` und `height` eine prozentige Angabe in Form einer Zeichenkette angegeben werden, z.B. "50%".

#### EINGABEN

`id` Animation, welche skaliert wird  
`width` gewünschte neue Breite der Animation  
`height` gewünschte neue Höhe der Animation  
`smooth` optional: Skalierung mit oder ohne Antialiasing (V2.5)

#### BEISPIEL

```
ScaleAnim(1, 320, 240)
```

Dieses Beispiel skaliert Animation 1 auf das Format 320x240 Bildpunkte.

## 17.22 SelectAnim

### BEZEICHNUNG

SelectAnim – wählt ein Animationseinzelbild als Ausgabeziel (V4.5)

### ÜBERSICHT

```
SelectAnim(id, frame[, mode, combomode])
```

### BESCHREIBUNG

Dieser Befehl wählt das angegebene Animationseinzelbild (Frame) als das aktuelle Ausgabeziel. Dies bedeutet, dass alle von Hollywood erzeugten Grafikdaten auf dieses Einzelbild geschrieben werden. Sie müssen die `id` der Animation sowie das Einzelbild angeben.

Das optionale Argument `mode` ist standardmäßig auf `#SELMODE_NORMAL` gesetzt, was bedeutet, dass nur die Farbkanäle der Animation geändert werden. Der Transparenzkanal der Animation (kann entweder eine Maske oder ein Alphakanal sein) wird nicht verändert. Wenn Sie hingegen beim optionalen Argument `mode` `#SELMODE_COMBO` angeben, werden alle Grafikbefehle von Hollywood, welche nach `SelectAnim()` aufgerufen werden, in den Farb- und Transparenzkanal der Animation zeichnen. Wenn die Animation keinen Transparenzkanal hat, verhält sich `#SELMODE_COMBO` gleich wie `#SELMODE_NORMAL`.

Ab Hollywood 5.0 können Sie das optionale Argument `combomode` verwenden, um festzulegen, wie sich `#SELMODE_COMBO` verhalten soll. Wenn `combomode` auf 0 gesetzt ist, werden die Farb- und Transparenzinformation aller Pixel in dem Quellenbild in jedem Fall zu dem Zielbild kopiert, selbst wenn die Pixel unsichtbar sind. Dies ist auch voreingestellt. Wenn `combomode` hingegen auf 1 gesetzt ist, werden nur die sichtbaren Pixel in das Zielbild kopiert. Dies bedeutet, dass, wenn der Alphawert eines Pixels in dem Quellenbild 0 ist (unsichtbar), wird es nicht in das Zielbild kopiert werden.

Ab Hollywood 6.0 steht die neue `combomode` 2 zur Verfügung. Wenn Sie 2 in `combomode` übergeben, wird Hollywood die Farbkanäle und den Alphakanal des Quellbildes mit der Farbe und Alpha der Zielbildkanäle mischen. Wenn Sie später das Zielbild zeichnen, wird es so aussehen, als ob ein Bild hintereinander auf das jeweils andere gezeichnet wurde. Bitte beachten Sie, dass das Argument `combomode` nur zusammen mit `#SELMODE_COMBO` unterstützt wird. Es hat keine Wirkung, wenn sie es mit den anderen Modi verwenden.

Ein alternativer Weg in die Transparenzkanäle einer Animation zu zeichnen, ist dies separat mit den Befehlen `SelectMask()` oder `SelectAlphaChannel()` zu erledigen. Diese beiden Befehle werden jedoch die Schreibdaten nur auf den Transparenzkanal anwenden und somit den Farbkanal nicht ändern. Also, wenn Sie beide Kanäle (Farbe und Transparenz) ändern wollen, müssen Sie `SelectBrush()` mit `mode` `#SELMODE_COMBO` verwenden.

Wenn Sie mit dem Ändern der Animation fertig sind und möchten, dass Ihr Display wieder das Ausgabeziel wird, rufen Sie einfach den Befehl `EndSelect()` auf.

Beachten Sie, dass Sie alle Befehle, die die Animation ändern, nicht aufrufen, während ein Einzelbild der Animation als Ausgabeziel ausgewählt ist (insbesondere `ScaleAnim()`, `FreeAnim()` oder `ModifyAnimFrames()`).

Nur Befehle von Hollywood können verwendet werden, welche Grafiken direkt zeichnen, wenn `SelectAnim()` aktiv ist. Sie können keine animierten Befehle wie `MoveAnim()` oder `DisplayBrushFX()` aufrufen, solange `SelectAnim()` aktiv ist.

Dieser Befehl funktioniert nur mit vollständig im Speicher gepufferten Animationen. Sie können ihn nicht für Animationen verwenden, die direkt von der Festplatte abgespielt werden.

Siehe auch `EndSelect()`, `SelectAlphaChannel()`, `SelectBrush()`, `SelectMask()`, `SelectLayer()` und `SelectBGPic()`.

#### EINGABEN

**id** Animation, welche als Ausgabeziel verwendet werden soll

**mode** optional: Darstellungsmodus (siehe oben); dies kann entweder `#SELMODE_NORMAL` oder `#SELMODE_COMBO` sein; voreingestellt ist `#SELMODE_NORMAL` (V4.5)

**combomode** optional: Wenn `#SELMODE_COMBO` aktiv ist, kann 0,1 oder 2 eingesetzt werden (siehe oben); voreingestellt ist 0 (V5.0)

#### BEISPIEL

```
SelectAnim(1, 5)
SetFillStyle(#FILLCOLOR)
Box(0, 0, 320, 256, #RED)
EndSelect()
```

Dieses Beispiel zeichnet ein 320x256 großes Rechteck in das Einzelbild der Animation 1.

## 17.23 SetAnimFrameDelay

#### BEZEICHNUNG

`SetAnimFrameDelay` – setzt die Verzögerung nach einem Einzelbild (V4.5)

#### ÜBERSICHT

```
SetAnimFrameDelay(id, frame, delay)
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die Verzögerung von einem Einzelbild (Frame) einer bestehenden Animation zu setzen. Übergeben Sie die `id` der Animation sowie das Einzelbild. Die Verzögerungszeit geben Sie in Millisekunden an.

#### EINGABEN

**id** Identifikator der Animation

**frame** Einzelbild, welches Sie verzögern möchten

**delay** neue Verzögerungszeit in Millisekunden

#### BEISPIEL

```
SetAnimFrameDelay(1, 5, 1500)
```

Setzt die Verzögerungszeit des Einzelbildes 5 der Animation 1 auf 1.5 Sekunden.

## 17.24 StopAnim

### BEZEICHNUNG

StopAnim – stoppt eine momentan spielende Animation (nur V1.0)

### ÜBERSICHT

StopAnim(id)

### BESCHREIBUNG

Achtung: Dieser Befehl wurde in Hollywood 1.5 entfernt. Er kann nicht mehr benutzt werden.

Dieser Befehl stoppt die in `id` angegebene Animation. Wird die Animation mehrmals wiedergegeben, wird die zuletzt gestartete angehalten.

### EINGABEN

`id`                    Identifikator der zu stoppenden Animation

## 17.25 Vektoranimationen

Hollywoods Animationsbibliothek unterstützt auch eine spezielle Art von Animation: Eine Vektoranimation. Um herauszufinden, ob `OpenAnim()`, `LoadAnim()` oder `@ANIM` eine Vektoranimation geladen haben, müssen Sie den Befehl `GetAttribute()` mit dem Attribut `#ATTRTYPE` verwenden.

Der Vorteil einer Vektoranimation im Gegensatz zu herkömmlichen Rasteranimationsformaten wie GIF ANIM und IFF ANIM besteht darin, dass Sie sie ohne Qualitätsverluste skalieren und/oder transformieren können. Beispielsweise erzeugt der Befehl `ScaleAnim()` bei Verwendung mit Vektoranimationen hochwertige Animationseinzelbilder. Wenn Ebenen und das Ebenenskalierungs-System aktiviert sind, werden Vektoranimationsebenen automatisch skaliert und ohne Qualitätsverluste transformiert. Wenn Sie in Ihrem Skript nur Vektoranimationen und TrueType-Text verwenden, kann diese auf jede Auflösung skaliert werden und erscheint dennoch perfekt gestochen scharf.

Der Nachteil von Vektoranimationen ist, dass sie nicht von allen Hollywood-Befehlen unterstützt werden. Sie können beispielsweise nicht mit `SelectAnim()` in Animationen zeichnen.

## 17.26 WaitAnimEnd

### BEZEICHNUNG

WaitAnimEnd – hält an, bis die abspielende Animation beendet ist (nur V1.0)

### ÜBERSICHT

WaitAnimEnd(id)

### BESCHREIBUNG

Achtung: Dieser Befehl wurde in Hollywood 1.5 entfernt. Er kann nicht mehr benutzt werden.

Dieser Befehl hält den Programmablauf an, bis die in `id` angegebene Animation beendet ist. Danach wird die Ausführung des Skripts fortgesetzt. Wenn Sie Befehle oder Funktionen während dem Abspielen der Animation aufrufen müssen, verwenden Sie den Befehl `IsAnimPlaying()` in Verbindung mit einer Schleife.

**EINGABEN**

`id`            Identifikator der abspielenden Animation

**BEISPIEL**

```
PlayAnim(1,#PLAYONCE)
WaitAnimEnd(1)
```

Dieser Code spielt die Animation 1 ab und wartet, bis sie beendet ist.

## 17.27 WriteAnimFrame

**BEZEICHNUNG**

WriteAnimFrame – hängt ein Einzelbild an eine sequentielle Animation (V4.5)

**ÜBERSICHT**

```
WriteAnimFrame(id, brush_id[, table])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um ein einziges neues Einzelbild an ein sequentielles Animationsobjekt anzuhängen, welches mit `BeginAnimStream()` erstellt wurde. Um das Einzelbild an die Animation anzuhängen, muss es wie ein Pinsel zur Verfügung stehen. Idealerweise sollte die Pinselgröße mit der Abmessung in `BeginAnimStream()` übereinstimmen, sonst wird dieser Befehl automatisch die Lücken füllen.

Mit dem optionalen Tabellenargument können Sie weitere Parameter konfigurieren:

- X, Y:**        Mit diesen beiden Tags können Sie die Position konfigurieren, bei der der Pinsel in das Einzelbild kopiert wird. Dies ist nützlich, wenn Sie ein Bild hinzufügen, das kleiner ist als die Dimension der Animation. Sie könnten zum Beispiel dieses Bild in dieser Animation zentrieren. Der Standardwert ist 0.0, was die linke obere Ecke bedeutet.
- Delay:**      Setzen Sie diesen Tag, wenn Sie eine Zeitverzögerung für dieses Einzelbild definieren möchten. Die Zeit muss in Millisekunden angegeben werden. Nicht alle Animationsformate unterstützen Einzelbildverzögerung. Siehe Tabelle unten. Der Standardwert ist 0, was keine Verzögerung bedeutet.
- Dither:**      Setzen Sie diesen Tag auf `True`, um Dithering zu aktivieren. Dieses Feld wird nur verarbeitet, wenn das Format palettenbasiert ist und die Quelldaten im RGB-Format vorliegen. GIF- und IFF-Animationen verwenden immer eine Farbpalette. Die Voreinstellung ist `False`, was bedeutet, dass kein Dithering erfolgt.
- Depth:**      Gibt die gewünschte Farbtiefe des Einzelbildes an. Dies wird nur berücksichtigt, wenn das Format palettenbasiert ist und die Quelldaten im RGB-Format vorliegen. Gültige Werte liegen zwischen 1 (= 2 Farben) und 8 (= 256 Farben). Die Voreinstellung ist 8. (V9.0)
- Colors:**      Dies ist eine Alternative zum Tag `Depth`. Anstelle einer Bittiefe können Sie hier angeben, wie viele Farben das Einzelbild verwenden soll. Auch dies wird nur berücksichtigt, wenn das Format palettenbasiert ist und die Quelldaten

im RGB-Format sind. Gültige Werte liegen zwischen 1 und 256. Die Voreinstellung ist 256.

**Optimize:**

Gibt an, ob Hollywood versuchen soll, die Animation zu optimieren. Optimiertes Speichern ist langsamer, aber in der Regel führt es zu kleineren Animationen. Der Standardwert ist **True**.

**FillColor:**

Wenn Sie ein RGB-Einzelbild mit transparenten Pixeln speichern, können Sie hier eine RGB-Farbe angeben, die in alle transparenten Pixel geschrieben werden soll. Dies ist wahrscheinlich von wenig praktischem Nutzen. Die Voreinstellung ist **#NOCOLOR**, was bedeutet, dass transparente Pixel so belassen werden, wie sie sind. (V9.0)

Hier ist eine Tabelle, die einen Überblick zeigt, welche Tabellenelemente mit den verschiedenen Animationsformate verwendet werden können:

	<b>GIF</b>	<b>IFF</b>	<b>MJPEG AVI</b>
X, Y	Ja	Ja	Ja
Delay	Ja	Ja	Nein
Dither	Ja	Ja	Nein
Colors	Ja	Ja	Nein
Optimize	Ja	Nein	Nein
Depth	Ja	Ja	Nein
FillColor	Ja	Ja	Ja

## EINGABEN

- id** ID des Animationsobjekt; muss mit **BeginAnimStream()** erstellt worden sein
- brush\_id** ID des Pinsels, welcher an die Animation angehängt wird
- table** optional: weitere Optionen fürs Speichern; siehe oben

## BEISPIEL

Siehe [Abschnitt 17.3 \[BeginAnimStream\]](#), Seite 189.



## 18 Anwendungsbibliothek

### 18.1 APPAUTHOR

#### BEZEICHNUNG

APPAUTHOR – gibt den Autor der Anwendung an (V2.0)

#### ÜBERSICHT

@APPAUTHOR author\$

#### BESCHREIBUNG

Mit dieser Präprozessor-Anweisung können Sie einfach eine Zeichenfolge in die Anwendung einfügen, die den Autor des Skripts enthält. Diese Anweisung hat keine tatsächliche Funktion. Sie nimmt nur die angegebene Zeichenfolge und speichert sie im Applet oder in der ausführbaren Datei.

Siehe auch @APPCOPYRIGHT, @APPDESCRIPTION, @APPICON, @APPIDENTIFIER, @APTITLE, APPVERSION und GetApplicationInfo().

#### EINGABEN

author\$    Ihren Namen

#### BEISPIEL

Siehe [Abschnitt 18.8 \[APPVERSION\]](#), Seite 223.

### 18.2 APPCOPYRIGHT

#### BEZEICHNUNG

APPCOPYRIGHT – gibt das Urheberrecht der Anwendung an (V2.0)

#### ÜBERSICHT

@APPCOPYRIGHT copyright\$

#### BESCHREIBUNG

Mit dieser Präprozessor-Anweisung können Sie einfach eine Zeichenfolge in die Anwendung einfügen, die das Urheberrecht des Skripts enthält. Diese Anweisung hat keine tatsächliche Funktion. Sie nimmt nur die angegebene Zeichenfolge und speichert sie im Applet oder in der ausführbaren Datei.

Siehe auch @APPAUTHOR, @APPDESCRIPTION, @APPICON, @APPIDENTIFIER, @APTITLE, APPVERSION und GetApplicationInfo().

#### EINGABEN

copyright\$  
                  Urheberrecht des Skripts

#### BEISPIEL

Siehe [Abschnitt 18.8 \[APPVERSION\]](#), Seite 223.

## 18.3 APPDESCRIPTION

### BEZEICHNUNG

APPDESCRIPTION – gibt die Anwendungsbeschreibung an (V2.0)

### ÜBERSICHT

@APPDESCRIPTION desc\$

### BESCHREIBUNG

Mit dieser Präprozessor-Anweisung können Sie einfach eine Zeichenfolge in die Anwendung einfügen, die eine Beschreibung des Skripts enthält. Diese Anweisung hat keine tatsächliche Funktion. Sie nimmt nur die angegebene Zeichenfolge und speichert sie im Applet oder in der ausführbaren Datei.

Unter AmigaOS wird die hier angegebene Beschreibung als Informationstext im Commodity Exchange verwendet.

Siehe auch @APPAUTHOR, @APPCOPYRIGHT, @APPICON, @APPIDENTIFIER, @APPTITLE, APPVERSION und GetApplicationInfo().

### EINGABEN

desc\$      Beschreibung des Skripts

### BEISPIEL

Siehe [Abschnitt 18.8 \[APPVERSION\]](#), Seite 223.

## 18.4 APPENTRY

### BEZEICHNUNG

APPENTRY – deklariert das Projekteingangsskript (V10.0)

### ÜBERSICHT

@APPENTRY file\$

### BESCHREIBUNG

Mit dieser Präprozessor-Anweisung können Sie das Eingangsskript für ein Projekt definieren. Dies ist nur nützlich, wenn Sie ein Projekt haben, das aus mehreren Skripten besteht. In einem solchen Projekt haben Sie normalerweise ein Hauptskript, das @INCLUDE verwendet, um mehrere andere Hilfsskripte einzubinden. Indem Sie @APPENTRY zu Ihren Hilfsskripten hinzufügen, können Sie Hollywood den Namen des Hauptskripts mitteilen. Immer wenn Sie eines der Hilfsskripte ausführen und Hollywood auf die Definition von @APPENTRY stößt, wird das Hauptskript anstelle des Hilfsskripts ausgeführt.

Nehmen wir zum Beispiel an, Ihr Projekt besteht aus den folgenden drei Skripten:

```
main.hws
engine.hws
utils.hws
```

Sie könnten dann die folgende Zeile an den Anfang der Hilfsskripte setzen, d.h. an den Anfang von engine.hws und utils.hws:

```
@APPENTRY "main.hws"
```

Mit dieser Definition wird Hollywood `main.hws` ausführen, wenn Sie `engine.hws` oder `utils.hws` starten, da `main.hws` als Einstiegsskript festgelegt wurde. Das kann sehr praktisch sein, z.B. beim Bearbeiten von Hilfsskripten in der Hollywood-IDE. Wenn Sie `@APPENTRY` verwenden, können Sie einfach auf "Run" klicken und die IDE führt stattdessen das Hauptskript aus.

## EINGABEN

`file$`      Projekteingangsskript-Datei

## 18.5 APPICON

### BEZEICHNUNG

APPICON – gibt das Piktogramm für die Anwendung an (V4.7)

### ÜBERSICHT

```
@APPICON table
@APPICON f$      (V8.0)
```

### BESCHREIBUNG

Mit dieser Präprozessor-Anweisung können Sie das Piktogramm für Ihre Anwendung angeben. Unter Windows, macOS und Linux wird dieses im Fensterrahmen und in mehreren Elementen des Fenstermanagers angezeigt, wie z.B. der Taskleiste. Unter AmigaOS 4 wird es im AmiDock angezeigt, wenn Ihr Skript als registrierte AmigaOS 4-Anwendung läuft. Die Piktogramme, die Sie hier angeben, werden auch mit den kompilierten Applets und ausführbaren Dateien verknüpft. Standardmäßig werden bei ausführbaren Dateien immer das Hollywood-Piktogramm verwendet (die Filmklappe). Wenn Sie lieber Ihr eigenes Piktogramm verwenden möchten, benutzen Sie diese Präprozessor-Anweisung.

Ab Hollywood 8.0 gibt es zwei verschiedene Möglichkeiten, diese Präprozessor-Anweisung zu verwenden: Sie können entweder einzelne Bilder für die verschiedenen Piktogrammgrößen in einer Tabelle angeben (siehe unten) oder Sie können einfach ein Hollywood-Piktogramm dieser Präprozessor-Anweisung übergeben, welches mit dem Befehl `SaveIcon()` erstellt wurde. In diesem Fall muss das Piktogramm im Parameter `f$` übergeben werden. Die Angabe eines einzelnen Piktogramme anstelle einer ganzen Tabelle führt zu Code, der besser lesbar ist, aber natürlich erfordert es, dass Sie das Piktogramm zuerst mit dem Befehl `SaveIcon()` erzeugen.

Wenn Sie einzelne Bilder für die verschiedenen Piktogrammgrößen übergeben wollen, müssen Sie dieser Präprozessor-Anweisung eine Tabelle im Argument `table` übergeben. Die Tabelle kann eines oder mehrere der folgenden Tags enthalten:

**Ic16x16:** Benutzerdefiniertes Piktogramm in der Auflösung von 16x16 Pixel.

**Ic16x16s:**  
Ausgewähltes Zustandspiktogramm in der Größe 16x16. (V6.0)

**Ic24x24:** Benutzerdefiniertes Piktogramm in der Auflösung von 24x24 Pixel.

**Ic24x24s:**  
Ausgewähltes Zustandspiktogramm in der Größe 24x24. (V6.0)

**Ic32x32:** Benutzerdefiniertes Piktogramm in der Auflösung von 32x32 Pixel.

- Ic32x32s:**  
Ausgewähltes Zustandspiktogramm in der Größe 32x32. (V6.0)
- Ic48x48:** Benutzerdefiniertes Piktogramm in der Auflösung von 48x48 Pixel.
- Ic48x48s:**  
Ausgewähltes Zustandspiktogramm in der Größe 48x48. (V6.0)
- Ic64x64:** Benutzerdefiniertes Piktogramm in der Auflösung von 64x64 Pixel.
- Ic64x64s:**  
Ausgewähltes Zustandspiktogramm in der Größe 64x64. (V6.0)
- Ic96x96:** Benutzerdefiniertes Piktogramm in der Auflösung von 96x96 Pixel.
- Ic96x96s:**  
Ausgewähltes Zustandspiktogramm in der Größe 96x96. (V6.0)
- Ic128x128:**  
Benutzerdefiniertes Piktogramm in der Auflösung von 128x128 Pixel.
- Ic128x128s:**  
Ausgewähltes Zustandspiktogramm in der Größe 128x128. (V6.0)
- Ic256x256:**  
Benutzerdefiniertes Piktogramm in der Auflösung von 256x256 Pixel.
- Ic256x256s:**  
Ausgewähltes Zustandspiktogramm in der Größe 256x256. (V6.0)
- Ic512x512:**  
Benutzerdefiniertes Piktogramm in der Auflösung von 512x512 Pixel.
- Ic512x512s:**  
Ausgewähltes Zustandspiktogramm in der Größe 512x512. (V6.0)
- Ic1024x1024:**  
Benutzerdefiniertes Piktogramm in der Auflösung von 1024x1024 Pixel.  
(V7.0)
- Ic1024x1024s:**  
Ausgewähltes Zustandspiktogramm in der Größe 1024x1024. (V7.0)

**DefaultIcon:**

Mit diesem Tag können Sie angeben, welches Piktogramm das Standardpiktogramm sein soll. Sie müssen eine Zeichenfolge übergeben, die eine Piktogrammgröße aus den oben aufgeführten Größen beschreibt, z.B. "64x64" kennzeichnet das in dem Tag **Ic64x64** als das Standardpiktogramm. Das Standardpiktogramm ist das Piktogramm, welches Hollywood in AmiDock auf AmigaOS 4 anzeigen wird, wenn der Tag **RegisterApplication** in **@OPTIONS** auf **True** gesetzt ist. So hat derzeit der Tag **DefaultIcon** nur eine Wirkung unter AmigaOS 4. (V6.0)

Der Grund, warum diese Präprozessor-Anweisung nicht einfach nur ein einziges 512x512-Piktogramm akzeptiert und es dann auf alle anderen Auflösungen skaliert, ist, dass sehr kleine Piktogramme wie 16x16 oder 24x24 nicht wirklich gut aussehen, wenn sie von

einem größeren Piktogramm aus verkleinert werden. Sie sehen viel besser aus, wenn Sie für jede Größe handgefertigt werden. Deshalb akzeptiert diese Präprozessor-Anweisung so viele verschiedene Tags.

Bitte beachten Sie, dass derzeit nicht alle Größen auf allen Plattformen unterstützt werden. Trotzdem sollten Sie sicherstellen, für alle Größen Piktogramme bereitzustellen. Wenn Sie eine Größe auslassen, könnte Hollywood für diese oder gar alle Größen auf das Standard-Piktogramm (Filmklappe) zurück greifen. Also, wenn Sie beabsichtigen, Ihre eigenen Piktogramme zu verwenden, stellen Sie sicher, dass Sie immer in allen oben aufgeführten Größen verfügbar sind.

Die Bilddatei, die als Parameter von den oben aufgeführten Tags benötigt wird, sollte ein PNG-Bild mit Alphakanal sein. Bilder ohne Alphakanal werden auch unterstützt, aber wird nicht empfohlen, weil sie nicht gut aussehen.

Bitte beachten Sie, dass wenn Sie unter AmigaOS 4 ein Piktogramm im AmiDock verwenden möchten, müssen Sie mit dem Tag `DefaultIcon` die Größe des Piktogramme angeben. Wenn Sie den Tag `DefaultIcon` nicht übergeben, wird Hollywood sein Standard-Piktogramm im AmiDock anzeigen (die Filmklappe). Wenn Sie ein Piktogramm im AmiDock anzeigen möchten, welches eine nicht aufgeführte benutzerdefinierte Größe verwendet, können Sie den Tag `DockyBrush` von der Präprozessor-Anweisung `@OPTIONS` verwenden. Siehe [Abschnitt 50.25 \[OPTIONS\]](#), [Seite 1107](#), für Details. Bitte beachten Sie, dass Hollywood zwei verschiedene Docky-Typen unter AmigaOS 4 unterstützt: Standard-Dockies und App-Dockies. Siehe [Abschnitt 16.1 \[Informationen über AmiDock\]](#), [Seite 173](#), für mehr Infos über die Unterschiede der beiden Dockies.

Ab Hollywood 6.0 können Sie für jede Piktogrammgröße zwei Bilder angeben, da auf AmigaOS und kompatiblen Plattformen Piktogramme in der Regel zwei verschiedene Zustände enthalten, während auf allen anderen Systemen Piktogramme nur einzelne statische Bilder haben. Wenn nur Nicht-Amiga-Systeme Ihr Ziel sind, müssen Sie keine Piktogramme für den ausgewählten Status bereitstellen, da sie ohnehin nicht verwendet werden.

Alternativ können Sie anstatt `@APPICON` auch eine der Konsolenargumente `-iconXXX` verwenden.

Beachten Sie, dass diese Präprozessor-Anweisung derzeit keine Auswirkungen auf Amiga-Systeme hat. Wenn Sie das Piktogramm ändern möchten, das Hollywood anzeigt, wenn es auf der Workbench minimiert wird, verwenden Sie den Befehl `SetWBIcon()`.

Siehe auch `@APPAUTHOR`, `@APPCOPYRIGHT`, `@APPDESCRIPTION`, `@APPIDENTIFIER`, `@APTITLE`, und `APPVERSION`.

## EINGABEN

<code>f\$</code>	Name eines Hollywood-Piktogramme ODER
<code>table</code>	eine Tabelle mit einem oder mehreren der unterstützten Tags (siehe oben)

## BEISPIEL

```
@APPICON "icon.png"
```

Der obige Code setzt das Piktogramm "icon.png" als Piktogramm für die Anwendung. Dieses Piktogramm muss mit `SaveIcon()` erstellt worden sein.

```
@APPICON {Ic16x16 = "my16x16icon.png",
Ic24x24 = "my24x24icon.png",
Ic32x32 = "my32x32icon.png",
Ic48x48 = "my48x48icon.png",
Ic64x64 = "my64x64icon.png",
Ic96x96 = "my96x96icon.png",
Ic128x128 = "my128x128icon.png",
Ic256x256 = "my256x256icon.png",
Ic512x512 = "my512x512icon.png",
Ic1024x1024 = "my1024x1024icon.png"}
```

Der obige Code ersetzt alle eingebauten Piktogramme mit den benutzerdefinierten, die in den angegebenen PNG-Bildern zur Verfügung gestellt werden.

## 18.6 APPIDENTIFIER

### BEZEICHNUNG

APPIDENTIFIER – deklariert die Anwendungskennung (V6.1)

### ÜBERSICHT

```
@APPIDENTIFIER id$
```

### BESCHREIBUNG

Geben Sie hier eine eindeutige Kennung für Ihre Anwendung in umgekehrter DNS-Notation an. Einige Befehle von Hollywood wie `LoadPrefs()` und `SavePrefs()` benötigen eine eindeutige Kennung für Ihre Anwendung in umgekehrter DNS-Notation, z.B. `com.airsoftsoftwair.hollywood`. Dies kann mit der Präprozessor-Anweisung `"@APPIDENTIFIER"` geschehen.

Auch beim Kompilieren von Programmpaketen für macOS wird die in `@APPIDENTIFIER` angegebene Kennung automatisch in die Datei `Info.plist` des App-Pakets geschrieben.

Siehe auch `@APPAUTHOR`, `@APPCOPYRIGHT`, `@APPDESCRIPTION`, `@APPICON`, `@APTITLE`, `APPVERSION` und `GetApplicationInfo()`.

### EINGABEN

`id$`            Anwendungskennung in umgekehrter DNS-Schreibweise

### BEISPIEL

Siehe [Abschnitt 18.8 \[APPVERSION\]](#), Seite 223.

## 18.7 APTITLE

### BEZEICHNUNG

APTITLE – gibt den Anwendungstitel an (V2.0)

### ÜBERSICHT

```
@APTITLE title$
```

**BESCHREIBUNG**

Mit dieser Präprozessor-Anweisung können Sie einfach eine Zeichenfolge mit dem Titel des Skripts in der Anwendung aufnehmen. Diese Anweisung hat keine tatsächliche Funktion. Sie nimmt nur die angegebene Zeichenfolge und speichert sie im Applet oder in der ausführbaren Datei.

Unter AmigaOS wird der hier angegebene Titel als Informationstext im Commodity Exchange verwendet.

Siehe auch `@APPAUTHOR`, `@APPCOPYRIGHT`, `@APPDESCRIPTION`, `@APPICON`, `@APPIDENTIFIER`, `APPVERSION` und `GetApplicationInfo()`.

**EINGABEN**

`title$` Titel für das Skript

**BEISPIEL**

Siehe [Abschnitt 18.8 \[APPVERSION\]](#), Seite 223.

## 18.8 APPVERSION

**BEZEICHNUNG**

APPVERSION – gibt die Anwendungsversion an (V2.0)

**ÜBERSICHT**

`@APPVERSION version$`

**BESCHREIBUNG**

Mit dieser Präprozessor-Anweisung können Sie einfach eine Zeichenfolge mit der Version des Skripts in die Anwendung aufnehmen. Diese Anweisung hat keine tatsächliche Funktion. Sie nimmt nur die angegebene Zeichenfolge und speichert sie im Applet oder in der ausführbaren Datei.

Unter AmigaOS wird die hier angegebene Version als Informationstext im Commodity Exchange angezeigt.

Siehe auch `@APPAUTHOR`, `@APPCOPYRIGHT`, `@APPDESCRIPTION`, `@APPICON`, `@APPIDENTIFIER`, `@APTITLE` und `GetApplicationInfo()`.

**EINGABEN**

`version$` Skript Version

**BEISPIEL**

```
@APTITLE "Run-away Randy"
@APPAUTHOR "Andreas Falkenhahn"
@APPCOPYRIGHT "(C) Copyright 2002-2005 by Airsoft Softwair."
@APPVERSION "$VER: Run-away Randy 1.0 (23-Dec-05)"
@APPDESCRIPTION "A really cool jump'n'run game done with Hollywood."
@APPIDENTIFIER "com.airsoftsoftwair.runawayrandy"
```

So sieht ein kompletter Anwendungsinformationssektor aus. Hollywood speichert alle diese Informationen im Applet oder in der ausführbaren Datei. Wenn der Benutzer "Version Run-away-Randy.exe" eingibt, wird die Versionszeichenfolge "\$VER: Run-away Randy 1.0 (23-Dec-05)" zurückgegeben.

## 18.9 DeletePrefs

### BEZEICHNUNG

DeletePrefs – löscht die Benutzereinstellungen (V6.1)

### ÜBERSICHT

DeletePrefs()

### BESCHREIBUNG

Dieser Befehl kann zum Löschen von Benutzereinstellungen verwendet werden, die mit dem Befehl `SavePrefs()` gespeichert wurden. Somit können Sie mit diesem Befehl eine Wiederherstellung der Standardeinstellungen vornehmen. Durch Aufruf von `DeletePrefs()` werden alle Benutzereinstellungen physisch von der Festplatte des Benutzers gelöscht.

Beachten Sie, dass dieser Befehl nur funktionieren wird, wenn Sie eine eindeutige Kennung für Ihre Anwendung mit der Präprozessor-Anweisung `@APPIDENTIFIER` angegeben haben. Siehe [Abschnitt 18.6 \[APPIDENTIFIER\]](#), [Seite 222](#), für Details.

Siehe auch `LoadPrefs()` und `SavePrefs()`.

### EINGABEN

keine

## 18.10 GetApplicationInfo

### BEZEICHNUNG

GetApplicationInfo – gibt Informationen über die Anwendung zurück (V6.0)

### ÜBERSICHT

`t = GetApplicationInfo()`

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um Informationen zu erhalten, welche in den Präprozessor-Anweisungen `@APPXXX` angegeben wurden. `GetApplicationInfo()` gibt eine Tabelle zurück, welche die folgenden Tags enthält:

**Title:** Dies ist festgelegt auf den Wert von `@APPTITLE`. Siehe [Abschnitt 18.7 \[APPTITLE\]](#), [Seite 222](#), für Details.

**Version:** Dies ist festgelegt auf den Wert von `@APPVERSION`. Siehe [Abschnitt 18.8 \[APPVERSION\]](#), [Seite 223](#), für Details.

**Author:** Dies ist festgelegt auf den Wert von `@APPAUTHOR`. Siehe [Abschnitt 18.1 \[APPAUTHOR\]](#), [Seite 217](#), für Details.

**Copyright:** Dies ist festgelegt auf den Wert von `@APPCOPYRIGHT`. Siehe [Abschnitt 18.2 \[APPCOPYRIGHT\]](#), [Seite 217](#), für Details.

**Description:** Dies ist festgelegt auf den Wert von `@APPDESCRIPTION`. Siehe [Abschnitt 18.3 \[APPDESCRIPTION\]](#), [Seite 218](#), für Details.



**Identifizier:**

Dies ist festgelegt auf den Wert von `@APPIDENTIFIER`. Siehe Abschnitt 18.6 [APPIDENTIFIER], Seite 222, für Details. (V6.1)

Siehe auch `@APPAUTHOR`, `@APPCOPYRIGHT`, `@APPDESCRIPTION`, `@APPIDENTIFIER`, `@APTITLE` und `APPVERSION`.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`t` eine Tabelle mit den oben beschriebenen Tags

## 18.11 GetCommandLine

**BEZEICHNUNG**

`GetCommandLine` – gibt die Argumente von der Befehlszeile zurück (V3.0)

**ÜBERSICHT**

```
t, n, console = GetCommandLine()
```

**BESCHREIBUNG**

Mit diesem Befehl können Sie die Argumente abrufen, mit denen Ihr Skript gestartet wurde. Dies ermöglicht es Ihrem Skript, Argumente vom Benutzer zu empfangen und dann entsprechend darauf zu reagieren. Alle Argumente, die nicht von Hollywood erkannt werden, werden an Ihr Skript weitergeleitet. Beachten Sie, dass Argumente mit einem Bindestrichzeichen (-) vorangestellt werden müssen. Ein Parameter kann auch nach jedem Argument folgen.

`GetCommandLine()` gibt drei Werte zurück: Der erste Rückgabewert `t` ist eine Tabelle, die alle Argumente und deren Parameter enthält. Die Tabelle ist ein Feld von "n"-Tabellen mit den Elementen `arg` und `param`. `arg` wird auf das Argument initialisiert mit Ausnahme des Bindestrichs. `param` erhält die Parameter für dieses Argument, wenn es einen gibt, sonst wird eine leere Zeichenfolge ("") empfangen. Der zweite Rückgabewert `n` ist eine Zahl, die einfach angibt, wie viele Argumente von der Befehlszeile aus an Ihr Skript übergeben wurde. Der dritte Rückgabewert `console` ist neu in Hollywood 4.7 und gibt an, ob Hollywood von einer Konsole (`True`) gestartet wurde oder nicht (`False`).

Unter AmigaOS berücksichtigt dieser Befehl auch die Tooltypes des Piktogramme des Skripts oder des Applets, wenn das Programm von der Workbench aus gestartet wurde.

Unter macOS schaut dieser Befehl in den Wörterbucheintrag `CFBundleExecutableArgs` in der Info.plist des Programmpakets, wenn das Programm vom Finder gestartet wurde.

Siehe auch `GetFileArgument()` und `GetRawArguments()`.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`t` Tabelle, welche alle Argumente und ihre Parameter enthält

`n` Anzahl der Argumente, die an die Programmkonsole übergeben wurden

`console`    `True`, wenn das Programm von einer Konsole gestartet wurde, andernfalls `False` (V4.7)

### BEISPIEL

```
args, count = GetCommandLine()
NPrint("Number of arguments:", count)
For Local k = 0 to count - 1
    NPrint("Arg #", k, ":", args[k].arg, "Param:", args[k].param)
Next
```

Der obige Code ruft alle Argumente von der Befehlszeile ab und gibt sie auf dem Bildschirm aus.

## 18.12 GetFileArgument

### BEZEICHNUNG

`GetFileArgument` – ruft das Datei-Argument von einem kompilierten Skript ab (V5.0)

### ÜBERSICHT

```
f$ = GetFileArgument([path])
```

### BESCHREIBUNG

Mit `GetFileArgument()` können Sie das Argument der Datei eines kompilierten Hollywood-Skripts abrufen. So funktioniert dieser Befehl nur bei der Verwendung von Programmen, die mit Hollywood kompiliert wurden, weil im Skript-Modus das Argument der Datei natürlich immer das Hollywood-Skript selbst ist. `GetFileArgument()` erweitert den Befehl `GetCommandLine()`, da letzteres nur erlaubt, Optionsargumente abzurufen, nicht das Argument der Datei selbst.

Dieser Befehl ist besonders beim Erstellen von Tools nützlich, die in der Lage sein sollten, als Betrachter für bestimmte Dateiformate mit Hollywood zu agieren. Mithilfe von `GetFileArgument()` können Sie Programme erstellen, die als Standardtools für bestimmte Dateiformate verwendet werden können.

Ab Hollywood 9.0 gibt es ein optionales Argument namens `path`. Wenn dieses auf `True` gesetzt ist, gibt `GetFileArgument()` einen vollständig qualifizierten Pfad zur Datei statt nur des Dateinamens zurück. Aus Kompatibilitätsgründen ist dieses Argument standardmäßig `False` gesetzt.

Siehe auch `GetCommandLine()` und `GetRawArguments()`.

### EINGABEN

`path`            optional: setzen Sie dies auf `True`, um einen vollständig qualifizierten Pfad zur Datei zu erhalten (Standardwert ist `False`) (V9.0)

### RÜCKGABEWERTE

`f$`              das Datei-Argument, welches an das kompilierte Hollywood-Skript übergeben wurde oder eine leere Zeichenfolge, wenn Hollywood im Interpretermodus ausgeführt wird

## 18.13 GetProgramInfo

### BEZEICHNUNG

GetProgramInfo – ruft Informationen über das aktuelle Programm ab (V3.0)

### ÜBERSICHT

```
type, name$[, hw$] = GetProgramInfo()
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einige Informationen über das derzeit laufende Hollywood-Programm zu erhalten. `GetProgramInfo()` gibt zwei Werte zurück: Der erste Rückgabewert `type` gibt den Programmtyp an, der gerade in Hollywood ausgeführt wird. Diese Variable kann `#PRGTYPE_SCRIPT` für Hollywood Skripte, `#PRGTYPE_APPLET` für Hollywood Applets oder `#PRGTYPE_PROGRAM` für kompilierte ausführbare Dateien sein. Der zweite Rückgabewert `name$` ist eine Zeichenfolge, der den Dateinamen des aktuell laufenden Programms enthält. Wenn das gerade ausgeführte Programm ein Applet oder ein Skript ist, wird sein Name im `namen$` zurückgegeben. Wenn hingegen das aktuell ausgeführte Programm eine kompilierte ausführbare Datei ist, erhält `name$` den Dateinamen dieser ausführbaren Datei.

Wenn das derzeit laufende Programm ein Applet oder ein Skript ist, gibt es einen dritten Rückgabewert `hw$`. Dieser Rückgabewert enthält den Dateinamen des Hollywood-Interpreters, der verwendet wird, um dieses Applet oder Skript auszuführen.

### EINGABEN

keine

### RÜCKGABEWERTE

<code>type</code>	Typ des aktuell laufenden Programms ( <code>#PRGTYPE_SCRIPT</code> , <code>#PRGTYPE_APPLET</code> oder <code>#PRGTYPE_PROGRAM</code> )
<code>name\$</code>	Dateiname des aktuell laufenden Programms
<code>hw\$</code>	optional: Dieser Wert wird nur zurückgegeben, wenn das gerade ausgeführte Programm ein Skript oder ein Applet ist; gibt den Pfad zum Hollywood-Interpreter an, der verwendet wird, um dieses Skript oder Applet auszuführen

## 18.14 GetRawArguments

### BEZEICHNUNG

GetRawArguments – ruft alle an das Programm übergebenen Argumente ab (V10.0)

### ÜBERSICHT

```
args = GetRawArguments()
```

### BESCHREIBUNG

Mit diesem Befehl können Sie alle Argumente abrufen, die Ihrem Programm oder Skript entweder über die Konsole oder den Desktopmanager des Hostsystems übergeben werden. Die Argumente werden in der Tabelle `args` zurückgegeben und sind in keiner Weise formatiert. Sie werden in ihrer Rohform so zurückgegeben, wie sie an Ihr Programm

oder Skript übergeben wurden. Das bedeutet, dass die zurückgegebenen Argumente auch spezielle, von Hollywood gehandhabte Argumente enthalten können, z.B. "-window" oder "-quiet".

Beachten Sie, dass der erste Tabelleneintrag immer den Namen des Programms enthält, dies muss jedoch nicht unbedingt einen qualifizierten Pfad beinhalten, sondern nur den Namen des Programms, wie er vom Aufrufer angegeben wurde.

`GetRawArguments()` kann nützlich sein, wenn Ihr Skript in der Lage sein soll, mehrere Dateiarumente zu verarbeiten. Mit `GetFileArgument()` können Sie nur die allererste Datei abrufen, die an Ihr Programm übergeben wird, und `GetCommandLine()` erwartet, dass Argumente auf eine bestimmte Weise formatiert sind, was es unmöglich macht, damit alle Dateiarumente abzurufen. Mit `GetRawArguments()` können Sie alle Dateiarumente abrufen, da Argumente niemals in irgendeiner Weise formatiert werden, sondern einfach ohne Änderung durch Hollywood an Ihr Skript weitergeleitet werden.

## EINGABEN

keine

## RÜCKGABEWERTE

`args`            alle Argumente in einer Tabelle; Der erste Eintrag in dieser Tabelle enthält den Namen des Programms (nicht unbedingt mit seinem Pfad).

## BEISPIEL

```
t = GetRawArguments()
DebugPrint("Program:", t[0])
For Local k = 1 To ListItems(t) - 1
    DebugPrint("Argument", k .. ":", t[k])
Next
```

Der obige Code ruft alle Argumente ab, die an das Skript übergeben werden, und gibt sie aus.

## 18.15 LoadPrefs

### BEZEICHNUNG

LoadPrefs – lädt Benutzereinstellungen (V6.1)

### ÜBERSICHT

LoadPrefs(prefs[, t])

### BESCHREIBUNG

Dieser Befehl lädt Benutzereinstellungen, die mit dem Befehl `SavePrefs()` gespeichert wurden, in die durch `prefs` angegebene Tabelle. Vor dem Aufrufen von `LoadPrefs()` sollte die Tabelle `prefs` mit den Standardeinstellungen initialisiert worden sein. `LoadPrefs()` überschreibt dann alle Tabellenelemente, für die benutzerdefinierte Benutzereinstellungen vorhanden sind, und behält die Standardwerte der Elemente bei, für die keine Benutzereinstellungen vorhanden sind.

Beachten Sie, dass dieser Befehl nur funktionieren wird, wenn Sie eine eindeutige Kennung für Ihre Anwendung mit der Präprozessor-Anweisung `@APPIDENTIFIER` angegeben haben. Siehe [Abschnitt 18.6 \[APPIDENTIFIER\]](#), [Seite 222](#), für Details.

Ab Hollywood 9.0 akzeptiert dieser Befehl ein neues optionales Tabellenargument, das verwendet werden kann, um weitere Optionen anzugeben.

Die folgenden Tags werden derzeit im optionalen Tabellenargument `t` unterstützt:

**Adapter:** Dieser Tabellen-Tag kann verwendet werden, um den Deserialisierer anzugeben, der zum Importieren der Voreinstellungen verwendet werden soll. Dies kann der Name eines externen Deserialisierungs-Plugins (z.B. `xml`) oder einer der folgenden integrierten Deserialisierer sein:

**Default:** Verwendet den Hollywood Standard-Deserialisierer. Dadurch werden Daten aus dem JSON-Format in eine Hollywood-Tabelle deserialisiert.

**Inbuilt:** Verwendet den in Hollywood integrierten alten Deserialisierer. Die Verwendung dieses Deserialisierers wird nicht mehr empfohlen, da die Daten in einem proprietären, nicht lesbaren Format vorliegen. Die Verwendung von JSON ist eine viel bessere Wahl.

Wenn der Tag **Adapter** nicht angegeben ist, wird standardmäßig der mit `SetDefaultAdapter()` eingestellte Adapter verwendet.

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Serializer-Plugins übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Siehe auch `SavePrefs()` und `DeletePrefs()`.

## EINGABEN

`prefs`      lädt die Benutzereinstellungen in eine Tabelle

`t`            optional: Tabelle mit weiteren Optionen (siehe oben) (V9.0)

## BEISPIEL

```
@APPIDENTIFIER "com.airsoftsoftwair.example"
prf = {lastfile$ = "Unnamed", lastxpos = 0, lastypos = 0}
LoadPrefs(prf)
```

Dies initialisiert die Tabelle `prf` auf Standardwerte und verwendet dann `LoadPrefs()`, um Benutzereinstellungen zu lesen. Zum Speichern in eine Tabelle verwenden Sie `SavePrefs()`. Siehe [Abschnitt 18.16 \[SavePrefs\]](#), [Seite 229](#), für Details.

## 18.16 SavePrefs

### BEZEICHNUNG

`SavePrefs` – speichert Benutzereinstellungen (V6.1)

### ÜBERSICHT

`SavePrefs(prefs[, t])`

## BESCHREIBUNG

Dieser Befehl speichert die in **prefs** angegebene Tabelle mit Benutzervoreinstellungen für Ihre Anwendung in einer externen Datei. Sie können diese Einstellungen dann beim nächsten Programmstart mit dem Befehl **LoadPrefs()** zurück in das Programm laden. Der aktuelle Speicherort, in dem die Voreinstellungsdatei gespeichert wird, ist plattformabhängig.

Beachten Sie, dass dieser Befehl nur funktionieren wird, wenn Sie eine eindeutige Kennung für Ihre Anwendung mit der Präprozessor-Anweisung **@APPIDENTIFIER** angegeben haben. Siehe [Abschnitt 18.6 \[APPIDENTIFIER\]](#), [Seite 222](#), für Details.

Ab Hollywood 9.0 akzeptiert dieser Befehl ein neues optionales Tabellenargument, das verwendet werden kann, um weitere Optionen anzugeben.

Die folgenden Tags werden derzeit im optionalen Tabellenargument unterstützt:

**Adapter:** Dieser Tabellen-Tag kann verwendet werden, um den Serialisierer anzugeben, der zum Exportieren der Voreinstellungen verwendet werden soll. Dies kann der Name eines externen Serialisierer-Plugins (z.B. **xml**) oder einer der folgenden integrierten Serialisierer sein:

**Default:** Verwendet Hollywoods Standard-Serialisierer. Dieser serialisiert die Voreinstellungen in das JSON-Format.

**Inbuilt:** Verwendet den in Hollywood integrierten alten Serialisierer. Dieser serialisiert die Tabelle in ein eigenes, proprietäres Format.

Wenn der Tag **Adapter** nicht angegeben ist, wird standardmäßig der mit **SetDefaultAdapter()** eingestellte Adapter verwendet.

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Serializer-Plugins übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Siehe auch **LoadPrefs()** und **DeletePrefs()**.

## EINGABEN

**prefs**      Tabelle mit den zu speichernden Benutzereinstellungen

**t**            optional: Tabelle mit weiteren Optionen (siehe oben) (V9.0)

## BEISPIEL

```
@APPIDENTIFIER "com.airsoftsoftwair.example"
prf = {lastfile$ = "test.txt", lastxpos = 100, lastypos = 200}
SavePrefs(prf)
```

Dadurch wird die Tabelle **prf** an einen plattformabhängigen Speicherort gesichert. Beim nächsten Programmstart können Sie die Einstellungen unter Verwendung des Befehls **LoadPrefs()** laden. Siehe [Abschnitt 18.15 \[LoadPrefs\]](#), [Seite 228](#), für Details.

## 19 Asynchrone Operationsbibliothek

### 19.1 AsyncDrawFrame

#### BEZEICHNUNG

AsyncDrawFrame – zeichnet das nächste Einzelbild eines asynchronen Objekts (V4.0)

#### ÜBERSICHT

```
finish = AsyncDrawFrame(id[, frame])
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um das nächste Einzelbild eines asynchronen Zeichnungsobjekts zu zeichnen, wenn es durch einen Übergangseffekt oder einem Verschiebungsbefehl aus den Objektbibliotheken erstellt wurde. `AsyncDrawFrame()` gibt `False` zurück, wenn es noch Einzelbilder in der Warteschlange hat. Daher sollten Sie in der Regel diesen Befehl solange aufrufen, bis er `True` zurückgibt. Somit haben Sie alle Einzelbilder gezeichnet, die in der Warteschlange des asynchronen Zeichnungsobjekts waren.

Wenn dieser Befehl `True` zurückgibt, löscht er automatisch das asynchrone Zeichnungsobjekt, so dass es nicht mehr gültig ist und somit nicht mehr verwendet werden kann. Wenn Sie eine asynchrone Zeichnungssequenz stoppen wollen, bevor alle Bilder gezeichnet wurden, können Sie den Befehl `CancelAsyncDraw()` verwenden. Wenn Sie hingegen ein asynchrone Zeichnungsobjekt stoppen wollen und es seinen letzten Code ausführen sollte, müssen Sie dem Befehl `FinishAsyncDraw()` verwenden.

Ab Hollywood 4.5 akzeptiert `AsyncDrawFrame()` das optionales Argument `frame`, womit Sie explizit angeben können, welches Bild Sie dargestellt haben möchten. Um die Anzahl der Einzelbilder eines asynchronen Zeichnungsobjekt herauszufinden, rufen Sie den Befehl `GetAttribute()` mit dem Argument `#ATTRNUMFRAMES` auf. Der Wert, den Sie mit diesem Befehl erhalten, ist die größte gültige Einzelbildnummer. Um das erste Bild zu zeichnen, müssen Sie 1 übergeben. Um das nächste Bild zu zeichnen, übergeben Sie den Wert 0, was auch die Standardeinstellung ist, wenn das Argument `frame` weggelassen wird.

Wenn Sie das zu zeichnende Einzelbild manuell festlegen, müssen Sie auch darauf achten, dass Ihr asynchrones Zeichnungsobjekt korrekt aus dem Speicher entfernt wird. Wenn Sie das optionale Argument nicht verwenden, wird das asynchrone Zeichnungsobjekt automatisch aus dem Speicher gelöscht, wenn `AsyncDrawFrame()` `True` liefert. Wenn Sie ein Einzelbild manuell angeben, wird das asynchrone Zeichnungsobjekt nie aus dem Speicher entfernt. Auch wenn Sie das letzte Einzelbild angeben, wird Hollywood das asynchrone Zeichnungsobjekt nicht aus dem Speicher löschen, weil es möglich wäre, dass Sie noch ein anderes Einzelbild manuell anzeigen möchten. Also, wenn Sie manuell das aktuellen Einzelbild auswählen, stellen Sie sicher, dass Sie den Befehl `FinishAsyncDraw()` aufrufen, wenn Sie mit dem asynchronen Zeichnungsobjekt fertig sind.

Bitte beachten Sie, dass derzeit einige Einschränkungen für diesen Befehl gelten:

1. Asynchrone Einzelbilder können nur in dem Hauptdisplay gezeichnet werden. Sie können diesen Befehl derzeit nicht fürs Zeichnen von Pinseln verwenden.

2. Während asynchrone Zeichnungsobjekte gezeichnet werden, können Sie nicht den Ebenenmodus wechseln. Die Befehle `EnableLayers()` oder `DisableLayers()` sind deaktiviert, während asynchrone Zeichnungsobjekte aktiv sind.
3. Sie können nicht das Hintergrundbild wechseln, während asynchrone Zeichnungsobjekte gezeichnet werden.
4. Zu Einzelbilder springen funktioniert nur mit asynchronen Zeichnungsobjekte, die einer Ebene zugeordnet sind.
5. Zu Einzelbilder springen arbeitet nur mit asynchronen Zeichnungsobjekte, welche vom Typ `#ADF_FX` sind.

## EINGABEN

<code>id</code>	Identifikator des asynchronen Zeichnungsobjektes
<code>frame</code>	optional: das gewünschte Einzelbild, welches gezeichnet werden soll; arbeitet nur mit asynchronen Zeichnungsobjekten, welche einer Ebenen zugewiesen sind. (V4.5)

## RÜCKGABEWERTE

<code>finish</code>	<code>False</code> , wenn es noch Einzelbilder in der Warteschlange hat oder <code>True</code> , wenn das asynchrone Zeichnungsobjekt fertig ist
---------------------	--

## BEISPIEL

```
obj = DisplayBrushFX(1, #CENTER, #CENTER, {Type = #WATER1, Async = True})
Repeat
    done = AsyncDrawFrame(obj)
    VWait
Until done = True
```

Der obige Code zeigt den Pinsel 1 mit dem Übergangseffekt `#WATER1` in der Mitte des Bildschirms. Wie Sie sehen können, wird der Effekt nicht durch `DisplayBrushFX()`, sondern durch `AsyncDrawFrame()` in der Schleife angezeigt, so dass Sie auch während dem Übergangseffekt ein paar andere Dinge tun könnten.

```
EnableLayers
DisplayBrush(1, #CENTER, #CENTER)
obj = ShowLayerFX(1, {Type = #WALLPAPERTOP, Async = True})
frames = GetAttribute(#ASYNCDRAW, obj, #ATTRNUMFRAMES)
For Local k = frames To 1 Step -1
    AsyncDrawFrame(obj, k)
    VWait
Next
For Local k = 1 To frames
    AsyncDrawFrame(obj, k)
    VWait
Next
FinishAsyncDraw(obj)
```

Der obige Code zeigt Pinsel Nummer 1 unter Verwendung des Übergangseffekts `#WALLPAPERTOP` an. Der Effekt wird zuerst rückwärts angezeigt und dann in normaler



Reihenfolge. Beachten Sie, dass wir das Zeichnungsobjekt manuell mit dem Befehl `FinishAsyncDraw()` freigeben müssen.

## 19.2 CancelAsyncDraw

### BEZEICHNUNG

CancelAsyncDraw – bricht ein asynchrones Zeichnungsobjekt ab (V4.0)

### ÜBERSICHT

CancelAsyncDraw(id)

### BESCHREIBUNG

Mit diesem Befehl kann ein asynchrones Zeichnungsobjekt vorläufig abgebrochen werden. Siehe [Abschnitt 19.1 \[AsyncDrawFrame\]](#), [Seite 231](#), für weitere Informationen über asynchrone Zeichnungsobjekte.

Wenn das asynchrone Zeichnungsobjekt, welches Sie stoppen möchten, einer Ebene zugeordnet ist, dann ruft `CancelAsyncDraw()` den Abschlußcode für Ihr Zeichnungsobjekt nicht auf. Zum Beispiel, wenn Sie eine Ebene mit einem asynchronen Effekt entfernen möchten (das heißt Sie haben das asynchrone Zeichnungsobjekt mit dem Befehl `RemoveLayerFX()` aufgerufen), dann wird die Ebene nicht entfernt werden, wenn Sie den Befehl `CancelAsyncDraw()` benutzen. Das gleiche gilt für `HideLayerFX()` (Ebene wird nicht ausgeblendet werden, wenn Sie das asynchrone Zeichnungsobjekt auf halbem Wege abbrechen) und `ShowLayerFX()` (Ebene wird nicht angezeigt werden, wenn Sie das asynchrone Zeichnungsobjekt abbrechen).

Wenn Sie dieses Verhalten nicht wünschen, verwenden Sie stattdessen `FinishAsyncDraw()`. `FinishAsyncDraw()` wird zuerst der Abschlusscode für das asynchrone Zeichnungsobjekt aufrufen und dann wird es aus dem Speicher entfernt. Dies bedeutet, dass wenn Sie `FinishAsyncDraw()` bei einem mit `RemoveLayerFX()` erstellten asynchronen Zeichnungsobjekt ausführen, wird die Ebene entfernt, auch wenn der Effekt noch nicht beendet ist. `FinishAsyncDraw()` wird zum letzten Einzelbild in der Warteschlange springen, den Abschlusscode aufrufen (das heißt die Ebene wird entfernt, ausgeblendet oder angezeigt) und das asynchrone Zeichnungsobjekt wird aus dem Speicher gelöscht.

Bitte beachten Sie, dass es überhaupt keinen Unterschied zwischen `CancelAsyncDraw()` und `FinishAsyncDraw()` gibt, wenn Ebenen deaktiviert sind. Bei deaktivierten Ebenen sollten Sie aber immer `CancelAsyncDraw()` verwenden.

### EINGABEN

id                    ID des Objekts, das Sie abbrechen wollen

## 19.3 CancelAsyncOperation

### BEZEICHNUNG

CancelAsyncOperation – bricht eine asynchrone Operation ab (V9.0)

### ÜBERSICHT

CancelAsyncOperation(id)

**BESCHREIBUNG**

Dieser Befehl bricht die in `id` angegebene asynchrone Operation ab. `id` muss auf einen Handler für eine asynchrone Operation gesetzt werden, die von Befehlen erstellt wurde, die asynchrone Operationen unterstützen, z.B. `CopyFile()` oder `DownloadFile()`.

**EINGABEN**

`id`                    Handler für asynchrone Operationen, erhalten von einem Befehl, der asynchrone Operationen unterstützt

**BEISPIEL**

Siehe [Abschnitt 19.4 \[ContinueAsyncOperation\]](#), Seite 234.

## 19.4 ContinueAsyncOperation

**BEZEICHNUNG**

`ContinueAsyncOperation` – setzt eine asynchrone Operation fort (V9.0)

**ÜBERSICHT**

`done, ... = ContinueAsyncOperation(id)`

**BESCHREIBUNG**

Dieser Befehl setzt die Verarbeitung des durch `id` angegebenen asynchronen Objekts fort. `id` muss auf einen asynchronen Operations-Handler gesetzt werden, der von Befehlen erstellt wurde, die asynchrone Operationen unterstützen, z.B. `CopyFile()` oder `DownloadFile()`.

Sobald die asynchrone Operation beendet ist, löscht `ContinueAsyncOperation()` den Handler für den asynchronen Vorgang und gibt `True` zurück. Wenn `ContinueAsyncOperation()` `False` zurückgibt, ist die Operation noch nicht abgeschlossen und Sie müssen `ContinueAsyncOperation()` erneut aufrufen, bis `True` zurückgegeben wird.

Wenn der Hollywood-Befehl, der den Handler für die asynchrone Operationen erstellt hat, nach Abschluss Werte an das Skript zurückgibt, leitet `ContinueAsyncOperation()` diese Werte an Ihr Skript weiter, sobald die asynchrone Operation beendet ist, d.h. wenn `done True` wird. In diesem Fall kann `ContinueAsyncOperation()` abhängig vom Befehl, der den Handler für asynchrone Operationen erstellt hat, zusätzliche Werte zurückgeben. Beispielsweise gibt `DownloadFile()` die heruntergeladenen Daten sowie deren Länge nach Abschluss an das Skript zurück.

Um eine asynchrone Operation abubrechen, können Sie den Befehl `CancelAsyncOperation()` verwenden. Siehe [Abschnitt 19.3 \[CancelAsyncOperation\]](#), Seite 233, für Details.

**EINGABEN**

`id`                    Handler für asynchrone Operationen, erhalten von einem Befehl, der asynchrone Operationen unterstützt

**RÜCKGABEWERTE**

`done`                `True`, wenn der Vorgang beendet ist, sonst `False`

... optional: nach Abschluss, d.h. wenn `done True` ist, alle zusätzlichen Rückgabewerte von dem Befehl, der den asynchronen Operations-Handler erstellt hat

**BEISPIEL**

```
handle = CopyFile("images", "sounds", {Async = True})
Repeat
    NextFrame(1)
```

```
Until ContinueAsyncOperation(handle) = True
```

Der obige Code zeigt, wie eine Animation angezeigt wird, während das Verzeichnis `images` in das Verzeichnis `sounds` kopiert wird.

## 19.5 FinishAsyncDraw

**BEZEICHNUNG**

`FinishAsyncDraw` – beendet ein asynchrones Zeichnungsobjekt (V4.5)

**ÜBERSICHT**

```
FinishAsyncDraw(id)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um ein asynchrones Zeichnungsobjekt zu beenden. Beim Aufruf von diesem Befehl wird Hollywood zum letzten Einzelbild in der Warteschlange des asynchronen Zeichnungsobjekts springen und es beenden. "Beenden" eines asynchronen Zeichnungsobjekts bedeutet, dass das letzte Einzelbild angezeigt und anschließend der Abschlusscode ausgeführt wird (zum Beispiel das Entfernen der Ebene, wenn das asynchrone Zeichnungsobjekt zu einem `RemoveLayerFX()` Aufruf gehört).

Es gibt einen Unterschied zwischen `FinishAsyncDraw()` und `CancelAsyncDraw()`. Siehe [Abschnitt 19.2 \[CancelAsyncDraw\]](#), [Seite 233](#), für Details.. Eine Ausnahme ist, wenn Ebenen aktiviert sind. In diesem Fall gibt es keinen Unterschied zwischen `CancelAsyncDraw()` und `FinishAsyncDraw()`. Bei deaktivierten Ebenen sollten Sie aber immer `CancelAsyncDraw()` verwenden.

**EINGABEN**

`id` ID des Objekts, das Sie beenden wollen



## 20 Dateisystembibliothek (DOS)

### 20.1 CanonizePath

#### BEZEICHNUNG

CanonizePath – konvertiert den Pfad in das kanonische Format (V9.0)

#### ÜBERSICHT

p\$ = CanonizePath(path\$)

#### BESCHREIBUNG

Mit diesem Befehl kann der in **path\$** angegebene Pfad in einen kanonischen Pfad umgewandelt werden. Das Kanonisieren eines Pfades umfasst die folgenden Operationen:

- Verknüpfungen wie ".." oder "." werden in den vollen Pfad aufgelöst
- Relative Pfade werden in vollständig Pfade konvertiert
- Auf Plattformen mit Dateisystemen, bei denen die Groß- und Kleinschreibung nicht berücksichtigt wird, wird die Schreibweise aller Pfadkomponenten an die Schreibweise angepasst, wenn sie im Dateisystem gespeichert wird
- Schrägstriche und Backslashes werden an die Konvention des Host-Betriebssystems angepasst
- Alle Zuweisungen im Pfad werden unter AmigaOS und kompatiblen Systemen aufgelöst

Beachten Sie, dass der an **CanonizePath()** übergebene Pfad nicht vorhanden sein muss. Wenn er nicht existiert, versucht **CanonizePath()**, so viele Komponenten im Pfad wie möglich aufzulösen. Beachten Sie jedoch, dass **CanonizePath()** keine Pfadüberprüfung durchführt. Wenn Sie einen Pfad übergeben, der aufgrund von syntaktischen Fehlern ungültig ist, ist das Ergebnis undefiniert.

Siehe auch **CanonizePath()**, **IsAbsolutePath()** und **MakeHostPath()**.

#### EINGABEN

path\$      Pfad, der konvertiert werden soll

#### RÜCKGABEWERTE

p\$      vollständiger Pfad im kanonischen Format des Hostsystems

#### BEISPIEL

```
Print(CanonizePath("../image.jpg"))
```

Der obige Code gibt den vollen Pfad der Datei **image.jpg** aus.

### 20.2 ChangeDirectory

#### BEZEICHNUNG

ChangeDirectory – wechselt das aktuelle Verzeichnis

#### ÜBERSICHT

ChangeDirectory(dir\$)

**BESCHREIBUNG**

Dieser Befehl wechselt das Verzeichnis auf das in `dir$` angegebene.

Ab Hollywood 10.0 akzeptiert dieser Befehl ein optionales Tabellenargument, das die folgenden Tags unterstützt:

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateisystemadapter angeben, diese Operation auszuführen. Dies muss auf eine Zeichenkette gesetzt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()` gesetzt wurde. Siehe [Abschnitt 7.9 \[Lader und Adapter\]](#), Seite 96, für Details. (V10.0)

**UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an den Dateisystemadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), Seite 100, für Details. (V10.0)

Siehe auch `GetCurrentDirectory()`, `GetProgramDirectory()`, `GetStartDirectory()`, `GetFileAttributes()` mit dem Tabellenfeld `Path` und `MakeDirectory()`.

**EINGABEN**

`dir$` Verzeichnis, das zum aktuellen gemacht werden soll

`t` optional: Tabellenargument mit weiteren Optionen (siehe oben) (V10.0)

**BEISPIEL**

```
ChangeDirectory("Data")
OpenFile(1, "Highscores.txt")
CloseFile(1)
```

Diese Zeilen ändern das aktuelle Verzeichnis auf "Data", öffnet und schließt dann die Datei "Highscores.txt" im Verzeichnis "Data".

## 20.3 CloseDirectory

**BEZEICHNUNG**

`CloseDirectory` – schließt ein offenes Verzeichnis (V4.0)

**ÜBERSICHT**

```
CloseDirectory(id)
```

**BESCHREIBUNG**

Dieser Befehl schließt ein Verzeichnis, welches zuvor mit dem Befehl `OpenDirectory()`, `@DIRECTORY` oder `MonitorDirectory()` geöffnet wurde. Sie sollten immer Verzeichnisse schließen, sobald Sie mit ihnen fertig sind. Dadurch wird sichergestellt, dass das Dateisystem nicht unnötig Verzeichnisse gesperrt hält.

**EINGABEN**

`id` ID des zu schließenden Verzeichnisses

**BEISPIEL**

Siehe [Abschnitt 20.47 \[OpenDirectory\]](#), Seite 289.

## 20.4 CloseFile

**BEZEICHNUNG**

CloseFile – schließt eine geöffnete Datei

**ÜBERSICHT**

CloseFile(id)

**BESCHREIBUNG**

Schließt die durch id identifizierte Datei, die mit [OpenFile\(\)](#) geöffnet wurde.

Siehe auch [@FILE](#) und [Exists\(\)](#).

**EINGABEN**

id            Nummer, mit der die Datei geöffnet wurde

**BEISPIEL**

Siehe [Abschnitt 20.48 \[OpenFile\]](#), Seite 290.

## 20.5 CompressFile

**BEZEICHNUNG**

CompressFile – komprimiert eine Datei (V4.0)

**ÜBERSICHT**

size = CompressFile(src\$, dst\$)

**BESCHREIBUNG**

Dieser Befehl komprimiert die in src\$ angegebene Datei und speichert die gepackten Daten in dst\$. Der Rückgabewert size gibt die Größe der komprimierten Datei an. Hollywood verwendet zlib für die Datenkompression.

Zum Entpacken von Dateien, welche mit CompressFile() komprimiert wurden, verwenden Sie den Befehl [DecompressFile\(\)](#).

**EINGABEN**

src\$            Datei, welche komprimiert werden soll

dst\$            Dateiname der gepackten (neuen) Datei

**RÜCKGABEWERTE**

size            Größe der gepackten Datei

**BEISPIEL**

```
CompressFile("image.bmp", "image.pak")
```

Der Code oben komprimiert die Datei "image.bmp" und speichert Sie als "image.pak" ab.

## 20.6 CopyFile

### BEZEICHNUNG

CopyFile – kopiert eine Datei oder ein Verzeichnis (V2.0)

### ÜBERSICHT

```
CopyFile(src$, dst$[, t])
```

### FRÜHERE SYNTAX

```
CopyFile(src$, dst$[, newname$, func, userdata, pattern$, matchdir])
```

### BESCHREIBUNG

Dieser Befehl kopiert die Datei oder das Verzeichnis in **src\$** in das in **dst\$** angegebene Verzeichnis. Beachten Sie, dass vorhandene Dateien standardmäßig ohne Nachfrage überschrieben werden. Sie können dieses Verhalten anpassen, indem Sie eine Callback-Funktion angeben (siehe unten). Beachten Sie auch, dass Sie in **dst\$** ein Verzeichnis und keine Datei angeben müssen.

Dieser Befehl ist mächtig. Es wird vollständig rekursiv alle Unterverzeichnisse erfassen und kopiert außerdem die Dateiattribute, Datumsstempel und Kommentare. Wenn das Zielverzeichnis nicht vorhanden ist, wird es für Sie erstellt werden (auch wenn es Verzeichnisse enthält, die noch nicht existieren). Alle Pfadangaben können lokal in das aktuelle Verzeichnis verweisen. Sie können somit auch Dateien in das aktuelle Verzeichnis kopieren, indem Sie "" als **dst\$** angeben.

**CopyFile()** unterstützt viele optionale Argumente. Vor Hollywood 9.0 mussten diese als optionale Parameter übergeben werden (siehe oben). Seit Hollywood 9.0 wird jedoch empfohlen, die neue Syntax zu verwenden, die ein einzelnes optionales Tabellenargument hat, mit dem ein oder mehrere optionale Argumente an **CopyFile()** übergeben werden können.

Die folgenden Tabellenfelder werden von diesem Befehl erkannt:

**newName:** Wenn Sie den Namen der Datei beim Kopieren ändern möchten, setzen Sie dieses Feld auf den gewünschten neuen Namen für die Datei. Das Setzen dieses Feldes ist natürlich nur sinnvoll, wenn Sie eine Datei in **src\$** angeben.

**Pattern:** Sie können in diesem Tabellenfeld ein Filtermuster übergeben. In diesem Fall kopiert **CopyFile()** nur die Dateien, die dem angegebenen Muster entsprechen. Wenn Sie beispielsweise **.jpg** in **Pattern** angeben, werden nur Dateien kopiert, die die Dateierweiterung **.jpg** verwenden. Natürlich macht die Verwendung eines Filtermusters nur dann Sinn, wenn Sie ein Verzeichnis in **src\$** übergeben. Beachten Sie, dass aus historischen Gründen das in **Pattern** angegebene Muster auch mit allen zu kopierenden Unterverzeichnissen abgeglichen wird, die kopiert werden sollen. Wenn Sie das nicht möchten, setzen Sie das Tabellen-Tag **MatchDir** auf **False** (siehe unten). Das in **Pattern** angegebene Muster muss den Musterregeln entsprechen, die in der Dokumentation des Befehls **MatchPattern()** beschrieben sind. Siehe [Abschnitt 20.42 \[MatchPattern\]](#), [Seite 280](#), für Details. (V5.0)

**MatchDir:**

Dieses Tabellenfeld gibt an, ob das in **Pattern** angegebene Filtermuster auch mit Unterverzeichnissen abgeglichen werden soll oder nicht. Wenn dies auf



**True** gesetzt ist, rekursiert `CopyFile()` nur in Unterverzeichnisse, die dem angegebenen Filtermuster entsprechen. Wenn es auf **False** gesetzt ist, rekursiert `CopyFile()` in alle Unterverzeichnisse. Aus Kompatibilitätsgründen ist `MatchDir` standardmäßig auf **True** gesetzt, aber meistens möchten Sie hier **False** übergeben, da es normalerweise keinen Sinn macht, ein Dateimuster mit einem Verzeichnisnamen abzugleichen. Es macht beispielsweise keinen Sinn, das obige Beispiel `*.jpg` auch mit Verzeichnissen abzugleichen. (V5.0)

**BufferSize:**

In diesem Tabellenfeld kann die Puffergröße eingestellt werden, die zum Kopieren von Dateien verwendet werden soll. Der hier übergebene Wert muss in Bytes angegeben werden. Der Standardwert ist 16384, also 16 Kilobyte. (V9.0)

**FailOnError:**

Standardmäßig schlägt `CopyFile()` fehl, wenn ein Fehler auftritt. Sie können dieses Verhalten ändern, indem Sie `FailOnError` auf **False** setzen. In diesem Fall schlägt `CopyFile()` bei einem Fehler nicht fehl, stattdessen wird Ihre Callback-Funktion, falls vorhanden, mit der Meldung `#COPYFILE_FAILED` benachrichtigt und Ihre Callback-Funktion muss `CopyFile()` mitteilen, wie es weitergeht (Wiederholen, Fortfahren, Abbrechen). Siehe unten, um zu erfahren, wie man eine Callback-Funktion für `CopyFile()` einrichtet. Die Voreinstellung für `FailOnError` ist **True**. (V9.0)

**Force:**

Wenn dieser Tag auf **True** gesetzt ist, werden schreib- oder löschgeschützte Dateien automatisch überschrieben, ohne vorher die Callback-Funktion abzufragen. Beachten Sie, dass `CopyFile()` fehlschlägt, wenn es keine Callback-Funktion gibt und `Force` auf **False** (Standard) gesetzt ist, wenn es eine Datei nicht überschreiben kann, da diese Datei schreib- oder löschgeschützt ist. Die Voreinstellung ist **False**. (V9.0)

**Async:**

Wenn dies auf **True** gesetzt ist, arbeitet `CopyFile()` im asynchronen Modus. Das bedeutet, dass der Befehl sofort zurückkehrt und falls Ihr Skript während des Vorgangs etwas anderes tun muss, Ihnen ein asynchroner Operationshandler übergibt. Sie können dann diesen asynchronen Operationshandler verwenden, um den Vorgang abzuschließen, indem Sie wiederholt `ContinueAsyncOperation()` aufrufen, bis **True** zurückgegeben wird. Dies ist sehr nützlich, z.B. für die Anzeige eines Fortschrittsbalken oder ähnlichem. Indem Sie `CopyFile()` in den asynchronen Modus versetzen, ist es Ihrem Skript leicht möglich, während der Verarbeitung des Vorgangs etwas anderes zu tun. Siehe [Abschnitt 19.4 \[ContinueAsyncOperation\]](#), [Seite 234](#), für Details. Voreingestellt ist **False**. (V9.0)

**Adapter:**

Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, welche die Quelldatei oder das Quellverzeichnis öffnen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Zeichenkette setzen, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()` gesetzt wurde. Siehe [Abschnitt 7.9 \[Lader und Adapter\]](#), [Seite 96](#), für Details. (V10.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an den Dateiadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

**DstAdapter:**

Mit diesem Tag können Sie einen oder mehrere Dateisystemadapter angeben, alle Operationen auf der Seite des Kopierziels auszuführen. Der hier angegebene Dateisystemadapter ist beispielsweise für das Erstellen von Verzeichnissen und das Setzen von Datei- und Verzeichnisattributen zuständig. Dies muss auf eine Zeichenkette festgelegt werden, die den Namen eines oder mehrerer Adapter enthält. Der Standardwert ist `default`. Siehe [Abschnitt 7.9 \[Lader und Adapter\]](#), [Seite 96](#), für Details. (V10.0)

**DstUserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Dateisystemadapter übergeben werden sollen, die in **DstAdapter** angegeben sind. Siehe oben für Details. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

**Callback:**

Dieses Tabellenfeld kann verwendet werden, um eine Callback-Funktion zu übergeben, die von `CopyFile()` bei verschiedenen Gelegenheiten aufgerufen wird, damit Sie beispielsweise einen Fortschrittsbalken aktualisieren können. Die Callback-Funktion wird auch aufgerufen, wenn bereits eine Zieldatei existiert oder sie schreib-/löschgeschützt ist. Die Callback-Funktion erhält ein Argument: Eine Tabelle, die weitere Informationen enthält.

Die folgenden Callback-Typen sind verfügbar:

**#COPYFILE\_OVERWRITE:**

`CopyFile()` führt diesen Callback aus, um zu fragen, ob eine Datei überschrieben werden kann. Ihre Callback-Funktion muss `True` zurückgeben, wenn die Datei überschrieben werden soll oder `False`, wenn sie übersprungen werden soll. Um den Kopiervorgang vollständig abubrechen, geben Sie `-1` zurück. Die folgenden Felder werden im Tabellenparameter gesetzt, der an Ihre Callback-Funktion übergeben wird:

**Action:** `#COPYFILE_OVERWRITE`

**Source:** Enthält den vollständigen Pfad der zu kopierenden Datei.

**Destination:**

Enthält den vollständigen Pfad der Datei, die bereits vorhanden ist.

**UserData:**

Enthält den Wert, den Sie im Tabellenfeld **UserData** übergeben haben (siehe unten).

**#COPYFILE\_UNPROTECT:**

Die Callback-Funktion vom Typ **#COPYFILE\_UNPROTECT** wird aufgerufen, wenn die zu überschreibende Datei schreib- oder löschgeschützt ist. Diese Callback-Funktion muss **True** zurückgeben, wenn der Schutz der Datei aufgehoben werden kann, oder **False**, wenn sie übersprungen werden soll. Wenn Sie **-1** zurückgeben, wird der Kopiervorgang vollständig abgebrochen.

Ab Hollywood 9.0 kann Ihre Callback-Funktion auch **-2** zurückgeben, um anzugeben, dass das Kopieren trotzdem versucht werden soll, obwohl die Datei schreib- oder löschgeschützt ist. Dies führt jedoch in der Regel zu einem Fehler, da schreib- oder löschgeschützte Dateien nicht überschrieben werden können, ohne den Schutz zuvor aufzuheben. Wenn Sie jedoch **-2** zurückgeben, können Sie den folgenden Fehler in Ihrem **#COPYFILE\_FAILED**-Callback abfangen, wenn Sie **FailOnError** auf **False** gesetzt haben (siehe oben).

Die folgenden Felder werden im Tabellenparameter gesetzt, der an Ihre Callback-Funktion übergeben wird:

**Action:** **#COPYFILE\_UNPROTECT**

**Destination:**

Enthält die schreib- oder löschgeschützte Zieldatei, deren Schutz aufgehoben werden soll.

**UserData:**

Enthält den Wert, den Sie im Tabellenfeld **UserData** übergeben haben (siehe unten).

**#COPYFILE\_STATUS:**

Diese Callback-Funktion wird von Zeit zu Zeit ausgeführt, damit Sie eine Statusleiste oder ähnliches aktualisieren können. Die Callback-Funktion vom Typ **#COPYFILE\_STATUS** sollte normalerweise **False** zurückgeben. Wenn **True** zurückgegeben wird, wird der Kopiervorgang abgebrochen. Die folgenden Felder werden im Tabellenparameter gesetzt, der an Ihre Callback-Funktion übergeben wird:

**Action:** **#COPYFILE\_STATUS**

**Source:** Enthält den vollständigen Pfad der Datei, die gerade kopiert wird (Quelle).

**Destination:**

Enthält den vollständigen Pfad der Datei, die gerade geschrieben wird (Ziel).

**Copied:** Enthält die Anzahl der Bytes, die bereits kopiert wurden.

**Filesize:**  
Enthält die Dateigröße der Quelldatei.

**UserData:**  
Enthält den Wert, den Sie im Tabellenfeld **UserData** übergeben haben (siehe unten).

#### **#COPYFILE\_FAILED:**

Dieser Callback kann nur aufgerufen werden, wenn der Tag **FailOnError** auf **False** gesetzt wurde (siehe oben). In diesem Fall wird die Callback-Funktion vom Typ **#COPYFILE\_FAILED** immer dann aufgerufen, wenn ein Kopiervorgang fehlgeschlagen ist. Es muss **True** zurückgegeben werden, um den Kopiervorgang abubrechen, **False**, um fortzufahren, obwohl ein Fehler aufgetreten ist, oder **-1**, um den gerade fehlgeschlagenen Kopiervorgang erneut zu versuchen. Die folgenden Felder werden im Tabellenparameter gesetzt, der an Ihre Callback-Funktion übergeben wird:

**Action:** **#COPYFILE\_FAILED**

**Source:** Enthält den vollständigen Pfad der Datei, die gerade kopiert wird (Quelle).

**Destination:**  
Enthält den vollständigen Pfad der Datei, die gerade geschrieben wird (Ziel).

**UserData:**  
Enthält den Wert, den Sie im Tabellenfeld **UserData** übergeben haben (siehe unten).

(V9.0)

#### **UserData:**

Dieses Feld kann verwendet werden, um Ihrer Callback-Funktion einen beliebigen Wert zu übergeben. Der hier angegebene Wert wird bei jedem Aufruf an Ihre Callback-Funktion übergeben. Dies ist nützlich, wenn Sie vermeiden möchten, mit globalen Variablen zu arbeiten. Mit dem Tag **UserData** können Sie ganz einfach Daten an Ihre Callback-Funktion übergeben. Sie können in **UserData** einen beliebigen Wert beliebigen Typs angeben. Als Benutzerdaten können Zahlen, Zeichenketten, Tabellen und sogar Funktionen übergeben werden. Ihre Callback-Funktion erhält diese Daten im Feld **UserData** in der ihm übergebenen Tabelle. (V5.1)

Siehe auch **DeleteFile()**, **MoveFile()**, **Rename()** und **Exists()**.

#### **EINGABEN**

**src\$** die zu kopierende Quelldatei oder Verzeichnis  
**dst\$** Zielverzeichnis

t optional: Tabelle mit zusätzlichen Optionen (siehe oben) (V9.0)

### BEISPIEL

```
CopyFile("image.png", "TestDir")
```

Kopiert die Datei "image.png" aus dem aktuellen Verzeichnis nach "TestDir".

```
CopyFile("Images", "Images_Bak")
```

Erstellt eine Sicherung des Verzeichnis "Images" im "Images\_Bak" Verzeichnis (das neue Verzeichnis wird durch diesen Befehl automatisch erstellt). Alle Dateien inklusive Unterverzeichnisse werden an den neuen Speicherort kopiert werden.

```
CopyFile("Hollywood_Sources/WarpOS", "HW_BAK", {Pattern = "*.c;*.h",  
MatchDir = False})
```

Kopiert alle Quellcode und Header-Dateien von Hollywood\_Sources/WarpOS nach HW\_BAK.

```
Function p_CopyCallback(msg)
```

```
Switch msg.action
```

```
Case #COPYFILE_STATUS:
```

```
    DebugPrint("Now copying", FilePart(msg.source), "to",  
                PathPart(msg.destination))
```

```
Case #COPYFILE_OVERWRITE:
```

```
    Return(SystemRequest("Hollywood", FilePart(msg.destination) ..  
        " does already exist!\nDo you want me to overwrite it?",  
        "Yes|No"))
```

```
Case #COPYFILE_UNPROTECT:
```

```
    Return(SystemRequest("Hollywood", FilePart(msg.destination) ..  
        " is write/delete protected!\nDo you want me to unprotect it?",  
        "Yes|No"))
```

```
EndSwitch
```

```
Return(False)
```

```
EndFunction
```

```
CopyFile("Images", "Copy_of_Images", {Callback = p_CopyCallback})
```

Demonstriert die Verwendung einer Callback-Funktion.

## 20.7 CountDirectoryEntries

### BEZEICHNUNG

CountDirectoryEntries – zählt die Anzahl Einträge im Verzeichnis (V8.0)

### ÜBERSICHT

```
n, ... = CountDirectoryEntries(id[, what, recursive])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um alle Einträge in dem durch id angegebenen Verzeichnis zu zählen. Dieses Verzeichnis muss zuvor mit **OpenDirectory()** oder **@DIRECTORY** geöffnet worden sein.

Mit dem optionalen Argument **what** kann angegeben werden, welche Art von Einträgen gezählt werden soll. Folgende Typen werden derzeit unterstützt:

**#COUNTFILES:**

Zählt alle Dateien im Verzeichnis. Dies ist die Standardeinstellung.

**#COUNTDIRECTORIES:**

Zählt alle Verzeichnisse im Verzeichnis.

**#COUNTBOTH:**

Zählt sowohl Dateien als auch Verzeichnisse.

**#COUNTSEPARATE:**

In diesem Modus werden Dateien und Verzeichnisse separat gezählt. Das bedeutet, dass zwei Werte zurückgegeben werden: Der erste Rückgabewert enthält die Anzahl der gezählten Dateien, der zweite die Anzahl der gezählten Verzeichnisse. (V9.0)

Ab Hollywood 9.0 gibt es ein neues optionales Argument namens **rekursive**. Wenn dieses auf **True** gesetzt ist, rekursiert **CountDirectoryEntries()** in alle Unterverzeichnisse und schließt diese auch in die Zählung mit ein.

Beachten Sie, dass **CountDirectoryEntries()** alle Einträge im Verzeichnis durchläuft, so dass es nicht während einer Iteration mit **NextDirectoryEntry()** verwendet werden darf. Dadurch werden alle vorhandenen Verzeichnis-Iterationen automatisch zurückgesetzt (an den Anfang gesetzt).

Siehe auch **DirectoryItems()**.

## EINGABEN

<b>id</b>	Identifikator des Verzeichnisses, dessen Einträge gezählt werden sollen
<b>what</b>	optional: was gezählt werden soll (siehe oben) (Standardeinstellung ist <b>#COUNTFILES</b> )
<b>recursive</b>	optional: ob das Zählen auch in Unterverzeichnisse rekursiv erfolgen soll (Voreingestellt ist <b>False</b> ) (V9.0)

## RÜCKGABEWERTE

<b>n</b>	Anzahl der Einträge des gewünschten Typs im Verzeichnis
<b>...</b>	optional: zusätzliche Rückgabewerte abhängig vom aktuellen Zählmodus (siehe oben)

## BEISPIEL

```
OpenDirectory(1, "data")
NPrint(CountDirectoryEntries(1))
```

Der obige Code gibt die Anzahl der Dateien im Verzeichnis **data** aus.

## 20.8 CRC32

### BEZEICHNUNG

CRC32 – berechnet die 32-Bit-Prüfsumme einer Datei (V2.0)

### ÜBERSICHT

```
chk = CRC32(f$)
```

### BESCHREIBUNG

Dieser Befehl berechnet die 32-bit zyklische Redundanzprüfsumme (CRC32) für eine bestimmte Datei. Mit dieser Prüfsumme können Sie Ihre Dateien identifizieren.

Wenn Sie die CRC32 einer Zeichenfolge berechnen, verwenden Sie stattdessen den Befehl `CRC32Str()`.

Siehe auch `MD5()` und `MD5Str()`.

### EINGABEN

f\$            Quelldatei

### RÜCKGABEWERTE

chk            32-bit CRC32 von f\$

## 20.9 DecompressFile

### BEZEICHNUNG

DecompressFile – dekomprimiert eine Datei (V4.0)

### ÜBERSICHT

```
size = DecompressFile(src$, dst$)
```

### BESCHREIBUNG

Dieser Befehl dekomprimiert die Datei `src$` und speichert die entpackten Daten in `dst$`. Die Datei muss vorher mit dem Befehl `CompressFile()` gepackt worden sein. Der Rückgabewert gibt die Größe der unkomprimierten Datei an.

### EINGABEN

src\$            Datei, welche dekomprimiert werden soll

dst\$            Dateiname der entpackten (neuen) Datei

### RÜCKGABEWERTE

size            Größe der dekomprimierten Datei

### BEISPIEL

```
DecompressFile("image.pak", "image.bmp")
```

Der Code oben dekomprimiert die Datei "image.pak" und speichert Sie als "image.bmp" ab.

## 20.10 DefineVirtualFile

### BEZEICHNUNG

DefineVirtualFile – definiert eine virtuelle Datei in einer echten Datei (V4.0)

### ÜBERSICHT

```
virtfile$ = DefineVirtualFile(file$, offset, size, name$)
```

### BESCHREIBUNG

Mit diesem Befehl können Sie eine virtuelle Datei in einer anderen Datei definieren, die in verschiedenen Situationen hilfreich sein kann. Sie arbeiten an einem Spiel und Sie wollen in einer großen Ressource-Datei alle Daten des Spiels speichern. Nun müssen Sie einige Daten aus dieser großen Ressource-Datei laden und das ist der Punkt, wenn DefineVirtualFile() ins Spiel kommt.

Im Argument `file$` übergeben Sie den Namen der Datei, die die Quelle der virtuellen Datei sein soll. In den Argumenten `offset` und `size` definieren Sie den Speicherort der virtuellen Datei in der Datei `file$`. Die virtuelle Datei beginnt innerhalb der Datei `file$` vom Dateianfang mit einem Versatz `offset` und endet bei `Dateianfang+offset+size`. Im Argument `name$` übergeben Sie schließlich den Dateinamen für die virtuelle Datei. Das einzige, was hier wichtig ist, ist die Dateierweiterung. Der Name spielt keine Rolle, aber die Dateierweiterung muss die richtige sein, weil nicht alle Dateien leicht durch ihren Header identifiziert werden können.

DefineVirtualFile() gibt eine Zeichenfolge zurück, die die virtuelle Datei beschreibt. Sie können diese Zeichenfolge an alle Hollywood-Befehle übergeben, die einen Dateinamen akzeptieren. Natürlich werden nur Lesezugriffe bei virtuellen Dateien unterstützt. Der Versuch zu schreiben, wird bei virtuellen Dateien nicht funktionieren.

Siehe auch [DefineVirtualFileFromString\(\)](#), [GetTempFileName\(\)](#) und [UndefineVirtualStringFile\(\)](#). ■

### EINGABEN

<code>file\$</code>	Quellendatei, in der eine virtuelle Datei definiert wird
<code>offset</code>	Startposition der virtuellen Datei innerhalb der Quellendatei
<code>size</code>	Größe in Bytes der virtuellen Datei
<code>name\$</code>	Name und Erweiterung der virtuellen Datei (siehe oben)

### RÜCKGABEWERTE

<code>virtfile\$</code>	Zeichenkette, die die virtuelle Datei beschreibt
-------------------------	--

### BEISPIEL

```
vf$ = DefineVirtualFile("hugeresource.dat", 100000, 32768, "image.png")
LoadBrush(1, vf$, {LoadAlpha = True})
```

Der obige Code definiert eine virtuelle Datei in "hugeresource.dat". Die virtuelle Datei hat eine Größe von 32768 Bytes und beginnt bei der Position 100000 der Datei "hugeresource.dat". Die virtuelle Datei ist ein PNG-Bild. Nachdem die virtuelle Datei beschrieben wurde, wird das Bild mit einem einfachen Aufruf von LoadBrush() geladen.



## 20.11 DefineVirtualFileFromString

### BEZEICHNUNG

DefineVirtualFileFromString – definiert eine virtuelle Datei mit einer Zeichenkette (V5.0)

### ÜBERSICHT

```
virtfile$ = DefineVirtualFileFromString(data$, name$[, writable])
```

### BESCHREIBUNG

Mit diesem Befehl können Sie eine virtuelle Datei aus einer Zeichenkettenquelle definieren. Eine virtuelle Datei ist eine Datei, die nur im Arbeitsspeicher vorhanden ist, aber Sie können alle Befehle von Hollywood anwenden und somit so tun, als ob die Datei wirklich auf einem physikalischen Laufwerk existiert.

**DefineVirtualFileFromString()** hat zwei obligatorische Argumente: Im ersten Argument **data\$** stellen Sie die Daten zur Verfügung, die ihre virtuelle Datei darstellen soll. Im zweiten Argument **name\$** übergeben Sie den Namen der virtuellen Datei. Das einzige, was hier wichtig ist, ist die Dateierweiterung. Der Name spielt keine Rolle, aber die Dateierweiterung sollte übergeben werden, da nicht alle Dateien leicht durch einen Blick auf ihre Header-Bytes identifiziert werden können.

Ab Hollywood 6.1 unterstützt **DefineVirtualFileFromString()** auch die Erstellung von virtuellen Dateien, in die geschrieben werden kann. Wenn die virtuelle Datei schreibbar sein soll, müssen Sie das Argument **writable** auf **True** setzen. In diesem Fall wird **DefineVirtualFileFromString()** eine beschreibbare virtuelle Datei für Sie erstellen. Die beschreibbare Datei wird mit dem im Argument **data\$** definierten Parameter initialisiert werden. Wenn Sie eine leere Zeichenkette im Argument **data\$** übergeben, wird eine neue, leere, beschreibbare virtuelle Datei erstellt.

**DefineVirtualFileFromString()** gibt eine Zeichenfolge zurück, die die virtuelle Datei beschreibt. Sie können diese Zeichenfolge allen Hollywood-Befehlen übergeben, die einen Dateinamen akzeptieren.

Bitte beachten Sie, dass der Inhalt der Datei sich nicht nur auf Text beschränkt. Sie können auch Binärdaten innerhalb des Argumentes **data\$** übergeben, da Zeichenketten von Hollywood auch spezielle Steuer- und das Nullzeichen enthalten können. So ist es durchaus möglich, dass Sie mit diesem Befehl virtuelle Dateien mit Binärdaten erstellen können.

Wenn Sie fertig mit der virtuellen Datei sind, sollten Sie die virtuelle Datei wieder löschen, indem Sie den Befehl **UndefineVirtualStringFile()** benutzen. Dies zu tun ist wichtig, weil Sie so jeden Arbeitsspeicher, welchen die virtuelle Datei belegt, wieder freigeben.

Siehe auch **DefineVirtualFile()**, **GetTempFileName()** und **UndefineVirtualStringFile()**.

### EINGABEN

<b>data\$</b>	Quellenzeichenkette, welche die Daten der virtuellen Datei hat
<b>name\$</b>	Name und Erweiterung der virtuellen Datei (siehe oben)
<b>writable</b>	optional: <b>True</b> , wenn Sie in die virtuelle Datei schreiben wollen, andernfalls <b>False</b> (voreingestellt ist <b>False</b> ) (V6.1)

### RÜCKGABEWERTE

<b>virtfile\$</b>	Zeichenkette, die die virtuelle Datei beschreibt
-------------------	--

**BEISPIEL**

```
vf$ = DefineVirtualFileFromString("This is a virtual file test.",
                                "test.txt")
```

```
OpenFile(1, vf$)
While Not Eof(1) Do Print(Chr(ReadChr(1)))
CloseFile(1)
UndefineVirtualStringFile(vf$)
```

Der obige Code erstellt eine virtuelle Textdatei und liest dann mit der Hollywood-DOS-Bibliothek aus dieser virtuellen Datei.

```
data$ = DownloadFile("http://www.airsoftsoftwair.de/images/" ..
                    "products/hollywood/47_shot1.jpg")
vf$ = DefineVirtualFileFromString(data$, "image.jpg")
LoadBrush(1, vf$)
DisplayBrush(1, 0, 0)
UndefineVirtualStringFile(vf$)
data$ = Nil
```

Der obige Code lädt ein JPEG-Bild in eine Zeichenkette und lädt dann das Bild direkt in Hollywood, ohne sie zuerst in einer externen Datei speichern zu müssen.

```
vf$ = DefineVirtualFileFromString("", "test.txt", True)
OpenFile(1, vf$, #MODE_WRITE)
WriteLine(1, "A virtual file test!")
CloseFile(1)
CopyFile(vf$, GetSystemInfo().UserHome)
UndefineVirtualStringFile(vf$)
```

Der obige Code schreibt eine Zeichenkette in eine virtuelle Datei und kopiert dann diese virtuelle Datei in das Home-Verzeichnis des Benutzers.

## 20.12 DeleteFile

**BEZEICHNUNG**

DeleteFile – löscht eine Datei oder ein Verzeichnis

**ÜBERSICHT**

```
DeleteFile(file$[, t])
```

**FRÜHERE SYNTAX**

```
DeleteFile(file$[, callback, userdata, pattern$, matchdir])
```

**BESCHREIBUNG**

Löscht die in `file$` angegebene Datei oder das Verzeichnis. Bitte beachten Sie, dass dieser Befehl standardmäßig ganze Verzeichnisse rekursiv löscht. Es wird nicht geprüft, ob das angegebene Verzeichnis leer ist oder nicht! Wenn Sie ein Verzeichnis angeben, wird es mit allen Unterverzeichnissen und allen darin enthaltenen Dateien gelöscht, es sei denn, dies wird ausdrücklich von Ihnen verboten. Seien Sie also sehr vorsichtig mit diesem Befehl!

`DeleteFile()` unterstützt mehrere optionale Argumente. Vor Hollywood 9.0 mussten diese als optionale Parameter übergeben werden (siehe oben). Seit Hollywood 9.0 wird jedoch empfohlen, die neue Syntax zu verwenden, die ein einzelnes optionales Tabellenargument hat, mit dem ein oder mehrere optionale Argumente an `DeleteFile()` übergeben werden können.

Die folgenden Tabellenfelder werden von diesem Befehl erkannt:

**Recursive:**

Standardmäßig rekursiert `DeleteFile()` in alle Unterverzeichnisse und löscht sie, wenn `file$` ein Verzeichnis angibt. Wenn Sie dies nicht möchten, setzen Sie diesen Tag auf `False`. (V9.0)

**Force:**

Wenn dieser Tag auf `True` gesetzt ist, werden schreib- oder löschgeschützte Dateien automatisch gelöscht, ohne vorher die Callback-Funktion abzufragen. Beachten Sie, dass `DeleteFile()` alle schreib- oder löschgeschützten Dateien überspringt, wenn es keine Callback-Funktion gibt und `Force` auf `False` (Voreinstellung) gesetzt ist. Die Voreinstellung ist `False`. (V9.0)

**MustExist:**

Standardmäßig schlägt `DeleteFile()` stillschweigend fehl, wenn Sie eine Datei oder ein Verzeichnis angeben, welche/s nicht in `file$` existiert. In diesem Fall wird kein Fehler generiert. Wenn `DeleteFile()` stattdessen einen Fehler anzeigen soll, setzen Sie diesen Tag auf `True`. (V9.0)

**Pattern:**

In diesem Tabellenfeld können Sie ein Filtermuster übergeben. In diesem Fall löscht `DeleteFile()` nur die Dateien, die dem angegebenen Muster entsprechen. Wenn Sie beispielsweise `*.jpg` in `Pattern` übergeben, werden nur Dateien gelöscht, die die Dateierweiterung `.jpg` verwenden. Natürlich macht die Verwendung eines Filtermusters nur dann Sinn, wenn Sie in `file$` ein Verzeichnis angeben. Beachten Sie, dass aus historischen Gründen das in `Pattern` angegebene Muster auch mit allen zu löschenden Unterverzeichnissen abgeglichen wird. Wenn Sie das nicht möchten, setzen Sie das Tabellen-Tag `MatchDir` auf `False` (siehe unten). Das in `Pattern` angegebene Muster muss den Musterregeln entsprechen, die in der Dokumentation des Befehls `MatchPattern()` beschrieben sind. Siehe [Abschnitt 20.42 \[MatchPattern\]](#), Seite 280, für Details. (V5.0)

**MatchDir:**

Dieses Tabellenfeld gibt an, ob das in `Pattern` angegebene Filtermuster auch mit Unterverzeichnissen abgeglichen werden soll oder nicht. Wenn dies auf `True` gesetzt ist, rekursiert `DeleteFile()` nur in Unterverzeichnisse, die dem angegebenen Filtermuster entsprechen. Wenn es auf `False` gesetzt ist, rekursiert `DeleteFile()` in alle Unterverzeichnisse. Aus Kompatibilitätsgründen ist `MatchDir` standardmäßig auf `True` voreingestellt, aber meistens möchten Sie hier `False` übergeben, da es normalerweise keinen Sinn macht, ein Dateimuster mit einem Verzeichnisnamen abzugleichen. Es macht beispielsweise keinen Sinn, das obige Beispiel `*.jpg` auch mit Verzeichnissen abzugleichen. (V5.0)

**FailOnError:**

Standardmäßig schlägt `DeleteFile()` fehl, wenn eine Datei oder ein Verzeichnis nicht gelöscht werden kann. Sie können dieses Verhalten ändern, indem Sie `FailOnError` auf `False` setzen. In diesem Fall schlägt `DeleteFile()` nicht fehl, wenn eine Datei oder ein Verzeichnis nicht gelöscht werden kann. Stattdessen wird Ihre Callback-Funktion, falls vorhanden, mit der Meldung `#DELETEFILE_FAILED` benachrichtigt und Ihre Callback-Funktion muss `DeleteFile()` mitteilen, wie es weitergehen soll (Wiederholen, Fortsetzen, Abbrechen). Siehe unten, um zu erfahren, wie Sie eine Callback-Funktion für `DeleteFile()` einrichten. Beachten Sie, dass `FailOnError` nicht verwendet wird, wenn `file$` nur eine einzelne Datei ist. Es wird nur verwendet, wenn ganze komplette Verzeichnisse oder mehrere Dateien mit Mustern gelöscht werden. `FailOnError` ist voreingestellt auf `True`. (V9.0)

**Async:**

Wenn dies auf `True` gesetzt ist, arbeitet `DeleteFile()` im asynchronen Modus. Das bedeutet, dass der Befehl sofort zurückkehrt und Ihnen ein asynchroner Operationshandler übergibt. Sie können dann diesen asynchronen Operationshandler verwenden, um den Vorgang abzuschließen, indem Sie wiederholt `ContinueAsyncOperation()` aufrufen, bis `True` zurückgegeben wird. Dies ist sehr nützlich, z.B. für die Anzeige eines Fortschrittsbalken oder ähnlichem. Indem Sie `DeleteFile()` in den asynchronen Modus versetzen, ist es Ihrem Skript leicht möglich, während der Verarbeitung des Vorgangs etwas anderes zu tun. Siehe [Abschnitt 19.4 \[ContinueAsyncOperation\]](#), [Seite 234](#), für Details. Voreingestellt ist `False`. (V9.0)

**Adapter:**

Mit diesem Tag können Sie einen oder mehrere Dateisystemadapter angeben, die die Operation ausführen. Dies muss auf eine Zeichenkette gesetzt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()` gesetzt wurde. Siehe [Abschnitt 7.9 \[Lader und Adapter\]](#), [Seite 96](#), für Details. (V10.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an den Dateisystemadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

**Callback:**

Zur genaueren Steuerung des Löschvorgangs können Sie eine Callback-Funktion angeben, die bei verschiedenen Gelegenheiten aufgerufen wird. `DeleteFile()` ruft sie beispielsweise von Zeit zu Zeit auf, damit Sie einen Fortschrittsbalken aktualisieren können. Sie wird auch aufgerufen, wenn eine Datei löschgeschützt ist, um Sie zu fragen, wie es weitergehen soll. Wenn keine Callback-Funktion vorhanden ist, überspringt `DeleteFile()` löschgeschützte Dateien stillschweigend. Die Callback-Funktion erhält ein Argument, eine Tabelle, die weitere Informationen enthält.

Die folgenden Callback-Typen sind verfügbar:

**#DELETEFILE\_UNPROTECT:**

Die Callback-Funktion vom Typ **#DELETEFILE\_UNPROTECT** wird aufgerufen, wenn eine zu löschende Datei löschgeschützt ist. Diese Callback-Funktion muss **True** zurückgeben, wenn der Schutz der Datei aufgehoben werden kann, oder **False**, wenn der Schutz nicht aufgehoben werden soll. Wenn Sie -1 zurückgeben, wird der Löschvorgang vollständig abgebrochen.

**Action:** **#DELETEFILE\_UNPROTECT**

**File:** Enthält die löschgeschützte Datei, deren Schutz aufgehoben werden soll (vollständiger Pfad).

**UserData:**  
Enthält den Wert, den Sie im Tabellenfeld **UserData** übergeben haben (siehe unten).

(V2.0)

**#DELETEFILE\_STATUS:**

Dieser Callback wird immer dann ausgeführt, wenn eine Datei gelöscht wird. Dies ist beispielsweise nützlich, um einen Fortschrittsbalken zu aktualisieren. Die Callback-Funktion vom Typ **#DELETEFILE\_STATUS** sollte normalerweise **False** zurückgeben. Wenn sie **True** zurückgibt, wird der Löschvorgang abgebrochen.

**Action:** **#DELETEFILE\_STATUS**

**File:** Enthält den vollständigen Pfad der Datei, die als nächstes gelöscht werden soll

**UserData:**  
Enthält den Wert, den Sie im Tabellenfeld **UserData** übergeben haben (siehe unten).

(V2.0)

**#DELETEFILE\_FAILED:**

Dieser Callback kann nur aufgerufen werden, wenn der Tag **FailOnError** auf **False** gesetzt wurde (siehe oben). In diesem Fall wird die Callback-Funktion vom Typ **#DELETEFILE\_FAILED** immer dann aufgerufen, wenn ein Löschvorgang fehlgeschlagen ist. Es muss **True** zurückgegeben werden, um den Löschvorgang abubrechen, **False**, um fortzufahren, obwohl ein Fehler aufgetreten ist, oder -1, um den gerade fehlgeschlagenen Löschvorgang erneut zu versuchen. Die folgenden Felder werden im Tabellenparameter gesetzt, der an Ihre Callback-Funktion übergeben wird:

**Action:** **#DELETEFILE\_FAILED**

**File:** Enthält den vollständigen Pfad der Datei, die nicht gelöscht werden konnte.

**UserData:**

Enthält den Wert, den Sie im Tabellenfeld **UserData** übergeben haben (siehe unten).

(V9.0)

**UserData:**

Dieses Feld kann verwendet werden, um Ihrer Callback-Funktion einen beliebigen Wert zu übergeben. Der hier angegebene Wert wird bei jedem Aufruf an Ihre Callback-Funktion übergeben. Dies ist nützlich, wenn Sie die Arbeit mit globalen Variablen vermeiden wollen. Mit dem Tag **UserData** können Sie ganz einfach Daten an Ihre Callback-Funktion übergeben. Sie können in **UserData** einen Wert beliebigen Typs angeben. Als Benutzerdaten können Zahlen, Zeichenketten, Tabellen und sogar Funktionen übergeben werden. Ihre Callback-Funktion erhält diese Daten im Feld **UserData** in der ihm übergebenen Tabelle. (V3.1)

Siehe auch **CopyFile()**, **MoveFile()**, **Rename()** und **Exists()**.

**EINGABEN**

**file\$**      Dateiname oder Verzeichnis, welches gelöscht werden soll  
**t**            optional: Tabelle mit weiteren Optionen (siehe oben) (V9.0)

**BEISPIEL**

```
DeleteFile("FooBar")
```

Löscht die Datei (oder Verzeichnis) "FooBar" im aktuellen Verzeichnis.

```
Function p_DeleteCallback(msg)
```

```
Switch msg.action
```

```
Case #DELETEFILE_STATUS:
```

```
  DebugPrint("Now deleting", FilePart(msg.file))
```

```
Case #DELETEFILE_UNPROTECT:
```

```
  Return(SystemRequest("Hollywood", FilePart(msg.file) ..
```

```
    " is delete protected!\nDo you want me to unprotect it?",
```

```
    "Yes|No"))
```

```
EndSwitch
```

```
Return(False)
```

```
EndFunction
```

```
DeleteFile("TestDir", {Callback = p_DeleteCallback})
```

Demonstriert die Verwendung einer Callback-Funktion. Es löscht das Verzeichnis "TestDir" aus dem aktuellen Verzeichnis und gibt Informationen über die Datei aus, die gerade gelöscht wird.

## 20.13 DIRECTORY

**BEZEICHNUNG**

**DIRECTORY** – bindet ein gesamtes Verzeichnis im Applet oder Programm ein (V8.0)

## ÜBERSICHT

@DIRECTORY id, dir\$[, table]

## BESCHREIBUNG

Diese Präprozessor-Anweisung kann verwendet werden, um das gesamte in `dir$` angegebene Verzeichnis in Ihr Applet oder Programm einzubinden, wenn Sie Ihr Skript kompilieren. Auf diese Weise können Sie beim Kompilieren Ihres Skripts eine Vielzahl von Dateien bequem in Ihr Applet oder Programm verlinken, da Sie nur eine zusätzliche Zeile zu Ihrem Skript anstelle von einzelnen Zeilen für jede Datei hinzufügen müssen.

Wenn Sie @DIRECTORY verwenden, müssen Sie den Befehl `GetDirectoryEntry()` verwenden, um auf einzelne Dateien und Unterverzeichnisse zuzugreifen, die in dem Verzeichnis gespeichert sind, das Sie mit Ihrem Applet oder Ihrem Programm verknüpft haben. Siehe unten für ein Beispiel. Falls Sie Ihr Skript nur mit dem Hollywood-Interpreter ausführen, gibt `GetDirectoryEntry()` die Zeichenkette zurück, die Sie an das Skript übergeben haben, sodass das Skript unabhängig davon identisch funktioniert, ob Sie es als Skript mit dem Hollywood-Interpreter ausführen oder wenn Sie es als Applet oder Programm kompiliert haben. Siehe [Abschnitt 20.29 \[GetDirectoryEntry\]](#), [Seite 270](#), für Details.

Diese Präprozessor-Anweisung akzeptiert auch ein optionales Tabellenargument, mit dem weitere Optionen konfiguriert werden können. Die folgenden Tags werden derzeit vom optionalen Tabellenargument unterstützt:

### Recursive:

Wenn dieser Tag auf `True` gesetzt ist, werden in @DIRECTORY alle Dateien in den Unterverzeichnissen von `dir$` ebenfalls eingebunden. Dies ist die Standardeinstellung. Setzen Sie den Wert auf `False`, wenn Sie nicht möchten, dass @DIRECTORY auch Dateien und Verzeichnisse in Unterverzeichnisse berücksichtigt soll.

**Link:** Setzen Sie dieses Feld auf `False`, wenn dieses Verzeichnis beim Kompilieren Ihres Skripts nicht mit Ihrem Programm/Applet verknüpft werden soll. Dieses Feld ist standardmäßig auf `True` eingestellt, was bedeutet, dass das Verzeichnis mit Ihrem Programm/Applet verknüpft wird, wenn sich Hollywood im Kompilierungsmodus befindet.

Beachten Sie, dass @DIRECTORY nicht nur alle Dateien und Unterverzeichnisse in `dir$` in Ihr Applet oder Programm einbindet, sondern auch ein Verzeichnisobjekt erstellt, das dann mit allen Befehlen verwendet werden kann, die Verzeichnisobjekte unterstützen, z.B. `NextDirectoryEntry()` und `RewindDirectory()`. Es ist sogar möglich, alle Dateien und Unterverzeichnisse zu durchlaufen, die über @DIRECTORY mit Ihrem Skript verknüpft sind. Siehe unten für ein Beispiel.

Beachten Sie schließlich, dass nur Datei-/Verzeichnisnamen, -größen und der tatsächliche Inhalt der Dateien mit Ihrem Applet oder Programm verknüpft werden. Dateiattribute wie Schutzbits, Datumsstempel und Kommentare werden nicht verknüpft. Wenn Sie also versuchen sie abzufragen, werden stattdessen einige Standardwerte angezeigt.

Wenn Sie zur Laufzeit Verzeichnisse öffnen möchten, verwenden Sie bitte den Befehl `OpenDirectory()`.

Siehe auch `CloseDirectory()` und `MonitorDirectory()`.

**EINGABEN**

<code>id</code>	ein Wert, der verwendet wird, um dieses Verzeichnis später im Code zu identifizieren
<code>dir\$</code>	das Verzeichnis, das Sie mit Ihrem Applet oder Programm verknüpfen möchten
<code>table</code>	optional: eine Tabelle mit weiteren Optionen (siehe oben)

**BEISPIEL**

```
@DIRECTORY 1, "data"
LoadBrush(1, GetDirectoryEntry("data/title.png"))
```

Der obige Code zeigt, wie Sie alle Dateien und Unterverzeichnisse im Verzeichnis `data` mit Ihrem Applet oder Programm verknüpfen und dann die Datei `title.png` aus diesem Verzeichnis in Pinsel 1 laden. Beachten Sie, dass das Skript nicht als Applet oder Programm kompiliert wurde. `LoadBrush()` lädt die Datei einfach aus `data/title.png`. Falls das Skript als Applet oder Programm kompiliert wurde, wird die Datei `title.png` jedoch direkt aus dem Applet oder Programm geladen, da sie mit diesem verknüpft wurde.

```
@DIRECTORY 1, "data"
```

```
Function p_DumpDirs(d$, indent)
    Local handle
    If d$ <> ""
        handle = OpenDirectory(Nil, GetDirectoryFile(1, d$))
    Else
        handle = 1
    EndIf
    Local e = NextDirectoryEntry(handle)
    While e <> Nil
        If e.Type = #DOSTYPE_DIRECTORY Then e.size = 0
        NPrint(RepeatStr(" ", indent) .. IIf(e.type = #DOSTYPE_FILE,
            "File:", "Directory:") .. " " .. e.name .. " " .. e.size
            .. " " .. HexStr(e.flags) .. " " .. e.time)
        If e.Type = #DOSTYPE_DIRECTORY Then p_DumpDirs(FullPath(d$,
            e.name), indent + 4)
        e = NextDirectoryEntry(handle)
    Wend
    If GetType(handle) = #LIGHTUSERDATA Then CloseDirectory(handle)
EndFunction
```

```
p_DumpDirs("", 0)
```

Der obige Code zeigt, wie alle Dateien und Verzeichnisse in einem Verzeichnis rekursiv ausgegeben werden, das mit dem Applet oder Programm verknüpft wurde.



## 20.14 DirectoryItems

### BEZEICHNUNG

DirectoryItems – durchläuft alle Elemente in einem Verzeichnis (V7.0)

### ÜBERSICHT

```
f = DirectoryItems(d$)
```

### BESCHREIBUNG

Dieser Befehl kann zusammen mit der generischen **For**-Anweisung verwendet werden, um alle Dateien und Unterverzeichnisse in einem Verzeichnis zu durchlaufen. Er gibt eine Iterator-Funktion zurück, die zwei Werte für jedes Verzeichnisobjekt zurückgibt: Der erste Rückgabewert ist der Name der Datei oder des Verzeichnisses, der zweite Rückgabewert ist eine Tabelle mit zusätzlichen Informationen über das Verzeichniselement. Sobald alle Verzeichniseinträge zurückgegeben wurden, gibt die Iterator-Funktion **Nil** zurück, um die generische **For**-Anweisung zu beenden.

Siehe [Abschnitt 11.4 \[Generische Version der For-Anweisung\]](#), [Seite 135](#), für Details.

Die Tabelle, die von `DirectoryItems()` als zweiter Rückgabewert zurückgegeben wird, wenn sie in einer generischen **For**-Schleife verwendet wird, werden die folgenden Tags initialisiert:

- Type:** Dies ist `#DOSTYPE_FILE`, wenn der Eintrag eine Datei ist oder `#DOSTYPE_DIRECTORY`, falls der Eintrag ein Verzeichnis ist.
- Size:** Dieser Tag ist nur vorhanden, wenn der Eintrag eine Datei ist. In diesem Fall enthält dieser Tag die Größe der Datei in Bytes.
- Flags:** Dieser Tag enthält eine Kombination von Schutzbits der Datei oder des Verzeichnisses. Siehe [Abschnitt 20.63 \[Schutzbits\]](#), [Seite 305](#), für Details.
- Time:** Dieser Tag enthält eine Zeichenkette, die die Zeit enthält, zu der die Datei oder das Verzeichnis zuletzt geändert wurde. Die Zeichenkette ist immer im Format `dd-mmm-yyyy hh:mm:ss`. Z.B: 08-Nov-2004 14:32:13.

**LastAccessTime:**

Dieser Tag erhält eine Zeichenkette, welche die Zeit enthält, wann zuletzt auf die Datei oder dem Verzeichnis zugegriffen wurde. Dieses Attribut wird bei AmigaOS nicht unterstützt.

**CreationTime:**

Dieser Tag erhält eine Zeichenkette, welche die Zeit enthält, wann die Datei oder das Verzeichnis erstellt wurde. Dieses Attribut wird nur bei Windows unterstützt.

**Comment:** Dieser Tag wird den Kommentar einer Datei enthalten. Dies wird nur durch die Amiga-Versionen unterstützt.

Beachten Sie, dass Sie alle Dateien und Unterverzeichnisse innerhalb eines Verzeichnisses auch manuell durchlaufen können, indem Sie die Befehle `OpenDirectory()`, `NextDirectoryEntry()` und `CloseDirectory()` benutzen. Die Verwendung von `DirectoryItems()` ist jedoch oft bequemer.

Siehe auch [CountDirectoryEntries\(\)](#).

**EINGABEN**

**d\$**            Verzeichnis, welches durchlaufen wird

**RÜCKGABEWERTE**

**f**            Iterator-Funktion für die generische For-Schleife

**BEISPIEL**

```
Function p_TraverseDir(d$, indent)
  For s$,t In DirectoryItems(d$)
    DebugPrint(RepeatStr(" ", indent) .. s$, t.time)
    If t.type = #DOSTYPE_DIRECTORY
      p_TraverseDir(FullPath(d$, s$), indent + 8)
    EndIf
  Next
EndFunction
```

```
p_TraverseDir("images", 0)
```

Die Funktion `p_TraverseDir()` kann rekursiv alle Dateien und Unterverzeichnisse im angegebenen Verzeichnis ausgeben. Der Beispielaufruf gibt den Inhalt eines Verzeichnisses mit dem Namen "images" aus, das relativ zum Pfad des Skripts gespeichert werden muss.

## 20.15 Eof

**BEZEICHNUNG**

Eof – prüft, ob das Dateiende erreicht wurde

**ÜBERSICHT**

```
result = Eof(id)
```

**BESCHREIBUNG**

Gibt **True** (WAHR) zurück, falls das Ende der durch **id** angegebenen Datei erreicht wurde, andernfalls **False** (FALSCH).

Siehe auch [FilePos\(\)](#), [FileLength\(\)](#), [FileSize\(\)](#) und [Seek\(\)](#).

**EINGABEN**

**id**            Identifikator der Datei

**RÜCKGABEWERTE**

**result**        **True**, falls das Ende der Datei erreicht ist, oder **False**

**BEISPIEL**

Siehe [Abschnitt 20.48 \[OpenFile\]](#), Seite 290.

## 20.16 Execute

### BEZEICHNUNG

Execute – führt eine Datei synchron aus

### ÜBERSICHT

Execute(file\$[, args\$, t])

### FRÜHERE SYNTAX

Execute(cmdline\$[, resetkeys])

### BESCHREIBUNG

Dieser Befehl führt die durch `file$` angegebene Datei synchron aus und übergibt ihm die in `args$` angegebenen Argumente. Falls Sie ein Programm asynchron ausführen möchten, müssen Sie den Befehl `Run()` benutzen. Siehe [Abschnitt 20.62 \[Run\]](#), [Seite 301](#), für Details.

Wenn synchrones Ausführen einer Datei vom Betriebssystem unterstützt wird, können Sie diesen Befehl fürs Anzeigen von Daten-Dateien wie Dokumente oder Bilder durch das Standardanzeigeprogramm verwenden. In diesem Fall kann `file$` auch eine nicht ausführbare Datei wie ein JPEG-Bild oder eine MP3-Datei sein.

Auf Android muss `file$` entweder eine Datendatei wie ein JPEG-Bild oder ein Paketname wie `com.airsoftsoftwair.hollywood` sein, wenn Sie mit diesem Befehl eine andere App starten wollen.

Beachten Sie, dass es aus historischen Gründen einige Fallstricke bei der Verwendung von diesem Befehl gibt. Vor Hollywood 9.0 erwartete dieser Befehl, dass Programm und Argumente in einer einzigen `cmdline$`-Zeichenkette zusammengefasst werden. In diesem Fall ist beim Umgang mit Leerzeichen besondere Vorsicht geboten (Details siehe unten). Ab Hollywood 9.0 gibt es eine neue Syntax, die es Ihnen ermöglicht, Programm und Argumente als zwei separate Argumente zu übergeben, was die Sache viel einfacher macht. Um die Kompatibilität mit früheren Versionen aufrechtzuerhalten, kann diese neue Syntax jedoch nur verwendet werden, wenn Sie explizit eine Zeichenkette im zweiten Argument übergeben. Wenn Sie also die neue Syntax verwenden möchten, stellen Sie sicher, dass Sie im zweiten Argument eine Zeichenkette übergeben. Wenn das Programm, das Sie starten möchten, keine Argumente benötigt, übergeben Sie einfach eine leere Zeichenkette (`""`), um Hollywood zu signalisieren, dass Sie die neue Syntax verwenden möchten.

Wenn Sie im zweiten Argument keine Zeichenkette übergeben, wird die alte Syntax verwendet, was bedeutet, dass Sie bei der Übergabe von Programmpfaden, die Leerzeichen enthalten, sehr vorsichtig sein müssen, da das allererste Leerzeichen in `cmdline$` als Trennzeichen zwischen Programm und Argumenten interpretiert wird. Wenn Sie ein Programm starten möchten, dessen Pfadangabe Leerzeichen verwendet, müssen Sie diese Pfadangabe in doppelte Anführungszeichen setzen, sonst funktioniert es nicht. Sie können diese Komplikationen leicht vermeiden, indem Sie im zweiten Argument einfach eine Zeichenkette übergeben, auch wenn diese leer ist (siehe oben für Details).

Ab Hollywood 9.0 ist es möglich, das Programm und seine Argumente in zwei separaten Argumenten anzugeben, was die Sache wesentlich komfortabler macht. Außerdem gibt es jetzt ein neues optionales Tabellenargument, das verwendet werden kann, um weitere Optionen anzugeben.

Die folgenden Optionen werden derzeit durch das optionale Tabellenargument unterstützt:

**Directory:**

Mit diesem Tabellenargument können Sie das aktuelle Verzeichnis für das zu startende Programm festlegen. (V9.0)

**ResetKeys:**

Das Tabellenargument **resetkeys** ist nur interessant für fortgeschrittene Anwender. Wenn dies auf **False** gesetzt ist, wird **Execute()** nicht alle internen Tastenzustände zurücksetzt, nachdem das Programm ausgeführt wurde. Standardmäßig werden alle Tastenzustände zurückgesetzt, wenn **Execute()** beendet ist, weil Programme unter Verwendung von **Execute()** übernehmen oft den Tastaturfokus und Hollywood wird möglicherweise nicht in der Lage sein, die internen Status-Flags zurückzusetzen. Aus diesem Grund werden standardmäßig mit **Execute()** alle internen Tastenzustände zurückgesetzt, wenn dieser Befehl beendet ist. Es könnte dann Sinn machen, dieses Verhalten zu deaktivieren, wenn Sie **Execute()** für den Start von Programmen verwenden, die nicht über eine GUI verfügen und nicht den Tastaturfokus wegnehmen. Voreingestellt ist **True**. (V5.1)

**ForceExe:**

Wenn dieser Tag auf **True** gesetzt ist, wird **Execute()** die in **file\$** übergebene Datei immer als ausführbare Datei behandeln. Dies ist nur unter Linux und macOS nützlich, da auf diesen Plattformen Dateien mit einer Erweiterung als Datendateien behandelt werden, sodass Hollywood stattdessen versucht, das entsprechende (Anzeige)Programm für die Datendatei zu starten. Daher funktioniert der Versuch, **Execute()** auf einer ausführbaren Datei namens "test.exe" zu verwenden, aufgrund der Erweiterung \*.exe unter Linux und macOS nicht. Indem Sie **ForceExe** auf **True** setzen, können Sie es jedoch zum Laufen bringen. Die Voreinstellung ist **False**. (V9.0)

**Verb:**

Unter Windows kann dies auf eine Zeichenkette gesetzt werden, der dem Befehl **Execute()** mitteilt, was mit der Datei geschehen soll. Dies kann eines der folgenden Verben sein:

<b>edit</b>	Öffnet die angegebene Datei in einem Editor.
<b>explore</b>	Öffnet den angegebenen Ordner im Explorer. Wenn Sie dieses Verb verwenden, müssen Sie statt einer Datei einen Ordner an <b>Execute()</b> übergeben.
<b>find</b>	Öffnet den Suchdialog für den angegebenen Ordner. Wenn Sie dieses Verb verwenden, müssen Sie statt einer Datei einen Ordner an <b>Execute()</b> übergeben.
<b>open</b>	Öffnet die angegebene Datei.
<b>print</b>	Gibt die angegebene Datei aus.
<b>runas</b>	Startet die angegebene Datei im Administratormodus.

Beachten Sie, dass der Tag **Verb** nur unter Windows unterstützt wird. (V9.1)

Siehe auch **Run()** und **Exists()**.

## EINGABEN

**file\$**      das zu startende Programm (oder die Datei)

**args\$**      optional: Argumente, die an das Programm übergeben werden sollen; Beachten Sie, dass Sie diesen Parameter übergeben müssen, um Hollywood zu signalisieren, die neue Syntax zu verwenden. Sie können dies tun, indem Sie einfach eine leere Zeichenfolge ("" ) angeben; siehe oben für Details (V9.0)

**t**            optional: Tabelle mit weiteren Argumenten (siehe oben) (V9.0)

## BEISPIEL

```
Execute("Sys:Prefs/Locale")
```

Der obige Code führt auf AmigaOS das Einstellungsprogramm Locale aus. Die Ausführung des Skripts wird angehalten, bis der Benutzer das Einstellungsprogramm Locale schließt (synchrone Ausführung).

```
Execute("Echo >Ram:Test \"Hello World\"")
```

Auf Systemen mit AmigaOS schreibt der obige Code "Hello World" in "Ram:Test".

```
Execute("\"C:\\Program Files (x86)\\Hollywood\\ide.exe\"")
```

Der obige Code führt die Hollywood-IDE auf Windows-Systemen aus. Beachten Sie, dass wir die Programmdekleration in doppelte Anführungszeichen eingebettet haben. Dies ist unbedingt erforderlich, da das erste Leerzeichen in der Zeichenkette, die an **Execute()** übergeben wird, normalerweise als Trennzeichen zwischen Programm und Argumenten interpretiert wird. Wenn wir keine doppelten Anführungszeichen in dem obigen Code verwendeten, würde **Execute()** versuchen, das Programm "C:\Program" zu starten und die Argumente "Files (x86)\Hollywood\ide.exe" zu übergeben, was wir offensichtlich nicht wollen. Beachten Sie, dass es seit Hollywood 9.0 jetzt viel einfacher ist, mit Leerzeichen in Pfaden umzugehen. Sie müssen nur die neue Syntax verwenden, die das Programm und seine Argumente in zwei separaten Argumenten verwendet. Mit Hollywood 9.0 können Sie einfach diesen Code verwenden:

```
Execute("C:\\Program Files (x86)\\Hollywood\\ide.exe", "")
```

Beachten Sie, dass die Übergabe der leeren Zeichenkette im zweiten Argument hier unbedingt erforderlich ist, um Hollywood zu signalisieren, dass Sie die neue Syntax verwenden möchten. Siehe oben für detaillierte Informationen darüber.

## 20.17 Exists

### BEZEICHNUNG

Exists – prüft, ob die angegebene Datei existiert

### ÜBERSICHT

```
result = Exists(filename$)
```

**BESCHREIBUNG**

Prüft, ob die durch `filename$` angegebene Datei existiert und gibt dann `True` in die Variable `result` zurück. Andernfalls erhält die Variable `result` den Wert `False`.

**EINGABEN**

`filename$`  
zu prüfende Datei

**RÜCKGABEWERTE**

`result`      `True`, falls die angegebene Datei existiert, andernfalls `False`

**BEISPIEL**

```
result=Exists("S:Startup-Sequence")
Print(result)
```

Das sollte immer "1" ausgeben, was der Wert von `True` ist.

## 20.18 FILE

**BEZEICHNUNG**

FILE – öffnet eine Datei zur späteren Verwendung (V2.0)

**ÜBERSICHT**

`@FILE id, filename$[, table]`

**BESCHREIBUNG**

Diese Präprozessor-Anweisung kann verwendet werden, um eine Datei zu öffnen, so dass Sie sie später verwenden können. Die Datei wird nicht vollständig in den Arbeitsspeicher geladen, sondern wird genauso geöffnet, wie wenn Sie den Befehl `OpenFile()` aufrufen. Die Datei wird immer im schreibgeschützten Modus geöffnet werden. Sie können diese Präprozessor-Anweisung nicht verwenden, um in Dateien zu schreiben.

Die innovative Funktionalität der Präprozessor-Anweisung `@FILE` ist, dass wenn Sie Ihr Skript kompilieren, wird die Datei mit eingebunden werden und Sie können immer noch auf die gleiche Weise auf sie zugreifen, als ob es sich um eine normale Datei auf Ihrer Festplatte handelt, d.h. Sie können die normalen Befehle der DOS-Bibliothek auf die Datei anwenden.

Das dritte Argument `table` ist optional. Es ist eine Tabelle, die weitere Möglichkeiten bereitstellt. Die folgenden Tags der Tabelle können verwendet werden:

**Link:**      Setzen Sie diesen Tag auf `False`, wenn Sie dieses Sample nicht in die ausführbare Datei/das Applet einbinden wollen, wenn Sie Ihr Skript kompilieren. Dieser Tag ist standardmäßig auf `True` gesetzt, was bedeutet, dass das Sample mit der ausführbaren Datei/dem Applet beim Kompilieren verknüpft wird.

**Adapter:**    Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen sollen. Dies muss als eine Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()`

gesetzt wurde. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

#### UserTags:

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an den Dateiadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Wenn Sie die Datei manuell öffnen möchten, nutzen Sie den Befehl `OpenFile()`.

Siehe auch `CloseFile()`, und `Exists()`.

### EINGABEN

<code>id</code>	einen Wert, der verwendet wird, um diese Datei später im Code zu identifizieren
<code>filename\$</code>	die zu öffnende Datei
<code>table</code>	optional: Eine Tabelle für weitere Optionen

### BEISPIEL

```
@FILE 1, "Highscore.txt"
```

Die Anweisung oben öffnet die Datei "Highscore.txt" für die Weiterverarbeitung im Skript.

## 20.19 FileAttributes

### BEZEICHNUNG

FileAttributes – gibt die Attribute einer Datei zurück (V6.0)

### ÜBERSICHT

```
t = FileAttributes(id)
```

### BESCHREIBUNG

Dieser Befehl gibt die Tabelle `t` zurück, die die Attribute einer Datei enthält, die mit dem Befehl `OpenFile()` geöffnet wurde. Dazu gehören Informationen wie die Dateizeit, der vollständige Pfad der Datei, Schutzbits und vieles mehr, je nach Host-Dateisystem.

Folgende Felder der Tabelle `t` werden initialisiert:

<b>Path:</b>	Dieses Feld wird eine Zeichenfolge mit dem vollständigen Pfad zu dieser Datei enthalten.
<b>Size:</b>	Dieses Feld wird auf die Größe der Datei in Byte gesetzt.
<b>Flags:</b>	Dieses Feld wird auf eine Kombination von Schutzbits der Datei gesetzt. Siehe <a href="#">Abschnitt 20.63 [Schutzbits]</a> , <a href="#">Seite 305</a> , für Details.
<b>Time:</b>	Dieses Feld wird eine Zeichenkette mit der Zeit des Verzeichnisses oder Datei enthalten, wann es/sie zuletzt geändert wurde. Die Zeichenkette wird immer im Format dd-mmm-jjjj hh: mm: ss übergeben. Z.B. 08-Nov-2004 14:32:13.

**LastAccessTime:**

Dieses Feld wird eine Zeichenkette mit der Zeit des Verzeichnisses oder Datei enthalten, wann zuletzt auf die Datei/dem Verzeichnis zugegriffen wurde. Dieses Attribut wird nicht auf AmigaOS unterstützt.

**CreationTime:**

Dieses Feld wird eine Zeichenkette mit der Zeit des Verzeichnisses oder Datei enthalten, wann es/sie erstellt wurde. Dieses Attribut wird nur auf Windows unterstützt.

**Comment:** Dieses Feld wird den Kommentar einer Datei enthalten. Wird nur von den Amigaversionen unterstützt.

**Streaming:**

Dieses Feld wird auf **True** gesetzt, wenn die Datei nicht von einem physischen Laufwerk, sondern von einer gestreamten Quelle gelesen wird.

**NoSeek:** Dieses Feld wird auf **True** gesetzt, wenn diese Datei nicht gefunden werden kann. Dies kann passieren, wenn die Datei von einer gestreamten Quelle durch einen Datei-Adaptermodul gelesen wird, die nur sequentielle Lesevorgänge ohne Suchfunktionen ermöglicht.

Wenn Sie die Attribute einer Datei abfragen möchten, die derzeit nicht geöffnet ist, verwenden Sie stattdessen den Befehl **GetFileAttributes()**. Siehe [Abschnitt 20.31 \[GetFileAttributes\]](#), Seite 272, für Details.

Siehe auch **SetFileAttributes()**.

**EINGABEN**

**id**            Identifikator der Datei

**RÜCKGABEWERTE**

**t**            eine Tabelle, die wie oben gezeigt initialisiert wird

**BEISPIEL**

```
OpenFile(1, "test.txt")
t = FileAttributes(1)
Print(t.time)
If t.flags & #FILEATTR_READ_USR
  Print("#FILEATTR_READ_USR is set.")
Else
  Print("#FILEATTR_READ_USR is not set.")
EndIf
```

Der obige Code untersucht die Datei "test.txt" und gibt die zuletzt geänderte Zeit auf den Bildschirm aus. Zusätzlich wird geprüft, ob das Schutzbit **#FILEATTR\_READ\_USR** gesetzt ist.

## 20.20 FileLength

**BEZEICHNUNG**

**FileLength** – gibt die Größe einer geöffneten Datei zurück (V3.0)



**ÜBERSICHT**

```
size = FileLength(id)
```

**BESCHREIBUNG**

Dieser Befehl gibt die aktuelle Größe der in `id` angegebene Datei zurück. Die von diesem Befehl zurückgegebene Größe wird immer auf dem neusten Stand sein. Zum Beispiel könnten Sie in die Datei schreiben und dann würde `FileLength()` die neue Größe der Datei zurückgeben.

Bitte beachten Sie, dass `FileLength()` auch -1 zurückgeben kann, wenn es die Größe der Datei nicht kennt. Dies kann zum Beispiel dann geschehen, wenn die Datei von einer gestreamten Quelle durch einen Datei-Adaptermodul gelesen wird.

Siehe auch `Eof()`, `FilePos()`, `FileSize()` und `Seek()`.

**EINGABEN**

`id`            Identifikator der Datei

**RÜCKGABEWERTE**

`size`            aktuelle Größe dieser Datei

**BEISPIEL**

```
OpenFile(1, "test.txt", #MODE_WRITE)
NPrint(FileLength(1))
WriteLine(1, "Hello World.")
NPrint(FileLength(1))
CloseFile(1)
```

Der obige Code öffnet die Datei "test.txt" zum Schreiben und ruft `FileLength()` zweimal auf. Der erste Aufruf gibt 0 zurück, da die Datei zu diesem Zeitpunkt noch leer ist, aber der zweite Aufruf gibt 13 zurück, da nun einige Zeichen in die Datei geschrieben wurden.

## 20.21 FileLines

**BEZEICHNUNG**

`FileLines` – gibt eine zeilenbasierte Iterator-Funktion zurück (V5.0)

**ÜBERSICHT**

```
f = FileLines(file$)
```

**BESCHREIBUNG**

Dieser Befehl kann in Verbindung mit der generischen `For`-Anweisung verwendet werden, um alle Zeilen einer Datei zu durchlaufen. Er gibt eine Iterator-Funktion zurück, die die nächste Zeile der in `file$` angegebene Datei zurückgibt. Wenn das Ende der Datei erreicht ist, gibt die Iterator-Funktion `Nil` zurück, um die generische `For`-Anweisung abubrechen.

Siehe [Abschnitt 11.4 \[Generische Version der For-Anweisung\]](#), Seite 135, für Details.

Siehe auch `ReadLine()`, `WriteLine()`, `ReadString()`, `WriteString()`, `SetFileEncoding()` und `UseCarriageReturn()`.

**EINGABEN**

file\$      Dateiname

**RÜCKGABEWERTE**

f              zeilenbasierte Iterator-Funktion

**BEISPIEL**

```
For s$ In FileLines("Highscores.txt") Do DebugPrint(s$)
```

Dieser Code wird alle Zeilen Ihrer Datei "Highscores.txt" ausgeben.

## 20.22 FilePart

**BEZEICHNUNG**

FilePart – gibt den Dateiname-Teil eines Pfades zurück

**ÜBERSICHT**

```
file$ = FilePart(path$)
```

**BESCHREIBUNG**

Dieser Befehl extrahiert den Dateinamen aus einem durch `path$` angegebenen Pfad und gibt ihn zurück.

Siehe auch `FullPath()`, `PathPart()` und `ReadDirectory()`.

**EINGABEN**

path\$      Quellpfad

**RÜCKGABEWERTE**

file\$      Dateinameteil

**BEISPIEL**

```
f$=FilePart("Data/Gfx/Test.jpg")
```

```
Print(f$)
```

Dieses Beispiel druckt "Test.jpg" auf den Bildschirm.

## 20.23 FilePos

**BEZEICHNUNG**

FilePos – gibt die Cursorposition der Datei zurück (V2.0)

**ÜBERSICHT**

```
pos = FilePos(id)
```

**BESCHREIBUNG**

Dieser Befehl gibt die Cursorposition der in `id` angegebenen Datei zurück. Der Cursor beginnt bei 0 (Anfang der Datei) und endet bei der Länge der Datei. Sie können mit diesem Befehl herauszufinden, wo Sie in einer Datei sind, weil alle Lese- und Schreiboperationen an dieser Cursorposition starten. Sie können mit dem Befehl `Seek()` die Cursorposition in der Datei ändern.

Siehe auch `Eof()`, `FileLength()` und `FileSize()`.

**EINGABEN**

`id`            Identifikator der Datei

**RÜCKGABEWERTE**

`pos`            Cursorposition in der Datei

**BEISPIEL**

```
OpenFile(1, "test.txt", #MODE_READ)
Seek(1, 1024)
Print(FilePos(1))
CloseFile(1)
Das gibt 1024 aus.
```

## 20.24 FileSize

**BEZEICHNUNG**

`FileSize` – gibt die Größe der angegebenen Datei zurück

**ÜBERSICHT**

```
size = FileSize(file$)
```

**BESCHREIBUNG**

Gibt die Größe der Datei `file$` zurück. Falls die Datei nicht existiert wird -1 zurückgegeben.

Bitte beachten Sie, dass `FileSize()` kann auch -1 zurückgeben, wenn es nicht die Größe der Datei kennt. Dies kann z.B. in dem Fall geschehen, wenn die Datei von einer gestreamten Quelle durch ein Datei-Adaptermodul gelesen wird.

Siehe auch `Eof()`, `FilePos()`, `FileLength()` und `Seek()`.

**EINGABEN**

`file$`            Dateiname der zu prüfenden Datei

**RÜCKGABEWERTE**

`size`            Größe der angegebenen Datei in Bytes

**BEISPIEL**

```
result=FileSize("test.jpg")
Print("The file test.jpg takes up", result, "bytes!")
Das wird die Größe der Datei "test.jpg" ausgeben.
```

## 20.25 FileToString

**BEZEICHNUNG**

`FileToString` – liest die ganze Datei in eine Zeichenkette

**ÜBERSICHT**

```
s$, len = FileToString(file$[, t])
```

## BESCHREIBUNG

Dieser Befehl ist ein Komfortbefehl, der einfach die in `file$` angegebene Datei in den Arbeitsspeicher liest und sie als Zeichenkette `s$` zurückgibt. Der zweite Rückgabewert `len` enthält die Dateilänge in Bytes. Beachten Sie, dass Sie diesen Befehl auch verwenden können, um Nicht-Text-Dateien in Zeichenketten einzulesen, da Hollywood-Zeichenketten auch Binärdaten enthalten können.

Ab Hollywood 10.0 akzeptiert `FileToString()` ein optionales Tabellenargument `t`, mit dem Sie zusätzliche Argumente an den Befehl übergeben können. Die folgenden Tags werden derzeit vom optionalen Tabellenargument unterstützt:

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen sollen. Dies muss als eine Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()` gesetzt wurde. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V10.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an den Dateiadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Siehe auch `StringToFile()`.

## EINGABEN

`file$`      Datei, welche in eine Zeichenkette gelesen wird  
`t`            optional: Tabelle mit zusätzlichen Optionen (siehe oben) (V10.0)

## RÜCKGABEWERTE

`s$`            Inhalt der angegebenen Datei als Zeichenkette  
`len`          Länge der Datei in Bytes

## 20.26 FlushFile

### BEZEICHNUNG

FlushFile – leert alle ausstehenden Puffer (V2.5)

### ÜBERSICHT

FlushFile(id)

### BESCHREIBUNG

Dieser Befehl leert alle ausstehenden Puffer der in `id` angegebenen Datei und justiert den Dateicursor neu. Es ist wichtig, dass Sie diesen Befehl aufrufen, wenn Sie zwischen gepuffertes und ungepuffertes IO bei der gleichen Datei wechseln. Wenn Sie `SetIOMode()` überhaupt nicht verwenden, müssen Sie sich keine Gedanken über die Pufferleerung machen, weil alles automatisch durch das Dateisystem durchgeführt wird, wenn Sie nur IO gepufferte verwenden.

**EINGABEN**

`id`            Identifikator der zu leerenden Datei

**20.27 FullPath****BEZEICHNUNG**

`FullPath` – kombiniert Verzeichnis und Datei in einen vollständigen Pfad (V2.0)

**ÜBERSICHT**

```
path$ = FullPath(dir$, file$[, ...])
```

**BESCHREIBUNG**

Dieser Befehl kombiniert das Verzeichnis `dir$` und die Datei `file$` und gibt den vollständigen Pfad in `path$` zurück.

Ab Hollywood 9.0 akzeptiert dieser Befehl eine unbegrenzte Anzahl von Argumenten, sodass Sie eine unbegrenzte Anzahl von Pfadbestandteilen zu einem einzigen Pfad kombinieren können.

Siehe auch `FilePart()`, `PathPart()` und `ReadDirectory()`.

**EINGABEN**

`dir$`            Quellverzeichnis

`file$`          Quelldatei

`...`           optional: zusätzliche Elemente, die an den Pfad angehängt werden (V9.0)

**RÜCKGABEWERTE**

`path$`          Pfadangabe

**BEISPIEL**

```
path$ = FullPath("/home/andreas", "image.jpg")
```

`path$` empfängt die Zeichenkette `"/home/andreas/image.jpg"`

```
path$ = FullPath("/home", "andreas", "image.jpg")
```

Dies entspricht dem ersten Beispiel, übergibt jedoch drei statt zwei Argumente an `FullPath()`.

**20.28 GetCurrentDirectory****BEZEICHNUNG**

`GetCurrentDirectory` – gibt den vollständigen Pfad des aktuellen Verzeichnisses zurück (V4.5)

**ÜBERSICHT**

```
dir$ = GetCurrentDirectory([t])
```

**BESCHREIBUNG**

Dieser Befehl gibt einfach den vollständigen Pfad des aktuellen Verzeichnis in `dir$` zurück. Sie können das aktuelle Verzeichnis mit dem Befehl `ChangeDirectory()` wechseln.

Ab Hollywood 10.0 akzeptiert dieser Befehl ein optionales Tabellenargument `t`, das die folgenden Tags unterstützt:

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateisystemadapter angeben, die die Operation ausführen. Dies muss auf eine Zeichenkette gesetzt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()` gesetzt wurde. Siehe [Abschnitt 7.9 \[Lader und Adapter\]](#), Seite 96, für Details. (V10.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an dem Dateisystemadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), Seite 100, für Details. (V10.0)

Siehe auch `ChangeDirectory()`, `GetProgramDirectory()`, `GetStartDirectory()`, `GetFileAttributes()` mit dem Tabellenfeld `Path` und `MakeDirectory()`.

**EINGABEN**

`t` optional: Tabellenargument mit weiteren Optionen (siehe oben) (V10.0)

**RÜCKGABEWERTE**

`dir$` Pfad des aktuellen Verzeichnis

## 20.29 GetDirectoryEntry

**BEZEICHNUNG**

`GetDirectoryEntry` – ermittelt den Eintrag vom verknüpften Verzeichnis (V8.0)

**ÜBERSICHT**

`e$ = GetDirectoryEntry(id, entry$)`

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um den durch `entry$` angegebenen Eintrag aus dem durch `id` angegebenen Verzeichnis zu ermitteln. Dieses Verzeichnis muss zuvor mit der Präprozessor-Anweisung `@DIRECTORY` geöffnet worden sein.

Wenn `GetDirectoryEntry()` von einem Applet oder einer ausführbaren Datei aufgerufen wird, die alle Dateien in dem durch `id` angegebenen Verzeichnis enthält, gibt er einen speziellen Handler zurück, der an alle Hollywood-Befehle übergeben werden kann, die einen Datei- oder Verzeichnisnamen akzeptieren, z.B. `LoadBrush()`.

Wenn dieser Befehl aufgerufen wird, während nur ein Skript mit dem Hollywood-Interpreter ausgeführt wird, d.h. nicht von einem eigenständigen Applet oder einer ausführbaren Datei, gibt `GetDirectoryEntry()` die in `entry$` übergebene Zeichenfolge zurück. Dadurch wird möglich, dass Skripts, die `@DIRECTORY` verwenden, sich genau

gleich verhalten, wie Applets oder kompilierte ausführbare Dateien. Wenn sie mit dem Hollywood-Interpreter ausgeführt werden, laden sie die Daten einfach aus einer echten Datei. Falls `GetDirectoryEntry()` aber von einem eigenständigen Applet oder einer ausführbaren Datei aufgerufen wird, werden jedoch die Daten direkt aus dem Applet oder der ausführbaren Datei geladen.

Siehe auch `NextDirectoryEntry()` und `RewindDirectory()`.

#### EINGABEN

`id`            Identifikator des abzufragenden Verzeichnisses

`entry$`        Eintrag des Verzeichnisses, das Sie erhalten möchten

#### RÜCKGABEWERTE

`e$`            spezieller Handler für diesen Eintrag, der an alle Hollywood-Befehle übergeben werden kann, die Dateien oder Verzeichnisse als Parameter akzeptieren

#### BEISPIEL

Siehe [Abschnitt 20.13 \[DIRECTORY\]](#), Seite 254.

## 20.30 GetEnv

#### BEZEICHNUNG

`GetEnv` – liest die Umgebungsvariable (V5.0)

#### ÜBERSICHT

`s$, ok = GetEnv(var$)`

#### BESCHREIBUNG

Dieser Befehl kann den Inhalt der in `var$` angegebenen Umgebungsvariable auslesen. Wenn die angegebene Umgebungsvariable nicht gefunden werden konnte, wird in `s$` eine leere Zeichenkette zurückgegeben und der zweite Rückgabewert `ok` wird auf `False` gesetzt. Wenn die Umgebungsvariable gefunden werden konnte, wird der Rückgabewert `ok` `True` sein.

Siehe auch `SetEnv()` und `UnsetEnv()`.

#### EINGABEN

`var$`            Umgebungsvariable

#### RÜCKGABEWERTE

`s$`            Inhalt der angegebenen Umgebungsvariable

`ok`            `True`, wenn die angegebene Umgebungsvariable gefunden werden konnte, andernfalls `False`

## 20.31 GetFileAttributes

### BEZEICHNUNG

GetFileAttributes – gibt die Attribute einer Datei oder eines Verzeichnisses zurück (V3.0)

### ÜBERSICHT

```
t = GetFileAttributes(f$, table)
```

### FRÜHERE SYNTAX

```
t = GetFileAttributes(f$, adapter$)
```

### BESCHREIBUNG

Dieser Befehl gibt die Tabelle `t` zurück, die die Attribute einer Datei oder eines Verzeichnisses enthält. Dazu gehören Informationen wie die Dateizeit, den vollständigen Pfad der Datei, Schutzkennzeichen und mehr (ist vom Host-Dateisystem abhängig). Übergeben Sie diesem Befehl den Namen einer Datei oder eines Verzeichnisses in `f$`. Sie können eine leere Zeichenkette ("" ) angeben, um Informationen über das aktuelle Verzeichnisses zu erhalten.

Dieser Befehl akzeptiert ein optionales Tabellenargument `table`, das verwendet werden kann, um zusätzliche Parameter zu übergeben. Die folgenden Tabellenelemente werden derzeit erkannt:

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen sollen. Dies muss als eine Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()` gesetzt wurde. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), Seite 96, für Details. (V6.0)

**UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an den Dateiadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), Seite 100, für Details. (V10.0)

Die Tabelle wird mit folgenden Feldern initialisiert:

**Type:** Dies wird `#DOSTYPE_FILE` sein, wenn `f$` eine Datei oder `#DOSTYPE_DIRECTORY`, falls `f$` ein Verzeichnis ist.

**Path:** Dieses Feld wird eine Zeichenfolge mit dem vollständigen Pfad zu dieser Datei oder dem Verzeichnis enthalten.

**Size:** Dieses Feld ist nur vorhanden, wenn in `f$` eine Datei übergeben wurde. In diesem Fall wird dieses Feld die Größe der Datei in Bytes enthalten.

**Flags:** Dieses Feld wird eine Kombination aus Schutzbits der Datei oder des Verzeichnisses enthalten. Siehe [Abschnitt 20.63 \[Schutzbits Informationen\]](#), Seite 305, für Details.

**Time:** Dieses Feld wird eine Zeichenkette mit der Datei- oder Verzeichniszeit enthalten, wann sie zuletzt geändert wurde. Die Zeichenkette wird immer im Format `dd-mmm-jjjj hh: mm: ss.` übergeben, Z. B . 08-Nov-2004 14:32:13.



**LastAccessTime:**

Dieses Feld wird eine Zeichenkette mit der Datei- oder Verzeichniszeit enthalten, wann zuletzt auf die Datei/dem Verzeichnis zugegriffen wurde. Dieses Attribut wird nicht auf AmigaOS unterstützt.

**CreationTime:**

Dieses Feld wird eine Zeichenkette mit der Zeit der Datei oder dem Verzeichnis enthalten, wann sie/es erstellt wurde. Dieses Attribut wird nur auf Windows unterstützt.

**Comment:** Dieses Feld wird den Kommentar einer Datei enthalten. Wird nur von den Amigaversionen unterstützt.

**Virtual:** Dieses Feld wird auf **True** gesetzt, wenn die Datei, die Sie diesem Befehl übergeben haben, eine virtuelle ist. d.h. es ist eine mit Ihrem Applet/ausführbaren Programm verknüpfte Datei oder eine mit dem Befehl **DefineVirtualFile()** erstellte Datei. (V5.2)

Wenn Sie die Attribute einer Datei abfragen möchten, die Sie mit dem Befehl **OpenFile()** geöffnet haben, verwenden Sie stattdessen den Befehl **FileAttributes()**. Siehe **Abschnitt 20.19 [FileAttributes], Seite 263**, für Details.

Siehe auch **SetFileAttributes()**.

**EINGABEN**

**f\$**            Name der Datei oder Verzeichnis

**table**       optional: Tabelle mit weiteren Optionen (siehe oben) (V10.0)

**RÜCKGABEWERTE**

**t**            eine Tabelle, die wie oben gezeigt initialisiert wird

**BEISPIEL**

```
t = GetFileAttributes("test.txt")
Print(t.time)
If t.flags & #FILEATTR_READ_USR
  Print("#FILEATTR_READ_USR is set.")
Else
  Print("#FILEATTR_READ_USR is not set.")
EndIf
```

Der obige Code prüft die Datei "test.txt" und gibt die Zeit, als sie zuletzt geändert wurde, auf dem Bildschirm aus. Außerdem prüft sie, ob der Schutzbit **#FILEATTR\_READ\_USR** gesetzt ist.

**20.32 GetProgramDirectory****BEZEICHNUNG**

**GetProgramDirectory** – gibt das Verzeichnis des Programms zurück (V9.0)

**ÜBERSICHT**

```
dir$ = GetProgramDirectory()
```

**BESCHREIBUNG**

Dieser Befehl gibt das Verzeichnis des Programms zurück. Beachten Sie, dass die Verwendung von diesem Befehl nur in kompilierten Programmen sinnvoll ist, da `GetProgramDirectory()` beim Ausführen von Skripten das Verzeichnis des Hollywood-Interpreters zurückgibt, weil derzeit kein anderes Programm vorhanden ist.

Siehe auch `ChangeDirectory()`, `GetCurrentDirectory()`, `GetStartDirectory()`, `GetFileAttributes()` mit dem Tabellenfeld `Path` und `MakeDirectory()`.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`dir$`        Pfad zum Programmverzeichnis

## 20.33 GetStartDirectory

**BEZEICHNUNG**

`GetStartDirectory` – gibt das aktuelle Startverzeichnis des Skripts zurück (V9.0)

**ÜBERSICHT**

`dir$ = GetStartDirectory()`

**BESCHREIBUNG**

Dieser Befehl gibt das Verzeichnis zurück, welches das aktuelle war, als das Hollywood-Skript gestartet wurde. Dies ist nur beim Ausführen von Hollywood-Skripten von Nutzen, da Hollywood in diesem Fall immer das aktuelle Verzeichnis in das Verzeichnis des Skripts ändert (es sei denn, Sie verwenden das Konsolenargument `-nochdir`, so dass es nicht einfach möglich ist, das Verzeichnis herauszufinden, welches aktuell war, bevor Hollywood es in das Verzeichnis des Skripts änderte.

Bei kompilierten Programmen ist das Startverzeichnis immer identisch mit dem Programmverzeichnis, das Sie mit `GetProgramDirectory()` ermitteln können. Siehe Abschnitt 20.32 [`GetProgramDirectory`], Seite 273, für Details.

Siehe auch `ChangeDirectory()`, `GetCurrentDirectory()`, `GetProgramDirectory()`, `GetFileAttributes()` mit dem Tabellenfeld `Path` und `MakeDirectory()`.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`dir$`        Pfad des Startverzeichnis

## 20.34 GetTempFileName

**BEZEICHNUNG**

`GetTempFileName` – gibt den Namen einer temporären Datei zurück (V3.0)

**ÜBERSICHT**

`f$ = GetTempFileName()`

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um eine Datei zu erstellen, die Sie vorübergehend nutzen können. Dies ist nützlich, wenn Sie für kurze Zeit einige Informationen in einer Datei speichern müssen, die Sie später löschen werden. Hollywood wird alle temporären Dateien automatisch löschen, wenn es beendet wird, aber Sie können auch manuell die vorübergehende Datei mit dem Befehl `DeleteFile()` löschen.

Wenn Sie mit temporären Dateien arbeiten müssen, ist es zu empfehlen, diesen Befehl zu verwenden, da jedes Betriebssystem seine temporären Dateien an einem anderen Ort speichert. Durch die Verwendung von `GetTempFileName()` können Sie sicher sein, dass Ihre temporären Dateien in dem richtigen Ordner landen.

Bitte beachten Sie, dass dieser Befehl nicht nur einen Dateinamen zurückgibt, sondern er wird auch eine leere Datei für Sie erstellen. Dies geschieht um mögliche Konflikte mit anderen Anwendungen zu vermeiden, die ihre eigene temporäre Datei unter dem gleichen Namen speichern möchten. Dies ist nicht möglich, wenn die Datei bereits vorhanden ist. Das ist der Grund, warum `GetTempFileName()` für Sie eine leere Datei erstellen wird.

Siehe auch `DefineVirtualFile()`, `DefineVirtualFileFromString()` und `UndefineVirtualStringFile()`. ■

**EINGABEN**

keine

**RÜCKGABEWERTE**

f\$           Dateinamen, den Sie für temporäre Operationen verwenden können

**BEISPIEL**

```
f$ = GetTempFileName()
OpenFile(1, f$, #MODE_WRITE)
WriteLine(1, "My temporary file")
CloseFile(1)
```

Der obige Code wird den Namen einer temporären Datei erhalten und schreibt dann einen Text hinein. Die Datei wird automatisch gelöscht, wenn Hollywood beendet wird.

**20.35 GetVolumeInfo****BEZEICHNUNG**

`GetVolumeInfo` – zeigt die Speicherplatzinformationen eines Datenträgers an

**ÜBERSICHT**

```
space = GetVolumeInfo(vol$, type)
```

**BESCHREIBUNG**

Dieser Befehl fragt den durch `vol$` benannten Datenträger nach den durch `type` angegebenen Informationen ab. Die folgenden Konstanten sind für `type` möglich:

**#FREESPACE:**

gibt den freien Platz des Datenträgers zurück

**#USEDSPACE:**

gibt den benutzten Platz des Datenträgers zurück

Siehe auch `GetVolumeName()` und `HaveVolume()`.

#### EINGABEN

`vol$`        Name eines Datenträgers  
`type`        eine der oben genannten Konstanten

#### RÜCKGABEWERTE

`info`        freier bzw. benutzter Platz des Datenträgers

#### BEISPIEL

```
space=GetVolumeInfo("SYS:",#FREESPACE)
Print(space, "bytes are free on SYS:!")
```

Dieses Beispiel gibt auf AmigaOS den freien Platz auf Ihrer Partition SYS: zurück.

## 20.36 GetVolumeName

#### BEZEICHNUNG

`GetVolumeName` – fragt einen Datenträgernamen ab

#### ÜBERSICHT

```
name$ = GetVolumeName(vol$)
```

#### BESCHREIBUNG

Dieser Befehl versucht, den Namen des durch `vol$` angegebenen Datenträgers zu bekommen. Wenn er erfolgreich ist, wird der Datenträgername in `name$` zurückgegeben.

Siehe auch `GetVolumeInfo()` und `HaveVolume()`.

#### EINGABEN

`vol$`        AmigaDOS Datenträgerbezeichnung

#### RÜCKGABEWERTE

`name$`        Name des Datenträgers

#### BEISPIEL

```
n$=GetVolumeName("df0:")
Print(n$)
```

Dieses Beispiel druckt den Namen der Diskette in Laufwerk df0: (falls eine eingelegt ist).

## 20.37 HaveVolume

#### BEZEICHNUNG

`HaveVolume` – überprüft, ob ein Datenträger im System vorhanden ist (V8.0)

#### ÜBERSICHT

```
r = HaveVolume(vol$)
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um zu prüfen, ob der in `vol$` angegebene Datenträger derzeit verfügbar ist. Dies ist besonders bei AmigaOS- und kompatiblen Systemen

hilfreich, da dadurch der Systemdialog "Benötige den Datenträger XXX in beliebigem Laufwerk" unterdrückt wird, der normalerweise bei einem AmigaOS-Systemen angezeigt wird, wenn versucht wird, auf einen nicht vorhandenen Datenträger zuzugreifen. Wenn Sie mit diesem Befehl zunächst prüfen, ob der Datenträger vorhanden ist, können Sie dieses lästige Dialogfenster auf AmigaOS einfach loswerden.

Siehe auch `GetVolumeInfo()` und `GetVolumeName()`.

#### EINGABEN

`vol$` Name eines DOS-Datenträgers, dessen Vorhandensein überprüft werden soll

#### RÜCKGABEWERTE

`r` `True` bei vorhandenem Datenträger, andernfalls `False`

#### BEISPIEL

```
Print(HaveVolume("FOOBAR:"))
```

Der obige Code sollte 0 zurückgeben, da der angegebene Typ normalerweise nicht vorhanden ist. Unter AmigaOS wird kein Systemdialog nach dem Datenträger "FOOBAR:" fragen, wenn Sie diesen Code verwenden.

## 20.38 IsAbsolutePath

#### BEZEICHNUNG

`IsAbsolutePath` – prüft, ob der Pfad absolut ist (V9.0)

#### ÜBERSICHT

```
result = IsAbsolutePath(p$)
```

#### BESCHREIBUNG

Dieser Befehl prüft, ob der durch `p$` angegebene Pfad ein absoluter Pfad ist und gibt `True` zurück, wenn dies der Fall ist, oder `False`, wenn dies nicht der Fall ist.

Beachten Sie, dass erwartet wird, dass der angegebene Pfad im kanonischen Format des Hosts vorliegt. Daher ist ein Pfad wie `"/home"` unter Linux (unter anderem) absolut, aber nicht unter AmigaOS, wobei `"/home"` nur auf ein Verzeichnis mit dem Namen `"home"` im übergeordneten Verzeichnis verweist. Um einen Pfad in das Hostformat zu konvertieren, können Sie `MakeHostPath()` verwenden.

Siehe auch `CanonizePath()`, `IsDirectory()` und `MakeHostPath()`.

#### EINGABEN

`p$` der zu überprüfende Pfad

#### RÜCKGABEWERTE

`result` `True`, wenn Pfad absolut ist, sonst `False`

## 20.39 IsDirectory

#### BEZEICHNUNG

`IsDirectory` – überprüft, ob es eine Datei oder ein Verzeichnis ist (V2.0)

**ÜBERSICHT**

```
r = IsDirectory(f$)
```

**BESCHREIBUNG**

Dieser Befehl überprüft, ob `f$` eine Datei oder ein Verzeichnis ist. Wenn es ein Verzeichnis ist, gibt dieser Befehl `True` zurück, andernfalls `False`.

Siehe auch `CanonizePath()`, `IsAbsolutePath()` und `MakeHostPath()`.

**EINGABEN**

`f$`            Systemobjekt

**RÜCKGABEWERTE**

`r`            `True`, wenn `f$` ein Verzeichnis ist, ansonsten `False`

**BEISPIEL**

```
r = IsDirectory("S:")
```

Das gibt `True` zurück.

## 20.40 MakeDirectory

**BEZEICHNUNG**

`MakeDirectory` – erstellt ein neues Verzeichnis (V1.5)

**ÜBERSICHT**

```
MakeDirectory(dir$, t)
```

**BESCHREIBUNG**

Dieser Befehl erstellt ein neues Verzeichnis, welches durch `dir$` angegeben wurde. Beachten Sie, dass dieser Befehl nicht fehlschlägt, wenn das durch `dir$` angegebene Verzeichnis bereits existiert.

Dieser Befehl kann auch mehrere Verzeichnisse erstellen, falls diese noch nicht existieren. Alle Angaben in `dir$` werden rekursiv durchgegangen und jedes nicht existente Verzeichnis wird erstellt (ab Hollywood 1.9).

Ab Hollywood 10.0 akzeptiert dieser Befehl ein optionales Tabellenargument, das die folgenden Tags unterstützt:

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateisystemadapter angeben, die die Operation ausführen. Dies muss auf eine Zeichenkette gesetzt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()` gesetzt wurde. Siehe [Abschnitt 7.9 \[Lader und Adapter\]](#), [Seite 96](#), für Details. (V10.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an den Dateisystemadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Siehe auch `ChangeDirectory()`, `GetCurrentDirectory()`, `GetProgramDirectory()`, `GetStartDirectory()`, und `GetFileAttributes()` mit dem Tabellenfeld `Path`.

#### EINGABEN

`dir$` zu erstellendes Verzeichnis

`t` optional: Tabelle mit weiteren Optionen (siehe oben) (V10.0)

#### BEISPIEL

```
MakeDirectory("Test")
```

Der obige Code erstellt das neue Verzeichnis "Ram:Test".

```
MakeDirectory("Ram:A/B/C/D/E")
```

Der obige Code erstellt fünf neue Verzeichnisse in Ram:

## 20.41 MakeHostPath

#### BEZEICHNUNG

`MakeHostPath` – konvertiert den Hollywood-Pfad in das Host-Pfad-Format (V9.0)

#### ÜBERSICHT

```
p$ = MakeHostPath(path$)
```

#### BESCHREIBUNG

Mit diesem Befehl kann ein plattformunabhängiger Hollywood-Pfad in einen Pfad im kanonischen Pfadformat des Hostsystems konvertiert werden. Der Pfad, der konvertiert werden soll, muss in `path$` übergeben werden.

Um die plattformübergreifende Kompatibilität sicherzustellen, können Hollywood-Pfade mehrere Bestandteile enthalten, die das zugrunde liegende Host-Betriebssystem nicht versteht. Beispielsweise ist es möglich, .. unter AmigaOS und kompatiblen Systemen zu verwenden, um das übergeordnete Verzeichnis anzugeben, obwohl AmigaOS dies nicht versteht. Es ist auch möglich, normale Schrägstriche in Pfaden unter Windows zu verwenden, obwohl dieses Betriebssystem normalerweise den umgekehrten Schrägstrich (Backslash) verwendet. Andererseits ist es auch möglich, umgekehrte Schrägstriche auf allen anderen Systemen zu verwenden, obwohl sie Schrägstriche verwenden, und so weiter.

`MakeHostPath()` stellt sicher, dass der zurückgegebene Pfad vollständig den Anforderungen des Host-Betriebssystems entspricht. Normalerweise müssen Sie diesen Befehl jedoch nicht verwenden, da alle Hollywood-Befehle plattformunabhängige Hollywood-Pfade verarbeiten können. Dieser Befehl muss möglicherweise nur aufgerufen werden, wenn Pfade an externe Programme übergeben werden, die das plattformunabhängige Pfadformat von Hollywood nicht verstehen.

Siehe auch `CanonizePath()`, `IsAbsolutePath()`, und `IsDirectory()`.

#### EINGABEN

`path$` Pfad, der konvertiert werden soll

#### RÜCKGABEWERTE

`p$` konvertierter Pfad im kanonischen Format des Hostsystems

**BEISPIEL**

```
Print(MakeHostPath("../image.jpg"))
```

Unter AmigaOS wird dies `"/image.jpg"` ausgegeben, da AmigaOS das Token `..` nicht versteht. Unter Windows wird `"..\image.jpg"` ausgegeben, da Windows umgekehrte Schrägstriche (Backslashes) anstelle von Schrägstrichen verwendet. Auf allen anderen Plattformen wird die Quellzeichenkette zurückgegeben, da keine Änderungen erforderlich sind.

**20.42 MatchPattern****BEZEICHNUNG**

MatchPattern – überprüft die Zeichenkette mit einem Muster (V2.0)

**ÜBERSICHT**

```
bool = MatchPattern(src$, pattern$)
```

**BESCHREIBUNG**

Dieser Befehl überprüft, ob die in `src$` angegebene Zeichenkette mit dem in `pattern$` angegebenen Muster übereinstimmt. Ist dies der Fall, wird `True` zurückgegeben, sonst `False`. `MatchPattern()` vergleicht `pattern$` mit `src$` Zeichen für Zeichen und bricht ab, sobald er einen Unterschied findet. Wenn er keinen Unterschied findet, gibt er `True` zurück.

Das Muster, das in `pattern$` angegeben wird, ist eine Zeichenkette, die normale Zeichen und Platzhalterzeichen enthalten kann. Ein Platzhalter ist ein Sonderzeichen, das verwendet werden kann, um mehr als ein Zeichen in der Quellzeichenkette anzupassen. Folgende Platzhalter werden derzeit unterstützt:

- `*`            Entspricht allen Zeichen.
- `?`            Entspricht nur einem einzelnen Zeichen.
- `#`            Entspricht allen Zahlen.
- `[]`           Entspricht einem oder mehreren Zeichen oder einer Reihe von Zeichen, wenn diese mit einem Bindestrich abgegrenzt werden. Beispielsweise stimmt `[a]` nur mit `a` überein, während `[af]` mit `a` und `f` übereinstimmt und `[a-f]` stimmt mit allen Zeichen im Bereich von `a` bis `f` überein. Sie können das `'!'` Präfix verwenden, um das Ergebnis zu negieren, d.h. `[! A]` passt zu jedem Zeichen außer `a`.

Sie können auch mehrere Muster in einer einzelnen Zeichenkette kombinieren, indem Sie sie mit einem Semikolon (;) trennen.

Wenn Sie eine anspruchsvollere Musteranpassung benötigen, dann benutzen Sie den Befehl `PatternFindStr()`. Siehe [Abschnitt 57.44 \[PatternFindStr\]](#), [Seite 1283](#), für Details.

**EINGABEN**

`src$`            Quellenzeichenkette

`pattern$`    Muster, womit die Zeichenkette verglichen wird



**RÜCKGABEWERTE**

`bool`            `True`, wenn die Zeichenkette überein stimmt, andernfalls `False`

**BEISPIEL**

```
r = MatchPattern("Pictures/JPG/Pic1.jpg", "*.jpg")
```

Gibt `True` zurück, da die Zeichenfolge mit dem Muster übereinstimmt.

```
r = MatchPattern("Pictures/JPG/Pic1.gif", "*.jpg;*.gif")
```

Gibt auch `True` zurück, da die Zeichenfolge mit dem Muster übereinstimmt.

```
r = MatchPattern("Hollywood 2.a", "Hollywood #.#")
```

Gibt `False` zurück, da `a` nicht mit dem numerischen Platzhalter (`#`) übereinstimmt.

**20.43 MD5****BEZEICHNUNG**

MD5 – berechnet die MD5-Prüfsumme einer Datei (V5.0)

**ÜBERSICHT**

```
sum$ = MD5(f$)
```

**BESCHREIBUNG**

Dieser Befehl berechnet die MD5-Prüfsumme der in `f$` angegebenen Datei und gibt sie zurück. Die 128-Bit-Prüfsumme wird als Zeichenfolge zurückgegeben, die 16 Hex-Ziffern enthält.

Wenn Sie die MD5-Prüfsumme einer Zeichenkette berechnen möchten, verwenden Sie stattdessen den Befehl `MD5Str()`.

Siehe auch `CRC32()` und `CRC32Str()`.

**EINGABEN**

`f$`            Datei, dessen Prüfsumme Sie berechnen wollen

**RÜCKGABEWERTE**

`sum$`            MD5-Prüfsumme der Datei

**20.44 MonitorDirectory****BEZEICHNUNG**

MonitorDirectory – überwacht Änderungen in einem Verzeichnis (V8.0)

**ÜBERSICHT**

```
[id] = MonitorDirectory(id, dir$[, table])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um Änderungen in dem durch `dir$` angegebenen Verzeichnis zu überwachen. Um Verzeichnisänderungen zu überwachen, erstellt

`MonitorDirectory()` ein neues Verzeichnisobjekt und weist ihm die angegebene `id` zu. Wenn Sie `Nil` in `id` übergeben, wählt `MonitorDirectory()` automatisch einen freien Identifikator und gibt diesen zurück.

Wenn sich etwas in dem von `dir$` angegebenen Verzeichnis ändert, sendet `MonitorDirectory()` ein `DirectoryChanged`-Ereignis an Ihr Skript. Um mit diesem Ereignis umgehen zu können, müssen Sie zunächst einen Ereignis-Handler installieren, indem Sie den Befehl `InstallEventHandler()` verwenden. Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), Seite 483, für Details.

`MonitorDirectory()` akzeptiert auch ein optionales Tabellenargument, mit dem Sie einige weitere Optionen konfigurieren können. Die folgenden Tags werden derzeit vom optionalen Tabellenargument erkannt:

**All:** Wenn dieser Tag auf `True` gesetzt ist, leitet `MonitorDirectory()` alle Verzeichnisänderungsbenachrichtigungen vom Betriebssystem an Ihr Skript weiter. Überlegen Sie sich dies zweimal, denn je nach Betriebssystem und Dateisystem erhalten Sie möglicherweise mehrere Meldungen für eine einzige Änderung, weil das Dateisystem intern ist. Standardmäßig versucht `MonitorDirectory()`, solche doppelten Benachrichtigungen für Sie zu filtern, damit Sie nicht mehrere Meldungen für eine einzige Änderung erhalten. Wenn Sie nicht möchten, dass `MonitorDirectory()` diesen Filter anwendet, d.h. wenn Sie alle Benachrichtigungen möchten, setzen Sie diesen Tag auf `True`. Der Standardwert ist `False`.

**UserData:**

Dieser Tag kann auf einen beliebigen Typ gesetzt werden. `MonitorDirectory()` speichert ihn im Feld `MonitorUserData` der Nachricht, die vom `InstallEventHandler()` gesendet wird. Dies ist nützlich, um globale Variablen zu vermeiden.

**ReportChanges:**

Wenn dieser Tag auf `True` gesetzt ist, wird Ihr Ereignis-Callback auch darüber informiert, was sich genau geändert hat. Ihr Ereignis-Callback erhält zwei neue Parameter: `Type` informiert Sie über die Art der Änderung, d.h. ob eine Datei oder ein Verzeichnis hinzugefügt, entfernt oder geändert wurde, und `Name` enthält den Namen der geänderten Datei oder des Verzeichnisses. Beachten Sie, dass das Tabellen-Tag `All` (siehe oben) ignoriert wird, wenn `ReportChanges` auf `True` gesetzt wird. (V9.0)

Beachten Sie, dass das von diesem Befehl erstellte Verzeichnisobjekt nur zur Überwachung von Verzeichnisänderungen verwendet werden darf. Es ist nicht möglich, es an andere Verzeichnisbefehle wie `NextDirectoryEntry()` oder `RewindDirectory()` zu übergeben. Eine Ausnahme ist der Befehl `CloseDirectory()`: Sie sollten `CloseDirectory()` aufrufen, sobald Sie die Überwachung des Verzeichnisses abgeschlossen haben. Dadurch wird sichergestellt, dass keine Ressourcen verschwendet und keine unnötigen Nachrichten in Ihrem Skript bereitgestellt werden.

Beachten Sie auch, dass einige Dateisysteme die Überwachung von Verzeichnissen nicht unterstützen. Dies kann insbesondere auf Netzwerkdatenträger oder Netzwerkdateisystemen der Fall sein. In diesem Fall kann `MonitorDirectory()` fehlschlagen.

Siehe auch [@DIRECTORY](#) und [OpenDirectory\(\)](#).

**EINGABEN**

**id**            Identifikator des Verzeichnis oder **Nil** für die automatische ID-Auswahl

**dir\$**        Name des zu überwachenden Verzeichnisses

**table**       optional: Tabelle mit weiteren Parametern (siehe oben)

**RÜCKGABEWERTE**

**id**            optional: Identifikator des Verzeichnis; wird nur zurückgegeben werden, wenn Sie **Nil** als **id** angegeben haben (siehe oben)

**BEISPIEL**

```
InstallEventHandler({DirectoryChanged = Function(msg)
    NPrint(msg.action, msg.id, msg.directory)
EndFunction})
MonitorDirectory(1, "Data")
Repeat
    WaitEvent
Forever
```

Der obige Code überwacht alle Änderungen im Verzeichnis "Data" und gibt eine Meldung aus, wenn sich in diesem Verzeichnis etwas ändert.

## 20.45 MoveFile

**BEZEICHNUNG**

MoveFile – verschiebt eine Datei oder ein Verzeichnis (V7.1)

**ÜBERSICHT**

```
MoveFile(src$, dst$[, t])
```

**FRÜHERE SYNTAX**

```
MoveFile(src$, dst$[, func, userdata])
```

**BESCHREIBUNG**

Dieser Befehl verschiebt die in **src\$** angegebene Datei oder Verzeichnis in die in **dst\$** angegebene Datei oder Verzeichnis. Beachten Sie, dass **dst\$** nicht existieren darf oder **MoveFile()** wird fehlschlagen. Außerdem darf **src\$** nicht das Stammverzeichnis eines Laufwerkes sein, da dieses offensichtlich nirgendwohin verschoben werden kann.

Das Verschieben von Dateien (oder Verzeichnissen) auf dem gleichen Laufwerk ist wirklich schnell und braucht fast keine Zeit. Beim Verschieben von Dateien und Verzeichnissen von einem Laufwerk auf ein anderes muss **MoveFile()** zuerst die Dateien kopieren und in einem zweiten Schritt vom Original-Laufwerk löschen. Dieser Vorgang ist viel langsamer als das Verschieben auf dem gleichen Laufwerk. Deshalb können Sie eine Callback-Funktion angeben, die den Fortschritt dieser Operation überwacht.

**MoveFile()** unterstützt mehrere optionale Argumente. Vor Hollywood 9.0 mussten diese als optionale Parameter übergeben werden (siehe oben). Seit Hollywood 9.0 wird jedoch empfohlen, die neue Syntax zu verwenden, die ein einzelnes optionales Tabellenargument hat, mit dem ein oder mehrere optionale Argumente an **MoveFile()** übergeben werden können.

Die folgenden Tabellenfelder werden von diesem Befehl verstanden:

**Force:** Wenn dieser Tag auf **True** gesetzt ist, werden schreib- oder löschgeschützte Dateien automatisch gelöscht, ohne vorher die Callback-Funktion abzufragen. Beachten Sie, dass **MoveFile()** alle schreib- oder löschgeschützten Dateien überspringt, anstatt sie zu löschen, wenn es keine Callback-Funktion gibt und **Force** auf **False** (Voreinstellung) gesetzt ist. Beachten Sie auch, dass **Force** nur verwendet wird, wenn **MoveFile()** tatsächlich Dateien löschen muss, d.h. wenn Dateien von einem Laufwerk auf ein anderes verschoben werden. Das Verschieben von Dateien auf demselben Laufwerk erfordert kein Löschen. Die Voreinstellung ist **False**. (V9.0)

**BufferSize:** In diesem Tabellenfeld kann die Puffergröße eingestellt werden, die beim Kopieren von Dateien verwendet werden soll. Der hier übergebene Wert muss in Bytes angegeben werden. Voreingestellt ist 16384, also 16 Kilobytes. Beachten Sie, dass **BufferSize** nur verwendet wird, wenn **MoveFile()** tatsächlich Dateien kopieren muss, d.h. wenn Dateien von einem Laufwerk auf ein anderes verschoben werden. Das Verschieben von Dateien auf demselben Laufwerk beinhaltet kein Kopieren. (V9.0)

**FailOnError:** Standardmäßig schlägt **MoveFile()** fehl, wenn eine Datei oder ein Verzeichnis nicht kopiert oder gelöscht werden kann. Sie können dieses Verhalten ändern, indem Sie **FailOnError** auf **False** setzen. In diesem Fall schlägt **MoveFile()** nicht fehl, wenn eine Datei oder ein Verzeichnis nicht kopiert oder gelöscht werden kann. Stattdessen wird Ihre Callback-Funktion, falls vorhanden, mit den Meldungen **#MOVEFILE\_COPYFAILED** und **#MOVEFILE\_DELETEFAILED** benachrichtigt und Ihre Callback-Funktion muss **MoveFile()** mitteilen, wie es weitergehen soll (Wiederholen, Fortfahren, Abbrechen). Siehe unten, um zu erfahren, wie man eine Callback-Funktion für **MoveFile()** einrichtet. Beachten Sie, dass **FailOnError** nur verwendet wird, wenn **MoveFile()** tatsächlich Dateien kopieren und löschen muss, d.h. wenn Dateien von einem Laufwerk auf ein anderes verschoben werden. Das Verschieben von Dateien auf demselben Laufwerk erfordert kein Kopieren oder Löschen. **FailOnError** ist voreingestellt auf **True**. (V9.0)

**Async:** Wenn dies auf **True** gesetzt ist, arbeitet **MoveFile()** im asynchronen Modus. Das bedeutet, dass der Befehl sofort zurückkehrt und Ihnen ein asynchroner Operationshandler übergibt. Sie können dann diesen asynchronen Operationshandler verwenden, um den Vorgang abzuschließen, indem Sie wiederholt **ContinueAsyncOperation()** aufrufen, bis **True** zurückgegeben wird. Dies ist sehr nützlich, z.B. für die Anzeige eines Fortschrittsbalken oder ähnlichem. Indem Sie **MoveFile()** in den asynchronen Modus versetzen, ist es Ihrem Skript leicht möglich, während der Verarbeitung des Vorgangs etwas anderes zu tun. Siehe [Abschnitt 19.4 \[ContinueAsyncOperation\]](#), [Seite 234](#), für Details. Voreingestellt ist **False**. (V9.0)

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateisystemadapter angeben, die die Operation ausführen. Dies muss auf eine Zeichenkette gesetzt werden,

die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()` gesetzt wurde. Siehe [Abschnitt 7.9 \[Lader und Adapter\]](#), [Seite 96](#), für Details. (V10.0)

#### UserTags:

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an den Dateisystemadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

#### Callback:

Zur genaueren Steuerung des Löschvorgangs können Sie eine Callback-Funktion angeben, die bei verschiedenen Gelegenheiten aufgerufen wird. `MoveFile()` ruft sie beispielsweise von Zeit zu Zeit auf, damit Sie einen Fortschrittsbalken aktualisieren können. Sie wird auch aufgerufen, wenn eine Datei löschgeschützt ist, um Sie zu fragen, wie es weitergehen soll. Wenn keine Callback-Funktion vorhanden ist, überspringt `MoveFile()` löschgeschützte Dateien stillschweigend. Die Callback-Funktion erhält ein Argument, eine Tabelle, die weitere Informationen enthält.

Beachten Sie, dass die Callback-Funktion nur aufgerufen wird, wenn Dateien zwischen Laufwerken verschoben werden. Das Verschieben von Dateien auf demselben Laufwerk kann sofort erfolgen und führt nicht zu einem Aufruf der Callback-Funktion. Beachten Sie auch, dass Dateien mit Löschschutz nicht gelöscht werden, sondern nur an den neuen Speicherort kopiert werden, ohne die alte Datei zu löschen.

Die folgenden Callback-Typen sind verfügbar:

#### #MOVEFILE\_UNPROTECT:

Die Callback-Funktion vom Typ `#MOVEFILE_UNPROTECT` wird aufgerufen, wenn `MoveFile()` eine löschgeschützte Datei löschen muss. Die Parametertabelle für diesen Callback-Typ enthält die folgenden Felder:

**Action:** `#MOVEFILE_UNPROTECT`

**File:** Enthält den vollständigen Pfad der Datei, die löschgeschützt ist.

**UserData:**

Enthält den Wert, den Sie im Tabellenfeld `UserData` übergeben haben (siehe unten).

Diese Callback-Funktion muss `True` zurückgeben, wenn es in Ordnung ist, die Datei auf ungeschützt zu setzen, oder `False`, wenn sie nicht ungeschützt sein soll. Wenn Sie `-1` zurückgeben, wird der Verschiebevorgang komplett abgebrochen.

#### #MOVEFILE\_DELETE:

Dieser Callback wird immer dann ausgeführt, wenn eine Datei gelöscht wird. Die Callback-Funktion vom Typ `#MOVEFILE_DELETE` sollte normalerweise `False` zurückgeben. Wenn `True`

zurückgegeben wird, bricht `MoveFile()` den gesamten Vorgang ab.

**Action:** `#MOVEFILE_DELETE`

**File:** Enthält den vollständigen Pfad der Datei, die als nächstes gelöscht werden soll.

**UserData:**  
Enthält den Wert, den Sie im Tabellenfeld `UserData` angegeben haben (siehe unten).

#### **#MOVEFILE\_COPY:**

Dieser Callback wird aufgerufen, während `MoveFile()` Dateien kopiert. Die Callback-Funktion vom Typ `#MOVEFILE_COPY` sollte normalerweise `False` zurückgeben. Wenn `True` zurückgegeben wird, bricht `MoveFile()` den gesamten Vorgang ab.

**Action:** `#MOVEFILE_COPY`

**Source:** Enthält den vollständigen Pfad der Datei, die gerade kopiert wird (Quelle).

**Destination:**  
Enthält den vollständigen Pfad der Datei, die gerade kopiert wird (Ziel).

**Copied:** Enthält die Anzahl der Bytes, die bereits kopiert wurden.

**Filesize:**  
Enthält die Dateigröße der Quelldatei.

**UserData:**  
Enthält den Wert, den Sie im Tabellenfeld `UserData` angegeben haben (siehe unten).

#### **#MOVEFILE\_DELETEFAILED:**

Dieser Callback kann nur aufgerufen werden, wenn der Tag `FailOnError` auf `False` gesetzt wurde (siehe oben). In diesem Fall wird die Callback-Funktion vom Typ `#MOVEFILE_DELETEFAILED` immer dann aufgerufen, wenn ein Löschvorgang fehlgeschlagen ist. Es muss `True` zurückgegeben werden, um den gesamten Vorgang abubrechen, `False`, um fortzufahren, obwohl ein Fehler aufgetreten ist, oder `-1`, um den gerade fehlgeschlagenen Löschvorgang erneut zu versuchen. Die folgenden Felder werden im Tabellenparameter gesetzt, der an Ihre Callback-Funktion übergeben wird:

**Action:** `#MOVEFILE_DELETEFAILED`

**File:** Enthält den vollständigen Pfad der Datei, die nicht gelöscht werden konnte.

**UserData:**

Enthält den Wert, den Sie im Tabellenfeld **UserData** angegeben haben (siehe unten).

(V9.0)

**#MOVEFILE\_COPYFAILED:**

Dieser Callback kann nur aufgerufen werden, wenn der Tag **FailOnError** auf **False** gesetzt wurde (siehe oben). In diesem Fall wird die Callback-Funktion vom Typ **#MOVEFILE\_COPYFAILED** immer dann aufgerufen, wenn ein Kopiervorgang fehlgeschlagen ist. Es muss **True** zurückgegeben werden, um den gesamten Vorgang abubrechen, **False**, um fortzufahren, obwohl ein Fehler aufgetreten ist, oder -1, um den gerade fehlgeschlagenen Kopiervorgang erneut zu versuchen. Die folgenden Felder werden im Tabellenparameter gesetzt, der an Ihre Callback-Funktion übergeben wird:

**Action:** **#MOVEFILE\_COPYFAILED**

**Source:** Enthält den vollständigen Pfad der Datei, die gerade kopiert wird (Quelle).

**Destination:**

Enthält den vollständigen Pfad der Datei, die gerade geschrieben wird (Ziel).

**UserData:**

Enthält den Wert, den Sie im Tabellenfeld **UserData** angegeben haben (siehe unten).

(V9.0)

**UserData:**

Dieses Feld kann verwendet werden, um Ihrer Callback-Funktion einen beliebigen Wert zu übergeben. Der hier angegebene Wert wird bei jedem Aufruf an Ihre Callback-Funktion übergeben. Dies ist nützlich, wenn Sie vermeiden möchten, mit globalen Variablen zu arbeiten. Mit dem Tag **UserData** können Sie ganz einfach Daten an Ihre Callback-Funktion übergeben. Sie können in **UserData** einen beliebigen Wert angeben. Als Benutzerdaten können Zahlen, Zeichenketten, Tabellen und sogar Funktionen übergeben werden. Ihre Callback-Funktion erhält diese Daten im Feld **UserData** in der ihm übergebenen Tabelle.

Siehe auch [CopyFile\(\)](#), [DeleteFile\(\)](#), [Rename\(\)](#) und [Exists\(\)](#).

**EINGABEN**

<b>src\$</b>	die zu verschiebende Quelldatei oder -verzeichnis
<b>dst\$</b>	Zieldatei oder -verzeichnis; darf nicht vorhanden sein
<b>t</b>	optional: Tabelle mit zusätzlichen Optionen (siehe oben) (V9.0)

**BEISPIEL**

```
MoveFile("image.png", "images/image.png")
```

Verschiebt die Datei "image.png" unter Beibehaltung von ihrem Namens in das Unterverzeichnis "images".

**20.46 NextDirectoryEntry****BEZEICHNUNG**

NextDirectoryEntry – den nächsten Eintrag aus einem geöffneten Verzeichnis erhalten (V4.0)

**ÜBERSICHT**

```
t = NextDirectoryEntry(id)
```

**BESCHREIBUNG**

Dieser Befehl wird in **t** den nächsten Eintrag aus einem Verzeichnis zurückgeben, welches zuvor mit dem Befehl **OpenDirectory()** oder **@DIRECTORY** geöffnet wurde. Der Befehl liefert eine Tabelle, die detaillierte Informationen über den Eintrag enthält. Wenn sich keine weiteren Einträge im angegebenen Verzeichnis befinden, wird dieser Befehl **Nil** zurückgeben. Normalerweise wird dieser Befehl in einer Schleife aufgerufen, bis **Nil** zurückgegeben wird. Auf diese Weise können Sie den gesamten Inhalt eines Verzeichnisses scannen.

Bei der Tabelle **t**, die von diesem Befehl zurückgegeben wird, werden folgende Felder initialisiert:

- Name:** Dieses Feld beinhaltet den Namen des Eintrags.
- Type:** Falls der Eintrag eine Datei ist, wird hier **#DOSTYPE\_FILE** stehen, bei einem Verzeichnis **#DOSTYPE\_DIRECTORY**.
- Size:** Dieses Feld ist nur vorhanden, wenn der Eintrag eine Datei ist. Hier steht die Größe der Datei in Bytes.
- Flags:** Dieses Feld wird eine Kombination aus Schutzbits der Datei oder des Verzeichnisses enthalten. Siehe **Abschnitt 20.63 [Schutzbits Informationen]**, **Seite 305**, für Details.
- Time:** Dieses Feld wird eine Zeichenkette mit der Datei- oder Verzeichniszeit enthalten, wann sie zuletzt geändert wurde. Die Zeichenkette wird immer im Format **dd-mmm-jjjj hh: mm: ss** übergeben, Z. B . 08-Nov-2004 14:32:13.
- LastAccessTime:** Dieses Feld wird eine Zeichenkette mit der Datei- oder Verzeichniszeit enthalten, wann zuletzt auf die Datei/dem Verzeichnis zugegriffen wurde. Dieses Attribut wird nicht auf AmigaOS unterstützt.
- CreationTime:** Dieses Feld wird eine Zeichenkette mit der Zeit der Datei oder dem Verzeichnis enthalten, wann sie/es erstellt wurde. Dieses Attribut wird nur auf Windows unterstützt.



**Comment:** Dieses Feld wird den Kommentar einer Datei enthalten. Wird nur von den Amigaversionen unterstützt.

Verwenden Sie den Befehl `RewindDirectory()`, um eine Verzeichnis-Iteration zurückzusetzen. Siehe [Abschnitt 20.61 \[RewindDirectory\]](#), [Seite 301](#), für Details.

Siehe auch `GetDirectoryEntry()`.

## EINGABEN

`id` ID des geöffneten Verzeichnisses

## RÜCKGABEWERTE

`t` eine Tabelle, die wie oben gezeigt initialisiert wird

## BEISPIEL

Siehe [Abschnitt 20.47 \[OpenDirectory\]](#), [Seite 289](#).

# 20.47 OpenDirectory

## BEZEICHNUNG

`OpenDirectory` – öffnet ein Verzeichnis für die Überprüfung (V4.0)

## ÜBERSICHT

```
[id] = OpenDirectory(id, dir$[, table])
```

## BESCHREIBUNG

Dieser Befehl öffnet das in `dir$` angegebene Verzeichnis und ordnet ihm die angegebene `id` zu. Wenn Sie `Nil` in `id` angeben, wird `OpenDirectory()` automatisch eine freie ID auswählen und ihnen zurückgeben. Das Verzeichnis kann dann anschließend unter Verwendung des Befehls `NextDirectoryEntry()` auf weitere Unterverzeichnisse überprüft werden, womit Sie Lowlevel-Zugriff auf das Verzeichnis haben. Das ist besonders für große Verzeichnisse nützlich, oder wenn Sie zusätzliche Informationen wie Größe/Attribute für die einzelnen Verzeichniseinträge benötigen. Sie können das sehr schnell mit einer Schleife erreichen, wie es im Beispiel unten dargestellt ist.

Ab Hollywood 6.0 hat dieser Befehl ein optionales Argument `table`, das verwendet werden kann, um zusätzliche Parameter zu übergeben. Zur Zeit werden folgende Elemente erkannt:

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Verzeichnisadapter angeben, die das angegebene Verzeichnis öffnen sollen. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()` gesetzt wurde. Siehe [Abschnitt 7.9 \[Lader und Adapter\]](#), [Seite 96](#), für Details. (V6.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an den Verzeichnisadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Sie sollten den Befehl `CloseDirectory()` aufrufen, sobald Sie mit dem Verzeichnis fertig sind. Dies stellt sicher, dass das Verzeichnis nicht durch das Dateisystem länger gesperrt bleibt als nötig.

Dieser Befehl ist auch als Präprozessor-Anweisung verfügbar: Verwenden Sie `@DIRECTORY`, um ganze Verzeichnisse mit Ihrem Applet oder der ausführbaren Datei zu verknüpfen.

Siehe auch `MonitorDirectory()`.

#### EINGABEN

`id`            Identifikator des Verzeichnis oder `Nil` für die automatische ID-Auswahl  
`dir$`           Name des zu öffnenden Verzeichnis  
`table`        optional: Tabellenargument für weitere Parameter (V6.0)

#### RÜCKGABEWERTE

`id`            optional: Identifikator des Verzeichnis; wird nur zurückgegeben werden, wenn Sie `Nil` als `id` angegeben haben (siehe oben)

#### BEISPIEL

```
OpenDirectory(1, "Data")
e = NextDirectoryEntry(1)
While e <> Nil
  NPrint(If(e.type = #DOSTYPE_FILE, "File:", "Directory:"), e.name)
  e = NextDirectoryEntry(1)
Wend
CloseDirectory(1)
```

Der obige Code öffnet das Verzeichnis "Data" und gibt alle Dateien und Verzeichnisse aus.

## 20.48 OpenFile

#### BEZEICHNUNG

`OpenFile` – öffnet eine Datei zum Lesen und/oder Schreiben

#### ÜBERSICHT

```
[id] = OpenFile(id, filename$, mode, table)
```

#### BESCHREIBUNG

Dieser Befehl versucht, die durch `filename$` angegebene Datei zu öffnen und weist Ihr die `id` zu. Wenn Sie `Nil` als `id` übergeben, wird `OpenFile()` automatisch eine freie ID auswählen und ihnen zurückgeben. Falls die Datei nicht existiert, wird sie automatisch angelegt und Sie können sie beschreiben.

Alle Schreib- und Leseoperationen starten an der aktuellen Cursorposition innerhalb der Datei. Sie können die Position innerhalb der Datei mit Hilfe vom Befehl `Seek()` manuell verändern, aber sie wird auch von anderen Befehlen erhöht, die aus der Datei lesen oder hineinschreiben.

Ab Hollywood 2.0 können Sie das optionale Argument `mode` verwenden, um die Datei im Lese- (voreingestellt), im Schreibmodus oder im gemeinsamen Modus zu öffnen, was

bedeutet, dass Sie aus der Datei lesen und auch schreiben können. Wenn eine Datei im Lesemodus geöffnet wird, werden alle Schreibvorgänge fehlschlagen. Wenn eine Datei im Schreibmodus geöffnet wird, schlagen alle Leseoperationen fehl. Beachten Sie, dass der Befehl `OpenFile()` mit `#MODE_WRITE` immer eine leere Datei erstellt. Um eine bestehende Datei zu ändern, muss `#MODE_READWRITE` verwendet werden.

Ab Hollywood 6.0 hat dieser Befehl ein optionales Argument `table`, das verwendet werden kann, um zusätzliche Parameter zu übergeben. Folgende Elemente werden zur Zeit erkannt:

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen sollen. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehreren Adaptermodulen enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()` gesetzt wurde. Siehe [Abschnitt 7.9 \[Lader und Adapter\]](#), Seite 96, für Details. (V6.0)

**Encoding:** Falls die Datei eine Textdatei ist, können Sie diesen Tag auf den von der Datei verwendeten Zeichensatz setzen. Hollywood verarbeitet dann Zeichensatzkonvertierungen automatisch beim Lesen von der oder Schreiben in die Datei mit Befehlen wie `ReadLine()`, `ReadString()`, `WriteLine()` oder `WriteString()`. Standardmäßig erwartet Hollywood, dass Textdateien im UTF-8-Zeichensatz vorliegen, da dies der Standardzeichensatz von Hollywood ist. Wenn Sie stattdessen aus einer Datei mit der ISO 8859-1-Kodierung lesen oder in diese Datei schreiben möchten, setzen Sie einfach `Encoding` auf `#ENCODING_ISO8859_1` und Hollywood wird alle Konvertierungen von und nach ISO 8859-1 automatisch durchführen. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), Seite 1162, für eine Liste der verfügbaren Zeichensätze. (V9.0)

**WriteBOM:** Setzen Sie diesen Tag auf `True`, wenn `OpenFile()` die UTF-8-BOM (Byte Order Mark/Bytereihenfolge-Markierung) am Anfang der Datei hinzufügen soll. Offensichtlich macht `OpenFile()` dies nur, wenn die Datei im Schreibmodus (`#MODE_WRITE`) geöffnet wurde und die Codierung der Datei auf `#ENCODING_UTF8` gesetzt wurde. (V9.0)

**UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an den Dateiadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), Seite 100, für Details. (V10.0)

Obwohl Hollywood alle geöffneten Dateien automatisch beim Beenden schließt, wird dringend empfohlen, dass Sie eine geöffnete Datei schließen. Dies erledigen Sie mit dem Befehl `CloseFile()`, so dass diese Datei wieder für das Betriebssystem verfügbar wird.

Ab Hollywood 9.0 kann `filename$` auch eine der speziellen Konstanten `#STDIN`, `#STDOUT` und `#STDERR` sein. Dies ist nützlich für fortgeschrittene Programmierer, die auf die Dateistreams `stdin`, `stdout` und `stderr` zugreifen möchten, die jedem Programm zugeordnet sind.

Dieser Befehl steht auch als Präprozessor-Anweisung zur Verfügung: Verwenden Sie **@FILE**, um eine Datei für den späteren Gebrauch bereits zu öffnen.

Siehe auch **CloseFile()** und **Exists()**.

#### EINGABEN

**id**            Identifikator für die Datei oder **Nil** für die **automatische ID-Auswahl**

**filename\$**        Name der Datei, die geöffnet wird

**mode**            Modus, wie die Datei geöffnet wird: **#MODE\_READ** (nur lesen), **#MODE\_WRITE** (nur schreiben) oder **#MODE\_READWRITE** (lesen, schreiben und ändern) (voreingestellt ist **#MODE\_READ**) (V2.0)

**table**           optional: Tabellenargument für weitere Parameter (V6.0)

#### RÜCKGABEWERTE

**id**            optional: Identifikator der Datei; wird nur zurückgegeben werden, wenn Sie **Nil** als **id** angegeben haben (siehe oben)

#### BEISPIEL

```
OpenFile(1, "Highscores.txt")
While Not Eof(1) Do NPrint(ReadLine(1))
CloseFile(1)
```

Dieser Code öffnet die Datei "Highscores.txt", weist ihr den Identifikator 1 zu und gibt alle ihre Zeilen auf dem Bildschirm aus.

## 20.49 PathPart

#### BEZEICHNUNG

PathPart – gibt den Pfadnamen-Teil eines Pfades zurück

#### ÜBERSICHT

```
p$ = PathPart(path$)
```

#### BESCHREIBUNG

Dieser Befehl extrahiert den Pfadnamen aus einem durch **path\$** angegebenen Pfad und gibt ihn zurück. Der zurückgegebene Teil endet immer mit einem "/" oder ":", so dass Sie sofort einen Dateinamen anhängen können.

Siehe auch **FilePart()**, **FullPath()** und **ReadDirectory()**.

#### EINGABEN

**path\$**            Quellpfad

#### RÜCKGABEWERTE

**p\$**            Pfadnamenteil

#### BEISPIEL

```
p$ = PathPart("Data/Gfx/Test.jpg")
Print(p$)
```

Diese Zeile gibt "Data/Gfx" auf den Bildschirm aus.

## 20.50 ReadByte

### BEZEICHNUNG

ReadByte – liest ein Byte aus einer Datei (V7.0)

### ÜBERSICHT

```
b = ReadByte(id[, flags])
```

### BESCHREIBUNG

Dieser Befehl liest ein einzelnes Byte aus der in `id` angegebenen Datei und gibt sie zurück. Das Lesen beginnt mit der aktuellen Cursorposition, die Sie mit dem Befehl `Seek()` ändern können. Nach dem Lesen wird `ReadByte()` die Position des Dateicursors um ein Byte erhöhen.

Der Parameter `flags` kann auf eines der folgenden Flags gesetzt werden:

`#IO_UNSIGNED:`

Der Rückgabewert ist vorzeichenlos und reicht von 0 bis 255. Dies ist der Standardwert.

`#IO_SIGNED:`

Der Rückgabewert ist vorzeichenbehaftet und reicht von -128 bis 127.

Siehe auch `ReadBytes()`, `WriteByte()` und `WriteBytes()`.

### EINGABEN

`id`            Datei, aus der gelesen wird

`flags`        optional: zusätzliche Flags (siehe oben) (Voreingestellt ist `#IO_UNSIGNED`) (V9.0)

### RÜCKGABEWERTE

`b`            Byte, welches aus der Datei gelesen wurde

## 20.51 ReadBytes

### BEZEICHNUNG

ReadBytes – liest Bytes aus einer Datei (V7.0)

### ÜBERSICHT

```
data$ = ReadBytes(id[, len])
```

### BESCHREIBUNG

Dieser Befehl liest `len` Bytes aus der in `id` angegebenen Datei. Wenn das Argument `len` weggelassen wird, liest `ReadBytes()` alle Bytes ab der aktuellen Cursorposition bis zum Ende der Datei. `ReadBytes()` wird die Cursorposition der Datei durch die Anzahl der gelesenen Bytes erhöhen.

Dieser Befehl eignet sich zum Lesen von Binärdaten aus einer Datei. Zeichenketten von Hollywood können sowohl binäre Daten als auch Text speichern. Somit kann `ReadBytes()` einfach alle Bytes aus der Datei lesen und sie in einer Hollywood-Zeichenkette zurückgeben.

Siehe auch `ReadByte()`, `WriteByte()` und `WriteBytes()`.

**EINGABEN**

**id** Datei, aus der gelesen wird

**len** optional: Anzahl zu lesende Bytes (voreingestellt ist 0, womit bis zum Dateiende gelesen wird)

**RÜCKGABEWERTE**

**data\$** Daten, welche aus der Datei gelesen wurden

**BEISPIEL**

Siehe [Abschnitt 20.74 \[WriteBytes\]](#), Seite 313.

## 20.52 ReadChr

**BEZEICHNUNG**

`ReadChr` – liest ein Zeichen aus der angegebenen Datei

**ÜBERSICHT**

```
chr = ReadChr(id[, encoding])
```

**BESCHREIBUNG**

`ReadChr()` liest ein einzelnes Zeichen aus der in `id` angegebenen Datei und gibt ihren Codepunktwert zurück. Beachten Sie, dass je nach Zeichencodierung bis zu 4 Bytes aus der Datei gelesen werden können, da UTF-8 Zeichen bis zu 4 Bytes verwenden. Die Cursorposition des Cursors wird um die Anzahl der gelesenen Bytes erhöht.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Standardcodierung gesetzt werden. Dies ist standardmäßig auf die Standardtextcodierung eingestellt `SetDefaultEncoding()`. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), Seite 158, für Details.

Wenn Sie ein einzelnes Byte aus einer Datei lesen möchten, verwenden Sie stattdessen `ReadByte()`. Siehe [Abschnitt 20.50 \[ReadByte\]](#), Seite 293, für Details.

Siehe auch [WriteChr\(\)](#).

**EINGABEN**

**id** Nummer die angibt, von welcher Datei gelesen werden soll

**encoding** optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

**RÜCKGABEWERTE**

**chr** nächstes Zeichen der Datei

**BEISPIEL**

```
OpenFile(1, "test", #MODE_READWRITE)
WriteLine(1, "Hello People! How are you?")
Seek(1, 0)
test = ReadChr(1)
CloseFile(1)
test$ = Chr(test)
Print(test$)
```

Dieses Beispiel gibt "H" auf den Bildschirm aus.

## 20.53 ReadDirectory

### BEZEICHNUNG

ReadDirectory – liest ein Verzeichnis in ein Zeichenkettenfeld

### ÜBERSICHT

```
fcount, dcount = ReadDirectory(dir$, files$, dirs$[, sort])
```

### BESCHREIBUNG

Dieser Befehl untersucht das durch `dir$` angegebene Verzeichnis und schreibt alle im Verzeichnisbaum gefundenen Dateinamen in das in `files$` angegebene Feld und alle Verzeichnisnamen in das durch `dirs$` angegebene Feld. Nach dem letzten Element wird dieser Befehl eine leere Zeichenkette in das Feld einfügen, so dass Sie wissen, wie viele Dateien/Verzeichnisse gefunden wurden.

Standardmäßig werden alle Datei- und Verzeichniseinträge automatisch durch diesen Befehl sortiert. Wenn Sie das nicht wünschen, können Sie das optionale Argument `sort` auf `False` setzen.

Ab 2.0 gibt dieser Befehl zwei Werte zurück: Der erste Rückgabewert gibt an, wie viele Dateien im Verzeichnis sind und der zweite gibt an, wie viele Unterverzeichnisse im Verzeichnis sind.

Siehe auch `FilePart()`, `FullPath()` und `PathPart()`.

### EINGABEN

<code>dir\$</code>	zu untersuchendes Verzeichnis
<code>files\$</code>	Feld, das die Dateinamen enthalten soll
<code>dirs\$</code>	Feld, das die Verzeichnisnamen enthalten soll
<code>sort</code>	optional: mit <code>False</code> werden die Dateien und Verzeichnisse nicht sortiert (V4.5)

### RÜCKGABEWERTE

<code>fcount</code>	Anzahl Dateien in <code>dir\$</code> (V2.0)
<code>dcount</code>	Anzahl Unterverzeichnisse in <code>dir\$</code> (V2.0)

### BEISPIEL

```
f$ = {}  
d$ = {}  
ReadDirectory("Data", f$, d$)
```

Dieses Beispiel liest den Inhalt Ihres "Data"-Verzeichnisses in die Felder `f$` und `d$`.

## 20.54 ReadFloat

### BEZEICHNUNG

ReadFloat – liest eine Fließkommazahl aus einer Datei (V2.0)

### ÜBERSICHT

```
float = ReadFloat(id[, width, le])
```

**BESCHREIBUNG**

Dieser Befehl liest ein mit Vorzeichen versehene Fließkommazahl aus der in `id` angegebenen Datei und gibt sie in `float` zurück. Das Lesen beginnt an der aktuellen Cursorposition der Datei, die Sie mit dem Befehl `Seek()` ändern können. Eine Fließkommazahl nimmt 8 Byte in Anspruch, womit sich wirklich große Zahlen mit vielen Dezimalstellen bilden lassen.

Ab Hollywood 6.0 gibt es das optionale Argument `width`, womit Sie die Byte-Breite der Fließkommazahl angeben können. Dies kann 8 für eine Doppel-Präzision-Fließkommazahl oder 4 für eine Einfach-Präzision-Fließkommazahl sein. Standardmäßig liest `ReadFloat()` mit doppelter Genauigkeit.

Standardmäßig erwartet dieser Befehl die Daten im Big-Endian-Format (höherwertiges Byte zuerst). Ab Hollywood 6.0 können Sie das optionale Argument `le` verwenden, um mit diesem Befehl explizit das Little-Endian-Format zu verwenden.

Siehe auch `WriteFloat()`, `ReadInt()`, `WriteInt()`, `ReadShort()` und `WriteShort()`.

**EINGABEN**

<code>id</code>	Datei, aus der die Daten gelesen werden
<code>width</code>	optional: Byte-Breite der Fließkommazahl (voreingestellt ist 8) (V6.0)
<code>le</code>	optional: <code>True</code> um das Little-Endian-Format oder <code>False</code> , um das Big-Endian-Format zu verwenden (voreingestellt ist <code>False</code> ) (V6.0)

**RÜCKGABEWERTE**

<code>float</code>	Fließkommazahl
--------------------	----------------

## 20.55 ReadFunction

**BEZEICHNUNG**

`ReadFunction` – liest eine Funktion aus einer Datei (V4.0)

**ÜBERSICHT**

```
func = ReadFunction(id)
```

**BESCHREIBUNG**

Dieser Befehl liest eine Funktion aus der in `id` angegebene Datei und gibt sie zurück. Der Lesevorgang beginnt an der aktuellen Cursorposition der Datei, die Sie mit dem Befehl `Seek()` ändern können.

Die Funktion muss zuvor mit dem Befehl `WriteFunction()` in eine Datei geschrieben worden sein.

Siehe auch `Exists()`.

**EINGABEN**

<code>id</code>	die zu lesende Datei
-----------------	----------------------

**RÜCKGABEWERTE**

<code>func</code>	die Funktion, die aus der Datei gelesen wurde
-------------------	---



**BEISPIEL**

Siehe [Abschnitt 20.77 \[WriteFunction\]](#), Seite 315.

**20.56 ReadInt****BEZEICHNUNG**

ReadInt – liest eine Ganzzahl aus einer Datei (V2.0)

**ÜBERSICHT**

```
int = ReadInt(id[, flags])
```

**BESCHREIBUNG**

Dieser Befehl liest eine Ganzzahl aus der in `id` angegebenen Datei und gibt sie zurück. Der Lesevorgang beginnt an der aktuellen Cursorposition der Datei, die Sie mit dem Befehl [Seek\(\)](#) ändern können. Standardmäßig liest `ReadInt()` eine 32-Bit-Ganzzahl, wobei der Dateicursor um 4 Bytes vorrückt.

Der Parameter `flags` kann eine Kombination der folgenden Flags sein:

**#IO\_SIGNED:**

Der Rückgabewert wird mit Vorzeichen versehen und reicht von -2147483648 bis 2147483647 (wenn `#IO_FAKE64` nicht gesetzt ist). Dies ist die Voreinstellung.

**#IO\_UNSIGNED:**

Der Rückgabewert ist ohne Vorzeichen und reicht von 0 bis 4294967295. Dies kann nicht mit `#IO_FAKE64` kombiniert werden.

**#IO\_LITTLEENDIAN:**

Standardmäßig erwartet dieser Befehl, dass die Daten im Big-Endian-Format gespeichert werden (das höchstwertige Byte zuerst). Sie können dieses Flag setzen, um diesen Befehl zu veranlassen, stattdessen das Little-Endian-Format zu verwenden.

**#IO\_FAKE64:**

Verwendet 64-Bit-Ganzzahlen. Dies wird "Fake 64" genannt, weil Hollywood nicht den vollen 64-Bit-Integer-Bereich verwenden kann, da sein numerischer Typ ein 64-Bit-Fließkommawert ist, der nicht genau den gleichen Bereich wie ein echter 64-Bit-Integer-Wert darstellen kann. Dennoch sollten Hollywoods unechte 64-Bit-Ganzzahlen für fast alles groß genug sein. Mit `#IO_FAKE64` können Sie ganze Zahlen im Bereich von -9007199254740992 bis 9007199254740992 lesen. Beachten Sie, dass `#IO_UNSIGNED` nicht mit `#IO_FAKE64` verwendet werden kann. Hollywoods unechte 64-Bit-Ganzzahlen werden immer signiert, d.h. vorzeichenbehaftet sein. (V9.0)

Siehe auch [ReadFloat\(\)](#), [WriteFloat\(\)](#), [WriteInt\(\)](#), [ReadShort\(\)](#) und [WriteShort\(\)](#).

**EINGABEN**

<code>id</code>	Datei, aus der die Daten gelesen werden
<code>flags</code>	optional: zusätzliche Flags (siehe oben) (Voreingestellt ist <code>#IO_SIGNED</code> ) (V9.0)

**RÜCKGABEWERTE**

`int`           Ganzzahl

**20.57 ReadLine****BEZEICHNUNG**

`ReadLine` – liest eine Zeile aus der angegebenen Datei

**ÜBERSICHT**

```
string$ = ReadLine(id[, lf])
```

**BESCHREIBUNG**

Dieser Befehl liest Zeichen aus der durch `id` angegebenen Datei, bis ein Zeilenumbruch auftritt. Dieser Zeilenumbruch ist nicht in der gelieferten Zeichenkette enthalten. Der Befehl bricht ebenfalls ab, wenn er eine Dateiende-Markierung erreicht. Die gelesene Zeichenkette wird in `string$` zurückgegeben.

Ab Hollywood 9.0 gibt es ein optionales Argument `lf`. Wenn dies auf `True` gesetzt ist, enthält `ReadLine()` auch Zeilenumbruchzeichen (d.h. Zeilenvorschub und Wagenrücklauf), falls diese in der Quelldatei vorhanden sind. Beachten Sie, dass in diesem Fall keine Plattformanpassung stattfindet. Wenn `lf` auf `True` gesetzt ist, gibt `ReadString()` genau die Zeilenumbruchzeichen zurück, die sich in der Quelldatei befinden, d.h. Sie erhalten je nach Inhalt der Datei Zeilenvorschub, Wagenrücklauf und Zeilenvorschub oder nur Wagenrücklauf.

Siehe auch `FileLines()`, `WriteLine()`, `ReadString()`, `WriteString()`, `SetFileEncoding()` und `UseCarriageReturn()`.

**EINGABEN**

`id`           Identifikator der geöffneten Datei

`lf`           optional: ob Zeilenumbruchzeichen in die Zeichenkette aufgenommen werden sollen (Voreingestellt ist `False`) (V9.0)

**RÜCKGABEWERTE**

`string$`   enthält die gelesene Zeichenkette

**BEISPIEL**

Siehe [Abschnitt 20.48 \[OpenFile\]](#), Seite 290.

**20.58 ReadShort****BEZEICHNUNG**

`ReadShort` – liest eine 16-Bit-Ganzzahl aus einer Datei (V2.0)

**ÜBERSICHT**

```
short = ReadShort(id[, flags])
```

**BESCHREIBUNG**

Dieser Befehl liest eine 16-Bit-Ganzzahl aus der in `id` angegebenen Datei und gibt sie in `short` zurück. Der Lesevorgang beginnt an der aktuellen Cursorposition der Datei, die

Sie mit dem Befehl `Seek()` ändern können. Da `ReadShort()` eine 16-Bit-Ganzzahl aus der Datei liest, wird der Dateicursor um 2 Byte vorgerückt.

Der Parameter `flags` kann eine Kombination der folgenden Flags sein:

**#IO\_UNSIGNED:**

Der Rückgabewert ist ohne Vorzeichen und reicht von 0 bis 65535. Dies ist die Voreingestellung.

**#IO\_SIGNED:**

Der Rückgabewert wird signiert und reicht von -32768 bis 32767.

**#IO\_LITTLEENDIAN:**

Standardmäßig erwartet dieser Befehl, dass die Daten im Big-Endian-Format gespeichert werden (das höchstwertige Byte zuerst). Sie können dieses Flag setzen, um diesen Befehl zu veranlassen, stattdessen das Little-Endian-Format zu verwenden.

Siehe auch `ReadFloat()`, `WriteFloat()`, `ReadInt()`, `WriteInt()` und `WriteShort()`.

## EINGABEN

<code>id</code>	Datei, aus der die Daten gelesen werden
<code>flags</code>	optional: zusätzliche Flags (siehe oben) (Voreingestellt ist <code>#IO_UNSIGNED</code> ) (V9.0)

## RÜCKGABEWERTE

<code>short</code>	kurze Ganzzahl im Bereich von 0 bis 65535
--------------------	---

## 20.59 ReadString

### BEZEICHNUNG

`ReadString` – liest Bytes aus einer Datei

### ÜBERSICHT

```
s$ = ReadString(id[, length, encoding])
```

### BESCHREIBUNG

Dieser Befehl liest eine Zeichenfolge aus der in `id` angegebenen Datei. Mit dem optionalen Argument `length` können Sie die Anzahl der Zeichen angeben, die aus der Datei gelesen werden sollen. Wenn `length` weggelassen wird, werden alle Zeichen von der aktuellen Position des Dateicursors bis zum Ende gelesen und zurückgegeben. Der Datei-Cursor wird um die Anzahl der aus der Datei gelesenen Bytes verschoben. Dies ist nicht zwangsläufig die gleiche Anzahl wie Zeichen in `length`, da in UTF-8 ein einzelnes Zeichen bis zu 4 Bytes verwenden kann.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung gesetzt werden. Dies ist standardmäßig auf die Standard-Zeichencodierung eingestellt, die mit `SetDefaultEncoding()` gesetzt wurde. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Dieser Befehl dient zum Lesen von Text aus Dateien. Wenn Sie Binärdaten aus einer Datei lesen müssen, verwenden Sie stattdessen den Befehl `ReadBytes()`. Siehe [Abschnitt 20.51 \[ReadBytes\]](#), [Seite 293](#), für Details.

Siehe auch [FileLines\(\)](#), [ReadLine\(\)](#), [WriteLine\(\)](#), [WriteString\(\)](#), [SetFileEncoding\(\)](#) und [UseCarriageReturn\(\)](#).

#### EINGABEN

<code>id</code>	ID der Datei, aus der gelesen wird
<code>length</code>	optional: zu lesende Zeichen oder 0, um alle Zeichen bis zum Ende der Datei zu lesen (voreingestellt ist 0)
<code>encoding</code>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

#### RÜCKGABEWERTE

<code>s\$</code>	Zeichenkette, die die gelesenen Bytes enthält
------------------	---

## 20.60 Rename

#### BEZEICHNUNG

Rename – ändert den Namen einer Datei oder eines Verzeichnisses (V2.0)

#### ÜBERSICHT

```
Rename(oldname$, newname$[, t])
```

#### BESCHREIBUNG

Dieser Befehl benennt eine Datei oder ein Verzeichnis um. `oldname$` ist der Name der umzubenennenden Datei oder des Verzeichnisses und kann eine Pfadangabe enthalten. `newname$` ist nur der gewünschte neue Name für die Datei/dem Verzeichnis und darf keine Pfadangabe enthalten.

Ab Hollywood 10.0 akzeptiert dieser Befehl ein optionales Tabellenargument `t`, das die folgenden Tags unterstützt:

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateisystemadapter angeben, die die Operation ausführen. Dies muss auf eine Zeichenkette gesetzt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der Adapter verwendet, der mit [SetDefaultAdapter\(\)](#) gesetzt wurde. Siehe [Abschnitt 7.9 \[Lader und Adapter\]](#), [Seite 96](#), für Details. (V10.0)

**UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an den Dateiadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Siehe auch [CopyFile\(\)](#), [DeleteFile\(\)](#), [MoveFile\(\)](#) und [Exists\(\)](#).

#### EINGABEN

<code>oldname\$</code>	Datei oder Verzeichnis, welche/s umbenannt wird
<code>newname\$</code>	neuer Name für die Datei/dem Verzeichnis
<code>t</code>	optional: Tabellenargument mit weiteren Optionen (siehe oben) (V10.0)

**BEISPIEL**

```
Rename("image1.png", "image2.png")
```

Ändert den Namen "image1.png" in "image2.png" um.

**20.61 RewindDirectory****BEZEICHNUNG**

RewindDirectory – setzt die Verzeichnis-Iteration zurück (V8.0)

**ÜBERSICHT**

```
RewindDirectory(id)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die Iteration eines Verzeichnisses zurückzusetzen, das mit dem Befehl `OpenDirectory()` oder der Präprozessor-Anweisung `@DIRECTORY` geöffnet wurde. Dieses Verzeichnis muss im Argument `id` an `RewindDirectory()` übergeben werden und die Iteration muss zuvor mit dem Befehl `NextDirectoryEntry()` gestartet worden sein. Nachdem dieser Befehl beendet wurde, gibt ein Aufruf von `NextDirectoryEntry()` den ersten Eintrag im Verzeichnis erneut zurück.

Siehe [Abschnitt 20.46 \[NextDirectoryEntry\]](#), Seite 288, für Details.

Siehe auch `GetDirectoryEntry()`.

**EINGABEN**

`id`            Identifikator des Verzeichnisses, das zurückgespult werden soll

**20.62 Run****BEZEICHNUNG**

Run – führt eine Datei asynchron aus

**ÜBERSICHT**

```
Run(file$, args$, t)
```

**FRÜHERE SYNTAX**

```
Run(cmdline$, resetkeys, userdata)
```

**BESCHREIBUNG**

Dieser Befehl führt die in `file$` angegebene Datei asynchron aus und übergibt ihr die in `args$` angegebenen Argumente. Falls Sie ein Programm synchron ausführen möchten, müssen Sie den Befehl `Execute()` benutzen.

Wenn asynchrones Ausführen einer Datei vom Betriebssystem unterstützt wird, können Sie diesen Befehl fürs Anzeigen von Daten-Dateien wie Dokumente oder Bilder durch das Standardanzeigeprogramm verwenden. In diesem Fall kann `file$` auch eine nicht ausführbare Datei wie ein JPEG-Bild oder eine MP3-Datei sein.

Auf Android muss `file$` entweder eine Datendatei wie ein JPEG-Bild oder ein Paketname wie `com.airsoftsoftwair.hollywood` sein, wenn Sie mit diesen Befehl eine andere App starten wollen.

Wenn Sie informiert werden möchten, wenn das mit `Run()` gestartete Programm beendet ist, können Sie den Ereignis-Handler `RunFinished` mit dem Befehl `InstallEventHandler()` installieren. Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), [Seite 483](#), für Details.

Es gibt auch einen Ereignis-Handler namens `RunOutput`, der mit dem Befehl `InstallEventHandler()` installiert werden kann. Der Ereignis-Handler `RunOutput` leitet die Programmausgabe an Ihr Programm um, was beispielsweise beim Schreiben von GUIs für Konsolenprogramme nützlich ist. Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), [Seite 483](#), für Details.

Beachten Sie, dass es aus historischen Gründen einige Fallstricke bei der Verwendung von diesem Befehl gibt. Vor Hollywood 9.0 erwartete dieser Befehl Programm und Argumente, die in nur einer einzigen `cmdline$`-Zeichenkette kombiniert wurden. In diesem Fall ist beim Umgang mit Leerzeichen besondere Vorsicht geboten (Details siehe unten). Ab Hollywood 9.0 gibt es eine neue Syntax, die es Ihnen ermöglicht, Programm und Argumente als zwei separate Argumente zu übergeben, was die Sache viel einfacher macht. Um die Kompatibilität mit früheren Versionen aufrechtzuerhalten, kann diese neue Syntax jedoch nur verwendet werden, wenn Sie explizit eine Zeichenkette im zweiten Argument übergeben. Wenn Sie also die neue Syntax verwenden möchten, stellen Sie sicher, dass Sie im zweiten Argument eine Zeichenkette übergeben. Wenn das Programm, das Sie starten möchten, keine Argumente benötigt, übergeben Sie einfach eine leere Zeichenkette (`""`), um Hollywood zu signalisieren, dass Sie die neue Syntax verwenden möchten.

Wenn Sie im zweiten Argument keine Zeichenkette übergeben, wird die alte Syntax verwendet, was bedeutet, dass Sie bei der Übergabe von Programmpfaden, die Leerzeichen enthalten, sehr vorsichtig sein müssen, da das allererste Leerzeichen in `cmdline$` als Trennzeichen zwischen Programm und Argumenten interpretiert wird. Wenn Sie ein Programm starten möchten, dessen Pfadangabe Leerzeichen verwendet, müssen Sie diese Pfadangabe in doppelte Anführungszeichen setzen, sonst funktioniert es nicht. Sie können diese Komplikationen leicht vermeiden, indem Sie im zweiten Argument einfach eine Zeichenkette übergeben, auch wenn diese leer ist (siehe oben für Details).

Ab Hollywood 9.0 ist es möglich, das Programm und seine Argumente in zwei separaten Argumenten anzugeben, was die Sache wesentlich komfortabler macht. Außerdem gibt es jetzt ein neues optionales Tabellenargument, das verwendet werden kann, um weitere Optionen anzugeben.

Die folgenden Optionen werden derzeit vom optionalen Tabellenargument unterstützt:

**Directory:**

Mit diesem Tabellenargument können Sie das aktuelle Verzeichnis für das zu startende Programm festlegen. (V9.0)

**ResetKeys:**

Dieses Tabellenargument `resetkeys` ist nur interessant für fortgeschrittene Anwender. Wenn dies auf `False` gesetzt ist, wird `Run()` nicht alle internen Tastenzustände zurücksetzen, nachdem das Programm ausgeführt wurde. Standardmäßig werden alle Tastenzustände zurückgesetzt, wenn `Run()` beendet ist, weil Programme unter Verwendung von `Run()` oft den Tastaturfokus übernehmen, wird Hollywood möglicherweise nicht in der Lage sein, die internen Status-Flags zurückzusetzen. Aus diesem Grund werden standardmäßig mit `Run()` alle internen Tastenzustände zurückgesetzt, wenn die-

ser Befehl beendet ist. Es könnte dann Sinn machen, dieses Verhalten zu deaktivieren, wenn Sie `Run()` für den Start von Programmen verwenden, die nicht über eine GUI verfügen und nicht den Tastaturfokus wegnehmen. Voreingestellt ist `True`. (V5.1)

**RawMode:** Dieser Tag wird nur verwendet, wenn der Ereignis-Handler `RunOutput` aktiv ist. Standardmäßig erwartet der Ereignis-Handler `RunOutput`, dass Programme nur Text ausgeben. Aus diesem Grund stellt `RunOutput` sicher, dass nur korrekt UTF-8-codierter Text an Ihre Callback-Funktion übergeben wird. Wenn Sie nicht möchten, dass `RunOutput` den Text als UTF-8 formatiert, müssen Sie beim Aufruf von `Run()` das Argument `RawMode` auf `True` setzen. In diesem Fall führt `RunOutput` keine Vorformatierung durch und leitet nur die Rohausgabe des Programms an Sie weiter. Das bedeutet, dass Ihr Callback-Ereignis-Handler bereit sein muss, auch Binärdaten zu verarbeiten. Der Standardwert ist `False`. (V9.0)

**IgnoreHandlers:**

Wenn die Ereignis-Handler für `RunFinished` oder `RunOutput` installiert sind, werden diese Handler automatisch ausgelöst, wenn `Run()` aufgerufen wird. Wenn Sie möchten, dass diese Ereignis-Handler nur für bestimmte Aufrufe von `Run()` ausgelöst werden, können Sie diesen Tag verwenden, um `Run()` mitzuteilen, welche Ereignis-Handler ignoriert werden sollen. Dies muss auf eine Zeichenkette gesetzt werden, der die zu ignorierenden Ereignis-Handler enthält. Mehrere Ereignis-Handler müssen durch einen vertikalen Strich `'|'` getrennt werden. Zum Beispiel würde das Setzen von `IgnoreHandlers` auf `RunFinished|RunOutput` `Run()` anweisen, keine Ereignisse für die beiden Ereignis-Handler `RunFinished` und `RunOutput` auszulösen. (V9.0)

**ReturnCode:**

Wenn Sie einen Ereignis-Handler für `RunFinished` installiert haben, können Sie diesen Tag auf `True` setzen, um anzugeben, dass Ihr Ereignis-Handler beim Beenden auch den Rückgabecode des Programms erhalten soll. Beachten Sie, dass Hollywood beim Setzen von diesem Tag auf `True` unter AmigaOS 4 und MorphOS nicht beendet werden kann, bevor das mit `Run()` gestartete Programm beendet ist. Die Voreinstellung ist `False`. (V9.0)

**ForceExe:**

Wenn dieser Tag auf `True` gesetzt ist, behandelt `Run()` die in `file$` übergebene Datei immer als ausführbare Datei. Dies ist nur unter Linux und macOS nützlich, da auf diesen Plattformen Dateien mit einer Erweiterung als Datendateien behandelt werden, sodass Hollywood stattdessen versucht, das entsprechende Anzeigeprogramm für die Datendatei zu starten. Daher funktioniert der Versuch, `Run()` auf eine ausführbare Datei mit dem Namen "test.exe" anzuwenden, aufgrund der Erweiterung \*.exe unter Linux und macOS nicht. Indem Sie `ForceExe` auf `True` setzen, können Sie sie jedoch zum Laufen bringen. Die Voreinstellung ist `False`. (V9.0)

**UserData:**

Dieses Argument kann verwendet werden, um Benutzerdaten anzugeben, die an die Ereignis-Handler `RunFinished` und `RunOutput` übergeben werden sollen, die über den Befehl `InstallEventHandler()` installiert werden können. Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), [Seite 483](#), für Details. Die hier angegebenen Benutzerdaten können beliebigen Typs sein. (V6.1)

**Verb:** Unter Windows kann dies auf eine Zeichenkette gesetzt werden, der dem Befehl `Run()` mitteilt, was mit der Datei geschehen soll. Dies kann eines der folgenden Verben sein:

<code>edit</code>	Öffnet die angegebene Datei in einem Editor.
<code>explore</code>	Öffnet den angegebenen Ordner im Explorer. Wenn Sie dieses Verb verwenden, müssen Sie statt einer Datei einen Ordner an <code>Run()</code> übergeben.
<code>find</code>	Öffnet den Suchdialog für den angegebenen Ordner. Wenn Sie dieses Verb verwenden, müssen Sie statt einer Datei einen Ordner an <code>Run()</code> übergeben.
<code>open</code>	Öffnet die angegebene Datei.
<code>print</code>	Gibt die angegebene Datei aus.
<code>runas</code>	Startet die angegebene Datei im Administratormodus.

Beachten Sie, dass der Tag **Verb** nur unter Windows unterstützt wird. (V9.1)

Siehe auch `Execute()` und `Exists()`.

**EINGABEN**

<code>file\$</code>	das zu startende Programm (oder die Datei)
<code>args\$</code>	optional: Argumente, die an das Programm übergeben werden sollen; Beachten Sie, dass Sie diesen Parameter übergeben müssen, um Hollywood zu signalisieren, die neue Syntax zu verwenden. Sie können dies tun, indem Sie einfach eine leere Zeichenfolge ("" ) angeben; siehe oben für Details (V9.0)
<code>t</code>	optional: Tabelle mit weiteren Argumenten (siehe oben) (V9.0)

**BEISPIEL**

```
Run("Sys:Prefs/Locale")
```

Der obige Code führt das Einstellungsprogramm `Locale` auf AmigaOS aus. Ihr Skript wird sofort nach dem Ausführen des Programmes `Locale` fortgesetzt (asynchrone Ausführung).

```
Run("\"C:\\Program Files (x86)\\Hollywood\\ide.exe\"")
```

Der obige Code wird die Hollywood IDE auf Windows-Systemen starten. Beachten Sie, dass wir die Programmdekleration in doppelte Anführungszeichen eingebettet haben. Dies ist unbedingt notwendig, da das erste Leerzeichen in der Zeichenfolge normalerweise als Trennzeichen zwischen Programm und Argumente interpretiert wird. Wenn wir keine doppelte Anführungszeichen im obigen Code verwenden würden, würde



`Run()` versuchen, das Programm "C:\Program" zu starten und die Argumente "Files (x86)\Hollywood\ide.exe" zu übergeben, das wir offensichtlich nicht wollen. Beachten Sie, dass es seit Hollywood 9.0 jetzt viel einfacher ist, mit Leerzeichen in Pfaden umzugehen. Sie müssen nur die neue Syntax verwenden, die das Programm und seine Argumente in zwei separaten Argumenten verwendet. Mit Hollywood 9.0 können Sie einfach diesen Code verwenden:

```
Run("C:\\Program Files (x86)\\Hollywood\\ide.exe", "")
```

Beachten Sie, dass die Übergabe der leeren Zeichenkette im zweiten Argument hier unbedingt erforderlich ist, um Hollywood zu signalisieren, dass Sie die neue Syntax verwenden möchten. Ausführliche Informationen dazu finden Sie oben.

## 20.63 Schutzbits-Informationen

Mit den Befehlen `GetFileAttributes()`, `FileAttributes()` und `SetFileAttributes()` können Sie die Schutzbits einer Datei oder eines Verzeichnisses abfragen und einstellen. Die Bits werden in einem einzigen Tabellenelement namens `flags` übergeben. Dieses Tabellenelement enthält eine Bitmaske, die eine Kombination der aktiven Bits ist.

Die Schutzbits sind von dem Host-Dateisystem abhängig, da nicht alle unten aufgeführten Bits auf allen Plattformen unterstützt werden. In den eckigen Klammern stehen für das jeweilige Bit die unterstützte Plattform.

Die folgenden Bits werden von Hollywood erkannt:

<code>#FILEATTR_READ_USR</code>	[AmigaOS, macOS, Linux, iOS, Android]
<code>#FILEATTR_WRITE_USR</code>	[AmigaOS, macOS, Linux, iOS, Android]
<code>#FILEATTR_DELETE_USR</code>	[AmigaOS]
<code>#FILEATTR_EXECUTE_USR</code>	[AmigaOS, macOS, Linux, iOS, Android]
<code>#FILEATTR_READ_GRP</code>	[macOS, Linux, iOS, Android]
<code>#FILEATTR_WRITE_GRP</code>	[macOS, Linux, iOS, Android]
<code>#FILEATTR_EXECUTE_GRP</code>	[macOS, Linux, iOS, Android]
<code>#FILEATTR_READ_OTH</code>	[macOS, Linux, iOS, Android]
<code>#FILEATTR_WRITE_OTH</code>	[macOS, Linux, iOS, Android]
<code>#FILEATTR_EXECUTE_OTH</code>	[macOS, Linux, iOS, Android]
<code>#FILEATTR_PURE</code>	[AmigaOS]
<code>#FILEATTR_ARCHIVE</code>	[AmigaOS, Windows]
<code>#FILEATTR_SCRIPT</code>	[AmigaOS]
<code>#FILEATTR_HIDDEN</code>	[AmigaOS, Windows]
<code>#FILEATTR_SYSTEM</code>	[Windows]
<code>#FILEATTR_READONLY</code>	[Windows]

Um mehrere dieser Attribute für eine Datei festzulegen, kombinieren Sie sie einfach mit dem **Bitweise-Oder-Operator**. Beispielsweise:

```
t = {}
t.flags = #FILEATTR_READ_USR | #FILEATTR_WRITE_USR
SetFileAttributes("test.txt", t)
```

Der obige Code erteilt die Erlaubnis, die Datei "test.txt" zu lesen und in sie zu schreiben. Bitte beachten Sie jedoch, dass dieser Code unter Windows nicht korrekt funktionieren würde, weil Windows diese beiden Bits nicht kennt.

Um zu überprüfen, ob ein Bit gesetzt ist, verwenden Sie den **Bitweise-Und-Operator**. Beispielsweise:

```
t = GetFileAttributes("test.txt")
If (t.flags & #FILEATTR_READ_USR)
    Print("#FILEATTR_READ_USR is set.")
EndIf
```

Es gibt noch ein anderes Bit namens **#FILEATTR\_NORMAL**. Dieses Bit hat eine besondere Bedeutung und kann nur mit dem Befehl **SetFileAttributes()** gesetzt und nicht mit anderen Bits kombiniert verwendet werden. Wenn Sie **#FILEATTR\_NORMAL** mit **SetFileAttributes()** übergeben, werden die Schutzbits der Datei in die Betriebssystem Einstellungen zurückgesetzt, die sich von Plattform zu Plattform unterscheiden.

## 20.64 Seek

### BEZEICHNUNG

Seek – setzt den Dateicursor auf eine neue Position

### ÜBERSICHT

```
Seek(id, newpos[, mode])
```

### BESCHREIBUNG

Dieser Befehl setzt den Dateicursor (von dem aus alle Schreib- und Leseoperationen starten) auf die Position **newpos**. Der Dateianfang ist bei Position 0. Wenn Sie an das Dateieinde springen möchten, setzen Sie **newpos** auf die Konstante **#EOF**.

Um die Cursorposition einer bestimmten Datei zu finden, können Sie den Befehl **FilePos()** verwenden.

Ab Hollywood 6.0 können Sie das optionale Argument **mode** verwenden, um den Sprungmodus einzustellen. Dies kann eine der folgenden Modus-Konstanten sein:

#### #SEEK\_BEGINNING:

Die angegebene Sprungposition ist relativ zum Anfang der Datei. Negative Positionen sind nicht erlaubt. Dies ist der Standardmodus.

#### #SEEK\_CURRENT:

Die angegebene Sprungposition ist relativ auf die aktuelle Position des Dateicursors. Hier sind negativ Positionsangaben erlaubt, womit von der aktuellen Cursorposition der Datei rückwärts gesprungen wird.

#### #SEEK\_END:

Die angegebene Sprungposition ist relativ zum Dateieinde. Sie können hier nur 0 oder negative Positionen übergeben. Wenn Sie einfach ans Ende der Datei springen möchten, geben Sie hier 0 ein.

Siehe auch **Eof()**, **FilePos()**, **FileLength()** und **FileSize()**.

### EINGABEN

**id**            Identifikator der Datei

**newpos**      Position, auf die Sie den Dateicursor setzen wollen  
**mode**        optional: Sprungmodus (voreingestellt ist `#SEEK_BEGINNING`) (V6.0)

**BEISPIEL**

Siehe [Abschnitt 20.52 \[ReadChr\]](#), Seite 294.

## 20.65 SetEnv

**BEZEICHNUNG**

SetEnv – schreibt in die Umgebungsvariable (V5.0)

**ÜBERSICHT**

SetEnv(var\$, s\$)

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die Umgebungsvariable **var\$** auf den in **s\$** angegebenen Wert zu setzen. Bitte beachten Sie, dass die Umgebungsvariable in Ihrem Hollywood-Skript lokal sein wird. Sie können keine globale Umgebungsvariablen mit diesem Befehl ändern.

Siehe auch [GetEnv\(\)](#) und [UnsetEnv\(\)](#).

**EINGABEN**

**var\$**        Umgebungsvariable  
**s\$**          gewünschter Wert für die Umgebungsvariable

## 20.66 SetFileAttributes

**BEZEICHNUNG**

SetFileAttributes – setzt die Attribute einer Datei/eines Verzeichnisses (V3.0)

**ÜBERSICHT**

SetFileAttributes(f\$, t)

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um ein oder mehrere Attribute einer Datei oder eines Verzeichnisses zu ändern. Dazu gehören Informationen wie die Dateizeit, Schutzbits und vieles mehr, ist abhängig vom jeweiligen Host-Dateisystem.

Die Datei (oder das Verzeichnis), dessen Attribute Sie ändern wollen, müssen Sie im Attribut **f\$** übergeben. Das zweite Attribut **t** ist eine Tabelle, die alle Attribute enthält, die Sie ändern möchten. Die folgenden Tags können in der Tabelle eingestellt werden:

**Flags:**      Verwenden Sie diesen Tag, um die Schutzbits der Datei oder des Verzeichnisses zu ändern. Setzen Sie diesen Tag auf eine Kombination von Schutzbits oder auf `#FILEATTR_NORMAL`, um alle Schutzbits zurückzusetzen. Siehe [Abschnitt 20.63 \[Schutzbits\]](#), Seite 305, für Details.

**Time:**      Dieser Tag kann verwendet werden, um die Zeit zu ändern, wann die Datei zuletzt geändert wurde. Sie müssen diesen Tag in einer Zeichenkette im

folgenden Format einstellen: dd-mmm-yyyy hh:mm:ss. Z.B.: 08-Nov-2004 14:32:13.

**LastAccessTime:**

Verwenden Sie diesen Tag, um die Zeit der Datei oder dem Verzeichnis zu ändern, wann zuletzt darauf zugegriffen wurde. Die Zeichenfolge, die Sie hier angeben, muss im Format dd-mmm-yyyy und hh:mm:ss sein. Dieser Tag wird auf AmigaOS nicht unterstützt.

**CreationTime:**

Verwenden Sie diesen Tag, um die Erstellungszeit der Datei oder des Verzeichnisses zu ändern. Die Zeichenfolge, die Sie hier angeben, muss im Format dd-mmm-yyyy und hh:mm:ss sein. Dieser Tag wird nur auf Windows unterstützt.

**Comment:** Verwenden Sie diesen Tag, um den Kommentar einer Datei zu ändern. Dieser Tag wird nur bei den Amigaversionen unterstützt.

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateisystemadapter angeben, die die Operation ausführen. Dies muss auf eine Zeichenkette gesetzt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()` gesetzt wurde. Siehe [Abschnitt 7.9 \[Lader und Adapter\]](#), [Seite 96](#), für Details. (V10.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an den Dateisystemadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Siehe auch `GetFileAttributes()` und `FileAttributes()`.

## EINGABEN

**f\$** Name der Datei/des Verzeichnisses, dessen Attribute Sie ändern wollen  
**t** eine Tabelle mit den Attributen, die Sie ändern möchten

## BEISPIEL

```
t = {}
t.time = "15-Dec-2006 23:30:12"
t.flags = #FILEATTR_READ_USR | #FILEATTR_WRITE_USR
SetFileAttributes("test.txt", t)
```

Der obige Code setzt den Zeitstempel der Datei "test.txt" auf den 15. Dezember 2006 um 23.30 Uhr und 12 Sekunden. Außerdem setzt es die Schutzbits `#FILEATTR_READ_USR` und `#FILEATTR_WRITE_USR`.

## 20.67 SetFileEncoding

### BEZEICHNUNG

`SetFileEncoding` – setzt den Zeichensatz der Datei (V9.0)

**ÜBERSICHT**

`SetFileEncoding(id, encoding)`

**BESCHREIBUNG**

Dieser Befehl ändert den in `id` angegebene Datei verwendeten Zeichensatz in den durch `encoding` angegebenen. Alle nachfolgenden Lese- und Schreibvorgänge werden dann mit dem neuen Zeichensatz ausgeführt.

Dies darf nur verwendet werden, wenn die Datei eine Textdatei ist. In diesem Fall kann die Angabe des Zeichensatzes sehr praktisch sein, da Hollywood dann alle Zeichensatzkonvertierungen automatisch durchführt, wenn mit den Befehlen `ReadLine()`, `ReadString()`, `WriteLine()` oder `WriteString()` aus der Datei gelesen oder in sie geschrieben wird. Standardmäßig erwartet Hollywood, dass Textdateien im UTF-8-Zeichensatz vorliegen, da dies der Standard-Zeichensatz von Hollywood ist. Wenn Sie stattdessen mit der ISO 8859-1-Codierung aus einer Datei lesen oder in diese schreiben möchten, übergeben Sie einfach `#ENCODING_ISO8859_1` in `encoding` und Hollywood führt alle Konvertierungen von und nach ISO 8859-1 automatisch durch.

Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für eine Liste der verfügbaren Zeichensätze.

Siehe auch `FileLines()`, `ReadLine()`, `WriteLine()`, `ReadString()`, `WriteString()` und `UseCarriageReturn()`.

**EINGABEN**

`id`            Identifikator der Datei  
`encoding`    gewünschter neu zu verwendender Zeichensatz

**20.68 SetIOMode****BEZEICHNUNG**

`SetIOMode` – wechselt zwischen gepuffertes und ungepuffertes IO (V2.5)

**ÜBERSICHT**

`SetIOMode(mode)`

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um den IO-Modus anzugeben, den die Befehle Hollywood DOS-Bibliothek verwenden sollen. Standardmäßig verwenden alle DOS-Befehle gepuffertes IO. Dies ist besonders für kleine Lese- und Schreiboperationen effizient. In einigen Fällen ist jedoch gepuffertes IO nicht sehr bequem und Sie möchten stattdessen ungepuffertes IO verwenden. Zum Beispiel, wenn Sie mit der DOS-Bibliothek in das Parallel-Gerät schreiben oder Sie öffnen ein Konsolenfenster mit dem Befehl `OpenFile()`. In diesen Fällen ist ungepuffertes IO vorzuziehen, da die Daten direkt dem Dateisystem übergeben werden.

Der Modus, den Sie mit diesem Befehl einstellen, wird von allen Befehlen der DOS-Bibliothek akzeptiert werden. Aber bitte beachten Sie, dass wenn Sie zwischen gepuffertes und ungepuffertes IO, bei der gleichen Datei wechseln, müssen Sie den Befehl `FlushFile()` benutzen, um alle ausstehenden Puffer zu leeren. Wenn Sie das vergessen, können die Daten an der falschen Positionen in der Datei landen.

Dieser Befehl ist für fortgeschrittene Anwender gedacht. Normalerweise müssen Sie sich nicht um den IO-Modus sorgen.

#### EINGABEN

**mode** gewünschter IO Modus für die DOS-Bibliothek; dies kann entweder `#IO_BUFFERED` oder `#IO_UNBUFFERED` sein (voreingestellt ist `#IO_BUFFERED`)

## 20.69 StringToFile

#### BEZEICHNUNG

`StringToFile` – speichert die Zeichenkette in einer Datei (V5.0)

#### ÜBERSICHT

`StringToFile(s$, file$[, t])`

#### BESCHREIBUNG

Dieser Befehl ist ein Komfortbefehl, der einfach die Zeichenkette im Argument `s$` in die Datei `file$` speichert. Seien Sie gewarnt, dass dieser Befehl die Zeichenkette nicht an die Datei anhängt. Wenn die Datei im Argument `file$` bereits vorhanden ist, wird sie ohne eine Warnung überschrieben. Beachten Sie, dass Sie diesen Befehl auch verwenden können, um Zeichenkette mit Rohdaten in Dateien zu schreiben, da Hollywood-Zeichenketten auch Binärdaten enthalten können.

Ab Hollywood 10.0 akzeptiert `StringToFile()` ein optionales Tabellenargument, mit dem Sie zusätzliche Argumente an den Befehl übergeben können. Die folgenden Tags werden derzeit vom optionalen Tabellenargument unterstützt:

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateisystemadapter angeben, die die Operation ausführen. Dies muss auf eine Zeichenkette gesetzt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()` gesetzt wurde. Siehe [Abschnitt 7.9 \[Lader und Adapter\]](#), [Seite 96](#), für Details. (V10.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Dateisystemadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Siehe auch [FileToString\(\)](#).

#### EINGABEN

**s\$** Zeichenkette, die in die Datei geschrieben wird

**file\$** Zieldatei

**t** optional: Tabelle mit weiteren Optionen (siehe oben)

## 20.70 `UndefinedVirtualStringFile`

### BEZEICHNUNG

`UndefinedVirtualStringFile` – entfernt die Definition einer virtuellen Datei aus einer Zeichenfolge (V5.0)

### ÜBERSICHT

`UndefinedVirtualStringFile(virtfile$)`

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um eine virtuelle Datei zu entfernen, die mit dem Befehl `DefineVirtualFileFromString()` erstellt wurde. Es ist wichtig, diesen Befehl aufzurufen, wenn Sie mit einer virtuellen Datei fertig sind, da sie den von der virtuellen Datei belegten Arbeitsspeicher freigibt.

Siehe [Abschnitt 20.11 \[DefineVirtualFileFromString\]](#), Seite 249, für mehr Informationen über virtuelle Dateien aus Zeichenketten.

Siehe auch `DefineVirtualFile()`, `DefineVirtualFileFromString()` und `GetTempFileName()`.

### EINGABEN

`virtfile$`

eine virtuelle Datei, welche mit `DefineVirtualFileFromString()` erstellt wurde

### BEISPIEL

Siehe [Abschnitt 20.11 \[DefineVirtualFileFromString\]](#), Seite 249.

## 20.71 `UnsetEnv`

### BEZEICHNUNG

`UnsetEnv` – löscht eine Umgebungsvariable (V5.0)

### ÜBERSICHT

`UnsetEnv(var$)`

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die angegebenen Umgebungsvariablen zu löschen. Bitte beachten Sie, dass Sie globale Umgebungsvariablen nicht mit diesem Befehl löschen können. Sie können nur Umgebungsvariablen löschen, die auch in Ihrem Hollywood-Skript lokal vorhanden sind.

Siehe auch `GetEnv()` und `SetEnv()`.

### EINGABEN

`var$` die zu löschende Umgebungsvariable

## 20.72 `UseCarriageReturn`

### BEZEICHNUNG

`UseCarriageReturn` – konfiguriert den Zeilenumbruch (V7.1)

**ÜBERSICHT**

`UseCarriageReturn(enable)`

**BESCHREIBUNG**

Mit diesem Befehl können Sie einstellen, ob `WriteLine()` ein Wagenrücklaufzeichen vor das Zeilenvorschubzeichen schreibt oder nicht. MS-DOS und sein Nachfolger Windows verwenden sowohl Wagenrücklauf- als auch Zeilenvorschubzeichen, um einen Zeilenumbruch anzuzeigen. Unix, Amiga und macOS erzwingen nur mit einem Zeilenvorschubzeichen einen Zeilenumbruch.

Wenn `enable` auf `True` gesetzt ist, schreibt `WriteLine()` vor jedem Zeilenvorschub ein Wagenrücklaufzeichen, ansonsten wird nur ein Zeilenvorschubzeichen benutzt. Standardmäßig wird `UseCarriageReturn()` auf Windows-Systemen auf `True` und auf allen anderen Systemen auf `False` gesetzt.

Siehe auch `FileLines()`, `ReadLine()`, `WriteLine()`, `ReadString()`, `WriteString()` und `SetFileEncoding()`.

**EINGABEN**

`enable`      `True` oder `False` geben an, ob `WriteLine()` einen Wagenrücklauf vor einem Zeilenvorschubzeichen ausgeben soll oder nicht.

## 20.73 WriteByte

**BEZEICHNUNG**

`WriteByte` – schreibt ein Byte in eine Datei (V7.0)

**ÜBERSICHT**

`WriteByte(id, b[, flags])`

**BESCHREIBUNG**

Dieser Befehl schreibt ein einzelnes Byte in die in `id` angegebene Datei an der aktuellen Cursorposition, die Sie mit dem Befehl `Seek()` ändern können. `WriteByte()` wird nach dem Schreibvorgang die Cursorposition der Datei um ein Byte erhöhen.

Der Parameter `flags` kann auf eines der folgenden Flags gesetzt werden:

**#IO\_UNSIGNED:**

Schreibt ein Byte ohne Vorzeichen in die Datei. Das bedeutet, dass `b` zwischen 0 und 255 liegen muss. Dies ist die Voreinstellung.

**#IO\_SIGNED:**

Schreibt ein vorzeichenbehaftetes Byte in die Datei. Das bedeutet, dass `b` zwischen -128 und 127 liegen muss.

Siehe auch `ReadByte()`, `ReadBytes()` und `WriteBytes()`.

**EINGABEN**

`id`            Datei, in die geschrieben wird  
`b`             Byte-Daten, die geschrieben werden  
`flags`        optional: zusätzliche Flags (siehe oben) (Voreingestellt ist `#IO_UNSIGNED`) (V9.0)



## 20.74 WriteBytes

### BEZEICHNUNG

WriteBytes – schreibt Bytes in eine Datei (V7.0)

### ÜBERSICHT

```
WriteBytes(id, data$[, len])
```

### BESCHREIBUNG

Dieser Befehl schreibt `len` Bytes aus der Zeichenkette `data$` in die in `id` angegebene Datei. Wenn das optionale Argument `len` weggelassen wird, schreibt dieser Befehl die komplette Zeichenkette in die Datei. `WriteBytes()` wird die Cursorposition der Datei um die Anzahl geschriebenen Bytes erhöhen.

Dieser Befehl ist nützlich, um binäre Daten in eine Datei zu schreiben. Die von `data$` angegebene Zeichenfolge wird als rohe Binärdaten anstatt als Text behandelt.

Siehe auch `ReadByte()`, `ReadBytes()` und `WriteByte()`.

### EINGABEN

<code>id</code>	Datei, in die geschrieben wird
<code>data\$</code>	Daten, die in die Datei geschrieben werden
<code>len</code>	optional: Anzahl zu schreibende Bytes (voreingestellt ist 0, womit die gesamte Zeichenkette in die Datei geschrieben wird)

### BEISPIEL

```
size = FileSize("test")
OpenFile(1, "test")
OpenFile(2, "copy_of_test", #MODE_WRITE)
data$ = ReadBytes(1, size)
WriteBytes(2, data$, size)
CloseFile(2)
CloseFile(1)
```

Dieser Code erstellt eine Kopie von der Datei "test" und speichert sie als "copy\_of\_test".

## 20.75 WriteChr

### BEZEICHNUNG

WriteChr – schreibt ein Zeichen in eine Datei

### ÜBERSICHT

```
WriteChr(id, chr[, encoding])
```

### BESCHREIBUNG

Schreibt das durch `chr` beschriebene Zeichen in die durch `id` angegebene Datei und erhöht die Cursorposition der Datei um die geschriebenen Bytes. Das zu schreibende Zeichen muss als Codepunktwert an `WriteChr()` übergeben werden. Beachten Sie, dass dieser Befehl je nach Codierung bis zu 4 Bytes in die Datei schreiben kann, da in UTF-8 für ein einzelnes Zeichen bis zu 4 Bytes verwendet werden kann.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung gesetzt werden. Dies ist standardmäßig auf die Standard-Zeichencodierung eingestellt, die mit `SetDefaultEncoding()` gesetzt wurde. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Wenn Sie ein einzelnes Byte in eine Datei schreiben müssen, verwenden Sie stattdessen den Befehl `WriteByte()`. Siehe [Abschnitt 20.73 \[WriteByte\]](#), [Seite 312](#), für Details.

Siehe auch `ReadChr()`.

#### EINGABEN

<code>id</code>	ID der zu benutzenden Datei
<code>chr</code>	Code-Punktwert des Zeichens, das in die Datei geschrieben werden soll
<code>encoding</code>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

#### BEISPIEL

```
OpenFile(1, "Test", #MODE_WRITE)
WriteChr(1, 65)
CloseFile()
```

Dieser Code schreibt das Zeichen "A" in die Datei Ram:Test.

## 20.76 WriteFloat

#### BEZEICHNUNG

`WriteFloat` – schreibt eine Fließkommazahl in eine Datei (V2.0)

#### ÜBERSICHT

```
WriteFloat(id, float[, width, le])
```

#### BESCHREIBUNG

Dieser Befehl schreibt die Fließkommazahl `float` mit Vorzeichen in die in `id` angegebene Datei an der aktuellen Cursorposition der Datei, die Sie mit dem Befehl `Seek()` ändern können. Eine Fließkommazahl nimmt 8 Byte in Anspruch, womit sich wirklich große Zahlen mit vielen Dezimalstellen bilden lassen.

Ab Hollywood 6.0 gibt es das optionale Argument `width`, womit Sie die Byte-Breite der Fließkommazahl angeben können. Dies kann 8 für eine Doppel-Präzision-Fließkommazahl oder 4 für eine Einfach-Präzision-Fließkommazahl sein. Standardmäßig schreibt `WriteFloat()` mit doppelter Genauigkeit.

Standardmäßig erwartet dieser Befehl die Daten im Big-Endian-Format (höherwertiges Byte zuerst). Ab Hollywood 6.0 können Sie das optionale Argument `le` verwenden, um mit diesem Befehl explizit das Little-Endian-Format zu verwenden.

Siehe auch `ReadFloat()`, `ReadInt()`, `WriteInt()`, `ReadShort()` und `WriteShort()`.

#### EINGABEN

<code>id</code>	Datei, in die geschrieben wird
<code>float</code>	Fließkommazahl, die in die Datei geschrieben wird

**width** optional: Byte-Breite der Fließkommazahl (voreingestellt ist 8) (V6.0)

**le** optional: **True** um das Little-Endian-Format oder **False**, um das Big-Endian-Format zu verwenden (voreingestellt ist **False**) (V6.0)

## 20.77 WriteFunction

### BEZEICHNUNG

WriteFunction – schreibt eine Funktion in eine Datei (V4.0)

### ÜBERSICHT

WriteFunction(id, func[, txtmode, nobrk])

### BESCHREIBUNG

Dieser Befehl schreibt die in **func** angegebene Funktion in die in **id** angegebene Datei an der aktuellen Cursorposition der Datei, die Sie mit dem Befehl **Seek()** ändern können. Die Funktion wird in die Datei als vorkompilierter Bytecode geschrieben werden, das heißt, für Menschen ist er nicht lesbar.

Sie können mit Hilfe des Befehls **ReadFunction()** gespeicherte Funktionen in andere Projekte laden. Das optionale Argument **txtmode** gibt an, ob die Funktion als binäre oder als Base64-codierte Daten in die Datei geschrieben wird. Letzteres ist nützlich für die Einbettung von Funktionen in für Menschen lesbare Text-Dateien wie zum Beispiel XML-Dateien. Für den Fall setzen Sie **txtmode** auf **True** und **WriteFunction()** wird dann für eine bessere Lesbarkeit jeweils nach 72 Zeichen automatisch einen Zeilenumbruch einfügen. Wenn Sie diesen Zeilenumbruch nicht wollen, setzen Sie das optionale Argument **nobrk** auf **True**. In diesem Fall werden keine Zeilenumbrüche eingefügt.

### EINGABEN

**id** Datei, in der geschrieben wird

**func** Funktion, die in die Datei geschrieben wird

**txtmode** optional: **True**, um die Funktion in Base64 oder **False**, um als Binärdaten zu schreiben (voreingestellt ist **False**)

**nobrk** optional: **True**, wenn Sie keine Zeilenumbrüche in für Menschen lesbaren Text wollen (voreingestellt ist **False**); dieses Argument wird bei Binärdaten ignoriert (V6.1)

### BEISPIEL

```
Function p_LittleTestFunc(a, b)
    Return(a+b)
EndFunction

OpenFile(1, "func.bin", #MODE_WRITE)
WriteFunction(1, p_LittleTestFunc)
CloseFile(1)

OpenFile(1, "func.bin", #MODE_READ)
p_MyAdd = ReadFunction(1)
```

```
CloseFile(1)
```

```
Print(p_MyAdd(5, 6)) ; gibt 11 aus
```

Der obige Code schreibt die Funktion `p_LittleTestFunc()` in die Datei "func.bin". Danach öffnet sich die Datei "func.bin" wieder und es wird die Funktion in Hollywood zurück gelesen. Die eingelesene Funktion wird der Variablen `p_MyAdd()` zugewiesen. schließlich wird die neue Funktion `p_MyAdd()` aufgerufen und es werden die Nummern 5 und 6 für uns summiert.

## 20.78 WriteInt

### BEZEICHNUNG

`WriteInt` – schreibt eine Ganzzahl in eine Datei (V2.0)

### ÜBERSICHT

```
WriteInt(id, int[, flags])
```

### BESCHREIBUNG

Dieser Befehl schreibt die Ganzzahl `int` in die in `id` angegebene Datei an der aktuellen Cursorposition der Datei, die Sie mit dem Befehl `Seek()` ändern können. Standardmäßig schreibt `WriteInt()` eine 32-Bit-Ganzzahl, die den Dateicursor um 4 Byte vorrückt.

Der Parameter `flags` kann eine Kombination der folgenden Flags sein:

#### #IO\_SIGNED:

Verwendet ganze Zahlen mit Vorzeichen. Das bedeutet, dass `int` im Bereich von -2147483648 bis 2147483647 liegen muss (falls `#IO_FAKE64` nicht gesetzt ist). Dies ist die Voreingstellung.

#### #IO\_UNSIGNED:

Verwendet ganze Zahlen ohne Vorzeichen. Dies bedeutet, dass `int` im Bereich von 0 bis 4294967295 liegen muss. Beachten Sie, dass `#IO_UNSIGNED` nicht mit `#IO_FAKE64` kombiniert werden kann.

#### #IO\_LITTLEENDIAN:

Standardmäßig erwartet dieser Befehl die Daten im Big-Endian-Format (höherwertiges Byte zuerst). Sie können dieses Flag setzen, um den Befehl aufzufordern, stattdessen das Little-Endian-Format zu verwenden.

#### #IO\_FAKE64:

Verwendet 64-Bit-Ganzzahlen. Dies wird "Fake 64" genannt, weil Hollywood nicht den vollen 64-Bit-Integer-Bereich verwenden kann, da sein numerischer Typ ein 64-Bit-Fließkommawert ist, der nicht genau den gleichen Bereich wie ein echter 64-Bit-Integer-Wert darstellen kann. Dennoch sollten Hollywoods unechte 64-Bit-Ganzzahlen für fast alles groß genug sein. Mit `#IO_FAKE64` können Sie Ganzzahlen im Bereich von -9007199254740992 bis 9007199254740992 schreiben. Beachten Sie, dass `#IO_UNSIGNED` nicht mit `#IO_FAKE64` verwendet werden kann. Hollywoods unechte 64-Bit-Ganzzahlen werden immer signiert, d.h. vorzeichenbehaftet sein. (V9.0)

Siehe auch `ReadFloat()`, `WriteFloat()`, `ReadInt()`, `ReadShort()` und `WriteShort()`.

**EINGABEN**

<code>id</code>	Datei, in die geschrieben wird
<code>int</code>	Ganzzahlenwert, der in die Datei geschrieben wird
<code>flags</code>	optional: zusätzliche Flags (siehe oben) (Standardwert ist <code>#IO_SIGNED</code> ) (V9.0)

**20.79 WriteLine****BEZEICHNUNG**

WriteLine – schreibt eine neue Zeile in eine Datei

**ÜBERSICHT**

`WriteLine(id, line$)`

**BESCHREIBUNG**

Schreibt die durch `line$` angegebene Zeile in die durch `id` angegebene Datei und erhöht den Dateicursor entsprechend.

Beachten Sie, dass `WriteLine()` auf Windows-Systemen sowohl ein Wagenrücklauf- als auch ein Zeilenvorschubzeichen an `line$` anfügt, während auf allen anderen Systemen nur das Zeilenvorschubzeichen an `line$` angehängt wird. Dieses Verhalten kann durch den Befehl `UseCarriageReturn()` geändert werden. Siehe [Abschnitt 20.72 \[UseCarriageReturn\]](#), [Seite 311](#), für Details.

Siehe auch `FileLines()`, `ReadLine()`, `ReadString()`, `WriteString()`, `SetFileEncoding()` und `UseCarriageReturn()`.

**EINGABEN**

<code>id</code>	ID der zu benutzenden Datei
<code>line\$</code>	Zeichenkette, die in die Datei geschrieben werden soll

**BEISPIEL**

Siehe [Abschnitt 20.52 \[ReadChr\]](#), [Seite 294](#).

**20.80 WriteShort****BEZEICHNUNG**

WriteShort – schreibt eine 16-Bit Ganzzahl in eine Datei (V2.0)

**ÜBERSICHT**

`WriteShort(id, short[, flags])`

**BESCHREIBUNG**

Dieser Befehl schreibt eine 16-Bit Ganzzahl in die in `id` angegebene Datei an der aktuellen Cursorposition, die Sie mit dem Befehl `Seek()` ändern können. Da eine kurze Ganzzahl 16 Bit lang ist, wird der Dateicursor um 2 Byte vorwärts bewegt.

Der Parameter **flags** kann eine Kombination der folgenden Flags sein:

**#IO\_UNSIGNED:**

Verwendet eine Ganzzahl ohne Vorzeichen. Das bedeutet, dass **short** im Bereich von 0 bis 65535 liegen muss. Dies ist die Voreinstellung.

**#IO\_SIGNED:**

Verwendet eine Ganzzahl mit Vorzeichen. Das bedeutet, dass **short** im Bereich von -32768 bis 32767 liegen muss.

**#IO\_LITTLEENDIAN:**

Standardmäßig erwartet dieser Befehl die Daten im Big-Endian-Format (höherwertiges Byte zuerst). Sie können dieses Flag setzen, um den Befehl aufzufordern, stattdessen das Little-Endian-Format zu verwenden.

Siehe auch [ReadFloat\(\)](#), [WriteFloat\(\)](#), [ReadInt\(\)](#), [WriteInt\(\)](#), und [ReadShort\(\)](#).

## EINGABEN

<b>id</b>	ID der Datei, in die geschrieben wird
<b>short</b>	kurze Ganzzahlenwert, der in die Datei geschrieben wird
<b>flags</b>	optional: zusätzliche Flags (siehe oben) (Standardwert ist <b>#IO_UNSIGNED</b> ) (V9.0)

## 20.81 WriteString

### BEZEICHNUNG

WriteString – schreibt eine Zeichenkette in eine Datei

### ÜBERSICHT

WriteString(id, s\$, len, encoding)

### BESCHREIBUNG

Dieser Befehl schreibt die Zeichenfolge **s\$** in die in **id** angegebene Datei. Das optionale Argument **len** kann verwendet werden, um die Anzahl der Zeichen festzulegen, die in die Datei geschrieben werden sollen. Wenn **len** weggelassen wird, wird die komplette Zeichenkette geschrieben. Die Cursorposition wird durch die Anzahl der in die Datei geschriebenen Bytes verschoben. Beachten Sie, dass dies nicht unbedingt der gleiche Wert wie in **len** ist, da bei UTF-8-Codierung ein einzelnes Zeichen bis zu 4 Bytes verwenden kann.

Mit dem optionalen Parameter **encoding** kann die zu verwendende Zeichencodierung gesetzt werden. Dies ist standardmäßig auf die Standard-Zeichencodierung eingestellt, die mit [SetDefaultEncoding\(\)](#) gesetzt wurde. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Dieser Befehl dient zum Schreiben von Text in Dateien. Wenn Sie binäre Daten in eine Datei schreiben müssen, verwenden Sie statt dessen den Befehl [WriteBytes\(\)](#). Siehe [Abschnitt 20.74 \[WriteBytes\]](#), [Seite 313](#), für Details.

Siehe auch [FileLines\(\)](#), [ReadLine\(\)](#), [WriteLine\(\)](#), [ReadString\(\)](#), [SetFileEncoding\(\)](#) und [UseCarriageReturn\(\)](#).

**EINGABEN**

<b>id</b>	ID der Datei, in der geschrieben wird
<b>s\$</b>	Zeichenkette, die in die Datei geschrieben werden soll
<b>len</b>	optional: Anzahl Zeichen, die in die Datei geschrieben werden oder 0, dann wird die ganze Zeichenkette in die Datei geschrieben (voreingestellt ist 0)
<b>encoding</b>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)





## 21 Datum- und Zeitbibliothek

### 21.1 CompareDates

#### BEZEICHNUNG

CompareDates – vergleicht zwei Datumszeichenketten (V4.5)

#### ÜBERSICHT

```
result = CompareDates(date1$, date2$[, notime])
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die Zeit von zwei Datumszeichenketten zu vergleichen und gibt ihre Beziehung zurück. Beide Datumszeichenketten müssen in der folgenden Darstellungsart von Hollywood verwendet werden:

```
dd-mmm-yyyy hh:mm:ss
```

Der Bestandteil `mmm` ist eine Zeichenkette mit drei Zeichen, der den Monat identifiziert. Das kann sein Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, oder Dec.

Wenn Sie das optionale Argument `notime` auf `True` setzen, werden nur Daten (Datum1 und Datum2) verglichen. In diesem Fall dürfen die beiden Zeichenfolgen, die Sie an `CompareDates()` übergeben, keine Zeitangaben enthalten.

Der Rückgabewert `result` zeigt an, wie die beiden Termine im Zusammenhang stehen. Folgende Rückgabewerte sind möglich:

- 0:            `date1$` und `date2$` haben genau die gleiche Zeit
- 1:            `date1$` ist später in der Zeit als `date2$`
- 2:            `date1$` ist früher in der Zeit als `date2$`

Siehe auch `MakeDate()`, `ParseDate()` und `ValidateDate()`.

#### EINGABEN

- `date1$`      Erste Datumszeichenkette in der Darstellungsart von Hollywood
- `date2$`      Zweite Datumszeichenkette in der Darstellungsart von Hollywood
- `notime`      optional: `True`, um nur Daten zu vergleichen (voreingestellt ist `False`)

#### RÜCKGABEWERTE

- `result`      Vergleichsergebnis

#### BEISPIEL

```
NPrint(CompareDates("10-Dec-2009 13:34:12", "09-Dec-2009 15:36:21"))
NPrint(CompareDates("12-Dec-2009 23:59:59", "13-Dec-2009 00:00:00"))
NPrint(CompareDates("24-Dec-2009 20:00:00", "24-Dec-2009 20:00:00"))
```

Der obige Code führt drei Datumsvergleiche durch und wird folgende Ergebnisse ausgegeben: 1,2,0

## 21.2 DateToTimestamp

### BEZEICHNUNG

DateToTimestamp – konvertiert das lokale Datum in Zeitstempel (V7.1)

### ÜBERSICHT

```
s = DateToTimestamp(d[, isdst])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Zeitstempel für das in `d$` übergebene Datum zu erhalten. Diese Zeichenkette muss im Hollywood-Standard-Datumsformat vorliegen, d.h. `dd-mmm-yyyy hh:mm:ss`. Siehe [Abschnitt 21.1 \[CompareDates\]](#), [Seite 321](#), für Details.

Beachten Sie, dass das Datum, das Sie an diesen Befehl übergeben, als lokale Zeit interpretiert wird, während der zurückgegebene Zeitstempel von der UTC-Zeit ausgeht, d.h. von der Unix-Epoche, die am 1. Januar 1970, 00:00:00 UTC beginnt. Das bedeutet, dass die Übergabe von `01-Jan-1970 00:00:00` nur dann 0 zurückgibt, wenn die lokale Zeitzone mit der UTC-Zeitzone identisch ist. Auf Systemen östlich von UTC führt das Übergeben von `01-Jan-1970 00:00:00` zu einem Fehler, da der 1. Januar 1970, 00:00:00, östlich von UTC der 31. Dezember 1969 UTC bedeutet, der in der Unix-Epoche nicht darstellbar ist.

Das optionale Argument `isdst` gibt an, ob die Sommerzeit zum angegebenen Datum aktiv ist oder nicht. Normalerweise müssen Sie dieses Argument nicht angeben, da Hollywood diese Informationen automatisch von der Zeitzonendatenbank abfragt. Diese Information muss nur dann weitergegeben werden, wenn die angegebene Zeit unklar ist, d.h. beim Umschalten von Sommer- auf Normalzeit wird eine bestimmte Zeitspanne (typischerweise eine Stunde) in der Nacht wiederholt. In Deutschland beispielsweise werden die Uhren beim Umschalten von Sommer- auf Normalzeit von 3 Uhr morgens auf 2 Uhr morgens zurückgestellt. Das bedeutet, dass die Stunde zwischen 2 Uhr morgens und 3 Uhr morgens zweimal durchlaufen wird: Einmal in der Sommer- und einmal in der Normalzeit. Mit dem Argument `isdst` können Sie angeben, auf welche Stunde Sie sich beziehen.

Um einen Zeitstempel wieder in ein Datum zu konvertieren, verwenden Sie den Befehl [TimestampToDate\(\)](#). Siehe [Abschnitt 21.20 \[TimestampToDate\]](#), [Seite 334](#), für Details.

Siehe auch [DateToUTC\(\)](#) und [UTCToDate\(\)](#).

### EINGABEN

<code>d\$</code>	Hollywood-Datum, welches in einen Zeitstempel konvertiert wird
<code>isdst</code>	optional: ob die Sommerzeit zum angegebenen Datum aktiv ist oder nicht (voreingestellt ist -1, d.h. diese Information sollte aus der lokalen Zeitzonendatenbank abgerufen werden).

### RÜCKGABEWERTE

<code>s</code>	Zeit in Sekunden, die seit der Unix-Epoche vergangen ist, oder -1, wenn das angegebene Datum nicht in der Unix-Zeit dargestellt werden kann.
----------------	--

## 21.3 DateToUTC

### BEZEICHNUNG

DateToUTC – konvertiert das lokale Datum ins UTC-Format (V7.1)

### ÜBERSICHT

```
u$ = DateToUTC(d$, isdst)
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um das in `d$` übergebene lokale Datum in ein UTC-Datum umzuwandeln. Der Parameter `d$` muss in Hollywoods Standard-Datumsformat vorliegen, d.h. `dd-mm-yyyy hh:mm:ss`. Siehe [Abschnitt 21.1 \[CompareDates\]](#), [Seite 321](#), für Details.

Das optionale Argument `isdst` gibt an, ob die Sommerzeit zum angegebenen Datum aktiv ist oder nicht. Normalerweise müssen Sie dieses Argument nicht angeben, da Hollywood diese Informationen automatisch von der Zeitzonendatenbank abfragt. Diese Information muss nur dann weitergegeben werden, wenn die angegebene Zeit unklar ist, d.h. beim Umschalten von Sommer- auf Normalzeit wird eine bestimmte Zeitspanne (typischerweise eine Stunde) in der Nacht wiederholt. In Deutschland beispielsweise werden die Uhren beim Umschalten von Sommer- auf Normalzeit von 3 Uhr morgens auf 2 Uhr morgens zurückgestellt. Das bedeutet, dass die Stunde zwischen 2 Uhr morgens und 3 Uhr morgens zweimal durchlaufen wird: Einmal in der Sommer- und einmal in der Normalzeit. Mit dem Argument `isdst` können Sie angeben, auf welche Stunde Sie sich beziehen.

Um ein UTC-Datum in ein lokales Datum umzuwandeln, verwenden Sie den Befehl `UTCToDate()`. Siehe [Abschnitt 21.21 \[UTCToDate\]](#), [Seite 334](#), für Details.

Siehe auch `DateToTimestamp()` und `TimestampToDate()`.

### EINGABEN

<code>d\$</code>	lokales Datum, welches ins UTC-Format konvertiert wird
<code>isdst</code>	optional: ob die Sommerzeit zum angegebenen Datum aktiv ( <code>True</code> ) ist oder nicht ( <code>False</code> ) (voreingestellt ist -1, d.h. diese Information sollte aus der lokalen Zeitzonendatenbank abgerufen werden).

### RÜCKGABEWERTE

<code>u\$</code>	UTC-Äquivalent des lokalen Datumsarguments
------------------	--

## 21.4 GetDate

### BEZEICHNUNG

GetDate – gibt eine Zeichenkette zurück, die das Datum beinhaltet

### ÜBERSICHT

```
date$ = GetDate([type])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um das aktuelle Systemdatum und die Uhrzeit abzufragen. Datum und Uhrzeit können abhängig vom in `type` übergebenen Wert in verschiedenen Formaten zurückgegeben werden.

Die folgenden Formate werden derzeit von **type** erkannt:

**#DATELOCALNATIVE:**

Dies ist das Standardformat. Wenn **type** weggelassen wird, wird **GetDate()** diesen Typ verwenden. **#DATELOCALNATIVE** gibt das Datum in der Systemsprache zurück. Beispielsweise wird auf einem deutschen System der 4. September 2002 als "04.09.02" zurückgegeben, aber auf einem System in den USA wäre es "09.04.02". Beachten Sie, dass die Uhrzeit für diesen Typ überhaupt nicht zurückgegeben wird.

**#DATELOCAL:**

Dadurch wird das Datum im Standardformat von Hollywood für Datum und Uhrzeit angezeigt. Es sieht wie folgt aus:

**dd-mmm-yyyy hh:mm:ss**

4. September 2002 um 3:16 und 23 Sekunden würde als Standardformat von Hollywood wie folgt aussehen:

**04-Sep-2002 15:16:23**

Diese Ländereinstellung wird auch von anderen Hollywood Befehlen verwendet, beispielsweise durch die folgenden: **GetFileAttributes()**, **SetFileAttributes()**, **FileAttributes()** und **CompareDates()**.

Beachten Sie, dass obwohl **#DATELOCAL** der gebräuchlichste Typ für diesen Befehl ist, dies aus historischen Gründen nicht der Standardwert ist. **#DATELOCALNATIVE** ist der Standardtyp. (V4.5)

**#DATEUTC:**

Wenn Sie **#DATEUTC** in **type** übergeben, gibt **GetDate()** das aktuelle UTC-Datum und die aktuelle Uhrzeit zurück. Das UTC-Datum und die UTC-Zeit werden im Standardformat und Zeitnotation von Hollywood übergeben (siehe oben). (V7.1)

Siehe auch **GetDateNum()**, **GetTime()**, **GetTimestamp()** und **GetWeekday()**.

## EINGABEN

**type** Datum- und Zeitformat, das verwendet werden soll (V4.5)

## RÜCKGABEWERTE

**date\$** aktuelles Datum und Uhrzeit im gewünschten Format

## 21.5 GetDateNum

### BEZEICHNUNG

**GetDateNum** – ruft Datumsinformation als Wert ab

### ÜBERSICHT

**info** = **GetDateNum(type)**

### BESCHREIBUNG

Mit diesem Befehl können Sie aktuelle Datumsinformationen als Wert mit Hollywood abrufen. Die folgenden Konstanten können als **type** angegeben werden:

**#DATEDAY:**  
gibt den Tag des Monats zurück (1-31)

**#DATEMONTH:**  
gibt den Monat zurück (1-12)

**#DATETIME:**  
gibt die Uhrzeit zurück (00hhmmss)

**#DATEYEAR:**  
gibt das Jahr zurück (yyyy)

Siehe auch [GetDate\(\)](#), [GetTime\(\)](#), [GetTimestamp\(\)](#) und [GetWeekday\(\)](#).

#### EINGABEN

**type** eine der Konstanten wie oben aufgelistet

#### RÜCKGABEWERTE

**info** Uhrzeit, Tag, Monat oder Jahr

## 21.6 GetTime

#### BEZEICHNUNG

`GetTime` – fragt die aktuelle Zeit ab

#### ÜBERSICHT

```
time$ = GetTime([secs])
```

#### BESCHREIBUNG

Dieser Befehl erhält die aktuelle Zeit und gibt sie in `time$` zurück. Falls das optionale Argument `secs True` (WAHR) ist, hängt Hollywood auch noch die Sekunden an die Zeichenkette an, die die Zeit enthält.

Siehe auch [GetDate\(\)](#), [GetDateNum\(\)](#), [GetTimestamp\(\)](#) und [GetWeekday\(\)](#).

#### EINGABEN

**secs** optional: setzen Sie dies auf `True`, wenn Sie auch die Sekunden abrufen möchten

#### RÜCKGABEWERTE

**time\$** aktuelle Zeit als Zeichenkette

## 21.7 GetTimer

#### BEZEICHNUNG

`GetTimer` – zeigt den Status eines Zeitmessers

#### ÜBERSICHT

```
time = GetTimer(id)
```

**BESCHREIBUNG**

Dieser Befehl gibt den Status eines Zeitmessers zurück, was der vergangenen Zeit entspricht, seit der Zeitmesser mit `StartTimer()` gestartet wurde. Die Zeit wird in Millisekunden zurückgegeben.

Siehe auch `StopTimer()`, `PauseTimer()`, `ResetTimer()`, `ResumeTimer()`, `SetTimerElapse()`, `TimerElapsed()` und `WaitTimer()`.

**EINGABEN**

`id` ID des Zeitmessers, der abgefragt werden soll

**RÜCKGABEWERTE**

`time` Anzahl von Millisekunden die vergangen sind, seit der Zeitmesser gestartet wurde (in Bezug auf `PauseTimer()` und `ResumeTimer()`)

**BEISPIEL**

Siehe [Abschnitt 21.17 \[StartTimer\]](#), Seite 332.

## 21.8 GetTimestamp

**BEZEICHNUNG**

`GetTimestamp` – gibt einen Zeitstempel zurück (V7.0)

**ÜBERSICHT**

```
s = GetTimestamp([type])
```

**BESCHREIBUNG**

Dieser Befehl gibt einen Zeitstempel zurück. Die Zeit wird in Sekunden als Bruchzahl zurückgegeben, was eine ausreichende Genauigkeit ermöglicht. Mit dem Parameter `type` können Sie angeben, welche Art von Zeitstempel Sie erhalten möchten. Dies kann eine der folgenden sein:

**#TIMESTAMP\_START:**

Gibt die Zeit in Sekunden zurück, seit Hollywood gestartet wurde. Dies ist der Standardwert.

**#TIMESTAMP\_UNIX:**

Gibt die Zeit zurück, die seit der Unix-Epoche vergangen ist, die am 1. Januar 1970, 00:00:00 UTC begann. Beachten Sie, dass dies von der Systemuhr abhängt, sodass Probleme auftreten können, wenn die Systemuhr geändert wird, während Ihr Skript läuft. Wenn Sie unabhängig von der Systemuhr sein möchten, verwenden Sie stattdessen `#TIMESTAMP_RAW` (siehe unten).

**#TIMESTAMP\_RAW:**

Gibt einen rohen Zeitwert zurück, der unabhängig von der Systemuhr ist und monoton ansteigt. Dies kann nützlich sein, da `#TIMESTAMP_UNIX` von der Systemuhr abhängt, weil es die Anzahl der Sekunden seit dem 1. Januar 1970 zurückgibt, sodass Sie Probleme bekommen könnten, falls die Systemuhr zwischen zwei Aufrufen von `GetTimestamp()` geändert wird. (V9.1)

`GetTimestamp()` ist besonders in Verbindung mit Hollywoods Ereignis-Handler nützlich. Alle Ereignismeldungen enthalten ein Feld mit dem Namen `Timestamp`, das den Zeitstempel des Ereignisses enthält, als es erzeugt wurde. Wenn Sie diesen Zeitstempel mit dem Rückgabewert von `GetTimestamp()` vergleichen, können Sie z.B. sehr alte Ereignisse herausfiltern. Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), [Seite 483](#), für Details.

Um einen Zeitstempel in ein Datum zu konvertieren, können Sie den Befehl `TimestampToDate()` verwenden. Um ein Datum in einen Zeitstempel zu konvertieren, verwenden Sie den Befehl `DateToTimestamp()`.

Siehe auch `GetDate()`, `GetDateNum()`, `GetTime()` und `GetWeekday()`.

#### EINGABEN

`type` optional: die Art des zu ermittelnden Zeitstempels; siehe oben für mögliche Typen (Voreingestellt ist `#TIMESTAMP_START`)

#### RÜCKGABEWERTE

`s` Zeitstempel in Sekunden als Bruchzahl

## 21.9 GetTimeZone

#### BEZEICHNUNG

`GetTimeZone` – gibt Informationen über die Zeitzone zurück (V7.1)

#### ÜBERSICHT

`off, dst = GetTimeZone()`

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um Informationen über die Zeitzone zu erhalten, in der sich das Host-System befindet. Er gibt zwei Werte zurück: `off` wird auf die Anzahl Minuten der Computerzeit von UTC gesetzt und `dst` ist ein boolescher Wert, der angibt, ob gerade die Sommerzeit in der Zeitzone des Host-Systems aktiv ist oder nicht.

Beachten Sie, dass `off` negativ ist, wenn das Host-System östlich von UTC liegt und positiv, wenn es westlich von UTC liegt.

#### EINGABEN

keine

#### RÜCKGABEWERTE

`off` Versatz in Minuten von UTC

`dst` `True`, wenn Sommerzeit aktiv ist, andernfalls `False`

#### BEISPIEL

```
Print(GetTimeZone())
```

Wenn dieser Code im Januar auf einem Computer in Deutschland ausgeführt wird, werden `"-60"` und `"0"` ausgegeben, da es in Deutschland im Januar keine Sommerzeit gibt und die MEZ im Winter 60 Minuten vor der UTC liegt.

## 21.10 GetWeekday

### BEZEICHNUNG

GetWeekday – ermittelt den Wochentag

### ÜBERSICHT

`day$ = GetWeekday()`

### BESCHREIBUNG

Dieser Befehl gibt den Wochentag in der Zeichenkette `day$` zurück. Beachten Sie, dass der Wochentag in der Landessprache des Benutzers zurückgegeben wird (abhängig von seinen Lokalen-Einstellungen).

Siehe auch [GetDate\(\)](#), [GetDateNum\(\)](#), [GetTime\(\)](#) und [GetTimestamp\(\)](#).

### EINGABEN

keine

### RÜCKGABEWERTE

`day$` aktueller Wochentag

## 21.11 MakeDate

### BEZEICHNUNG

MakeDate – erstellt ein Hollywood-Datum aus Komponenten (V7.1)

### ÜBERSICHT

`d$ = MakeDate(t)`

### BESCHREIBUNG

Dieser Befehl setzt ein Hollywood-Datum aus einzelnen Datumskomponenten zusammen, die in der Tabelle `t` übergeben werden müssen. Das von diesem Befehl zurückgegebene Datum entspricht dem Standard-Datumsformat von Hollywood, d.h. `dd-mm-yyyy hh:mm:ss`. Siehe [Abschnitt 57.15 \[CompareStr\]](#), [Seite 1262](#), für Details.

An `MakeDate()` müssen Sie eine Tabelle übergeben, die die folgenden Felder initialisiert hat:

<b>MDay:</b>	Tag (1-31).
<b>Mon:</b>	Monat (1-12).
<b>Year:</b>	Jahreszahl (z.B. 2018).
<b>Hour:</b>	Stunden seit Mitternacht (0-23).
<b>Min:</b>	Minuten (0-59).
<b>Sec:</b>	Sekunden (0-59).

Um ein Hollywood-Datum in seine einzelnen Komponenten zu zerlegen, verwenden Sie den Befehl [ParseDate\(\)](#). Siehe [Abschnitt 21.12 \[ParseDate\]](#), [Seite 329](#), für Details.

Siehe auch [CompareDates\(\)](#) und [ValidateDate\(\)](#).

### EINGABEN

`t` Tabelle mit den einzelnen Datumskomponenten (siehe oben)



**RÜCKGABEWERTE**

**d\$** Datumszeichenkette im Standard-Datumsformat von Hollywood

**21.12 ParseDate****BEZEICHNUNG**

ParseDate – zerlegt ein Hollywood-Datum in seine Komponenten (V7.1)

**ÜBERSICHT**

**t** = ParseDate(**d\$**)

**BESCHREIBUNG**

Dieser Befehl analysiert das in **d\$** übergebene Hollywood-Datum und zerlegt es in seine einzelnen Komponenten. Diese Komponenten werden dann in einer Tabelle zurückgegeben. Der an diesen Befehl übergebene Datumszeichenkette muss in Hollywoods Standard-Datumsformat vorliegen, d.h. **dd-mmm-yyyy hh:mm:ss**. Siehe [Abschnitt 57.15 \[CompareStr\]](#), [Seite 1262](#), für Details.

ParseDate() gibt eine Tabelle zurück, in der die folgenden Felder initialisiert sind:

**MDay:** Tag (1-31).  
**Mon:** Monat (1-12).  
**Year:** Jahreszahl (z.B. 2018).  
**Hour:** Stunden seit Mitternacht (0-23).  
**Min:** Minuten (0-59).  
**Sec:** Sekunden (0-59).  
**WDay:** Tage seit Sonntag (0-6).  
**YDay:** Tage seit dem 1. Januar (0-365).

Um aus den einzelnen Komponenten ein Hollywood-Datum zu erstellen, verwenden Sie den Befehl [MakeDate\(\)](#). Siehe [Abschnitt 21.11 \[MakeDate\]](#), [Seite 328](#), für Details.

Siehe auch [CompareDates\(\)](#) und [ValidateDate\(\)](#).

**EINGABEN**

**d\$** Hollywood-Datum, welches zerlegt wird

**RÜCKGABEWERTE**

**t** Tabelle mit den einzelnen Datumskomponenten (siehe oben)

**21.13 PauseTimer****BEZEICHNUNG**

PauseTimer – hält einen Zeitmesser vorübergehend an

**ÜBERSICHT**

PauseTimer(**id**)

**BESCHREIBUNG**

Mit diesem Befehl hält den durch `id` angegebenen Zeitmesser an. Wird ein Zeitmesser nur hiermit angehalten aber nicht gestoppt, zählt er zwar die Zeit nicht mehr weiter, aber Sie können immer noch den Zeitmesserstatus mit `GetTimer()` abfragen. Der Zeitmesser kann durch den Aufruf von `ResumeTimer()` fortgesetzt werden.

Siehe auch `StartTimer()`, `StopTimer()`, `ResetTimer()`, `SetTimerElapse()`, `TimerElapsed()` und `WaitTimer()`.

**EINGABEN**

`id` ID des anzuhaltenden Zeitmessers

**BEISPIEL**

keine

## 21.14 ResetTimer

**BEZEICHNUNG**

`ResetTimer` – löscht den Zeitmesser oder stellt die Zeit ein (V4.5)

**ÜBERSICHT**

```
ResetTimer(id[, time])
```

**BESCHREIBUNG**

Sie können diesen Befehl verwenden, um einen vorhandenen Zeitmesser auf Null oder eine angegebene Zeit zurücksetzen. Wenn Sie den Zeitmesser auf Null zurücksetzen möchten, lassen Sie einfach das zweite Argument `time` weg. Ansonsten verwenden Sie dieses Argument und geben für den Zeitmesser eine Zeit in Millisekunden an.

Benutzen Sie `ResetTimer()`, um den Zeitmesser zu löschen. Dies ist generell schneller als das Starten eines neuen Zeitmessers mit `StartTimer()`.

Siehe auch `GetTimer()`, `StopTimer()`, `PauseTimer()`, `ResumeTimer()`, `SetTimerElapse()`, `TimerElapsed()` und `WaitTimer()`.

**EINGABEN**

`id` ID des Zeitmessers, der zurückgesetzt werden soll

`time` optional: erwünschter Zeitwert für den Zeitmesser in Millisekunden (Standard ist 0, was bedeutet, keine Zeit)

**BEISPIEL**

```
StartTimer(1)
Wait(1000, #MILLISECONDS)
Print(GetTimer(1))
ResetTimer(1, 2000)
Print(GetTimer(1))
```

Der obige Code startet einen neuen Zeitmesser 1, wartet eine Sekunde und gibt dann den Zustand des Zeitmessers aus, diese sollte 1000 oder einige Millisekunden mehr sein. Dann wird der Zustand des Zeitmessers der auf 2000 Millisekunden eingestellt ist erneut ausgegeben. Diese Zeit sollte 2000 oder einige Millisekunden mehr sein.

## 21.15 ResumeTimer

### BEZEICHNUNG

ResumeTimer – setzt einen angehaltenen Zeitmesser fort

### ÜBERSICHT

ResumeTimer(id)

### BESCHREIBUNG

Dieser Befehl setzt den durch `id` angegebenen Zeitmesser fort. Der Zeitmesser muss sich im Pausemodus befinden (gesetzt durch `PauseTimer()`). Wenn `ResumeTimer()` erfolgreich war, macht der Zeitmesser damit weiter, die Zeit zu zählen.

Siehe auch `GetTimer()`, `StartTimer()`, `StopTimer()`, `ResetTimer()`, `SetTimerElapse()`, `TimerElapsed()` und `WaitTimer()`.

### EINGABEN

`id` ID des Zeitmessers, der fortgesetzt wird

### BEISPIEL

keine

## 21.16 SetTimerElapse

### BEZEICHNUNG

SetTimerElapse – stellt den Schwellenwert für den Ablauf des Zeitmessers ein (V9.0)

### ÜBERSICHT

SetTimerElapse(id, elapse[, reset])

### BESCHREIBUNG

Dieser Befehl setzt den Schwellenwert für den Ablauf des von `id` angegebenen Zeitmessers auf die in `elapse` angegebene Zeit. Diese Zeit muss in Millisekunden angegeben werden. Sie können dann `TimerElapsed()` aufrufen, um herauszufinden, wann der Zeitmesser abgelaufen ist. Mit dem Befehl `WaitTimer()` können Sie auf den Ablauf des Zeitmessers warten.

Standardmäßig wird mit `SetTimerElapse()` auch der Zeitmesser zurückgesetzt. Wenn Sie das nicht wollen, übergeben Sie `False` im Argument `reset`.

Beachten Sie, dass wenn Sie 0 im Argument `elapse` übergeben, das Ablaufen für diesen Zeitmesser deaktiviert wird, d.h. `TimerElapsed()` gibt für Zeitmesser mit einem Ablaufschwellenwert von 0 niemals `True` zurück.

Siehe auch `GetTimer()`, `StartTimer()`, `StopTimer()`, `PauseTimer()`, `ResetTimer()` und `ResumeTimer()`.

### EINGABEN

<code>id</code>	Identifikator des zu ändernden Zeitmessers
<code>elapse</code>	Ablaufzeitschwelle in Millisekunden oder 0, um die Ablaufzeit zu deaktivieren
<code>reset</code>	optional: ob der Zeitmesser zurückgesetzt werden soll ( <code>True</code> ) oder nicht ( <code>False</code> ) (voreingestellt ist <code>True</code> )

## 21.17 StartTimer

### BEZEICHNUNG

StartTimer – startet einen neuen Zeitmesser

### ÜBERSICHT

```
[id] = StartTimer(id[, elapse])
```

### BESCHREIBUNG

Dieser Befehl erstellt einen neuen Zeitmesser und weist ihm die `id` zu. Wenn Sie `Nil` als ID übergeben, wird `StartTimer()` automatisch eine freie ID wählen und gibt diese zurück. Dieser Zeitmesser wird solange laufen, bis Sie `PauseTimer()` oder `StopTimer()` aufrufen. Sie können den aktuellen Status des Zeitmessers durch den Aufruf des Befehls `GetTimer()` abfragen.

Ab Hollywood 9.0 gibt es ein neues optionales Argument `elapse`. Wenn Sie dies auf eine Zeit in Millisekunden setzen, gibt `TimerElapsed()` `True` zurück, sobald der Zeitmesser für die angegebene Zeitdauer gelaufen ist. Alternativ können Sie auch `WaitTimer()` verwenden, um auf den Ablauf eines Zeitmessers zu warten. Schließlich kann der Zeitablaufwert des Zeitmessers auch mit `SetTimerElapse()` eingestellt oder geändert werden. Siehe [Abschnitt 21.16 \[SetTimerElapse\]](#), Seite 331, für Details.

Siehe auch `ResetTimer()` und `ResumeTimer()`.

### EINGABEN

`id` ID für Ihren Zeitmesser oder `Nil` für die automatische ID-Auswahl

### RÜCKGABEWERTE

`id` optional: ID des Zeitmessers; wird nur zurückgegeben, wenn Sie `Nil` als `id` übergeben haben (siehe oben)

### BEISPIEL

```
StartTimer(1)
Wait(200)
t = GetTimer(1)
Print(t)
```

Der obige Code startet einen neuen Zeitmesser, wartet 4 Sekunden und ruft den Zeitmesserzustand ab. Der Zeitmesserzustand wird in die Variable `t` kopiert und sollte den Wert von 4000 Millisekunden haben.

## 21.18 StopTimer

### BEZEICHNUNG

StopTimer – stoppt den Zeitmesser

### ÜBERSICHT

```
StopTimer(id)
```

**BESCHREIBUNG**

Dieser Befehl stoppt den Zeitmesser, welcher in `id` angegeben ist. Wenn Sie den Zeitmesser stoppen, wird er vollständig aus dem System entfernt, deshalb können Sie ihn nicht fortsetzen. Wenn Sie einen Zeitmesser anhalten möchten, benutzen Sie stattdessen `PauseTimer()`.

Siehe auch `GetTimer()`, `StartTimer()`, `ResetTimer()`, `ResumeTimer()`, `SetTimerElapse()`, `TimerElapsed()` und `WaitTimer()`.

**EINGABEN**

`id` ID für den Zeitmesser, der gestoppt werden soll

**BEISPIEL**

Siehe [Abschnitt 21.17 \[StartTimer\]](#), Seite 332.

**21.19 TimerElapsed****BEZEICHNUNG**

`TimerElapsed` – überprüft, ob der Zeitmesser abgelaufen ist (V9.0)

**ÜBERSICHT**

```
elapsed = TimerElapsed(id[, reset])
```

**BESCHREIBUNG**

Dieser Befehl prüft, ob der von `id` angegebene Zeitmesser abgelaufen ist und gibt `True` zurück, falls dies der Fall ist, andernfalls `False`. Standardmäßig wird der Zeitmesser nach Ablauf auf 0 zurückgesetzt. Wenn Sie das nicht möchten, übergeben Sie `False` im Argument `reset`.

Der Schwellenwert für das Ablaufen eines Zeitmessers kann entweder beim Erstellen eines Zeitmessers mit `StartTimer()` oder später mit `SetTimerElapse()` festgelegt werden.

Siehe auch `GetTimer()`, `StopTimer()`, `PauseTimer()`, `ResetTimer()`, `ResumeTimer()` und `WaitTimer()`.

**EINGABEN**

`id` Identifikator des zu untersuchenden Zeitmessers

`reset` optional: `True`, wenn der abgelaufene Zeitmesser auf 0 zurückgesetzt werden soll, andernfalls `False` (Standardwert ist `True`)

**RÜCKGABEWERTE**

`elapsed` `True`, wenn der Zeitmesser abgelaufen ist, sonst `False`

**BEISPIEL**

```
StartTimer(1, 10000)
Repeat
    VWait
Until TimerElapsed(1) = True
```

## 21.20 TimestampToDate

### BEZEICHNUNG

TimestampToDate – wandelt Zeitstempel in Datum um (V7.1)

### ÜBERSICHT

```
d$ = TimestampToDate(s[, unixtime])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen Zeitstempel in eine Datumszeichenkette in Hollywoods Standard-Datumsformat zu konvertieren, d.h. `dd-mmm-yyyy hh:mm:ss`. Siehe [Abschnitt 57.15 \[CompareStr\]](#), [Seite 1262](#), für Details. Das optionale Argument `unixtime` gibt an, ob der Zeitstempel ab dem Beginn der Unix-Epoche (d.h. 1. Januar 1970, 00:00:00 UTC) oder ab dem Zeitpunkt des Hollywood-Starts gemessen wird. Standardmäßig werden die Zeitstempel ab dem Zeitpunkt gemessen, zu dem Hollywood gestartet wurde.

Um ein Datum wieder in einen Zeitstempel umzuwandeln, verwenden Sie den Befehl `DateToTimestamp()`. Siehe [Abschnitt 21.2 \[DateToTimestamp\]](#), [Seite 322](#), für Details.

Siehe auch `DateToUTC()` und `UTCToDate()`.

### EINGABEN

<code>s</code>	Zeitstempel, der in ein Datum konvertiert wird
<code>unixtime</code>	optional: wenn <code>True</code> gesetzt ist, wird der Zeitstempel relativ zum Beginn der Unix-Epoche interpretiert; andernfalls relativ zur Zeit, zu der Hollywood gestartet wurde (voreingestellt ist <code>False</code> ).

### RÜCKGABEWERTE

<code>d\$</code>	Datum im Hollywood-Standard-Datumsformat
------------------	--

## 21.21 UTCToDate

### BEZEICHNUNG

UTCToDate – konvertiert das UTC-Datum in das lokale Datum (V7.1)

### ÜBERSICHT

```
d$ = UTCToDate(u$)
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um das in `u$` übergebene UTC-Datum in ein lokales Datum zu konvertieren. Der Parameter `u$` muss im Standard-Datumsformat von Hollywood sein, d.h. `dd-mmm-yyyy hh:mm:ss`. Siehe [Abschnitt 57.15 \[CompareStr\]](#), [Seite 1262](#), für Details.

Um ein lokales Datum in ein UTC-Datum zurück zu konvertieren, verwenden Sie den Befehl `DateToUTC()`. Siehe [Abschnitt 21.3 \[DateToUTC\]](#), [Seite 323](#), für Details.

Siehe auch `DateToTimestamp()` und `TimestampToDate()`.

### EINGABEN

<code>u\$</code>	UTC-Datum, welches in ein lokales Datum konvertiert wird
------------------	--

**RÜCKGABEWERTE**

`d$`            lokales Datum

**21.22 ValidateDate****BEZEICHNUNG**

`ValidateDate` – überprüft, ob das Datum gültig ist (V7.1)

**ÜBERSICHT**

`b = ValidateDate(d$)`

**BESCHREIBUNG**

Mit diesem Befehl kann überprüft werden, ob das in `d$` angegebene Datum gültig ist. `ValidateDate()` stellt sicher, dass alle individuellen Datums- und Zeitkomponenten innerhalb ihrer gültigen Bereiche liegen, z. B. ist der 29. Februar nur ein gültiges Datum in einem Schaltjahre. Der Parameter `d$` muss im Standard-Datumsformat von Hollywood vorliegen, d.h. `dd-mmm-yyyy hh:mm:ss`. Siehe [Abschnitt 57.15 \[CompareStr\]](#), [Seite 1262](#), für Details.

Siehe auch [CompareDates\(\)](#), [MakeDate\(\)](#) und [ParseDate\(\)](#).

**EINGABEN**

`d$`            Datum, welches überprüft werden soll

**RÜCKGABEWERTE**

`b`            `True`, wenn das Datum gültig ist, andernfalls `False`

**21.23 WaitTimer****BEZEICHNUNG**

`WaitTimer` – wartet, bis der Zeitmesser eine bestimmte Zeit erreicht hat (V2.0)

**ÜBERSICHT**

`WaitTimer(id[, time, reset])`  
`t = WaitTimer(table[, reset])` (V9.0)

**BESCHREIBUNG**

Dieser Befehl wartet, bis der durch `id` angegebene Zeitmesser für die im Argument `time` angegebene Zeit abgelaufen ist. Diese Zeit muss in Millisekunden angegeben werden. Wenn Sie das Argument `time` weglassen oder auf -1 setzen, wartet `WaitTimer()`, bis der Zeitmesser seinen mit [SetTimerElapse\(\)](#) oder [StartTimer\(\)](#) eingestellten Schwellenwert erreicht hat.

Bevor dieser Befehl zurückkehrt, wird auch der angegebene Zeitmesser zurückgesetzt, so dass Sie diesen Befehl problemlos in einer Schleife verwenden können. Sie können dieses Verhalten ändern, indem Sie das optionale Argument `reset` auf `False` setzen. In diesem Fall wird der Zeitmesser nicht zurückgesetzt.

Ab Hollywood 9.0 gibt es eine alternative Möglichkeit, `WaitTimer()` zu verwenden. Anstelle der ID eines einzelnen Zeitmessers können Sie auch eine Tabelle mit mehreren

Zeitmesser-Identifikatoren übergeben. In diesem Fall wartet `WaitTimer()`, bis mindestens einer der Zeitmesser aus der im Argument `table` angegebenen Liste abgelaufen ist. Sobald dies geschieht, gibt `WaitTimer()` eine Tabelle an Sie zurück. Diese Tabelle ist eine Liste mit den Identifikatoren aller abgelaufenen Zeitmesser. Wenn das Argument `reset` `True` ist, was auch der Standardwert ist, werden alle abgelaufenen Zeitmesser von `WaitTimer()` zurückgesetzt. Beachten Sie, dass die Tabelle, die Sie an `WaitTimer()` übergeben, auch leer sein kann. In diesem Fall wartet `WaitTimer()` einfach darauf, dass ein Zeitmesser von allen aktuell laufenden Zeitmesser abgelaufen ist.

`WaitTimer()` kann sehr nützlich sein, um die Ausführung von Schleifen zu drosseln, damit sie nicht die gesamte CPU-Leistung verbrauchen. Wenn Sie beispielsweise eine Schleife haben, die ein Sprite vom linken zum rechten Rand des Displays bewegt, sollten Sie eine Art Drosselung hinzufügen, da es nicht sinnvoll ist, den Bildschirm häufiger zu aktualisieren, als der Monitor aktualisiert wird. Dies ist sehr wichtig. Auch wenn das Skript ohne `WaitTimer()` mit perfekter Geschwindigkeit läuft, sollten Sie nicht vergessen, dass es schnellere Rechner als Ihren gibt. Die Verwendung von `WaitTimer()` in Ihren Schleifen stellt sicher, dass Ihr Programm auf jeder Plattform mit der gleichen Geschwindigkeit ausgeführt wird.

Siehe [Abschnitt 15.3 \[Zeitsteuerung des Skripts\]](#), Seite 170, für Details.

Siehe auch [GetTimer\(\)](#), [StartTimer\(\)](#), [StopTimer\(\)](#), [PauseTimer\(\)](#), [ResetTimer\(\)](#), [ResumeTimer\(\)](#), [SetTimerElapse\(\)](#) und [TimerElapsed\(\)](#).

## EINGABEN

<code>id</code>	syntax 1: ID des Zeitmessers
<code>time</code>	syntax 1, optional: Zeit in Millisekunden, die der Zeitmesser erreichen muss (Standardwert ist der Schwellenwert für das Ablaufen des Zeitmessers)
<code>table</code>	syntax 2: übergeben Sie hier eine Tabelle mit einer Liste von Zeitmessern und <code>WaitTimer()</code> wird zurückkehren, sobald ein Zeitmesser aus der Liste abgelaufen ist (siehe oben); wenn Sie eine leere Tabelle übergeben, werden alle laufenden Zeitmesser berücksichtigt
<code>reset</code>	optional: gibt an, ob der Zeitmesser nach erreichter Zeit zurückgesetzt wird oder nicht (Standard ist <code>True</code> was bedeutet, dass der Zeitmesser zurückgesetzt wird)

## BEISPIEL

```

StartTimer(1)
For k = 0 To 640
    DisplaySprite(1, k, 0)
    WaitTimer(1, 40)
Next

```

Im obigen Code läuft Sprite 1 von links nach rechts. Mit jedem Aufruf von `DisplaySprite()` und `WaitTimer()` ist sichergestellt, dass mindestens 40 Millisekunden gewartet wird, bis zum nächsten `DisplaySprite()`. Daher wird diese Schleife nicht mehr als 25 Mal pro Sekunde ausgeführt werden, da  $40 * 25 = 1000$  Millisekunden (1 Sekunde).



## 22 Debugbibliothek

### 22.1 Assert

#### BEZEICHNUNG

Assert – schlägt fehl, wenn angegebener Ausdruck falsch ist (V5.0)

#### ÜBERSICHT

Assert(expr)

#### BESCHREIBUNG

Dieser Befehl überprüft, ob der angegebene Ausdruck **False** ist (oder **Nil**) und verursacht einen Fehler, wenn dies der Fall ist. Dies ist vor allem nützlich für Debugging-Zwecke. Sie kann auch eine Tabelle oder eine Funktion in **expr** übergeben. In diesem Fall wird **expr** wie **True** betrachtet werden.

Dieser Aufruf kann durch Angabe der **-nodebug** deaktiviert werden, wenn das Konsolenargument als Skript oder Applet ausgeführt wird. In diesem Fall, wird der Aufruf **Assert()** überhaupt nichts tun. Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), [Seite 33](#), für Details.

#### EINGABEN

**expr**            Ausdruck überprüfen

#### BEISPIEL

```
a = 5
b = 0
Assert(b <> 0)
c = a / b
```

In dem obigen Code wird **Assert()** verwendet, um sicherzustellen, dass wir nicht durch Null teilen. **Assert()** wird einen solchen Fehler durch die Überprüfung von **b** gegen Null verhindern.

### 22.2 CloseResourceMonitor

#### BEZEICHNUNG

CloseResourceMonitor – schließt den Ressourcenmonitor von Hollywood (V4.5)

#### ÜBERSICHT

CloseResourceMonitor()

#### BESCHREIBUNG

Dieser Befehl schließt den integrierten Ressourcenmonitor von Hollywood. Sie können ihn wieder durch den Aufruf von **OpenResourceMonitor()** öffnen.

Siehe [Abschnitt 22.7 \[OpenResourceMonitor\]](#), [Seite 340](#), für mehr Informationen über den Hollywood Ressourcenmonitor.

#### EINGABEN

keine

## 22.3 DebugOutput

### BEZEICHNUNG

DebugOutput – aktiviert/deaktiviert die Debug-Ausgabe

### ÜBERSICHT

DebugOutput(enable)

### BESCHREIBUNG

Dieser Befehl aktiviert bzw. deaktiviert die Debug-Ausgabe auf ein Standard Debug-Gerät. Wenn die Debug-Ausgabe aktiviert ist, wird Hollywood Informationen jeder Funktion oder jedem Befehl drucken, die/den es kennt, so dass Sie leichter Probleme aufspüren können.

Dieser Aufruf kann durch Angabe von `-nodebug` deaktiviert werden, wenn das Konsolenargument als Skript oder Applet ausgeführt wird. In diesem Fall, wird der Aufruf `DebugOutput()` überhaupt nichts tun. Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), [Seite 33](#), für Details.

### EINGABEN

`enable` Zustandsflag: `True` um die Debug-Ausgabe zu ermöglichen, `False` um sie zu deaktivieren

### BEISPIEL

DebugOutput(TRUE)

Der obige Code ermöglicht die Debug-Ausgabe.

## 22.4 DebugPrint

### BEZEICHNUNG

DebugPrint – gibt Informationen über die Debug-Ausgabe aus (V2.0)

### FRÜHERER NAME

DebugVal (V1.x) ; DebugStr (V1.x)

### ÜBERSICHT

DebugPrint(...)

### BESCHREIBUNG

Dieser Befehl gibt alle Argumente, die Sie angeben, auf dem aktuellen Debug-Gerät aus. Dies ist in der Regel Ihr Konsolenfenster. Sie können festlegen, wie viele Argumente Sie übergeben wollen und diese können von jeder Art sein. Wenn Sie mehrere Argumente diesem Befehl übergeben, werden diese um sie zu trennen mit einem Zwischenraum ausgegeben.

`DebugPrint()` fügt auch automatisch ein Zeilenumbruchzeichen an das Ende der Ausgabe an. Wenn Sie das nicht wollen, verwenden Sie stattdessen `DebugPrintNR()`. Siehe [Abschnitt 22.5 \[DebugPrintNR\]](#), [Seite 339](#), für Details.

Dieser Befehl ersetzt die Befehle `DebugStr()` und `DebugVal()`, Sie können jetzt einfach diesen Befehl hier verwenden.

Durch Angabe von `-nodebug` kann dieser Befehl deaktiviert werden, wenn das Konsolenargument als Skript oder Applet ausgeführt wird. In diesem Fall, wird der Aufruf `DebugPrint()` überhaupt nichts tun. Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), Seite 33, für Details.

Beachten Sie außerdem, dass das Debuggen beim Kompilieren eines Applets oder eines Programmes automatisch deaktiviert wird, sofern Sie es nicht ausdrücklich aktivieren, indem Sie den Tag `EnableDebug` beim Befehl `@OPTIONS` auf `True` setzen. Wenn Sie also ein Applet oder ein Programm kompiliert haben und feststellen, dass `DebugPrint()` nichts tut, liegt der Grund wahrscheinlich darin, dass das Debuggen deaktiviert ist.

#### EINGABEN

... mindestens einen Wert, der auf dem Debug-Gerät ausgegeben wird

#### BEISPIEL

```
DebugPrint("The user entered", name$, "as his name and", age,
           "as his age!")
```

## 22.5 DebugPrintNR

#### BEZEICHNUNG

`DebugPrintNR` – gibt Informationen über die Debug-Ausgabe ohne neue Zeile aus (V6.1)

#### ÜBERSICHT

```
DebugPrintNR(...)
```

#### BESCHREIBUNG

Dies ist das gleiche wie `DebugPrint()` aber hängt keinen neuen Zeilenumbruch an die Zeichenkette.

Dieser Aufruf kann durch Angabe der `-nodebug` deaktiviert werden, wenn das Konsolenargument als Skript oder Applet ausgeführt wird. In diesem Fall, wird der Aufruf `DebugPrintNR()` überhaupt nichts tun. Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), Seite 33, für Details.

Beachten Sie außerdem, dass das Debuggen beim Kompilieren eines Applets oder eines Programmes automatisch deaktiviert wird, sofern Sie es nicht ausdrücklich aktivieren, indem Sie den Tag `EnableDebug` beim Befehl `@OPTIONS` auf `True` setzen. Wenn Sie also ein Applet oder ein Programm kompiliert haben und feststellen, dass `DebugPrint()` nichts tut, liegt der Grund wahrscheinlich darin, dass das Debuggen deaktiviert ist.

Siehe [Abschnitt 22.4 \[DebugPrint\]](#), Seite 338, für Details.

#### EINGABEN

... mindestens einen Wert, der auf dem Debug-Gerät ausgegeben wird

#### BEISPIEL

```
DebugPrintNR("Hello ")
DebugPrintNR("World!")
DebugPrintNR("\n")
```

Dies ist das gleiche wie `DebugPrint("Hello World!")`.

## 22.6 DebugPrompt

### BEZEICHNUNG

DebugPrompt – liest die Benutzereingabe vom Debug-Gerät aus (V5.0)

### ÜBERSICHT

s\$ = DebugPrompt(p\$)

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Benutzer aufzufordern, eine Zeichenfolge in das aktuelle Debug-Gerät einzugeben. DebugPrompt() wird die Zeichenfolge in p\$ als Eingabeaufforderung angeben und stoppt die Skriptausführung, bis der Benutzer eine Zeichenfolge eingibt und mit der RETURN-Taste bestätigt. Die Zeichenfolge wird dann in s\$ zurückgegeben.

Dieser Aufruf kann deaktiviert werden, indem beim Ausführen eines Skripts oder eines Applets das Argument -nodebug angegeben wird. In diesem Fall gibt der Aufruf von DebugPrompt() nur eine leere Zeichenkette zurück. Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), [Seite 33](#), für Details.

Beachten Sie außerdem, dass das Debuggen beim Kompilieren eines Applets oder eines Programmes automatisch deaktiviert wird, sofern Sie es nicht ausdrücklich aktivieren, indem Sie den Tag EnableDebug beim Befehl @OPTIONS auf True setzen. Wenn Sie also ein Applet oder ein Programm kompiliert haben und feststellen, dass DebugPrint() nichts tut, liegt der Grund wahrscheinlich darin, dass das Debuggen deaktiviert ist.

### EINGABEN

p\$           Text mit der Aufforderung an den Benutzer etwas einzugeben

### RÜCKGABEWERTE

s\$           Zeichenfolge, welche vom Benutzer eingegeben wird

### BEISPIEL

```
name$ = DebugPrompt("Wie ist dein Name? ")
age$ = DebugPrompt("Und dein Alter? ")
home$ = DebugPrompt("Wo lebst du? ")
DebugPrint("Dein Name ist", name$, "und du bist", age$,
           "Jahre alt und lebst in", home$, "!")
```

Der obige Code zeigt die Verwendung des Befehls DebugPrompt().

## 22.7 OpenResourceMonitor

### BEZEICHNUNG

OpenResourceMonitor – öffnet den Ressourcenmonitor von Hollywood (V4.5)

### ÜBERSICHT

OpenResourceMonitor()

### BESCHREIBUNG

Dieser Befehl öffnet den in Hollywood eingebauten Ressourcenmonitor. Der Ressourcenmonitor wird eine Liste aller Ressourcen anzeigen, die Hollywood im Arbeitsspeicher hat.

Die Liste wird mehrmals pro Sekunde aktualisiert, sodass diese immer auf dem neuesten Stand ist.

Der Ressourcenmonitor ist sehr nützlich, um die Speicherverwaltung des Skripts richtig zu handhaben. Obwohl Hollywood über Tracking-Ressourcen verfügt und automatisch alle Ressourcen freigibt, wenn es beendet wird, ist es wichtig, für größere Projekte ihr Ressourcenmanagement im Auge zu behalten, da sonst Ihr Programm mehr und mehr Arbeitsspeicher verbrauchen wird. Wenn Sie nicht ein Auge auf Ihre Ressourcen haben, kann es oft passieren, dass je länger Ihr Programm läuft, immer mehr Arbeitsspeicher verbraucht wird und das darf nie passieren.

Der Ressourcenmonitor ermöglicht es Ihnen, bequem Ihre Ressourcen im Auge zu behalten. Wenn Sie den Ressourcenmonitor immer öffnen, während Sie entwickeln, werden Sie leicht feststellen, ob es irgendwo ein Ressourcenproblem gibt. Zum Beispiel, wenn Sie feststellen, dass Pinsel oder Ebenennummern stetig zunehmen, während das Skript ausgeführt wird, dann sollten Sie alarmiert sein und es ist wahrscheinlich, dass etwas falsch mit Ihrem Code ist, das Sie dann beheben müssen.

Sie können auch den Ressourcenmonitor direkt beim Start aktivieren, indem Sie das Konsole Argument `-resourcemonitor` benutzen.

Um den Ressourcenmonitor zu beenden, schließen Sie einfach das Fenster oder rufen Sie den Befehl `CloseResourceMonitor()` auf.

## EINGABEN

keine

## 22.8 WARNING

### BEZEICHNUNG

WARNING – sendet eine Warnmeldung an die Debug-Ausgabe (V6.1)

### ÜBERSICHT

`@WARNING msg$`

### BESCHREIBUNG

Diese Präprozessor-Anweisung wird die angegebene Warnmeldung direkt an die Debug-Ausgabe senden, bevor das Skript ausgeführt wird. Auf diese Weise können Sie bequem wichtig Informationen speichern wie z.B. eine "Zu tun Liste" oder eine Fehlerbereinigungsliste und Sie werden wieder daran erinnert, wenn Sie Ihr Skript ausführen.

Diese Präprozessor-Anweisung kann durch Angabe des Konsolenargumentes `-nodebug` deaktiviert werden, wenn ein Skript oder Applet ausgeführt wird. In diesem Fall wird diese Präprozessor-Anweisung nichts tun. Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), [Seite 33](#), für Details.

## EINGABEN

`msg$` zeigt diese Fehlermeldung an

## BEISPIEL

`@WARNING "FIXME: unterstützt andere Bildformate"`

Der obige Code wird die Zeichenfolge "FIXME: unterstützt andere Bildformate" zur Debug-Ausgabe senden, bevor Hollywood das Skript ausführt.

## 23 Dialogbibliothek

### 23.1 ColorRequest

#### BEZEICHNUNG

ColorRequest – fordert den Benutzer auf, eine Farbe zu wählen (V5.0)

#### ÜBERSICHT

```
ret = ColorRequest(title$[, t])
```

#### FRÜHERE SYNTAX

```
ret = ColorRequest(title$[, color])
```

#### BESCHREIBUNG

Dieser Befehl öffnet das Farbdialogfenster und fordert den Benutzer auf, aus einer Palette von vordefinierten Farben eine auszuwählen. Zusätzlich kann der Benutzer auch eine eigene Farbe mischen. Das erste Argument `title$` gibt den Titel des Dialogfensters an. Wenn eine leere Zeichenkette ("" ) als `title$` übergeben wird, verwendet das Farbdialogfenster den im Präprozessor-Anweisung `@APPTITLE` angegebenen Titel.

`ColorRequest()` unterstützt mehrere optionale Argumente. Vor Hollywood 9.0 mussten diese als optionale Parameter übergeben werden (siehe oben). Seit Hollywood 9.0 wird jedoch empfohlen, die neue Syntax zu verwenden, die ein einzelnes optionales Tabellenargument hat, mit dem ein oder mehrere optionale Argumente an `ColorRequest()` übergeben werden können.

Die folgenden Tabellenfelder werden von diesem Befehl erkannt:

<b>Color:</b>	Mit diesem Tabellen-Tag kann eine Farbe angegeben werden, die anfangs ausgewählt wird. Wenn Sie diesen Tabellen-Tag nicht setzen, wird eine zufällige Farbe vorgewählt.
<b>X:</b>	Anfängliche x-Position für das Farbdialogfenster auf dem Bildschirm. Nicht alle Plattformen unterstützen dies. (V9.0)
<b>Y:</b>	Anfängliche y-Position für das Farbdialogfenster auf dem Bildschirm. Nicht alle Plattformen unterstützen dies. (V9.0)
<b>Width:</b>	Anfangsbreite für das Farbdialogfenster. Nicht alle Plattformen unterstützen dies. (V9.0)
<b>Height:</b>	Anfangshöhe für das Farbdialogfenster. Nicht alle Plattformen unterstützen dies. (V9.0)

Bitte beachten Sie, dass dieser Befehl die Installation der `reqtools.library` auf AmigaOS und kompatiblen Geräten erfordert.

#### EINGABEN

<code>title\$</code>	Titel für das Dialogfenster
<code>t</code>	optional: Tabelle mit weiteren Argumenten (siehe oben) (V9.0)

#### RÜCKGABEWERTE

<code>ret</code>	die Auswahl des Benutzers oder -1, wenn der Benutzer den Dialog abgebrochen hat
------------------	---

**BEISPIEL**

```

r = ColorRequest("Select a color")
If r = -1
    Print("Requester cancelled!")
Else
    SetFillStyle(#FILLCOLOR)
    Box(0, 0, 640, 480, r)
EndIf

```

Im obigen Code wird der Benutzer nach der Farbe gefragt und dann wird unter der Verwendung der ausgewählten Farbe ein gefülltes Rechteck gezeichnet.

**23.2 FileRequest****BEZEICHNUNG**

FileRequest – öffnet ein Dateidialogfenster

**ÜBERSICHT**

```
f$ = FileRequest(title$[, t])
```

**FRÜHERE SYNTAX**

```
f$ = FileRequest(title$[, filter$, mode, defdir$, deffile$])
```

**BESCHREIBUNG**

Dieser Befehl öffnet die Dateiauswahl, die es dem Benutzer erlaubt, eine Datei auszuwählen. Sie können den Titel für das Dialogfenster durch die Zuordnung des Arguments `title$` angeben.

Die Datei, die der Benutzer auswählt, wird in `f$` zurückgegeben, einschließlich des Pfades, in dem sie sich befindet. Wenn der Benutzer den Dialog abbricht, bleibt die Zeichenfolge `f$` leer.

`FileRequest()` unterstützt viele optionale Argumente. Vor Hollywood 9.0 mussten diese als optionale Parameter übergeben werden (siehe oben). Seit Hollywood 9.0 wird jedoch empfohlen, die neue Syntax zu verwenden, die ein einzelnes optionales Tabellenargument hat, mit dem ein oder mehrere optionale Argumente an `FileRequest()` übergeben werden können.

Die folgenden Tabellenfelder werden von diesem Befehl erkannt:

**Mode:** Dieses Tabellenargument ermöglicht es Ihnen, das Dateidialogfenster in den Speicher- oder Mehrfachauswahlmodus zu setzen. Für den Speichermodus benutzen Sie `#REQ_SAVEMODE` und für den Mehrfachauswahlmodus `#REQ_MULTISELECT`. Im Mehrfachauswahlmodus wird dieser Befehl keine Zeichenkette sondern eine Tabelle zurückgeben. Sie enthält alle Dateien, die der Benutzer ausgewählt hat und endet mit einer leeren Zeichenkette. Ab Hollywood 6.0 können Sie mit `#REQ_HIDEICONS *.info`-Dateien unter AmigaOS verstecken. Beachten Sie, dass `#REQ_HIDEICONS` mit anderen Modi durch Verknüpfung in einer Oder-Bitmaske `'|'` kombiniert werden kann. `#REQ_HIDEICONS` unterstützt nur AmigaOS. (V2.0)



- Path:** Dieser Tabellen-Tag kann verwendet werden, um den Anfangspfad für das Dateialogfenster anzugeben. (V3.0)
- File:** Dieser Tabellen-Tag kann verwendet werden, um die Anfangsdatei für das Dateialogfenster anzugeben. (V3.0)
- Filters:** Mit diesem Tabellen-Tag können Sie Filter angeben, die definieren, welche Dateien im Dateialogfenster angezeigt werden sollen. Dies kann entweder eine Zeichenkette oder eine Tabelle sein.

Wenn es sich um eine Zeichenkette handelt, muss sie die Erweiterungen der Dateien enthalten, die im Dateialogfenster angezeigt werden sollen. Diese Erweiterungen müssen durch '|' Zeichen getrennt werden. Zum Beispiel: "voc|wav|8svx|16sv|iff|aiff" zeigt nur Dateien an, die eine dieser Erweiterungen haben. Stellen Sie sicher, dass Sie nicht den . (Punkt) vor der Dateierweiterung einfügen, sondern nur die eigentliche Erweiterung. Der Standardwert ist "\*", was bedeutet, dass alle Dateien angezeigt werden sollen.

Ab Hollywood 9.0 können Sie auch **Filters** auf eine Tabelle setzen, um einzelne Dateigruppen und eine Beschreibung für jede Gruppe zu definieren. Setzen Sie dazu **Filters** auf eine Tabelle, die eine beliebige Anzahl von Untertabellen enthält, von denen jede eine einzelne Gruppe von Dateien beschreibt. Jede Untertabelle muss die folgenden Elemente enthalten:

**Filter:** Eine Zeichenkette, die die Dateierweiterungen dieser Gruppe enthält. Diese Zeichenkette muss das gleiche Format haben wie oben beschrieben, d.h. die einzelnen Erweiterungen müssen durch den senkrechten Strich (|) getrennt werden, zum Beispiel "jpg|jpeg".

**Description:**

Eine Zeichenkette zur Beschreibung der Filtergruppe, z.B. "JPEG-Bilder".

**HideFilter:**

Dieses Tabellenelement ist optional. Wenn Sie es auf **True** setzen, zeigt **FileRequest()** nicht die einzelnen Dateierweiterungen an, die zur Filtergruppe gehören, sondern nur deren Beschreibung. Beachten Sie, dass dies nicht von allen Plattformen unterstützt wird. Die Voreinstellung ist **False**.

- X:** Anfängliche x-Position für das Dateialogfenster auf dem Bildschirm. Nicht alle Plattformen unterstützen dies. (V9.0)
- Y:** Anfängliche y-Position für das Dateialogfenster auf dem Bildschirm. Nicht alle Plattformen unterstützen dies. (V9.0)
- Width:** Anfangsbreite für das Dateialogfenster. Nicht alle Plattformen unterstützen dies. (V9.0)
- Height:** Anfangshöhe für das Dateialogfenster. Nicht alle Plattformen unterstützen dies. (V9.0)

**EINGABEN**

- title\$** Titel für das Dialogfenster; eine leere Zeichenfolge ("" ) verwendet den Standardtitel von **@APPTITLE**
- t** optional: Tabelle mit weiteren Argumenten (siehe oben) (V9.0)

**RÜCKGABEWERTE**

- f\$** die Auswahl des Benutzers oder eine leere Zeichenkette, wenn der Dialog abgebrochen wurde; Wenn der Dialog im Mehrfachauswahlmodus geöffnet wurde, wird eine Tabelle mit allen ausgewählten Dateien zurückgegeben

**BEISPIEL**

```
f$ = FileRequest("Select a picture", {Filters = "png|jpg|jpeg|bmp"})
If f$ = ""
    Print("Requester cancelled!")
Else
    Print("Your selection:", f$)
EndIf
```

Der obige Code fragt den Benutzer nach einer Datei und gibt das Resultat aus.

```
files = FileRequest("Select some files", {Mode = #REQ_MULTISELECT})
If files[0] = ""
    Print("Requester cancelled!")
Else
    NPrint("Path:", PathPart(files[0]))
    NPrint("Files selected:", ListItems(files) - 1)
    While files[c] <> ""
        NPrint(FilePart(files[c]))
        c = c + 1
    Wend
EndIf
```

Der Code öffnet die Dateiauswahl im Mehrfachauswahlmodus und gibt alle Dateien aus, die der Benutzer ausgewählt hat.

```
f$ = FileRequest("Select file", {
    {Description = "Image files", Filter = "png|jpg|jpeg|bmp"},
    {Description = "Audio files", Filter = "wav|mp3|mp4"},
    {Description = "All files", Filter = "*"}
})
```

Der obige Code zeigt, wie Sie mehrere Filtergruppen mit Beschreibungen verwenden.

**23.3 FontRequest****BEZEICHNUNG**

**FontRequest** – fordert den Benutzer auf, eine Schriftart zu wählen (V5.0)

**ÜBERSICHT**

```
t = FontRequest(title$[, t])
```

**FRÜHERE SYNTAX**

```
t = FontRequest(title$[, font$, size])
```

**BESCHREIBUNG**

Dieser Befehl öffnet ein Dialogfenster, welches alle Schriftarten auflistet, die aktuell im System verfügbar sind. Der Benutzer wird dann aufgefordert, eine Schrift aus dieser Liste zu wählen. Der Benutzer kann auch eine Größe für die Schrift wählen, als auch die Schriftart und Schriftfarbe. Beachten Sie, dass die Farbauswahl nicht auf jeder Plattform unterstützt wird. Das Argument `title$` gibt den Titeltext für das Dialogfenster an.

`FontRequest()` unterstützt mehrere zusätzliche Argumente. Vor Hollywood 9.0 mussten diese dann als optionale Parameter übergeben werden (siehe oben). Seit Hollywood 9.0 wird jedoch empfohlen, die neue Syntax zu verwenden, die ein einzelnes optionales Tabellenargument hat, mit dem ein oder mehrere optionale Argumente an `FontRequest()` übergeben werden können.

Die folgenden Tabellenfelder werden von diesem Befehl erkannt:

<b>Font:</b>	Verwenden Sie diesen Tabellen-Tag, um den Namen einer Schriftart anzugeben, die anfangs ausgewählt werden soll.
<b>Size:</b>	Verwenden Sie diesen Tabellen-Tag, um die Schriftgröße anzugeben, die anfänglich ausgewählt werden soll.
<b>X:</b>	Anfängliche x-Position für das Schriftarten-Dialogfenster auf dem Bildschirm. Nicht alle Plattformen unterstützen dies. (V9.0)
<b>Y:</b>	Anfängliche y-Position für das Schriftarten-Dialogfenster auf dem Bildschirm. Nicht alle Plattformen unterstützen dies. (V9.0)
<b>Width:</b>	Anfängliche Breite für das Schriftarten-Dialogfenster. Nicht alle Plattformen unterstützen dies. (V9.0)
<b>Height:</b>	Anfängliche Höhe für das Schriftarten-Dialogfenster. Nicht alle Plattformen unterstützen dies. (V9.0)

Bei der Rückkehr initialisiert `FontRequest()` eine Tabelle mit allen vom Benutzer ausgewählten Parametern und gibt diese Tabelle an das Skript zurück. In der Rückgabetable werden die folgenden Felder initialisiert:

<b>Name:</b>	Der komplette Schriftname (Familiennamen und Stil). Zum Beispiel, "Arial Bold Italic". Dies ist die Zeichenfolge, die direkt an <code>SetFont()</code> oder <code>OpenFont()</code> weitergegeben werden kann.
<b>Family:</b>	Der Familienname dieser Schrift, z.B. "Arial".
<b>Size:</b>	Enthält die gewählte Schriftgröße (z.B. 36).
<b>Weight:</b>	Die Strichdicke der Schriftart. Die folgenden Strichdickenkonstanten kommen vor:

```
#FONTWEIGHT_THIN
#FONTWEIGHT_EXTRALIGHT
```

```
#FONTWEIGHT_ULTRALIGHT
#FONTWEIGHT_LIGHT
#FONTWEIGHT_BOOK
#FONTWEIGHT_NORMAL
#FONTWEIGHT_REGULAR
#FONTWEIGHT_MEDIUM
#FONTWEIGHT_SEMIBOLD
#FONTWEIGHT_DEMIBOLD
#FONTWEIGHT_BOLD
#FONTWEIGHT_EXTRABOLD
#FONTWEIGHT_ULTRABOLD
#FONTWEIGHT_HEAVY
#FONTWEIGHT_BLACK
#FONTWEIGHT_EXTRABLACK
#FONTWEIGHT_ULTRABLACK
```

**Slant:** Die Neigung dieser Schriftart. Folgende Neigungskonstanten kommen vor:

```
#FONTSLANT_ROMAN
#FONTSLANT_ITALIC
#FONTSLANT_OBLIQUE
```

**Bold:** True wenn der Benutzer eine fette Schrift wählt.

**Italic:** True wenn der Benutzer eine kursive Schriftart wählt.

**Underline:**

True wenn der Benutzer eine unterstrichene Schriftart wählt.

**StrikeOut:**

True wenn der Benutzer einen durchgestrichenen Schriftstil wählt.

**Color:** Die Schriftfarbe im **RGB-Format**, die durch den Benutzer ausgewählt wurde.

Bitte beachten Sie, dass die Felder **Underline**, **StrikeOut** und **Color** nicht auf allen Plattformen unterstützt werden. Wenn das der Host-Schriftdialog des Betriebssystems nicht unterstützt, werden alle diese Felder auf **False** gesetzt.

## EINGABEN

**title\$** Titel des Dialogfensters

**t** optional: Tabelle mit weiteren Argumenten (siehe oben) (V9.0)

## RÜCKGABEWERTE

**t** eine Tabelle, die alle vom Benutzer gewählten Parameter enthält (siehe oben für die Beschreibung der Tabellenfelder)

## BEISPIEL

```
t = FontRequest("Select a font")
NPrint("Font:", t.name)
NPrint("Family:", t.family)
NPrint("Size:", t.size)
NPrint("Weight:", t.weight)
```

```
NPrint("Slant:", t.slant)
NPrint("Underline:", t.underline)
NPrint("Strike:", t.strikeout)
NPrint("Color:", HexStr(t.color))
```

Der obige Code öffnet ein Schriftdialogfenster und gibt alle gesammelten Informationen aus.

## 23.4 ImageRequest

### BEZEICHNUNG

ImageRequest – öffnet ein Dialogfenster, um ein Bild auszuwählen (V8.0)

### ÜBERSICHT

```
[id] = ImageRequest(id[, type])
```

### PLATTFORMEN

Nur Android

### BESCHREIBUNG

Mit diesem Befehl kann der Benutzer über einen Dialog ein Bild auswählen. Das Bild wird dann als der in `id` angegebene Pinsel gespeichert. Wenn Sie `Nil` im Argument `id` angeben, wählt `ImageRequest()` automatisch einen Identifikator für den Pinsel aus und gibt diesen an Sie zurück.

Mit dem optionalen Argument `type` können Sie die Bildquelle angeben, die verwendet werden soll, wenn der Benutzer zur Eingabe eines Bildes aufgefordert wird. Dies kann derzeit auf eine der folgenden vordefinierten Konstanten gesetzt werden:

#### #REQ\_GALLERY:

Öffnet die Galerie des Geräts und fordert den Benutzer auf, ein Bild auszuwählen, das dann als Hollywood-Pinsel an Ihr Skript zurückgegeben wird.

#### #REQ\_CAMERA:

Öffnet die Kamera des Geräts und fordert den Benutzer auf, ein Foto aufzunehmen, das dann als Hollywood-Pinsel in Ihr Skript zurückgegeben wird.

Der Standardmodus ist `#REQ_GALLERY`, d.h. `ImageRequest()` fordert den Benutzer auf, ein Bild aus der Galerie des Geräts auszuwählen.

Um herauszufinden, ob dieser Befehl fehlgeschlagen ist, weil der Benutzer den Bilddialog abgebrochen hat, verwenden Sie den Befehl `HaveObject()`, um festzustellen, ob das Pinsel-Objekt nach dem Beenden von `ImageRequest()` vorhanden ist. Wenn es nicht existiert, hat der Benutzer den Bild-Dialog abgebrochen. Siehe [Abschnitt 40.8 \[HaveObject\]](#), [Seite 837](#), für Details.

### EINGABEN

<code>id</code>	ID für den Pinsel oder <code>Nil</code> für die <b>automatische ID-Auswahl</b>
<code>type</code>	optional: zu verwendende Bildquelle; siehe oben für mögliche Modi; der Standardwert ist <code>#REQ_GALLERY</code>

**RÜCKGABEWERTE**

`id` optional: ID des Pinsels; wird nur zurückgegeben, wenn Sie `Nil` als `id` übergeben haben (siehe oben)

**BEISPIEL**

```
ImageRequest(1, #REQ_CAMERA)
If HaveObject(#BRUSH, 1)
    DisplayBrush(1, #CENTER, #CENTER)
Else
    NPrint("Requester cancelled!")
EndIf
```

Der obige Code fordert den Benutzer auf, ein Foto mit der Kamera aufzunehmen und zeigt dieses Bild an. Er prüft auch, ob der Dialog abgebrochen wurde.

**23.5 ListRequest****BEZEICHNUNG**

`ListRequest` – öffnet ein Dialogfenster mit einer Liste von Optionen (V5.0)

**ÜBERSICHT**

```
choice = ListRequest(title$, body$, choices[, t])
```

**FRÜHERE SYNTAX**

```
choice = ListRequest(title$, body$, choices[, active])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um eine Liste von Optionen für den Benutzer zu präsentieren und fordert ihn auf, einen der Listeneinträge auszuwählen. Das erste Argument `title$` ist der Titeltext für das Dialogfenster. Das zweite Argument `body$` gibt den Haupttext an, der über der Auswahlliste erscheinen soll. Das dritte Argument `choices` muss eine Tabelle sein, die eine beliebige Anzahl von Zeichenketten enthält, aus der der Anwender wählen kann.

In `choice` wird der Benutzer den Index des Listeneintrags als Rückgabewert erhalten, den er ausgewählt hat. Wenn der Benutzer kein Element auswählt oder er bricht den Dialog ab, wird -1 zurückgegeben.

`ListRequest()` unterstützt mehrere zusätzliche Argumente. Vor Hollywood 9.0 mussten diese als optionale Parameter übergeben werden (siehe oben). Seit Hollywood 9.0 wird jedoch empfohlen, die neue Syntax zu verwenden, die ein einzelnes optionales Tabellenargument hat, mit dem ein oder mehrere optionale Argumente an `ListRequest()` übergeben werden können.

Die folgenden Tabellenfelder werden von diesem Befehl erkannt:

**Active:** Mit diesem Tabellen-Tag kann eine der Auswahlmöglichkeiten in der Liste vorab ausgewählt werden. Übergeben Sie einfach den Index des Eintrags zur Vorauswahl im Tabellen-Tag **Active**. Indizes beginnen bei 0 für den ersten Eintrag und durchlaufen die Anzahl der Einträge minus 1. Wird **Active** weggelassen oder ist außerhalb des Bereichs, wird nichts vorausgewählt.

<b>X:</b>	Anfängliche x-Position für das Listendialogfenster auf dem Bildschirm. Nicht alle Plattformen unterstützen dies. (V9.0)
<b>Y:</b>	Anfängliche y-Position für das Listendialogfenster auf dem Bildschirm. Nicht alle Plattformen unterstützen dies. (V9.0)
<b>Width:</b>	Anfängliche Breite für das Listendialogfenster. Nicht alle Plattformen unterstützen dies. (V9.0)
<b>Height:</b>	Anfängliche Höhe für das Listendialogfenster. Nicht alle Plattformen unterstützen dies. (V9.0)

Ab 6.0 können Sie eine leere Zeichenfolge ("" ) im Parameter `title$` weitergeben. In diesem Fall wird das Dialogfenster den Titel aus der angegebenen Präprozessor-Anweisung `@APPTITLE` verwenden.

### EINGABEN

<code>title\$</code>	Titel des Dialogfensters
<code>body\$</code>	Haupttext, der über der Listenansicht angezeigt wird
<code>choices</code>	Tabelle, die eine Reihe von Zeichenketteneinträge der verfügbaren Optionen darstellt
<code>t</code>	optional: Tabelle mit weiteren Argumenten (siehe oben) (V9.0)

### RÜCKGABEWERTE

<code>choice</code>	Index der Benutzerauswahl oder -1, wenn der Benutzer den Dialog abbricht; der erste Eintrag beginnt bei 0 und endet bei der Anzahl der Einträge minus 1.
---------------------	--

### BEISPIEL

```
r = ListRequest("User prompt", "Which of these is not an island?",
{"Australia", "Fiji", "New Zealand", "Easter Island", "Hawaii",
"Goa", "Madagascar", "Maldives", "Seychelles"})
If r = -1
  Print("You chose the chicken exit!")
ElseIf r = 5
  Print("That's right, congratulations!")
Else
  Print("Sorry, but that is an island...")
EndIf
```

Der obige Code zeigt, wie `ListRequest()` für ein kleines Quiz verwendet werden kann.

## 23.6 PathRequest

### BEZEICHNUNG

`PathRequest` – öffnet ein Pfaddialogfenster (V2.0)

### ÜBERSICHT

```
p$ = PathRequest(title$[, t])
```

**FRÜHERE SYNTAX**

```
p$ = PathRequest(title$[, mode, defdir$])
```

**BESCHREIBUNG**

Dieser Befehl öffnet ein Pfaddialogfenster, welches dem Benutzer ermöglicht, einen Pfad auszuwählen. Sie können den Titel des Dialogfensters mit dem Argument `title$` angeben. Dies kann auch eine leere Zeichenkette ("" ) sein, um den Standardtitel zu verwenden. `PathRequest()` gibt die Pfadauswahl des Benutzers in `p$` zurück oder "", wenn der Benutzer den Pfaddialog abgebrochen hat.

`PathRequest()` unterstützt mehrere optionale Argumente. Vor Hollywood 9.0 mussten diese als optionale Parameter übergeben werden (siehe oben). Seit Hollywood 9.0 wird jedoch empfohlen, die neue Syntax zu verwenden, die ein einzelnes optionales Tabellenargument hat, mit dem ein oder mehrere optionale Argumente an `PathRequest()` übergeben werden können.

Die folgenden Tabellenfelder werden von diesem Befehl erkannt:

<b>Mode:</b>	Mit dem Tabellen-Tag <b>Mode</b> können Sie den Modus des Pfaddialogfensters festlegen. Dies kann entweder <b>#REQ_SAVEMODE</b> für den Speichermodus oder <b>#REQ_NORMAL</b> für den Normalmodus sein. Voreingestellt ist <b>#REQ_NORMAL</b> .
<b>Path:</b>	Dieser Tabellen-Tag kann verwendet werden, um den anfänglichen Pfad für das Pfaddialogfenster anzugeben. Standardmäßig ist dies das aktuelle Verzeichnis. (V3.0)
<b>X:</b>	Anfängliche x-Position für das Pfaddialogfenster auf dem Bildschirm. Nicht alle Plattformen unterstützen dies. (V9.0)
<b>Y:</b>	Anfängliche y-Position für das Pfaddialogfenster auf dem Bildschirm. Nicht alle Plattformen unterstützen dies. (V9.0)
<b>Width:</b>	Anfängliche Breite für das Pfaddialogfenster. Nicht alle Plattformen unterstützen dies. (V9.0)
<b>Height:</b>	Anfängliche Höhe für das Pfaddialogfenster. Nicht alle Plattformen unterstützen dies. (V9.0)

**EINGABEN**

<b>title\$</b>	Titel für das Dialogfenster; eine leere Zeichenfolge ("" ) verwendet den Standardtitel von <b>@APPTITLE</b>
<b>t</b>	optional: Tabelle mit weiteren Argumenten (siehe oben) (V9.0)

**RÜCKGABEWERTE**

<b>p\$</b>	die Auswahl des Benutzers oder eine leere Zeichenfolge, wenn der Dialog abgebrochen wurde
------------	---

**BEISPIEL**

```
p$ = PathRequest("Select a path")
If p$ = ""
    Print("Requester cancelled!")
Else
    Print("Your selection: ")
```



```
Print(p$)
EndIf
```

Der obige Code macht eine Benutzeranfrage für einen Pfad und gibt den Namen aus.

## 23.7 PermissionRequest

### BEZEICHNUNG

PermissionRequest – fragt den Benutzer um Erlaubnis (V8.0)

### ÜBERSICHT

```
ok = PermissionRequest(perms)
```

### PLATTFORMEN

Nur Android

### BESCHREIBUNG

Mit diesem Befehl können Sie bestimmte Berechtigungen vom Benutzer anfordern. Aus Sicherheitsgründen müssen Android-Apps zuerst die Erlaubnis des Benutzers einholen, bevor bestimmte Aktionen ausgeführt werden können. Dieser Befehl kann verwendet werden, um solche Berechtigungen vom Benutzer anzufordern. Android zeigt dann ein Dialogfeld an, in dem der Benutzer die Berechtigungen entweder annehmen oder ablehnen kann. Wenn er dies ablehnt, gibt `PermissionRequest()` `False` zurück, andernfalls wird `True` zurückgegeben.

Die Berechtigungen, die Sie anfordern möchten, müssen im Argument `perms` übergeben werden. Dies kann auf eine oder mehrere der folgenden Berechtigungsflags gesetzt werden:

#### #PERMREQ\_READEXTERNAL:

Wenn Ihre App über diese Berechtigung verfügt, kann sie Dateien vom externen Speichergerät lesen. Auf das externe Speichergerät kann über das Feld `SDCard` in der Tabelle zugegriffen werden, die von `GetSystemInfo()` zurückgegeben wird. Android-Apps dürfen standardmäßig nicht vom externen Speichergerät lesen.

#### #PERMREQ\_WRITEEXTERNAL:

Wenn Ihre App über diese Berechtigung verfügt, kann sie Dateien auf/von dem externen Speichergerät schreiben und lesen. Auf das externe Speichergerät kann über das Feld `SDCard` in der Tabelle zugegriffen werden, die von `GetSystemInfo()` zurückgegeben wird. Standardmäßig dürfen Android-Apps nicht auf das externe Speichergerät schreiben. Beachten Sie, dass `#PERMREQ_WRITEEXTERNAL` `#PERMREQ_READEXTERNAL` impliziert, sodass Sie bei Verwendung dieses Flags `#PERMREQ_READEXTERNAL` nicht einstellen müssen.

Um mehrere Berechtigungen gleichzeitig anzufordern, kombinieren Sie sie einfach mit dem **Bitweise-Oder-Operator**.

Beachten Sie, dass dieser Befehl nur benötigt wird, wenn Sie eigenständige APKs mit dem Hollywood APK Compiler kompilieren. Wenn Sie den Hollywood Player verwenden, fordert der Hollywood Player automatisch die Berechtigung `#PERMREQ_WRITEEXTERNAL` für Sie an, sodass Sie dies nicht manuell tun müssen.

**EINGABEN**

**perms** eine oder mehrere Berechtigungen zum Anfordern (mögliche Werte siehe oben)

**RÜCKGABEWERTE**

**ok** **True**, wenn der Benutzer die Berechtigung erteilt hat, **False**, wenn er sie abgelehnt hat

**BEISPIEL**

```
If PermissionRequest(#PERMREQ_WRITEEXTERNAL)
    t = GetSystemInfo()
    StringToFile("Hello World", FullPath(t.SDCard, "test.txt"))
Else
    NPrint("Sorry, no permission!")
EndIf
```

Der obige Code versucht vom Benutzer eine Erlaubnis zu erhalten, auf das externe Speichergerät zu schreiben. Wenn der Benutzer diese Berechtigung erteilt, schreibt der Code eine Datei mit dem Namen `test.txt` auf das externe Speichergerät.

## 23.8 StringRequest

**BEZEICHNUNG**

**StringRequest** – bittet den Benutzer eine Zeichenkette einzugeben (V2.0)

**ÜBERSICHT**

```
s$, ok = StringRequest(title$, body$[, t])
```

**FRÜHERE SYNTAX**

```
s$, ok = StringRequest(title$, body$[, def$, type, maxchars, password])
```

**BESCHREIBUNG**

Dieser Befehl öffnet ein Dialogfenster, welches den Benutzer auffordert, eine Zeichenkette einzugeben. Sie können den Titel für das Zeichenketten-Dialogfenster in `title$` und den Haupttext in `body$` angeben. Wenn Sie eine leere Zeichenfolge ("" ) in `title$` übergeben, verwendet das Zeichenketten-Dialogfenster den in der Präprozessor-Anweisung **@APPTITLE** angegebenen Titel.

**StringRequest()** gibt den vom Benutzer eingegebene Zeichenkette in `s$` zurück, wenn der Benutzer das Zeichenketten-Dialogfenster bestätigt. Wenn der Benutzer diesen Dialog abbricht, wird eine leere Zeichenkette zurückgegeben. Mit dem zweiten Rückgabewert `ok` können Sie feststellen, ob der Benutzer die Schaltfläche 'OK' gedrückt hat oder nicht. Dies ist normalerweise nur erforderlich, wenn Ihr Programm eine leere Zeichenkette bei 'OK' zulässt. In diesem Fall müssen Sie auch den zweiten Rückgabewert überprüfen.

**StringRequest()** unterstützt mehrere optionale Argumente. Vor Hollywood 9.0 mussten diese als optionale Parameter übergeben werden (siehe oben). Seit Hollywood 9.0 wird jedoch empfohlen, die neue Syntax zu verwenden, die ein einzelnes Tabellenargument hat, mit dem ein oder mehrere optionale Argumente an **StringRequest()** übergeben werden können.

Die folgenden Tabellenfelder werden von diesem Befehl erkannt:

**Type:** Mit diesem Tabellen-Tag kann festgelegt werden, welche Zeichen der Benutzer eingeben darf. Dies kann derzeit **#NUMERICAL** nur für Zahlen oder **#ALL** für keine Beschränkung der eingebbaren Zeichen sein. Die Voreinstellung ist **#ALL**.

**Password:** Setzen Sie diesen Tabellen-Tag auf **True**, um den Dialog in den Passwortmodus zu versetzen. In diesem Fall wird die Eingabe des Benutzers ausgeblendet. Der Standardwert ist **False**.

**MaxLength:** Dieser Tabellen-Tag kann verwendet werden, um die Anzahl der Zeichen anzugeben, die der Benutzer eingeben darf. Der Standardwert ist 0, was bedeutet, dass die Anzahl der Zeichen, die der Benutzer eingeben kann, unbegrenzt ist.

**Text:** Dieser Tabellen-Tag kann verwendet werden, um den Standardtext für das Zeichenketten-Dialogfenster anzugeben. Der hier angegebene Text wird zunächst im Texteingabe-Widget des Zeichenketten-Dialogfenster angezeigt.

**X:** Anfängliche x-Position für das Zeichenketten-Dialogfenster auf dem Bildschirm. Nicht alle Plattformen unterstützen dies. (V9.0)

**Y:** Anfängliche y-Position für das Zeichenketten-Dialogfenster auf dem Bildschirm. Nicht alle Plattformen unterstützen dies. (V9.0)

**Width:** Anfängliche Breite für das Zeichenketten-Dialogfenster. Nicht alle Plattformen unterstützen dies. (V9.0)

**Height:** Anfängliche Höhe für das Zeichenketten-Dialogfenster. Nicht alle Plattformen unterstützen dies. (V9.0)

Bitte beachten Sie, dass dieser Befehl die Installation der `reqtools.library` auf AmigaOS 3, MorphOS und AROS erfordert. Unter AmigaOS 4 funktioniert der `StringRequest()` ohne die `reqtools.library`.

## EINGABEN

**title\$** Titel für das Dialogfenster

**body\$** Text für den Haupttext

**t** optional: Tabelle mit weiteren Argumenten (siehe oben) (V9.0)

## RÜCKGABEWERTE

**s\$** die Zeichenfolge, die vom Benutzer eingegeben wurde oder "", wenn der Dialog abgebrochen wurde

**ok** **True** wenn der Benutzer die Schaltfläche "OK" drückt, sonst **False** (V4.5)

## BEISPIEL

```
a$ = StringRequest("My Program", "Please enter your name!")
Print("Hello,", a$, "!")
```

Der Code fragt den Benutzer nach seinem Namen und gibt diesen aus.

## 23.9 SystemRequest

### BEZEICHNUNG

SystemRequest – öffnet ein Auswahldialogfenster

### ÜBERSICHT

```
res = SystemRequest(title$, body$, buttons$[, icon])
```

### BESCHREIBUNG

Dieser Befehl öffnet ein Standard-Systemdialogfenster, welches die Nachricht **body\$** zeigt und dem Benutzer auch erlaubt, eine Auswahl über die Schaltflächen in **buttons\$** zu treffen. Trennen Sie diese Schaltflächen mit einem "|". Der Rückgabewert zeigt Ihnen, welche Schaltfläche vom Benutzer ausgewählt wurde. Bitte beachten Sie, dass die am weitesten rechts stehende Schaltfläche immer den Wert **False** (0) hat, weil sie in der Regel als "Abbruch"-Schaltfläche verwendet wird. Wenn Sie zum Beispiel drei Schaltflächen "Eins|Zwei|Drei" zur Auswahl haben, hat der Knopf "Drei" den Rückgabewert 0, "Zwei" Rückgabewert 2 und "Eins" Rückgabewert 1.

Neu in Hollywood 4.0: Sie können mit dem optionalen Argument **icon** dem Fenster ein kleines Piktogramm hinzufügen. Die folgenden Piktogramme sind möglich:

```
#REQICON_NONE:
    Kein Piktogramm

#REQICON_INFORMATION:
    Ein Informationszeichen

#REQICON_ERROR:
    Ein Fehlerzeichen

#REQICON_WARNING:
    Ein Warnungszeichen

#REQICON_QUESTION:
    Ein Fragezeichen
```

Bitte beachten Sie, dass derzeit das Auswahldialogfenster nicht auf jeder Plattform unterstützt wird, auf der Hollywood läuft.

Ab Hollywood 6.0 kann **SystemRequest()** die folgenden Schaltflächenangaben für bestimmte systemspezifische Schaltflächen abbilden: "OK", "OK|Abbrechen", "Ja|Nein" und "Ja|Nein|Abbrechen". Dies kann dazu führen, dass die Schaltflächen plötzlich in der Sprache des Benutzers anstatt der Versionen erscheinen, die Sie an **SystemRequest()** übergeben haben. Es ist aber auch möglich, dass die Reihenfolge der Knöpfe geändert wird. Z.B. unter macOS wird die Schaltfläche "OK" in der Regel rechts neben dem Knopf "Abbrechen" platziert, während es auf anderen Systemen umgekehrt ist. Trotzdem sind die Rückgabewerte immer konsistent, d.h. "OK" oder "Ja" hat immer einen Rückgabewert von 1, während "Abbrechen" oder "Nein" einen Rückgabewert von 0 hat. Die einzige Ausnahme ist "Ja|Nein|Abbrechen", wo "Nein" einen Rückgabewert von 2 hat, da es auch eine "Abbrechen"-Schaltfläche gibt, die einen Rückgabewert von 0 hat.

Seit Hollywood 6.0 ist es auch möglich, eine leere Zeichenfolge ("" ) im ersten Parameter **title\$** zu übergeben. In diesem Fall wird das Dialogfenster den Titel aus der angegebenen Präprozessor-Anweisung **@APTITLE** verwenden.

**EINGABEN**

`title$`      Title für das Dialogfenster  
`body$`      Text, der als Nachricht im Dialogfenster erscheint  
`buttons$`    eine oder mehrere Schaltflächen, die der Benutzer drücken kann  
`icon`       optional: Piktogramm, welches im Dialogfenster angezeigt werden kann  
              (Standardeinstellung ist `#REQICON_NONE`) (V4.0)

**RÜCKGABEWERTE**

`res`         die Schaltfläche, die durch den Benutzer gedrückt wurde

**BEISPIEL**

```
sel = SystemRequest("Pizza Service", "Select your pizza!",  
                    "Prosciutto e funghi|Calzone|Margerita|Hawaii")  
  
Switch sel  
Case 1:  
    Print("1x Prosciutto e funghi = 8 Euro")  
Case 2:  
    Print("1x Calzone = 10 Euro")  
Case 3:  
    Print("1x Margerita = 9 Euro")  
Case 0:  
    Print("1x Hawaii = 12 Euro")  
EndSwitch
```

Der obige Code fragt den Benutzer nach einer Pizza und zeigt den Preis der Pizza an.



## 24 Displaybibliothek

### 24.1 Übersicht

Das Display ist ein Bereich auf dem Bildschirm, in den Ihr Skript zeichnen kann. In der Regel ist ein Display ein Fenster auf Ihrem Desktop-Bildschirm, aber es kann auch im Vollbildmodus den gesamten Monitorbereich ausfüllen. In Hollywood ist einem Display immer ein Hintergrundbild (BGPic) zugewiesen. Das Hintergrundbild ist das, was anfangs dem Benutzer gezeigt wird, wenn Ihr Display sichtbar wird.

Das Hintergrundbild muss immer die gleiche Größe haben wie das Display. Wenn Sie also die Displaygröße zum Beispiel mit `ChangeDisplaySize()` ändern, wird Ihr Hintergrundbild automatisch auf die neuen Dimensionen skaliert. Wie schon geschrieben, die Displaygröße ist immer die gleiche wie die aktuelle Hintergrundbildgröße. Wenn Ihr Fenster in der Größe veränderbar ist, dann kann der Benutzer auch die Displaygröße anpassen. Wenn er das tut, wird Hollywood intern mit `ChangeDisplaySize()` das Hintergrundbild an die neuen Abmessungen anzupassen.

Wenn ein neues Hintergrundbild z.B. durch den Befehl `DisplayBGPic()` angezeigt wird und sich die Dimensionen des neuen Hintergrundbild von den Dimensionen des aktuellen Hintergrundbildes unterscheiden, dann wird Ihr Display auch an die neuen Dimensionen angepasst werden.

Beim starten des Programmes zeigt Hollywood das Hintergrundbild an, dem die ID 1 zugewiesen wurde. Wenn Sie mit der Präprozessor-Anweisung `@BGPIC` kein Hintergrundbild mit dem Identifikator 1 deklariert haben, wird Hollywood automatisch dieses Hintergrundbild für Sie erstellen und Ihrem Display zuweisen. Das Hintergrundbild verwendet den selben Füllstil und die gleiche Abmessungen, welche Sie mit der Präprozessor-Anweisung `@DISPLAY` für das Display 1 in Ihrem Skript angegeben haben.

Displays können auch transparente Bereiche haben. Hollywood unterstützt Displays mit Alpha-Transparenz (256 Transparenzstufen) und monochrome Transparenz.

Hier ist ein minimales Skript, das ein Display in der Größe 1024x768 mit der Farbe Rot erzeugt:

```
@DISPLAY {Width = 1024, Height = 768, Color = #RED}
WaitLeftMouse
End
```

Displays können auch automatisch ihre Inhalte mit einem der in Hollywood eingebauten Skalierungssysteme skalieren: Autoskalierung oder Ebenenskalierung. Bei der Aktivierung eines dieser Skalierungssysteme wird das Skript davon ausgehen, dass es in seiner ursprünglichen Auflösung läuft, obwohl das Display mit dem ausgewählten Skalierungssystem in einer völlig anderen Auflösung ausgeführt wird. Siehe [Abschnitt 24.24 \[Skalierungssysteme\]](#), [Seite 402](#), für Details.

Hollywood-Displays können auch im Palettenmodus ausgeführt werden. Die Displays wechseln in den Palettenmodus, wenn ein BGPic angezeigt wird, das eine Palette verwendet. Palettenmodus-Displays verhalten sich ganz anders als normale True-Color-Displays, daher sind einige wichtige Dinge zu beachten, wenn Sie Displays im Palettenmodus verwenden. Siehe [Abschnitt 24.16 \[Palettenmodus-Displays\]](#), [Seite 393](#), für Details.

Die Displaybibliothek von Hollywood unterstützt auch mehrere Monitore. Sie können den Monitor angeben, auf dem ein Display geöffnet werden soll. Siehe [Abschnitt 24.13 \[Unterstützung mehrerer Monitore\]](#), [Seite 389](#), für Details.

## 24.2 ActivateDisplay

### BEZEICHNUNG

ActivateDisplay – aktiviert ein Display (V4.5)

### ÜBERSICHT

ActivateDisplay(id[, nofront])

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um das festgelegte Display zu aktivieren. "Aktivierung" eines Display bedeutet, dass Hollywood den Window-Manager des Host-Betriebssystems mitteilt, dass es den Fokus auf dieses Display richten soll. Eine Displayaktivierung bedeutet nicht, dass dieses Display das aktuelle Ausgabeziel für die Grafikbibliothek von Hollywood wird. Wenn Sie ein Display als aktuelles Ausgabeziel auswählen möchten, verwenden Sie den Befehl [SelectDisplay\(\)](#), der auch das Display aktiviert, wenn Sie dies nicht explizit verbieten.

Siehe [Abschnitt 24.19 \[SelectDisplay\]](#), [Seite 396](#), für mehr Informationen über den Unterschied zwischen aktiven Displays und Displays, die als das aktuelle Ausgabeziel ausgewählt wurde.

Ab Hollywood 5.0 gibt es ein neues optionales Argument namens `nofront`. Wenn Sie dies auf `True` setzen, wird das Display aktiviert, aber es wird nicht auf das vorderste Fenster der Stapelreihenfolge verschoben werden. Dieses Argument wird nur auf AmigaOS und kompatiblen Systemen behandelt, weil aktive Fenster im Hintergrund auf anderen Betriebssystemen nicht unterstützt werden.

### EINGABEN

<code>id</code>	Identifikator des Display, welches aktiviert wird
<code>nofront</code>	optional: <code>True</code> , wenn das Display nicht hervorgeholt werden soll (voreingestellt ist <code>False</code> , womit das Display nach vorne geholt wird) (V5.0)

## 24.3 BACKFILL

### BEZEICHNUNG

BACKFILL – konfiguriert die Hintergrundfüllung des Skripts / VERALTET (V4.5)

### ÜBERSICHT

@BACKFILL table

### BESCHREIBUNG

Wichtiger Hinweis: Diese Präprozessor-Anweisung ist seit Hollywood 6.0 veraltet. Als Hollywood 6.0 die Unterstützung von mehreren Monitoren eingeführt hat, könnte es auch mehrere Hintergrundfüllungen (eine für jedes Display auf separaten Monitoren) haben. Deshalb sollte von nun an Hintergrundfüllungen mit der Präprozessor-Anweisung [@DISPLAY](#) oder dem Befehl [CreateDisplay\(\)](#) definiert werden. Sie können immer noch



diesen **@BACKFILL** Präprozessor-Anweisung verwenden, aber sie wird nur das erste Display beeinflussen.

Diese Präprozessor-Anweisung kann verwendet werden, um die Einstellungen der Hintergrundfüllung für das Skript zu konfigurieren. Die Hintergrundfüllung kann je nach **Type** (siehe Tabellen-Felder unten) ein Teil oder der den ganzen Bereich des Displays einnehmen. Sie können eine statische Farbe, einen Farbverlauf, ein Bild oder eine Textur als Hintergrundfüllung verwenden. Vor Hollywood 4.5 wurden Hintergrundfüllungen mit der Präprozessor-Anweisung **@DISPLAY** konfiguriert (was ab Hollywood 6.0 wieder empfohlen wird). Ab Hollywood 4.5 wurden jedoch mehrere Displays eingeführt. Deshalb wurde es erforderlich, eine eigene Präprozessor-Anweisung zu integrieren, weil es nur eine einzige Hintergrundfüllung pro Skript geben kann.

Sie übergeben diesem Befehl eine Tabelle. Die folgenden Tabellen-Felder können Sie derzeit verwenden:

**Type:** Dieses Feld ist obligatorisch und muss als Zeichenkette **Color**, **Gradient**, **Texture** oder **Picture** beinhalten.

**Color:** Wenn Sie **Color** als **Type** für die Hintergrundfüllung angegeben, übergeben Sie hier die gewünschte Hintergrundfüllfarbe.

**StartColor, EndColor:**

Wenn Sie **Gradient** als **Type** angegeben haben, verwenden Sie diese beiden Tags, um die Start- und Endfarben für den Farbverlauf zu definieren.

**Brush:** Wenn Sie **Texture** oder **Picture** als **Type** angegeben haben, geben Sie den Identifikator des Pinsels hier an. Wenn Sie den Dateinamen direkt übergeben möchten, verwenden Sie stattdessen den Tag **BrushFile**.

**X,Y:** Wenn Sie **Picture** als **Type** für die Hintergrundfüllung angegeben haben, können Sie diese beiden Tags verwenden, um das Bild auf dem Bildschirm zu positionieren. Beide sind standardmäßig auf **#CENTER** voreingestellt.

**BrushFile:**

Wenn Sie **Texture** oder **Picture** als **Type** angegeben haben, können Sie hier den Dateinamen des Pinsel verwenden. Die hier angegebene Datei wird mit dem Applet/der ausführbaren Datei bei der Kompilierung verknüpft werden, außer Sie setzen **LinkBrushFile** auf **False**. Wenn Sie statt einem Dateinamen eine Pinsel-ID übergeben möchten, verwenden Sie den Tag **Brush**. (V4.0)

**LinkBrushFile:**

Wenn **BrushFile** angegeben wurde, kann dieser Tag verwendet werden, um die Pinseldatei in das Applet/die ausführbare Datei beim Kompilierung einzubinden/verknüpfen. Der Standardwert ist **True**, was bedeutet, dass die Pinseldatei verknüpft wird. (V4.0)

**Transparency:**

Wenn Sie als **Type Picture** gewählt haben, können Sie hier eine **RGB-Farbe** angeben, welche transparent dargestellt werden soll. Der Standardwert ist **#NOTTRANSPARENCY**. (V4.0)

**ScalePicture:**

Wenn Sie als **Type Picture** gewählt haben, können Sie diesen Tag verwenden, um festzulegen, ob das Hintergrundfüllbild an die Dimensionen des Displays skaliert wird (**True**) oder nicht (**False**). Der Standardwert ist **False**. (V4.0)

**SmoothScale:**

Setzen Sie diesen Tag auf **True**, wenn das Hintergrundfüllbild bei der Skalierung Antialias für die Kantenglättung benutzen soll. Dieser Tag wird nur dann berücksichtigt, wenn **ScalePicture** auf **True** gesetzt wurde. Der Standardwert ist **False**. (V6.0)

Alternativ kann die Hintergrundfüllung auch von der **Konsole** aus konfiguriert werden. Falls Sie diese Möglichkeit deaktivieren möchten, sollten Sie Ihre Skripte mit dem **-locksettings** Konsolenargument kompilieren.

Vielleicht wollen Sie auch den Tag **HideTitleBar** bei **@SCREEN** verwenden. Wenn Sie **HideTitleBar** angeben, wird die Hintergrundfüllung auch die aktuelle Bildschirm-Titelleiste (Amiga) oder Finder-Menüleiste einnehmen (macOS).

**EINGABEN**

**table**      Tabelle, die den Stil der Hintergrundfüllung definiert

**BEISPIEL**

```
@BACKFILL {Type = "Gradient", StartColor = #BLACK, EndColor = #BLUE}
```

Dies installiert einen Farbverlauf von Schwarz nach Blau.

## 24.4 ChangeDisplayMode

**BEZEICHNUNG**

ChangeDisplayMode – wechselt zwischen Fenster- und Vollbildmodus (V4.5)

**ÜBERSICHT**

```
ChangeDisplayMode(mode[, table])
```

**FRÜHERE SYNTAX**

```
ChangeDisplayMode(mode[, width, height, table])
```

**BESCHREIBUNG**

Mit diesem Befehl kann der Anzeigemodus auf den im Parameter **mode** angegebenen Modus geändert werden. Dies kann einer der folgenden Modi sein:

**#DISPMODE\_FULLSCREEN:**

Wechselt in den Vollbildmodus. Beachten Sie, dass dadurch die Auflösung des Monitors umgeschaltet wird, was möglicherweise nicht auf allen Systemen unterstützt wird. Alternativ können Sie auch **#DISPMODE\_FULLSCREENSCALE** (siehe unten) verwenden, wodurch das Display einfach auf die Auflösung des Monitors skaliert wird. Wenn Sie **#DISPMODE\_FULLSCREEN** verwenden, können Sie die gewünschte Bildschirmauflösung in den Tags **Width** und **Height** des optionalen Tabellenarguments übergeben (siehe unten). Wenn

Sie `Width` und `Height` nicht angeben, wird automatisch die beste Bildschirmauflösung für die aktuellen Abmessungen des Displays gewählt. Ist das Display bereits im Vollbildmodus und Sie übergeben `#DISPMODE_FULLSCREEN` im Argument `mode`, kann `ChangeDisplayMode()` verwendet werden, um die aktuelle Bildschirmauflösung in eine andere zu wechseln.

**#DISPMODE\_WINDOWED:**

Wechselt in den Fenstermodus. Dieser Modus kann verwendet werden, um ein Display wieder in den Fenstermodus zu schalten. Es ist natürlich nur sinnvoll, diesen Modus auf Displays zu verwenden, die derzeit im Vollbildmodus sind.

**#DISPMODE\_FULLSCREENSCALE:**

Wechselt in den skalierten Vollbildmodus. In diesem Modus wird das Display im Vollbildmodus angezeigt, ohne die Auflösung des Monitors zu ändern. Stattdessen wird die Grafik des Displays auf die aktuelle Auflösung des Monitors skaliert. Somit füllen sie den gesamten Bildschirm aus, obwohl der Monitor seine Auflösung nicht geändert hat. Standardmäßig wird die Autoskalierung von Hollywood für die Skalierung verwendet, aber Sie können den Tag `LayerScale` im optionalen Tabellenargument (siehe unten) auf `True` setzen, um stattdessen die Ebenenskalierung zu verwenden. Siehe [Abschnitt 24.24 \[Skalierungssysteme\]](#), [Seite 402](#), für Details. Beachten Sie jedoch, dass `#DISPMODE_FULLSCREENSCALE` auf Plattformen, die keine hardwarebeschleunigte Skalierung unterstützen, ziemlich langsam werden kann. (V9.0)

**#DISPMODE\_MODESWITCH:**

Dies schaltet zwischen den Display-Modi um. Wenn das Display derzeit im Fenstermodus ist, wechselt es zum Vollbild. Wenn das Display hingegen derzeit im Vollbildmodus angezeigt wird, schaltet es in den Fenstermodus um. Beachten Sie, dass der von `#DISPMODE_MODESWITCH` gewählte Vollbildmodus sowohl `#DISPMODE_FULLSCREEN` als auch `#DISPMODE_FULLSCREENSCALE` sein kann. `#DISPMODE_MODESWITCH` verwendet dieselbe Logik wie das Tastaturkürzel `ALT+RETURN`, der ein Hollywood-Display zwischen Vollbild- und Fenstermodus umschaltet. Weitere Informationen finden Sie in der Dokumentation zum Tag `ScaleSwitch` in der Dokumentation der Präprozessor-Anweisung `@DISPLAY`. Siehe [Abschnitt 24.8 \[DISPLAY\]](#), [Seite 370](#), für Details. (V9.0)

Ab Hollywood 6.0 können Sie `ChangeDisplayMode()` eine optionale Tabelle übergeben, womit Sie folgende Optionen konfigurieren können:

**Height:** Wenn `mode #DISPMODE_FULLSCREEN` ist, kann dieser Tag verwendet werden, um die Höhe der Auflösung anzugeben, auf die der Monitor umgeschaltet werden soll. Sie können hier auch die spezielle Konstante `#NATIVE` übergeben, um anzugeben, dass Hollywood die native Höhe des Monitors verwenden soll.

**LayerScale:**

Wenn `mode #DISPMODE_FULLSCREENSCALE` ist, kann dieser Tag verwendet werden, damit Hollywood die Ebenenskalierung anstelle der Autoskalierung verwendet. Siehe [Abschnitt 24.24 \[Skalierungssysteme\]](#), [Seite 402](#), für Details. Voreingestellt ist `False`. (V9.0)

**KeepProportions:**

Wenn mode `#DISPMODE_FULLSCREENSCALE` ist, können Sie die proportionale Skalierung aktivieren, indem Sie diesen Tag auf `True` setzen. Die Voreinstellung ist `False`. (V9.0)

**SmoothScale:**

Wenn mode `#DISPMODE_FULLSCREENSCALE` ist, können Sie die interpolierte Skalierung aktivieren, indem Sie diesen Tag auf `True` setzen. Die Voreinstellung ist `False`. (V9.0)

**Monitor:** Mit diesem Tag können Sie den Monitor angeben, der verwendet werden soll. Die Monitore werden von 1 an gezählt, das ist der primäre Monitor. Standardmäßig wird der Monitor verwendet, der momentan als aktives Display zugeordnet ist.

**Backfill:**

Dieser Tag kann die Hintergrundfüllung für dieses Display konfigurieren. Die Tabelle, die Sie hier angeben müssen, ist dieselbe, wie sein Pendant des Tags `Backfill` von der Präprozessor-Anweisung `@DISPLAY`. Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

Beachten Sie, dass es ab Hollywood 6.0 möglich ist, mehr als ein Display im Vollbildmodus zu haben, da Hollywood 6.0 die Unterstützung für mehrere Monitore einführt. Dadurch ist es möglich, mehrere Displays im Vollbildmodus auf separaten Bildschirmen laufen zu lassen.

Um herauszufinden, ob der gewünschte Display-Modus vom aktuellen Monitor verarbeitet werden kann, verwenden Sie den Befehl `GetDisplayModes()`.

**EINGABEN**

`mode`            Modus zum Umschalten (siehe oben)

`table`           optional: Tabelle für weitere Optionen (siehe oben) (V6.0)

**BEISPIEL**

```
ChangeDisplayMode(#DISPMODE_FULLSCREEN, {Width = 1024, Height = 768})
NPrint("We are now in full screen mode. Press left mouse to return\n" ..
      "to windowed mode.")
WaitLeftMouse
ChangeDisplayMode(#DISPMODE_WINDOWED)
NPrint("We are back in windowed mode now.")
WaitLeftMouse
```

Der obige Code geht in 1024x768 Vollbildmodus, wartet, bis die linke Maustaste gedrückt wird und wechselt wieder zurück in den Fenstermodus.

## 24.5 ChangeDisplaySize

**BEZEICHNUNG**

`ChangeDisplaySize` – ändert die Größe des aktuellen Displays

**ÜBERSICHT**

`ChangeDisplaySize(width, height[, args])`

**BESCHREIBUNG**

Dieser Befehl ändert die Abmessungen des aktuell aktiven Displays auf die angegebenen Dimensionen. Das Hintergrundbild wird auf die neue Dimension skaliert werden. Deshalb können Sie mit diesem Befehl auch die Hintergrundbilder skalieren.

Neu in V2.0: Sie können `#KEEPASPRAT` entweder in Breite oder Höhe einsetzen. Hollywood wird dann die Größe automatisch anpassen, indem es das Seitenverhältnis des Hintergrundbildes berechnet.

Ab Version 2.0 können Sie für die Höhe und Breite auch eine Prozentzahl in Form einer Zeichenkette angeben. Z.B. "50%".

Neu in Hollywood 4.0: Sie können eine Tabelle als drittes optionales Argument übergeben. Derzeit kann die Tabelle folgende Tags enthalten:

**X:** Gibt die neue X-Position für das Display an. Wenn das Display seine aktuelle X-Position behalten soll, verwenden Sie die spezielle Konstante `#KEEPPOSITION`. Der Standardwert ist `#CENTER`.

**Y:** Gibt die neue Y-Position für das Display an. Wenn das Display seine aktuelle Y-Position behalten soll, verwenden Sie die spezielle Konstante `#KEEPPOSITION`. Der Standardwert ist `#CENTER`.

**Smooth:** Bei `True` wird die Grafik mit Antialias skaliert. Der Standardwert ist `False`.

Ab Hollywood 7.0 können Sie den Argumenten `width` und `height` auch auf die spezielle Konstante `#NATIVE` setzen. In diesem Fall wird Hollywood die Dimensionen des Displays vom Host-Gerät verwenden.

**EINGABEN**

`width` neue Breite des Displays

`height` neue Höhe des Displays

`args` optional: weitere Optionen zum Konfigurieren (V4.0)

**BEISPIEL**

`ChangeDisplaySize(320, 240)`

Das ändert die Displaygröße auf 320x240.

## 24.6 CloseDisplay

**BEZEICHNUNG**

`CloseDisplay` – schließt ein Display (V4.5)

**ÜBERSICHT**

`CloseDisplay(id)`

**BESCHREIBUNG**

Dieser Befehl schließt ein geöffnetes Display. Bitte beachten Sie, dass es nicht das Display aus dem Speicher entfernt. Sie können es immer noch mit `SelectDisplay()` als Ausgabeziel

festlegen, auch wenn es geschlossen ist. Wieder sichtbar wird das Display mit dem Befehl `OpenDisplay()`.

Wenn Sie das Display nicht vollständig schließen, sondern nur minimieren wollen, können Sie den Befehl `HideDisplay()` verwenden. Wenn Sie ein Display ganz aus dem Speicher entfernen wollen, müssen Sie `FreeDisplay()` anstelle von `Close()` benutzen.

## EINGABEN

`id`            Identifikator des Displays, das geschlossen werden soll

## 24.7 CreateDisplay

### BEZEICHNUNG

`CreateDisplay` – erstellt ein neues Display (V4.5)

### ÜBERSICHT

```
[id] = CreateDisplay(id[, table])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um ein neues Display zu erstellen, welches Sie mit dem Befehl `OpenDisplay()` öffnen dürfen. Sie können entweder eine ID oder `Nil` für das neue Display übergeben. Wenn Sie `Nil` angegeben haben, wird `CreateDisplay()` eine ID-Nummer für dieses Display zurückgeben.

Darüber hinaus sollten Sie eine Tabelle als zweites Argument übergeben, mit welcher Sie den Stil für das neue Display konfigurieren können. Bitte beachten Sie, dass jedes Display ein `BGPic` haben muss. Daher ist es ratsam, dass Sie den Tag `BGPic` in der optionalen Tabelle angeben, wenn Sie ein neues Display erstellen. Wenn Sie den `BGPic`-Tag nicht angeben, wird `CreateDisplay()` automatisch ein neues `BGPic` für das neue Display erstellen. Dieses neue `BGPic` wird die in `Width/Height` (Breite/Höhe) angegebenen Größe übernehmen und es wird nach dem in `FillStyle` angegebenen Stil gefüllt werden. Wenn Sie aber den `BGPic`-Tag angeben, werden `Width`, `Height` und `FillStyle` ignoriert.

Beachten Sie auch, dass das gleiche `BGPic` nicht mit mehreren Displays in Verbindung gebracht werden kann. Jedes `BGPic` darf nur einem einzigen Display zugewiesen werden. Es ist zum Beispiel nicht möglich, `BGPic 1` Display 1 und 2 zuzuordnen. Machen Sie sonst einfach eine Kopie des `BGPic` mit dem Befehl `CopyBGPic()`, wenn Sie ein einzelnes `BGPic` mit mehreren Displays verwenden müssen.

Die optionale Tabelle erkennt die folgenden Tags:

**BGPic:**        Gibt das `BGPic` an, welches mit dem neuen Display arbeiten soll. Sie müssen entweder diesen Tag oder `Width/Height` und `FillStyle` angeben. Siehe oben für weitere Hinweise.

**Width, Height:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. Wird ignoriert, wenn Sie den `BGPic`-Tag gesetzt haben.

**X, Y:**        Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**Mode:**        Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**Title:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**Borderless:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**Sizeable:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**Fixed:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**Backfill:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**ScrWidth, ScrHeight:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**ScrDepth:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**NoHide:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**NoModeSwitch:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**NoClose:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**Active:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**HidePointer:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**UseQuartz:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**ScaleMode:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**ScaleWidth, ScaleHeight:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**SmoothScale:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**DragRegion:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**SizeRegion:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**Layers:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**FitScale:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V4.7)

**KeepProportions:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V4.7)

**FillStyle:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.. Wird ignoriert, wenn Sie den `BGPic`-Tag gesetzt haben. Voreingestellt ist `#FILLCOLOR`. (V5.0)



- Color:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.
- TextureBrush:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.0)
- TextureX, TextureY:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.0)
- GradientStyle:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.0)
- GradientAngle:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.0)
- GradientStartColor, GradientEndColor:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.0)
- GradientCenterX, GradientCenterY:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.0)
- GradientBalance:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.0)
- GradientBorder:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.0)
- GradientColors:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.0)
- KeepScreenOn:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.1)
- PubScreen:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.2)
- HideFromTaskbar:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.3)
- HideOptionsMenu:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.3)
- Orientation:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.3)
- DisableBlanker:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V6.0)
- Menu:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V6.0)
- Monitor:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V6.0)
- XServer:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V6.0)
- ScreenTitle:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V6.0)
- ScreenName:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V6.0)



**RememberPosition:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V6.1)

**Maximized:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V7.0)

**TrapRMB:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V7.0)

**NoScaleEngine:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V7.0)

**NoLiveResize:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V7.0)

**NativeUnits:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V7.0)

**AlwaysOnTop:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V7.1)

**NoCyclerMenu:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V8.0)

**HideTitleBar:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V8.0)

**Subtitle:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V8.0)

**SingleMenu:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V8.0)

**ScaleFactor:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V8.0)

**ImmersiveMode:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V9.0)

**Palette:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V9.0)

**FillPen:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V9.0)

**SoftwareRenderer:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V9.0)

**VSync:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V9.0)

**ScaleSwitch:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V9.0)

**UserTags:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V10.0)

Nachdem das Display erfolgreich erstellt wurde, können Sie es öffnen, indem Sie [OpenDisplay\(\)](#) aufrufen. Möchten Sie mit Grafikbefehlen hineinzeichnen, verwenden Sie den Befehl [SelectDisplay\(\)](#) und zum Schließen den Befehl [CloseDisplay\(\)](#).

Siehe [Abschnitt 24.19 \[SelectDisplay\]](#), Seite 396, für die Verwendung mehrerer Displays mit Hollywood.

Dieser Befehl ist auch als Präprozessor vorhanden: Verwenden Sie **@DISPLAY** um beim Start ein Display zu erstellen!

### EINGABEN

**id**            Identifikator für das Display oder **Nil** für die **automatische ID-Zuweisung**  
**table**        optional: weitere Konfigurationsmöglichkeiten

### RÜCKGABEWERTE

**id**            optional: Identifikator des neuen Displays; wird nur zurückgegeben, wenn Sie **Nil** als erstes Argument übergeben haben (siehe oben)

### BEISPIEL

```
CreateDisplay(2, {BGPic = 2, Active = True})
OpenDisplay(2)
NPrint("Hello World")
```

Der obige Code erstellt ein neues Display und weist ihm BGPic Nummer 2 zu. Das Display zeigt die Größe und Grafiken von BGPic an. Dann öffnen wir das Display und geben "Hello World" aus.

```
CreateDisplay(2, {Width = 800, Height = 600, Borderless = True,
  Color = #WHITE, Active = True})
OpenDisplay(2)
```

Der obige Code erstellt und öffnet eine 800x600 großes randloses Display mit weißem Hintergrund. Weil wir kein BGPic für dieses Display angegeben haben, wird **CreateDisplay()** eines automatisch erstellen und dem neuen Display zuweisen. Sie können mit dem Befehl **GetAttribute()** und dem Argument **#ATTRBGPIC** die automatisch erstellte ID das BGPic abfragen.

## 24.8 DISPLAY

### BEZEICHNUNG

**DISPLAY** – erstellt ein neues Display (V2.0)

### ÜBERSICHT

**@DISPLAY** [id,] table

### BESCHREIBUNG

Diese Präprozessor-Anweisung erstellt ein neues Display. Mit Hilfe der optionalen Attribute in der Tabelle **table** kann das Erscheinungsbild des Displays beeinflusst werden. Eine ID wird nur dann benötigt, wenn Ihr Skript mehrere Displays verwendet. Wenn Sie nur ein einzelnes verwenden, können Sie den Identifikator auslassen und Hollywood wird dann den Identifikator 1 für das Display verwenden. Wenn Sie mehrere Monitore verwenden, wird gemäß Styleguide empfohlen, für das erste/Haupt-Display die ID 1 zu wählen.

Möchten Sie Displays dynamisch zur Laufzeit erstellen, können Sie den Befehl **CreateDisplay()** benutzen.

Das Erscheinungsbild des neuen Displays wird durch die Einstellung einer Anzahl von Tags in der optionalen Tabelle `table` konfiguriert. Die folgenden Tags werden derzeit von `@DISPLAY` unterstützt:

**Title:** Verwenden Sie diesen Tag, um den Titel für dieses Display einzustellen. Dieser Titel wird dann in der Fensterleiste des Displays angezeigt (wenn das Display einen Rahmen hat). Standardtitel ist "Hollywood".

**X,Y:** Mit `X` und `Y` können Sie definieren, wo auf dem Host-Bildschirm das Display geöffnet wird. Hier mit absoluten Werten zu arbeiten ist eher kontraproduktiv, weil man meist nicht wissen kann, wie groß der Bildschirm ist (außer der Vollbildmodus wird verwendet). Es ist klüger, hier die speziellen **Positionkonstanten** von Hollywood zu gebrauchen. Wenn Sie diese Tags nicht verwenden, wird das Display immer in der Mitte des Bildschirms geöffnet werden. Sie können später auch das Display mit dem Befehl `MoveDisplay()` bewegen.

**BGPic:** Definiert das `BGPic`, welches zusammen mit dem neuen Display angezeigt werden soll. Jedes Display muss ein zugehöriges `BGPic` haben. Wenn Sie also diesen Tag nicht angeben, wird Hollywood automatisch ein neues `BGPic` für dieses Display erstellen, wobei der Füllstil vom Tag `FillStyle` sowie die beiden Größen in den Tags `Width` (Breite) und `Height` (Höhe) verwendet werden (siehe unten). Eine Ausnahme wird für das Display mit der ID 1 gemacht: Aus Gründen der Kompatibilität für die Displaynummer 1 wird automatisch dem `BGPic` die Nummer 1 zugeordnet werden, wenn es eines gibt. (V4.5)

**Width, Height:** Diese Tags müssen Sie nur verwenden, wenn Sie nicht ein Hintergrundbild in `BGPic` (siehe oben) angegeben haben und Ihre Displaygröße eine andere Abmessung als die Standardgröße 640x480 haben soll. Falls dies zutrifft, setzen Sie diese beiden Tags auf die gewünschten Abmessungen. Diese Tags werden ignoriert, wenn Sie ein `BGPic` angeben. Ab Hollywood 7.0 können Sie diese Tags auch auf die spezielle Konstante `#NATIVE` setzen. In diesem Fall wird Hollywood die Dimensionen des Displays vom Host-Geräts verwenden.

**Desktop:** Wenn diesen Tag auf `True` gesetzt ist, wird das Hintergrundbild eine Kopie Ihres Desktop-Bildschirms sein. Dies kann für einige nette Effekte mit diesem Bildschirm verwendet werden. Hollywood wird auch automatisch ein randloses Fenster öffnen, wenn dieser Tag auf `True` gesetzt ist.

**Mode:** Mit diesem Tag können Sie einstellen, in welchem Modus das Display geöffnet werden soll. Sie müssen diesem Tag eine der folgenden Zeichenfolgen übergeben:

**Windowed** Öffnet das Display im Fenstermodus.

**FullScreen**

Öffnet das Display im Vollbildmodus. Dies kann die Auflösung Ihres Monitors auf die Abmessungen ändern, die am besten zu den Dimensionen Ihres Displays passen. Wenn Sie das nicht

wollen, werfen Sie einen Blick auf die nachfolgenden Modi **FullScreenScale** und **FakeFullScreen**.

#### **FullScreenScale**

Dies ist ein spezieller Vollbildmodus, der die Auflösung Ihres Monitors nicht ändert. Stattdessen wird das Display von Hollywood so dimensioniert, dass es den Dimensionen Ihres Monitors entspricht. Zusätzlich aktiviert dieser Vollbildmodus die Autoskalierung, so dass Ihr Display automatisch skaliert wird, um sich den Dimensionen Ihres Monitors anzupassen. **FullScreenScale** verwendet standardmäßig Autoskalierung. Wenn Sie möchten, dass Ebenenskalierung verwendet wird, müssen Sie auch **ScaleMode** auf **#SCALEMODE\_LAYER** setzen. **FullScreenScale** ist besonders bei mobilen Geräten nützlich, deren Display-Hardware eine hartcodierte Auflösung hat und keine Auflösungsänderungen in gleicher Weise wie ein externer Monitor unterstützt, der mit einem Desktop-Computer verbunden ist. Der Nachteil von **FullScreenScale** ist, dass es langsamer ist, weil Hollywood alle Wiedergabe-Operationen auf die Dimensionen des Monitors skalieren muss. (V7.0)

#### **AutoFullScreen:**

Dadurch wird das Display mithilfe des automatischen Skalierungssystems in den Vollbildmodus versetzt, anstatt die Auflösung des Monitors zu ändern, aber nur, wenn Hollywood auf Systemen ausgeführt wird, die GPU-beschleunigte Skalierung unterstützen. Auf allen anderen Plattformen wird ein normaler Vollbildmodus verwendet, d.h. Hollywood passt die Auflösung des Monitors an die aktuellen Bildschirmgröße an. Derzeit wird die GPU-beschleunigte Skalierung unter Windows, macOS, Android und iOS unterstützt, was bedeutet, dass auf diesen Plattformen keine Änderung der Monitorauflösung stattfindet, da Hollywood die Grafiken einfach skalieren kann, um sie an die aktuelle Bildschirmgröße anzupassen. Auf AmigaOS-kompatiblen Systemen und Linux wird es bei diesem Modus jedoch immer noch eine Änderung der Monitorauflösung geben, weil Hollywood auf diesen Plattformen keine GPU-beschleunigte Skalierung unterstützt. (V9.1)

#### **LayerFullScreen:**

Dadurch wird das Display mithilfe der Ebenenskalierung in den Vollbildmodus versetzt, anstatt die Auflösung des Monitors zu ändern, aber nur, wenn Hollywood auf Systemen ausgeführt wird, die GPU-beschleunigte Skalierung unterstützen. Auf allen anderen Plattformen wird ein normaler Vollbildmodus verwendet, d.h. Hollywood passt die Auflösung des Monitors an die aktuelle Bildschirmgröße an. Derzeit wird die GPU-beschleunigte Skalierung unter Windows,

macOS, Android und iOS unterstützt, was bedeutet, dass auf diesen Plattformen keine Änderung der Bildschirmauflösung stattfindet, da Hollywood die Grafiken einfach skalieren kann, um sie an die aktuellen Bildschirmabmessungen anzupassen. Auf AmigaOS-kompatiblen Systemen und Linux wird es bei diesem Modus jedoch immer noch eine Änderung der Bildschirmauflösung geben, weil Hollywood auf diesen Plattformen keine GPU-beschleunigte Skalierung unterstützt. (V9.1)

**FakeFullScreen**

Öffnet das Display in einem unechten vollen Bildschirmmodus. Das bedeutet, dass Hollywood die Auflösung des Monitors nicht ändert, aber die Hintergrundfüllung den ganzen Desktop abdecken wird. Somit erhält der Anwender den Eindruck, als ob Hollywood im Vollbildmodus ausgeführt wird, obwohl es auf dem Desktop läuft.

**ModeRequester**

Dies öffnet einen Anzeigemodus-Dialogfenster, in dem der Benutzer den gewünschten Vollbildmodus auswählen kann.

**Ask**

Es öffnet sich ein Dialogfenster, wo der Benutzer zwischen Fenster- und Vollbildmodus auswählen kann.

**SystemScale:**

Wenn Sie diesen Display-Modus wählen, wird der Skalierungsfaktor des Hostsystems automatisch auf Ihr Display angewendet. Dies kann bei Systemen mit High-DPI-Monitoren nützlich sein. Wenn Ihr Display beispielsweise normalerweise in 640 x 480 Pixeln öffnet und Sie es auf einem Monitor ausführen, der doppelt so viele Punkte pro Zoll (DPI) verwendet, wird Ihr Skript im Modus **SystemScale** automatisch auf 1280 x 960 Pixel skaliert, damit es nicht winzig aussieht, nur weil das System einen High-DPI-Monitor verwendet. Beachten Sie, dass durch die Verwendung von **SystemScale** standardmäßig die Autoskalierung aktiviert wird. Wenn Sie stattdessen die Ebenenskalierung verwenden möchten, benutzen Sie einfach den Tag **ScaleMode**, um dies in Ebenenskalierung zu ändern. Beachten Sie, dass **SystemScale** intern denselben Skalierungsmodus wie **ScaleFactor** verwendet, so dass sich Displays, die **SystemScale** verwenden, so verhalten, als wäre **ScaleFactor** angegeben worden. Es ist sogar möglich, den Tag **ScaleFactor** über **SystemScale** anzugeben. In diesem Fall wird der in **ScaleFactor** angegebene Wert mit dem Standard-Skalierungsfaktor des Hostsystems multipliziert. Siehe [Abschnitt 24.24 \[Skalierungssysteme\]](#), [Seite 402](#), für Details. Beachten Sie, dass Sie unter Windows auch den Tag **DPIAware** in der Präprozessor-Anweisung **@OPTIONS** auf **True** setzen

müssen, um `SystemScale` zu verwenden. Siehe [Abschnitt 50.25 \[OPTIONS\]](#), [Seite 1107](#), für Details. (V8.0)

Standardmäßig wird der Fenstermodus verwendet.

**Borderless:**

Setzen Sie diesen Tag auf `True`, wenn das Display ein randloses Fenster sein soll. Der Standardwert ist `False`.

**Sizeable:**

Setzen Sie diesen Tag auf `True`, wenn die Displaygröße durch den Benutzer verstellbar sein soll. Wenn `Borderless` auch auf `True` gesetzt ist, wird das Größesymbol in der unteren rechten Ecke des Fensters unsichtbar. Der Standardwert ist `False`.

**Fixed:**

Wenn dieser Tag auf `True` gesetzt ist, wird Hollywood ein festes, nichtziehbares Fenster erstellen. Dies ist besonders nützlich, wenn das Display als Vollbild angezeigt wird. Der Standardwert ist `False`.

**Backfill:**

Dieser Tag kann verwendet werden, um die Einstellung der Hintergrundfüllung für dieses Display zu konfigurieren. `Backfill` kann ein Schutzfenster schaffen, welches das gesamte Gebiet von Ihrem Hauptdisplay besetzt. Das wird allerdings nur dann unterstützt, wenn `Mode` auf `FullScreen` oder `FakeFullScreen` gesetzt wurde. Sie können eine statische Farbe, einen Farbverlauf, ein Bild oder eine Textur als Hintergrundfüllung verwenden. Sie müssen eine Tabelle an diesen Tag übergeben. Die folgenden Tabellen-Felder stehen derzeit zur Verfügung:

**Type:** Dieses Feld ist obligatorisch. Es kann `Color` (Farbe), `Gradient` (Farbverlauf), `Texture` (Struktur) oder `Picture` (Bild) sein. Der Typ muss hier als Zeichenkette übergeben werden (z.B. "Texture")

**Color:** Wenn Sie `Color` als `Type` angegeben, übergeben Sie hier die gewünschte Hintergrundfarbe.

**StartColor, EndColor:**

Wenn Sie `Gradient` als `Type` angegeben, verwenden Sie diese beiden Felder, um die Start- und Endfarben für den Farbverlauf zu definieren.

**Brush:** Wenn Sie `Texture` oder `Picture` als `Type` verwenden, geben Sie hier die ID des Pinsels an. Wenn Sie den Dateinamen direkt übergeben möchten, verwenden Sie das `BrushFile`-Feld.

**X,Y:** Wenn Sie `Picture` als `Type` angegeben haben, können Sie diese beiden Felder verwenden, um das Bild auf dem Bildschirm zu positionieren. Beide sind auf `#CENTER` voreingestellt.

**HideTitleBar:**

Wenn Sie dieses Feld auf `True` setzen, wird die Hintergrundfüllung auch die Titelleiste des Host-Bildschirm

abdecken (zum Beispiel Finder-Titelleiste auf macOS oder die Workbench-Titelleiste auf AmigaOS kompatible Geräte). Beachten Sie, dass Sie auch `HideTitleBar` außerhalb des `Backfill`-Feldes verwenden können, weil auf Android und iOS `HideTitleBar` auch ohne Hintergrundfüllung verwendet werden kann. Wenn `HideTitleBar` ohne Hintergrundfüllung benutzt wird, versteckt `HideTitleBar` die Statusleiste des Gerätes auf Android sowie iOS.

**BrushFile:**

Wenn Sie `Texture` oder `Picture` als `Type` angeben, können Sie hier den Dateinamen des Pinsels verwenden. Diese Datei wird dann beim Kompilieren mit dem Applet/der ausführbaren Datei verknüpft werden, außer Sie setzen `LinkBrushFile` auf `False`. Wenn Sie die ID des Pinsels übergeben möchten, verwenden Sie das `Brush`-Feld. (V4.0)

**LinkBrushFile:**

Wenn `BrushFile` angegeben ist, kann mit diesem Feld eingestellt werden, ob die Pinseldatei beim Kompilieren in das Applet/die ausführbare Datei eingebunden werden soll oder nicht. Der Standardwert ist `True`, was bedeutet, dass der Pinsel mit der Datei verknüpft wird. (V4.0)

**Transparency:**

Beim `Type Picture` können Sie hier eine `RGB-Farbe` angeben, die transparent dargestellt werden soll. Der Standardwert ist `#NOTRANSparency`. (V4.0)

**ScalePicture:**

Beim `Type Picture` können Sie dieses Feld verwenden, um festzulegen, ob das Bild an die Dimensionen des Displays angepasst/skaliert wird (`True`) oder nicht (`False`). Der Standardwert ist `False`. (V4.0)

**SmoothScale:**

Setzen Sie dieses Feld auf `True`, wenn Sie bei der Skalierung des Hintergrundbildes die Kantenglättung (Antialias) verwenden wollen. Dieses Feld wird nur dann berücksichtigt, wenn `ScalePicture` auf `True` gesetzt wurde. Der Standardwert ist `False`. (V6.0)

**ScrWidth, ScrHeight:**

Wenn `Mode` auf `FullScreen` gesetzt ist, können Sie mit diesen Tags die gewünschten Dimensionen für den Vollbildmodus einstellen. Voreinstellung ist, was beim Erstellen des Displays eingestellt wurde. Ab Hollywood 7.0 können Sie diese Tags auch auf die spezielle Konstante `#NATIVE` setzen. In diesem Fall wird Hollywood die Dimensionen des Host-Gerätes des Displays verwenden. (V3.0)

**ScrDepth:**

Wenn **Mode** auf **FullScreen** gesetzt worden ist, ermöglicht dieser Tag, die gewünschte Farbtiefe für den Vollbildmodus zu setzen. Der Standardwert ist, was eingestellt wurde, als das Display erstellt wurde. (V3.0)

**HidePointer:**

Wenn Sie diesen Tag angeben, wird der Mauszeiger automatisch wieder ausgeblendet werden, wenn Hollywood in Vollbild- oder unechten vollen Bildschirmmodus wechselt. Dieses Argument hat gegenüber dem Befehl **HidePointer()** den Vorteil, dass der Mauszeiger nur im Vollbildmodus versteckt wird. Wenn Hollywood sich wieder im Fenstermodus befindet, wird der Mauszeiger wieder sichtbar. Wenn Sie den Mauszeiger im Fenstermodus verstecken, kann der Benutzer verwirrt werden. Der Standardwert ist **False**. (V3.0)

**NoModeSwitch:**

Wenn Sie dieses Argument angeben, wird es nicht möglich sein, bei diesem Display zwischen Fenster- und Vollbildmodus zu wechseln, indem Sie die Tastenkombination **CMD+RETURN** (macOS und Amiga) oder **ALT+RETURN** (Windows) drücken. Wenn **NoModeSwitch** angegeben ist, wird dieses Display immer in seinem ursprünglichen Anzeigemodus bleiben. Der Standardwert ist **False**. (V3.0)

**NoHide:** Setzen Sie diesen Tag auf **True**, wenn Sie für dieses Display kein Iconify-Gadget haben wollen. Wenn Sie diesen Tag nicht angeben, wird Ihr Display immer über ein Iconify-Gadget verfügen. Der Standardwert ist **False**. (V4.5)

**ScaleMode:**

Mit diesem Argument können Sie ein Skalierungssystem für dieses Display auswählen. **Scalemode** kann auf einen der folgenden Parameter eingestellt werden: **#SCALEMODE\_LAYER** (verwendet Ebenenskalierung), **#SCALEMODE\_AUTO** (verwendet Autoskalierung) oder **#SCALEMODE\_NONE** (verwendet kein Skalierungssystem). Wenn **Scalemode** nicht angegeben wird, ist beim Display der Skalierungsmodus **#SCALEMODE\_NONE** eingestellt, dementsprechend ist kein Skalierungssystem aktiv. Bei der Angabe von **ScaleMode** möchten Sie in der Regel entweder die Argumente **ScaleWidth** und **ScaleHeight** oder den Display-Modus **ScaleFactor** oder **SystemScale** festlegen, um die Skalierungsdimensionen anzugeben (siehe unten). Siehe [Abschnitt 24.24 \[Skalierungssysteme\]](#), [Seite 402](#), für weitere Informationen über die Skalierungssysteme von Hollywood. (V4.5)

**ScaleWidth, ScaleHeight:**

Diese beiden Tags können verwendet werden, um die gewünschten Skalierungsmaße anzugeben, wenn ein Skalierungssystem aktiv ist (siehe Dokumentation von **ScaleMode** oben). Sie können die Größe entweder als direkten Wert übergeben oder eine prozentuale Zeichenfolge (d.h. **ScaleWidth="200%"**) angeben. Wenn Sie eine prozentuale Zeichenfolge übergeben, wird die Skalierungsgröße relativ zur ursprünglichen Größe festgelegt (d.h. **ScaleWidth="200%"** bedeutet zweimal die ursprüngliche Breite). Ab Hollywood 7.0 können Sie diese Tags auch auf die spezielle



Konstante **#NATIVE** setzen. In diesem Fall wird Hollywood die Dimensionen des Displays vom Host-Gerät verwenden. (V4.5)

**SmoothScale:**

Wenn **ScaleMode** eingestellt ist, können Sie mit diesem Argument festlegen, ob Hollywood Antialiasing bei der Skalierung verwenden soll (**True**) oder nicht (**False**). Der Standardwert ist **False**, somit wird kein Antialiasing verwendet. Beachten Sie, dass die Skalierung mit Antialiasing viel langsamer als die normale Skalierung ist. (V4.5)

**Hidden:** Setzen Sie dies auf **True**, wenn dieses Display zunächst ausgeblendet werden soll. Somit wird das Display erst angezeigt, wenn Sie **OpenDisplay()** aufrufen. Sie können diesen Tag auch verwenden, um ein Hollywood-Skript auszuführen, das kein Display öffnet. Aber bedenken Sie, dass einige Befehle (z.B. **WaitLeftMouse()**) nur mit sichtbarem Display funktionieren. (V4.5)

**Active:** Mit diesem Tag können Sie dem Display angeben, ob es beim Start aktiv sein soll. Bitte beachten Sie, dass nur ein Display aktiv sein kann, so dass es nicht gültig ist, für mehrere Displays **Active** auf **True** zu setzen. Dies wird zu nicht definierten Ergebnisse führen. Wenn Sie **Active** bei keinem Display auf **True** setzen, wird Hollywood die Displaynummer 1 standardmäßig aktivieren. (V4.5)

**DragRegion:**

Mit diesem Tag können Sie eine benutzerdefinierte Ziehregion (Drag-Bar) für dieses Display definieren. Benutzerdefinierte Ziehbereiche werden nur bei randlosen Displays unterstützt, so müssen Sie vorher **Borderless** auf **True** setzen, wenn Sie diesen Tag verwenden. Sie können mehrere Ziehregionen mit diesem Tag definieren. Aus diesem Grund müssen Sie eine Tabelle übergeben, die eine Liste von Tabellen enthält, die jeweils einen einzelnen rechteckigen Bereich definiert. Jede Tabelle in der Liste muss die folgenden Felder beinhalten: **Type**, **X**, **Y**, **Width** und **Height**. Momentan muss **Type** immer auf **#BOX** eingestellt werden, da derzeit nur rechteckige Bereiche unterstützt werden. Dies hört sich evtl. ziemlich kompliziert an, aber in Wirklichkeit ist es wirklich einfach. Alles, woran Sie denken müssen, ist eine Liste der Tabelle diesem Feld zu übergeben. Selbst wenn Sie nur eine einzige rechteckige Ziehregion wollen, müssen Sie eine Liste übergeben. Siehe unten für ein Beispiel. (V4.5)

**SizeRegion:**

Mit diesem Tag können Sie eine benutzerdefinierte Größenregion für dieses Display definieren. Benutzerdefinierte Größenbereiche werden nur von randlosen Displays unterstützt, so müssen Sie vorher **Borderless** auf **True** setzen, wenn Sie diesen Tag verwenden. Sie können mehrere Größenregionen mit diesem Tag definieren. Aus diesem Grund müssen Sie eine Tabelle übergeben, die eine Liste von Tabellen enthält, die jeweils einen einzelnen rechteckigen Bereich definiert. Jede Tabelle in der Liste muss die folgenden Felder beinhalten: **Type**, **X**, **Y**, **Width** und **Height**. Momentan muss **Type** immer **#BOX** eingestellt werden, da derzeit nur rechteckige Bereiche unterstützt werden. Dies hört sich evtl. ziemlich kompliziert an, aber in Wirklichkeit ist es wirklich einfach. Alles, woran Sie denken müssen, ist eine Liste der Tabelle diesem

Tag zu übergeben. Selbst wenn Sie nur eine einzige rechteckige Größenregion wollen, müssen Sie eine Liste übergeben. Siehe unten für ein Beispiel. (V4.5)

**Layers:** Setzen Sie diesen Tag auf **True**, wenn Sie für dieses Display Ebenen aktivieren wollen. Wenn Sie diesen Tag auf **True** setzen, müssen Sie für dieses Display nicht den Befehl **EnableLayers()** aufrufen. Siehe **Abschnitt 25.1 [Einführung in die Ebenentechnik]**, Seite 405, für weitere Informationen zu den Ebenen.. Dieser Tag wird standardmäßig auf **False** gesetzt. (V4.5)

**UseQuartz:** Dieser Tag wird nur unterstützt, wenn Hollywood unter macOS läuft. Wenn Sie diesen Tag auf **True** setzen, wird das Display seine Grafiken mit der Quartz 2D-API zeichnen. Wenn es auf **False** gesetzt ist, wird QuickDraw verwendet werden. Beachten Sie, dass dieses Argument nur von Hollywood in der PowerPC-Version unterstützt wird. Die x86/x64-Versionen von Hollywood für macOS verwenden immer Quartz 2D. Voreingestellt ist **False**. (V4.5)

**NoClose:** Setzen Sie diesen Tag auf **True**, wenn das Display kein Schließknopf in seinem Fensterrahmen haben soll. Denken Sie zweimal nach, bevor Sie diesen Tag verwenden. Es könnte den Benutzer verwirren und Sie müssen einen Ersatz für das Schließen des Displays bereit halten (z.B. **EscapeQuit()** oder **CtrlCQuit()**). Voreingestellt ist **False**. (V4.5)

**FitScale:** Dieser Tag wird nur berücksichtigt, wenn entweder **#SCALEMODE\_AUTO** oder **#SCALEMODE\_LAYER** in **ScaleMode** (siehe oben) aktiv ist. Ist dies der Fall und **FitScale** wird auf **True** gesetzt, füllt das Skript den gesamten Bildschirm aus. Dies geschieht, indem die Auflösung des Displays auf die aktuelle Bildschirmauflösung skaliert wird. Dies ist im Wesentlichen das Gleiche wie die aktuelle Bildschirmabmessung in **ScaleWidth** und **ScaleHeight**. Beachten Sie, dass **FitScale** das Aussehen Ihres Skripts verzerren könnte, falls die aktuelle Bildschirmauflösung ein anderes Seitenverhältnis als das Skript hat. Um diese Verzerrung zu verhindern, müssen Sie **KeepProportions** (siehe unten) verwenden. Der Standardwert ist **False**. (V4.7)

**KeepProportions:** Dieser Tag wird nur berücksichtigt, wenn entweder **#SCALEMODE\_AUTO** oder **#SCALEMODE\_LAYER** in **ScaleMode** (siehe oben) aktiv ist. Falls dies der Fall ist und **KeepProportions** wird auf **True** gesetzt, wird die Auflösung des aktuellen Skripts durch Verändern der Größe des Fensters nicht verzerrt. Stattdessen werden schwarze Ränder für die nichtproportionalen Fensterbereiche verwendet. Das Display selbst wird immer das Seitenverhältnis beibehalten. Dies ist sehr nützlich für Skripte, die nicht verzerrt werden sollten. (V4.7)

**FillStyle:** Dieser Tag ermöglicht es Ihnen, einen gefüllten Hintergrund mit einem Füllstil für dieses Display zu definieren. Dieser Tag wird nur berücksichtigt, wenn **BGPic** nicht angegeben wurde. Die Standardeinstellung für diesen Tag ist **#FILLCOLOR**. Siehe **Abschnitt 29.14 [SetFillStyle]**, Seite 610, für Informationen über alle verfügbaren Füllstile. (V5.0)

**Color:** Dieser Tag wird nur dann ausgeführt, wenn Sie **BGPic** nicht verwendet haben und bei **FillStyle** **#FILLCOLOR** gesetzt haben. In diesem Fall können Sie mit diesem Tag die Farbe für die Hintergrundfüllung festlegen, die automatisch für dieses Display erstellt wird.

**TextureBrush:**  
Wenn bei **FillStyle** **#FILLTEXTURE** gesetzt wurde, können Sie mit diesem Tag die ID des Pinsels angeben, die für die Texturierung verwendet werden soll. (V5.0)

**TextureX, TextureY:**  
Diese Tags steuern den Startpunkt innerhalb des Texturpinsels und werden nur unterstützt, wenn **FillStyle** auf **#FILLTEXTURE** gesetzt wurde. Siehe [Abschnitt 29.14 \[SetFillStyle\]](#), [Seite 610](#), für Details. (V5.0)

**GradientStyle:**  
Wenn **FillStyle** auf **#FILLGRADIENT** gesetzt wurde, können Sie diesen Tag nutzen, um den Farbverlauftyp anzugeben. Dies kann **#LINEAR**, **#RADIAL** oder **#CONICAL** sein. (V5.0)

**GradientAngle:**  
Gibt die Ausrichtung des Farbverlaufs an, wenn bei **FillStyle** **#FILLGRADIENT** eingestellt ist. Der Winkel wird in Grad ausgedrückt. Nur für die beiden **GradientStyle** **#LINEAR** und **#CONICAL**. (V5.0)

**GradientStartColor, GradientEndColor:**  
Verwenden Sie diese beiden Farben, um den Farbverlauf zu konfigurieren, wenn **FillStyle** auf **#FILLGRADIENT** gesetzt wurde. (V5.0)

**GradientCenterX, GradientCenterY:**  
Legt den Mittelpunkt für die **GradientStyle** des Typs **#RADIAL** oder **#CONICAL** fest. Muss ein Fließkommawert zwischen 0.0 und 1.0 sein. Siehe [Abschnitt 30.6 \[CreateGradientBGPic\]](#), [Seite 624](#), für Details. (V5.0)

**GradientBalance:**  
Dieser Tag steuert den Balancepunkt für Farbverläufe des Typs **#CONICAL**. Muss ein Fließkommawert zwischen 0.0 und 1.0 sein. Siehe [Abschnitt 30.6 \[CreateGradientBGPic\]](#), [Seite 624](#), für Details. (V5.0)

**GradientBorder:**  
Dieser Tag steuert die Rahmengröße der Farbverläufe des Typs **#RADIAL**. Muss ein Fließkommawert zwischen 0.0 und 1.0 sein. Siehe [Abschnitt 30.6 \[CreateGradientBGPic\]](#), [Seite 624](#), für Details. (V5.0)

**GradientColors:**  
Dieser Tag kann verwendet werden, um einen Farbverlauf zwischen mehr als zwei Farben zu erzeugen. Dies muss mit einer Tabelle gesetzt werden, die Sequenzen von abwechselnden Farben und Stoppwerte enthält. Wenn dieser Tag verwendet wird, werden die Tags **GradientStartColor** und **GradientEndColor** ignoriert. Siehe [Abschnitt 30.6 \[CreateGradientBGPic\]](#), [Seite 624](#), für Details. (V5.0)

**KeepScreenOn:**

Dieser Tag wird nur dann unterstützt, wenn Hollywood auf Android und iOS läuft. Wenn Sie ihn auf **True** setzten, wird der Batteriesparmodus auf mobilen Geräten deaktiviert. Dies bedeutet, dass der Bildschirm des Geräts nie gedimmt oder ausgeschaltet wird, um Energie zu sparen. Nützlich für Skripte, die keine Benutzereingaben erfordern. Der Standardwert ist **False**. (V5.1)

**PubScreen:**

Dieser Tag wird nur auf AmigaOS und kompatiblen Betriebssystemen unterstützt. Er kann verwendet werden, um den öffentlichen Bildschirm anzugeben, auf dem dieses Display geöffnet wird. Sie müssen eine Zeichenfolge übergeben, die den Namen des öffentlichen Bildschirms enthält. Bitte beachten Sie, dass wenn Sie mehrere öffentlichen Bildschirme auf verschiedenen Monitoren verwenden, ist es zwingend erforderlich, dass die einzelnen öffentlichen Bildschirme das gleiche Pixelformat verwenden. Es ist also nicht erlaubt, ein Display auf einem 16-Bit öffentlichen Bildschirm zu haben, während das andere Display auf einem 32-Bit-öffentlichen Bildschirm angezeigt wird. Das Pixelformat muss für alle öffentlichen Bildschirme identisch sein. (V5.2)

**HideFromTaskbar:**

Dieser Tag wird nur dann unterstützt, wenn Hollywood unter Windows ausgeführt wird. Falls Sie diesen Tag auf **True** setzen, wird Ihr Display keinen Eintrag in der Windows-Taskleiste erhalten. Dies ist nützlich, wenn Sie bereits ein Piktogramm der Taskleiste hinzugefügt haben und Sie kein Knopf in der Taskleiste wollen. Der Standardwert ist **False**. (V5.3)

**HideOptionsMenu:**

Dieser Tag wird nur auf Android-Geräten unterstützt. Wenn der Benutzer das Optionsmenü auf Android-Geräten öffnet, ermöglicht Hollywood dem Benutzer, mehrere Displayparameter zu konfigurieren, wie das Aktivieren oder Deaktivieren der automatischen Skalierung oder der Ebenenskalierung. Wenn der Benutzer dieses Optionsmenü nicht öffnen darf, setzen Sie diesen Tag auf **True**. Der Standardwert ist **False**. (V5.3)

**Orientation:**

Diesen Tag wird nur auf mobilen Plattformen unterstützt. Es ermöglicht Ihnen, eine hartcodierte Orientierung für Ihr Skript anzugeben. Wenn Sie diesen Tag setzen, wird Hollywood nicht reagieren, wenn sich die Ausrichtung beim Drehen des Gerätes ändert. Stattdessen wird es die Ausrichtung halten, die Sie hier angegeben. Die folgenden Werte sind möglich:

```
#ORIENTATION_NONE  
#ORIENTATION_PORTRAIT  
#ORIENTATION_LANDSCAPE  
#ORIENTATION_PORTRAITREV  
#ORIENTATION_LANDSCAPEREV
```

Der Standardwert ist `#ORIENTATION_NONE`, was bedeutet, dass keine feste Orientierung eingestellt ist und dass Hollywood sich dynamisch den Orientierungsänderungen anpasst. (V5.3)

**NoHardwareScale:** VERALTET

Dieser Tag wird nur auf Android unterstützt. Aus Performancegründen wird Hollywood versuchen, die hardwarebeschleunigte Skalierung zu verwenden, wenn automatische Skalierung auf Android-Geräten standardmäßig aktiviert ist. Einige Geräte haben keine Grafikhardware und es kann zu seltsamen Ergebnissen führen, wenn Sie das automatische Skalierungssystem benutzen. Ist dies der Fall, dann setzen Sie NoHardwareScale auf `True` und schauen, ob das hilft. Dieser Tag ist seit Hollywood 8.0 veraltet. Hollywood wird jetzt immer die hardwarebeschleunigte Skalierung verwenden. (V5.3) [Nur Android]

**DisableBlanker:**

Setzen Sie diesen Tag auf `True`, wenn Sie den Bildschirm-Blanker deaktivieren möchten, während dieses Display geöffnet ist. Der Standardwert ist `False`. (V6.0)

**Menu:** Dieser Tag kann verwendet werden, um diesem Display eine Menüleiste zuzuweisen. Sie übergeben die ID einer Menüleiste, die mit der Präprozessor-Anweisung `@MENU` oder dem Befehl `CreateMenu()` erstellt wurde. Es ist auch möglich, eine einzelne Menüleiste mehreren Displays zuzuweisen. Siehe [Abschnitt 37.8 \[MENU\]](#), [Seite 754](#), für Details. (V6.0)

**Monitor:** Mit diesem Tag können Sie den Monitor angeben, auf dem dieses Display geöffnet werden soll. Monitore werden von 1 bis zur Anzahl Monitore von dem System gezählt. Bitte beachten Sie, wenn Sie diesen Tag setzen: Befehle, die Displaykoordinaten annehmen, z.B. `MoveDisplay()`, orientieren sich nach dem Ursprung des im Monitor angegebenen Monitor mit relativen Werten. Voreingestellt ist 1, was bedeutet, dass das Display auf dem primären Monitor geöffnet werden soll. (V6.0)

**XServer:** Dieser Tag kann verwendet werden, um den X-Server anzugeben, dem dieses Display zugewiesen wird. Standardmäßig verwendet Hollywood den X-Server, der in der `DISPLAY` Umgebungsvariable angegeben wurde. Wenn Sie einen anderen X-Server für dieses Display verwenden möchten, verwenden Sie diesen Tag, welcher nur in der Linux-Version von Hollywood zur Verfügung steht. (V6.0)

**ScreenTitle:**

Auf AmigaOS kann dieser Tag verwendet werden, um den Text in der Titelleiste des Bildschirms zu setzen, wenn das Display angezeigt wird. In der Standardeinstellung wird "Workbench screen" angezeigt. Dieser Tag steht nur den AmigaOS-kompatiblen Versionen von Hollywood zur Verfügung. (V6.0)

**ScreenName:**

Wenn dieses Display im Vollbildmodus geöffnet werden soll, können Sie mit diesem Tag den gewünschten öffentlichen Bildschirmnamen für das Display

einstellen. Standardmäßig wird Hollywood "HOLLYWOOD.X" verwenden, wobei "X" durch einen leeren Index ersetzt wird. Dieser Tag steht den AmigaOS und kompatiblen Versionen von Hollywood zur Verfügung. (V6.0)

**RememberPosition:**

Setzen Sie hier den Tag auf **True**, wenn sich das Display an seine Position und Größe erinnern soll. Dies ist natürlich nur im Fenstermodus möglich. **RememberPosition** arbeitet nicht im Vollbildmodus. Sie müssen auch eine eindeutige ID für Ihre Anwendung angeben, welche mit der Präprozessor-Anweisung **@APPIDENTIFIER** festgelegt wird. Das Display kann auch eine numerische ID verwenden, das heißt Sie können mit diesem Tag auch Displays verwenden, welche die ID mit der **automatischen ID-Zuweisung** erhalten haben. Beachten Sie, dass dieser Tag durch das Argument **-overrideplacement** außer Kraft gesetzt werden kann. Wenn Sie Hollywood oder ein kompiliertes Skript mit dem Argument **-overrideplacement** starten, wird jede gespeicherte Position oder Größe ignoriert. Siehe **Abschnitt 3.2 [Konsolenargumente]**, **Seite 33**, für Details. (V6.1)

**Maximized:**

Wenn Sie diesen Tag auf **True** setzen, öffnet sich das Display im maximierten Modus. Dies ist nur möglich, wenn das Display größenveränderlich ist. Dieser Tag wird derzeit nur unter Windows unterstützt. (V7.0)

**TrapRMB:** Wenn dies unter AmigaOS auf **True** gesetzt ist, liefert Hollywood auch die rechten Maustastenereignisse, wenn eine Menüleiste einem Display zugewiesen wurde. Der Nachteil ist, dass der Menüzugriff nur über die Titelleiste des Bildschirms möglich ist. Dieser Tag wird nur dann berücksichtigt, wenn Ihr Display über eine Menüleiste verfügt, sonst hat es überhaupt keine Wirkung. **TrapRMB** ist standardmäßig auf **False** gesetzt, das heißt wenn eine Menüleiste einem Display zugeordnet ist, werden rechte Maustastenereignisse nicht erzeugt. Dieser Tag ist nur in den AmigaOS-kompatiblen Versionen von Hollywood verfügbar. (V7.0)

**NoScaleEngine:**

Dieser Tag wird nur berücksichtigt, wenn **Mode** für Ihr Display auf **FullScreenScale** gesetzt ist. In diesem Fall wird Hollywood keine Skalierungssysteme verwenden, sondern wird Ihr Display einfach in den gleichen Dimensionen wie die Auflösung des Monitors öffnen. Ihr Skript muss sich dann manuell an die Auflösung des Monitors anpassen. Dies ermöglicht es Ihnen Skripte zu schreiben, die sich dynamisch an verschiedene Auflösungen anpassen können, ohne einfach ihre Grafiken zu skalieren. (V7.0)

**NoLiveResize:**

Auf vielen Plattformen wird Hollywood Live-Größenänderung verwenden, wenn der Benutzer die Dimensionen eines Displays verändert. Dies bedeutet, dass der Inhalt des Displays automatisch skaliert wird, während der Benutzer das Display verkleinert. Wenn Sie das nicht wollen, können Sie diesen Tag auf **True** setzen. (V7.0)

**NativeUnits:**

Wenn Sie diesen Tag auf **True** setzen, verwendet Hollywood das native Koordinatennetz des Hostsystems und dessen Einheiten anstelle von Pixeln. Dies hat derzeit nur einen Einfluss auf macOS und iOS, da beide Betriebssysteme benutzerdefinierte Einheiten anstelle von Pixeln verwenden, wenn sie auf einem Retina-Gerät laufen. Standardmäßig wird Hollywood die Verwendung von Pixeln auf Retina Macs und iOS-Geräten wegen plattformübergreifende Kompatibilitätsgründe erzwingen, aber Sie können diese Einstellung mit diesem Tag überschreiben. (V7.0)

**AlwaysOnTop:**

Wenn Sie diesen Tag auf **True** setzen, bleibt das Display immer im Vordergrund. Verwenden Sie diesen Tag mit Vorsicht, da er für den Benutzer sehr lästig sein kann. (V7.1)

**NoCyclerMenu:**

Unter Android fügt Hollywood dem Optionsmenü in der Aktionsleiste der App automatisch ein Auswahl-Listenfeld-Menü hinzu, wenn mehr als ein Hollywood-Display geöffnet ist. Sie können dieses Auswahl-Listenfeld-Menü dann verwenden, um bequem zu anderen Displays zu wechseln. Wenn Sie nicht möchten, dass Hollywood ein solches Auswahl-Listenfeld-Menü hinzufügt, setzen Sie diesen Tag auf **False**. Dies wird nur auf Android unterstützt. Der Standardwert ist **False**. (V8.0)

**HideTitleBar:**

Setzen Sie diese Option auf **True**, um die Statusleiste unter iOS oder die Aktionsleiste unter Android auszublenden. Standardmäßig sind sowohl die Statusleiste als auch die Aktionsleiste immer sichtbar. Der Standardwert ist **False**. (V8.0)

**Subtitle:**

Mit diesem Tag können Sie den Untertitel des Displays einstellen. Dies wird nur auf Android unterstützt. Der Untertitel des Displays wird in der Aktionsleiste der App unter dem Anzeigetitel angezeigt, der mit dem oben genannten Tag **Title** festgelegt wurde. Standardmäßig gibt es keine Untertitel. (V8.0)

**SingleMenu:**

Mit diesem Tag können Sie Menüelemente in der Stammebene des Optionsmenüs der Aktionsleiste von Android platzieren. Normalerweise ist dies nicht möglich, da auf Desktopsystemen Menüelemente immer Mitglieder bestimmter Stammgruppen sein müssen (z.B. "Datei", "Bearbeiten", "Ansicht" usw.). Bei der Verwendung von Menüleisten unter Android wird Hollywood natürlich das Verhalten der Desktop-Menüs durch Erstellen einzelner Untermenüs für diese Stammgruppen replizieren. Das bedeutet, dass der Benutzer mindestens zweimal auftippen muss, um ein Menüelement auszuwählen, da es in der Stammebene keine Menüelemente gibt. Stattdessen befinden sie sich immer in Untermenüs. Wenn Sie nicht möchten, dass Hollywood diese Untermenüs erstellt, sondern alle Elemente in der Stammebene platziert, setzen Sie diesen Tag auf **True**. Dies ist besonders nützlich, wenn es nur wenige Menüpunkte gibt und es nicht sinnvoll ist, sie in Untermenüs

zu platzieren. Dieser Tag ist nur für Android verfügbar. Der Standardwert ist `False`. (V8.0)

#### **ScaleFactor:**

Wenn ein Skalierungssystem mit dem Tag `ScaleMode` (siehe oben) aktiviert wurde, können Sie mit diesem Tag einen globalen Skalierungsfaktor auf Ihr Display anwenden. Der Skalierungsfaktor muss als Bruchzahl angegeben werden, die den gewünschten Skalierungskoeffizienten angibt, z.B. ein Wert von 0.5 verkleinert alles auf die Hälfte seiner Größe, während ein Wert von 2.0 alles auf das Doppelte skaliert. Beachten Sie, dass sich das Skript durch die Einstellung von `ScaleFactor` geringfügig von der Einstellung von `ScaleWidth` und `ScaleHeight` (siehe oben) unterscheidet. Letzteres erzwingt eine feste Displaygröße, die niemals geändert wird, es sei denn, der Benutzer verwendet die Maus manuell zum Ändern der Displaygröße. Wenn Sie `ScaleFactor` einstellen, wird der Skalierungsfaktor jedoch auf alle neuen BGPics- und Displaygrößen angewendet, sodass sich die Displaygröße ändern kann, wenn sich die BGPic-Größe ändert oder das Skript die Displaygröße ändert. Daher ist die Verwendung von `ScaleFactor` ideal für die Skalierung eines Skripts für ein Display mit hoher Auflösung, da sichergestellt ist, dass sich das Skript genau gleich verhält, aber nur größer erscheint (oder kleiner, wenn Sie möchten!). Sie können den Tag `SystemScale` auch verwenden, um den Skalierungsfaktor des Hostsystems automatisch auf Ihr Display anzuwenden (siehe unten). Siehe [Abschnitt 24.24 \[Skalierungssysteme\]](#), [Seite 402](#), für Details. (V8.0)

#### **ImmersiveMode:**

Auf Android können Sie mit diesem Tag das Display in den immersiven Modus versetzen. Der Immersive-Modus ist ein spezieller Vollbildmodus, der die meisten System-Benutzeroberfläche (wie die Statusleiste) ausblendet, um Ihren Programmen so viel Platz wie möglich auf dem Bildschirm zu geben. Wenn Sie möchten, dass Ihr Display den Immersive-Modus verwendet, müssen Sie `ImmersiveMode` auf eines der folgenden Tags setzen. Alle diese Modi unterscheiden sich nur darin, wie der Benutzer die Systemleisten einblenden kann und ob Ihr Skript darüber benachrichtigt wird oder nicht.

##### **#IMMERSIVE\_NORMAL:**

Normaler immersiver Modus. Benutzer können die Systemleisten wieder einblenden, indem sie von einer beliebigen Kante aus wischen, an der sich eine ausgeblendete Systemleiste befindet. Über die Ereignis-Handler `ShowSystemBars` und `HideSystemBars` werden Sie über Änderungen der Sichtbarkeit der Systemleiste benachrichtigt.

##### **#IMMERSIVE\_LEANBACK:**

Lean back immersiver Modus. In diesem Modus können Systemleisten durch Tippen auf eine beliebige Stelle des Bildschirms wieder eingeblendet werden. Sie werden über Änderungen der Sichtbarkeit der Systemleisten mit den



Ereignis-Handlern `ShowSystemBars` und `HideSystemBars` benachrichtigt.

#### `#IMMERSIVE_STICKY:`

Sticky immersiver Modus. Dies ist dasselbe wie `#IMMERSIVE_NORMAL`, außer dass keine Benachrichtigung erfolgt, wenn sich die Sichtbarkeit der Systemleiste ändert. Stattdessen werden die rohen/unbearbeiteten Swipe-Ereignisse an Sie weitergeleitet, auch wenn sie dazu führen, dass die Systemleisten wieder angezeigt wurden.

Beachten Sie, dass sowohl `#IMMERSIVE_NORMAL` als auch `#IMMERSIVE_LEANBACK` Ihr Skript mit den Ereignis-Handlern `ShowSystemBars` und `HideSystemBars` über Änderungen der Sichtbarkeit der Systemleiste benachrichtigen. Sie können diese Ereignisse mit dem Befehl `InstallEventHandler()` überwachen. Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), Seite 483, für Details. (V9.0)

**Palette:** Wenn dieser Tag auf den Identifikator einer Palette gesetzt ist, erstellt Hollywood eine Palettenmodus-Display für Sie. Paletten können mit Befehlen wie `CreatePalette()` oder `LoadPalette()` erstellt werden. Alternativ können Sie diesen Tag auch auf eine von Hollywoods integrierten Paletten setzen, z.B. `#PALETTE_AGA`. Siehe [Abschnitt 41.36 \[SetStandardPalette\]](#), Seite 872, für eine Liste der integrierten Paletten. (V9.0)

**FillPen:** Wenn der Tag `Palette` gesetzt ist (siehe oben), können Sie mit dem Tag `FillPen` den Stift einstellen, mit dem der Hintergrund des Displays gefüllt werden soll. (V9.0)

#### **SoftwareRenderer:**

Auf Windows-Systemen können Sie diesen Tag auf `True` setzen, um die GPU-beschleunigte Direct2D-Wiedergabe von Hollywood auf Windows-Systemen zu deaktivieren. Hollywood wird dann seine CPU-basierten Wiedergabe für maximale Kompatibilität verwenden. Dies ist vor allem für Testzwecke nützlich. Normalerweise sollte es keinen Grund geben, diesen Tag auf `True` zu setzen. (V9.0)

**VSync:** Auf Windows-Systemen kann dieser Tag auf `True` gesetzt werden, um die Hollywood-Wiedergabe zu zwingen, die Aktualisierung auf die Bildwiederholrate des Monitors zu drosseln. Das bedeutet, dass Sie keine Befehle wie `VWait()` mehr verwenden müssen, um das Zeichnen zu drosseln. Beachten Sie jedoch, dass Sie, wenn Sie dies auf `True` setzen, darauf achten müssen, nur in Vollbildern zu zeichnen, da sonst das Zeichnen extrem langsam wird. Das Zeichnen in Vollbildern kann z.B. durch die Verwendung eines doppelten Puffers oder durch die Verwendung der Befehle `BeginRefresh()` und `EndRefresh()` beschleunigt werden. Beachten Sie auch, dass `VSync` derzeit nur unter Windows und nur dann unterstützt wird, wenn Hollywood seine Direct2D-Wiedergabe verwendet. Direct2D ist vor Windows Vista SP2 nicht verfügbar. (V9.0)

**ScaleSwitch:**

Wenn Sie ein Display zwischen Fenster- und Vollbildmodus umschalten, indem Sie das Tastaturkürzel `CMD+RETURN` (`LALT+RETURN` unter Windows) drücken oder den Modus `#DISPMODE_MODESWITCH` an `ChangeDisplayMode()` übergeben, ändert Hollywood den Bildschirmmodus des Monitors nicht auf Systemen, auf denen hardwarebeschleunigte Skalierung verfügbar ist. Auf diesen Systemen simuliert Hollywood einfach den Vollbildmodus, indem das Display auf die aktuelle Auflösung des Monitors skaliert wird. Nur auf älteren Systemen oder Plattformen, die keine hardwarebeschleunigte Skalierung unterstützen, schaltet Hollywood den Bildschirm auf eine neue Auflösung um. Der Grund, warum das Umschalten der Bildschirmauflösung nicht mehr standardmäßig erfolgt, ist, dass dies oft sehr lange dauert und nicht alle Monitore dies unterstützen (z.B. Laptop-Bildschirme unterstützen dies oft nicht). Wenn Sie Hollywood zwingen möchten, die Auflösung des Monitors beim Vollbildmodus immer zu ändern und niemals den Vollbildmodus durch Skalierung zu simulieren, setzen Sie diesen Tag einfach auf `True`. Die Voreinstellung ist `False`. (V9.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Display-Plugins übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Viele der Tags von oben - vor allem diejenigen, die das Erscheinungsbild des Displays konfigurieren - sind auch als **Konsolenargumente** verfügbar, welche dem Benutzer eine gewisse Flexibilität bei der Steuerung der Erscheinung des Skripts geben. Zum Beispiel ist es möglich, den Anzeigemodus des Skripts (Fenster, Vollbild, etc.) oder Fensterstil (randlos, Größe veränderbar, etc.) auch von der Kommandozeile aus zu ändern. Wenn Sie das nicht wollen, müssen Sie beim kompilieren Ihrer Skripte das Argument `-locksettings` verwenden. Damit wird verhindert, dass der Benutzer Änderungen an den von Ihnen definierten Einstellungen der Präprozessor-Anweisung vornimmt.

**EINGABEN**

`table` eine Tabelle, die ein oder mehrere der oben aufgeführten Tags enthält

**BEISPIEL**

```
@DISPLAY {Title = "My App", X = #LEFT, Y = #TOP, Width = 320,
          Height = 240, Color = #WHITE}
```

Dieser Code öffnet ein 320x240 großes Display mit einem weißen Hintergrund. Das Display wird auf den obersten linken Rand des Host-Bildschirms positioniert. Der Fenstertitel lautet: "My App".

```
@DISPLAY {Width = 640, Height = 480, Borderless = True,
DragRegion = {
  {Type = #BOX, X=0, Y=460, Width=640, Height=20}, ; untere Ziehregion
  {Type = #BOX, X=0, Y=0, Width=20, Height=480} ; linke Ziehregion
}
```

```
}
```

Der obige Code wird ein randloses Display mit der Abmessung 640x480 öffnen. Im oberen Bereich des Displays wird es keine Ziehregion (Drag-Bar) haben, aber es wird durch die Angabe einer benutzerdefinierten Ziehregion mit dem Tag `DragRegion` im unteren und linken Bereiche des Displays jeweils eine geben. Beachten Sie, dass den `DragRegion`-Tag erfordert, dass Sie immer eine Liste von rechteckigen Regionen übergeben, auch wenn Sie nur eine einzige Region brauchen. Siehe oben für weitere Informationen.

## 24.9 FreeDisplay

### BEZEICHNUNG

`FreeDisplay` – löscht ein Display aus dem Speicher (V4.5)

### ÜBERSICHT

`FreeDisplay(id)`

### BESCHREIBUNG

Dieser Befehl schließt das angegebene Display und gibt alle Ressourcen frei. Nach `FreeDisplay()` ist das angegebene Display nicht mehr verfügbar.

Bitte beachten Sie, dass Sie `FreeDisplay()` nicht auf das aktuelle Display aufrufen können. Sie müssen zuerst ein anderes Display mit `SelectDisplay()` wählen, bevor Sie `FreeDisplay()` aufrufen können. Dies liegt daran, dass Hollywood immer ein aktuelles Display erfordert. Selbst wenn alle Displays geschlossen sind und es kein sichtbares mehr gibt, wird Hollywood intern noch ein aktuelles Display haben, welches die Grafiken verarbeiten kann.

### EINGABEN

<code>id</code>	ID des Displays, welches aus dem Speicher gelöscht werden soll. Darf nicht das aktuelle sein.
-----------------	---

## 24.10 GetDisplayModes

### BEZEICHNUNG

`GetDisplayModes` – findet alle verfügbaren Bildschirmmodi (V5.0)

### ÜBERSICHT

`t = GetDisplayModes([monitor])`

### BESCHREIBUNG

Dieser Befehl kann dazu verwendet werden, um alle Bildschirmmodi zu erhalten, die vom System unterstützt werden. `GetDisplayModes()` speichert dann alle unterstützten Modi in einer Tabelle ab. Die zurückgegebene Tabelle ist eine Sammlung von mehreren Untertabellen, die alle über `Width` und `Height` (Breite/Höhe) initialisierte Elemente haben.

Dieser Befehl ist nützlich um herauszufinden, ob ein bestimmter Bildschirmmodus tatsächlich von diesem System unterstützt wird, bevor Sie versuchen, mit dem Befehl `ChangeDisplayMode()` in diesen Modus zu wechseln.

**EINGABEN**

**monitor** optional: Monitor-ID, dessen Modi abgefragt werden (Standardwert ist 1 für den primären Monitor)

**RÜCKGABEWERTE**

**t** Tabelle mit allen verfügbaren Anzeigemodi

**BEISPIEL**

```
t = GetDisplayModes()
For Local k = 0 To ListItems(t) - 1
    DebugPrint("Mode", k + 1, "Width:", t[k].Width, "Height:", t[k].Height)
Next
```

Der obige Code fragt die Bildschirmmodi ab und gibt dann alle verfügbaren Modi aus.

## 24.11 GetMonitors

**BEZEICHNUNG**

GetMonitors – gibt Informationen über die verfügbaren Monitore zurück (V6.0)

**ÜBERSICHT**

```
t = GetMonitors()
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um Informationen über alle zur Verfügung stehenden Monitoren zu erhalten. Eine Tabelle wird zurückgegeben, die eine Anzahl von Untertabellen über die Abmessung und die Position jedes Monitors auf dem Desktopbildschirm.

Bitte beachten Sie, dass die beschriebene Art der Monitorpositionen auf dem Desktopbildschirm plattformabhängig ist. Siehe [Abschnitt 24.13 \[Unterstützung mehrerer Monitore\]](#), [Seite 389](#), für Details.

**EINGABEN**

Keine

**RÜCKGABEWERTE**

**t** Tabelle mit allen verfügbaren Monitoren und ihre Desktopausrichtung

**BEISPIEL**

```
t = GetMonitors()
For Local k = 0 To ListItems(t) - 1
    DebugPrint("Monitor", k + 1, "X:", t[k].X, "Y:", t[k].Y,
        "Width:", t[k].Width, "Height:", t[k].Height)
Next
```

Der obige Code fragt die Monitor-Datenbank des Betriebssystems ab und gibt dann die Informationen über alle verfügbaren Monitore aus.

## 24.12 HideDisplay

### BEZEICHNUNG

HideDisplay – minimiert das aktuelle Display (V3.0)

### ÜBERSICHT

HideDisplay([toback])

### BESCHREIBUNG

Dieser Befehl minimiert das aktuelle Display. Die Skriptausführung wird fortgesetzt, während das Display minimiert ist. Sie können den Befehl `ShowDisplay()` verwenden, um ein minimiertes Display wieder zurück zu bringen.

Wenn Sie das Display anstelle der Minimierung schließen möchten, verwenden Sie stattdessen den Befehl `CloseDisplay()`.

Ab Hollywood 8.0 gibt es ein optionales Argument namens `toback`, das nur von AmigaOS und kompatiblen Systemen unterstützt wird. Wenn Sie den Wert auf `True` setzen, wird das Display nicht minimiert. Stattdessen wird es ausgeblendet, indem Sie es ganz nach unten (hinten) im Fensterstapel des aktuellen Bildschirms schieben, d.h. alle anderen Fenster werden dann davor angezeigt.

### EINGABEN

**toback** optional: Bei `True` wird das Fenster durch Verschieben an den unteren Rand des Fensterstapels (oder ganz nach hinten) ausgeblendet, anstatt es zu minimieren; dies wird nur auf AmigaOS und kompatiblen Systemen unterstützt (Standard ist `False`) (V8.0).

### BEISPIEL

```
HideDisplay()
Wait(100)
ShowDisplay()
```

Dieser Code verbirgt das Display, wartet zwei Sekunden und das Display erscheint wieder.

## 24.13 Mehrerer Monitore

Beginnend mit Hollywood 6.0 werden Multi-Monitor-Systeme unterstützt. Sie können bei dem Präprozessor-Anweisung `DISPLAY` oder den Befehlen `CreateDisplay()` oder `OpenDisplay()` mit dem Tag `Monitor` festlegen, auf welchem Monitor das Display geöffnet wird. Wenn Sie erfahren wollen, wie viele verfügbaren Monitore im System enthalten sind, können Sie das mit dem Befehl `GetMonitors()` herausfinden.

Bitte beachten Sie, dass es von der Plattform abhängt, wie die erweiterten Desktop-Koordinaten mit mehreren Monitoren behandelt werden. Zum Beispiel wird unter Windows der Bereich des primären Monitors auf dem erweiterten Desktop immer beim Versatz 0:0 starten. Wenn es vom primären Monitor nach links oder oben geht, wird das mit negativen Koordinaten ausgeglichen. Unter Linux jedoch werden negativ Koordinaten nicht verwendet. So kann es unter Linux vorkommen, dass der Bereich des primären Monitors in den erweiterten Desktop nicht mit Versatz 0:0 startet, sondern bei einem höheren Versatz, wenn es einen Monitor nach links oder oben von dem primären Monitor geht.

Hollywood kann diese Plattform Abhängigkeiten für Sie nicht umgehen, weil die Koordinatensysteme ein integraler Bestandteil der Fenster-Manager sind. Es könnte für den Programmierer in einigen Fällen sehr verwirrend werden, wenn Hollywood versucht, sein eigenes Koordinatennetz auf die Oberseite des Fenster-Managers zu legen. Normalerweise müssen Sie sich ohnehin nicht mit den absoluten Versätzen beschäftigen, da Befehle wie `MoveDisplay()` eh relativ auf die Versatz-Display-Monitore arbeiten. Somit geben Sie an, um wieviel sich das Display verschiebt und nicht die genauen Koordinaten.

## 24.14 MoveDisplay

### BEZEICHNUNG

MoveDisplay – verschiebt das Display an eine neue Position (V2.0)

### ÜBERSICHT

MoveDisplay(x, y)

### BESCHREIBUNG

Mit diesem Befehl wird das Display auf die neue Position verschoben, welche durch `x` und `y` angegeben wurden. Sie können auch Hollywoods besonderen Positionskonstanten verwenden. Um herauszufinden, welche `x`- und `y`-Positionen gültig sind, können Sie mit dem Befehl `GetAttribute()` die Attribute `#ATTRMAXWIDTH` und `#ATTRMAXHEIGHT` abfragen. Die Ausgangslage des Displays wird mit der `@DISPLAY` Präprozessor-Anweisung eingestellt.

### EINGABEN

<code>x</code>	neue <code>x</code> -Position für das Display
<code>y</code>	neue <code>y</code> -Position für das Display

### BEISPIEL

```
MoveDisplay(#LEFT, #TOP)
```

Dieser Code verschiebt das Display an die obere linke Ecke des Bildschirms.

## 24.15 OpenDisplay

### BEZEICHNUNG

OpenDisplay – öffnet ein Display (V4.5)

### ÜBERSICHT

OpenDisplay(id[, table])

### BESCHREIBUNG

Mit diesem Befehl wird ein Display geöffnet, welches zuvor mit `CreateDisplay()` erstellt wurde. Ab Hollywood 6.0 kann eine optionale Tabelle angegeben werden, mit welcher einige erweiterte Optionen konfiguriert werden können. Die folgenden Tags werden derzeit unterstützt:

<b>Mode:</b>	Mit diesem Tag können Sie einstellen, wie das Display geöffnet werden soll. Sie müssen diesem Tag eine der folgenden Zeichenfolgen übergeben:  <code>Windowed</code> Öffnet das Display im Fenstermodus.
--------------	--

**FullScreen**

Öffnet im Vollbildmodus. Dies kann die Auflösung Ihres Monitors auf die Abmessungen ändern, die am besten zu den Dimensionen Ihres Displays passen. Wenn Sie das nicht wollen, werfen Sie einen Blick auf die nachfolgenden Modi **FullScreenScale** und **FakeFullScreen**.

**FullScreenScale**

Dies ist ein spezieller Vollbildmodus, der die Auflösung Ihres Monitors nicht ändert. Stattdessen wird das Display von Hollywood so dimensioniert, dass es den Dimensionen Ihres Monitors entspricht. Zusätzlich aktiviert dieser Vollbildmodus die Autoskalierung, so dass Ihr Display automatisch skaliert wird, um sich den Dimensionen Ihres Monitors anzupassen. **FullScreenScale** verwendet standardmäßig Autoskalierung. Wenn Sie möchten, dass Ebenenskalierung verwendet wird, müssen Sie auch **ScaleMode** auf **#SCALEMODE\_LAYER** setzen. **FullScreenScale** ist besonders bei mobilen Geräten nützlich, deren Display-Hardware eine hartcodierte Auflösung hat und keine Auflösungsänderungen in gleicher Weise wie ein externer Monitor unterstützt, der mit einem Desktop-Computer verbunden ist. Der Nachteil von **FullScreenScale** ist, dass es langsamer ist, weil Hollywood alle Wiedergabe-Operationen auf die Dimensionen des Monitors skalieren muss. (V7.0)

**FakeFullScreen**

Öffnet das Display in einem unechten vollen Bildschirmmodus. Das bedeutet, dass Hollywood die Auflösung des Monitors nicht ändert, aber die Hintergrundfüllung deckt den ganzen Desktop ab. Somit erhält der Anwender den Eindruck, als ob Hollywood im Vollbildmodus ausgeführt wird, obwohl es auf dem Desktop läuft.

**ModeRequester**

Dies öffnet einen Anzeigemodus-Dialogfenster, in dem der Benutzer den gewünschten Vollbildmodus auswählen kann.

**Ask**

Es öffnet sich ein Dialogfenster, wo der Benutzer zwischen Fenster- und Vollbildmodus auswählen kann.

Voreingestellt ist, dass **OpenDisplay()** den Modus verwenden wird, welcher beim Erstellen des Displays angegeben wurde.

**ScrWidth, ScrHeight:**

Wenn **Mode** auf **FullScreen** gesetzt ist, können Sie mit diesen Tags die gewünschten Dimensionen für den Vollbildmodus einstellen. Voreinstellung ist, was beim Erstellen des Displays eingestellt wurde. Ab Hollywood 7.0 können Sie diese Tags auch auf die spezielle Konstante **#NATIVE** setzen. In diesem Fall wird Hollywood die Dimensionen des Host-Geräts des Displays verwenden.

**ScrDepth:**

Wenn **Mode** auf **FullScreen** gesetzt worden ist, ermöglicht dieser Tag, die gewünschte Farbtiefe für den Vollbildmodus zu setzen. Der Standardwert ist, was eingestellt wurde, als das Display erstellt wurde.

**Backfill:**

Dieser Tag kann die Hintergrundfüllung für dieses Display konfigurieren. Die Tabelle, die Sie hier angeben müssen, ist dieselbe, wie sein Pendant des **Backfill**-Tags von der Präprozessor-Anweisung **@DISPLAY**. Siehe [Abschnitt 24.8 \[DISPLAY\]](#), [Seite 370](#), für Details.

**NoSelect:**

Dieser Tag ermöglicht Ihnen, das neu geöffnete Display zum aktuellen Ausgabeziel auszuwählen. **NoSelect** ist standardmäßig auf **False** gesetzt, womit **OpenDisplay()** das angegebene Display öffnet und als aktuelles Ausgabeziel auswählt. Wenn Sie dieses Verhalten nicht wollen, übergeben Sie **True** in **NoSelect**. In diesem Fall müssen Sie manuell **SelectDisplay()** aufrufen, bevor Sie in das Display zeichnen können.

**Monitor:** Mit diesem Tag können Sie den Monitor auswählen, auf dem das Display geöffnet werden soll. Monitore werden von 1 bis zur Anzahl der zur Verfügung stehenden Monitore im System gezählt. Bitte beachten Sie, dass wenn Sie diesen Tag setzen, Befehle mit Displaykoordinaten, z.B. **MoveDisplay()**, sich nach dem Ursprung des im **Monitor** angegebenen Monitor mit relativen Werten orientieren. Voreingestellt ist der **Monitor**, auf welchem das Display erstellt wurde.

**XServer:** Dieser Tag kann verwendet werden, um den X-Server anzugeben, dem dieses Display zugewiesen wird. Standardmäßig verwendet Hollywood den X-Server, der beim Erstellen des Display angegeben wurde. Dieser Tag steht nur in der Linux-Version von Hollywood zur Verfügung.

**PubScreen:**

Dieser Tag wird nur auf AmigaOS und kompatiblen Betriebssystemen unterstützt. Er kann verwendet werden, um den öffentlichen Bildschirm anzugeben, auf dem dieses Display geöffnet wird. Sie müssen eine Zeichenfolge übergeben, die den Namen des öffentlichen Bildschirms enthält. Voreingestellt ist der öffentliche Bildschirm, welcher beim Erstellen des Displays angegeben wurde.

**ScreenName:**

Wenn dieses Display im Vollbildmodus geöffnet werden soll, können Sie den gewünschten öffentlichen Bildschirmnamen für das Display mit diesem Tag einstellen. Dieser Tag steht den AmigaOS und kompatiblen Versionen von Hollywood zur Verfügung.

**EINGABEN**

<b>id</b>	Identifikator des Displays, welches geöffnet wird
<b>table</b>	optional: Tabelle mit weiteren Optionen (siehe oben) (V6.0)



**BEISPIEL**

Siehe [Abschnitt 24.7 \[CreateDisplay\]](#), Seite 366.

## 24.16 Palettenmodus-Displays

Beim Anzeigen eines Paletten-Hintergrundbilds mit `DisplayBGPic()` oder beim Zuweisen eines Paletten-Hintergrundbilds zu einem Display mit der Präprozessor-Anweisung `@DISPLAY` oder `SetDisplayAttributes()`, versetzt Hollywood das Display in den Palettenmodus. Dies bedeutet, dass die einzigen Farben, die zum Zeichnen auf diesem Display verfügbar sind, die in der Palette sind. Dies ist sehr ähnlich zu alter Grafikhardware aus den 1980er und frühen 1990er Jahren, die nur eine feste Anzahl im Bereich von 2 bis 256 Farben auf dem Bildschirm anzeigen konnte.

Ein Vorteil des Palettenmodus besteht darin, dass durch Ändern der Farbe eines Stifts in der Palette alle Pixel, die diesen Stift verwenden, auch sofort ihre Farbe ändern. Dadurch ist es möglich, auf einfache Weise Code zu schreiben, der die Palettenfarben ausblendet oder sie zyklisch durchläuft. Farbwechseleffekte waren in den 1980er und frühen 1990er Jahren eine sehr beliebte Methode, um Grafiken wie Wasserfälle oder Regen zu animieren. Durch die Verwendung eines Hollywood-Displays im Palettenmodus können diese Effekte in Hollywood leicht nachgebildet werden.

Ein weiterer Vorteil von Palettenmodus-Displays besteht darin, dass der Speicherverbrauch viel geringer als bei der Verwendung von Echtfarb-Displays ist. Im 32-Bit-True-Color-Modus benötigt ein einzelnes Pixel 4 Byte Speicher, während im Palettenmodus ein einzelnes Pixel nur 1 Byte Speicher benötigt. So benötigt ein 1920x1080-Bild im 32-Bit-Modus etwa 8 Megabyte Speicher, im Palettenmodus jedoch nur 2 Megabyte.

Bei der Verwendung von Displays im Palettenmodus ist jedoch große Vorsicht geboten, da sonst das Zeichnen sehr langsam werden kann. Dies liegt daran, dass standardmäßig alle Grafiken, die Sie in einem Display im Palettenmodus zeichnen, der Palette des Displays neu zugeordnet werden, damit ihr Erscheinungsbild dem Original so nahe wie möglich kommt. Genauer gesagt, bedeutet die Neuuzuordnung, dass Hollywood alle Pixel der Quellgrafiken scannen und die beste Übereinstimmung für jedes Pixel in der Palette des Displays finden muss. Dies ist natürlich ein sehr zeitaufwändiger Vorgang und macht das Zeichnen sehr langsam. Aus diesem Grund können Sie am schnellsten auf einem Palettenmodus-Display zeichnen, indem Sie sicherstellen, dass alle Grafiken dieselbe Palette verwenden. Dann muss keine Neuuzuordnung durchgeführt werden und die Grafikdaten können einfach ohne langsamen Pixel-Anpassungsalgorithmus auf das Display kopiert werden.

Um die Neuuzuordnung eines Palettenmodus-Displays zu deaktivieren, müssen Sie `#PALETTE_MODE_PEN` an `SetPaletteMode()` übergeben. Dies ermöglicht das stiftbasierte Zeichnen. Wenn Sie Palettengrafiken auf das Display zeichnen, geht Hollywood davon aus, dass sie dieselbe Palette wie das Display verwenden. Sie können also einfach so auf das Display kopiert werden, wie sie sind.

Wenn der Palettenmodus auf `#PALETTE_MODE_PEN` eingestellt wurde, ignorieren einfarbige Zeichnungsbefehle wie `Box()` oder `Circle()` die an sie übergebene RGB-Farbe. Stattdessen zeichnen sie die Formen einfach mit dem Stift, der mit `SetDrawPen()` festgelegt wurde. Möglicherweise ignorieren auch aktive Schatten- und Rahmeneffekte die Standard-Schatten- und Rahmenfarbe, die mit `SetFormStyle()` oder `SetFontStyle()` eingestellte wur-

den, und zeichnet den Schatten und den Rahmen mit dem Stift, der mit `SetShadowPen()` und `SetBorderPen()` gesetzt wurde. Außerdem wird das Antialiasing deaktiviert, wenn `#PALETTEMODE_PEN` aktiv ist, da Paletten in den meisten Fällen nicht genug Farben haben, um Kanten zufriedenstellend mit Antialiasing zu glätten.

Beachten Sie jedoch, dass RGB-Grafiken auch dann noch neu zugeordnet werden müssen, wenn `#PALETTEMODE_PEN` aktiv ist, da es unmöglich ist, RGB-Grafiken auf andere Weise in ein Palettenmodus-Display zu zeichnen. Daher sollte das Zeichnen von 32-Bit-Echtfarbgrafiken auf Palettenmodus-Displays vermieden werden, da dies immer langsam ist, weil für diese Grafiken eine Neuuzuordnung erforderlich ist und daran kein Weg vorbei führt.

Die Art und Weise, wie Grafikdaten der Palette eines Displays neu zugeordnet werden, kann durch den Befehl `SetDitherMode()` konfiguriert werden. Auf diese Weise können Sie das Dithering aktivieren oder deaktivieren und den zu verwendenden Dithering-Algorithmus angeben.

Palettenmodus-Displays unterstützen alle Befehle von normalen Displays. Es ist auch möglich, Ebenen, Sprites, Übergangseffekte, Beschneidungsbereiche, Videos und Doppelpuffer mit Palettenmodus-Displays zu verwenden. Beachten Sie jedoch die oben genannten Hinweise, da sonst das Zeichnen sehr langsam werden kann.

Siehe [Abschnitt 41.1 \[Paletten-Übersicht\]](#), [Seite 841](#), um mehr über Paletten zu erfahren.

## 24.17 RefreshDisplay

### BEZEICHNUNG

RefreshDisplay – erzwingt die Aktualisierung des Displays (V9.0)

### ÜBERSICHT

RefreshDisplay([t])

### BESCHREIBUNG

Mit diesem Befehl kann die Aktualisierung des Displays erzwungen werden. Es ist normalerweise nicht erforderlich, diesen Befehl aufzurufen, da die Aktualisierung des Displays von Hollywood automatisch durchgeführt wird. Dieser Befehl dient nur zu Debugging-Zwecken. Das optionale Tabellenargument kann verwendet werden, um zusätzliche Optionen zu übergeben. Dies ist jedoch derzeit nur für den internen Gebrauch und nicht öffentlich dokumentiert.

### EINGABEN

t	optional: Tabelle mit weiteren undokumentierten Optionen (nicht dokumentiert)
---	---

## 24.18 SCREEN

### BEZEICHNUNG

SCREEN – konfiguriert den Bildschirmmodus für das Skript / VERALTET (V4.5)

### ÜBERSICHT

@SCREEN table

## BESCHREIBUNG

Wichtiger Hinweis: Dieser Präprozessor-Anweisung ist seit Hollywood 6.0 veraltet. Als Hollywood 6.0 die Unterstützung von mehreren Monitoren eingeführt hat, kann es auch mehrere Displays im Vollbildmodus auf separaten Monitoren darstellen. Deshalb ist eine einzelne `@SCREEN` Präprozessor-Anweisung nicht mehr ausreichend. Stattdessen sollten nun die Parameter mit der Präprozessor-Anweisung `@DISPLAY` oder dem Befehl `CreateDisplay()` konfiguriert werden. Sie können immer noch diese Präprozessor-Anweisung verwenden, aber es wird nur das erste Display beeinflussen.

Diese Präprozessor-Anweisung kann verwendet werden, um den Bildschirmmodus für das Skript zu konfigurieren. Standardmäßig werden alle Hollywoodskripte in einem Fenster geöffnet. Wenn Sie Ihr Skript im Vollbildmodus öffnen möchten, können Sie diesen Präprozessor verwenden.

Vor Hollywood 4.5 wurde der Bildmodus mit der Präprozessor-Anweisung `@DISPLAY` konfiguriert (was ab Hollywood 6.0 wieder empfohlen wird). Hollywood 4.5 führte jedoch mehrere Displays ein, was erforderlich machte, die Bildmoduseinstellungen in einer eigenen Präprozessor-Anweisung unterzubringen, weil mehrere Displays nicht im Vollbildmodus laufen können.

Sie übergeben dieser Präprozessor-Anweisung eine Tabelle, welche folgende Tabellentags enthalten kann:

**Mode:** Legt fest, mit welchem Anzeigemodus Ihr Skript gestartet werden soll. Dies kann entweder `Windowed`, `Fullscreen`, `FakeFullscreen` oder `Ask` sein. Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), [Seite 33](#), für Informationen, was die verschiedenen Modi bedeuten. Wenn Sie `Ask` angeben, wird Hollywood den Benutzer fragen, ob das Skript im Vollbild- oder Fenstermodus ausgeführt werden soll. Der Standardwert ist `Windowed` (Fenstermodus).

**HideTitleBar:** Dieser Tag kann verwendet werden, um die Titelleiste des Bildschirms zu verbergen (`True`). Es zeigt aber nur Wirkung, wenn Hollywood auf seinem eigenen Bildschirm geöffnet wurde oder wenn Sie eine Hintergrundfüllung verwenden. Der Standardwert ist `False`.

**Desktop:** Wenn Sie diesen Tag auf `True` setzen, wird das Hintergrundbild eine Kopie Ihres Desktopbildschirms beinhalten. Dies kann für einige nette Effekte mit diesem Bildschirm verwendet werden. Hollywood wird auch automatisch ein randloses Fenster öffnen, wenn dieser Tag auf `True` gesetzt ist. Beachten Sie, dass diese Einstellung Hollywood in einem speziellen Modus versetzt und Sie andere Displays nicht öffnen müssen.

**Width, Height:** Wenn **Mode** auf `Fullscreen` gesetzt ist, können Sie **Width** und **Height** verwenden, um die Dimensionen für den Bildschirm festzulegen, den Hollywood öffnen soll. Wenn Sie hier 0 übergeben, wird Hollywood die Dimensionen des Desktop-Bildschirms verwenden. Wenn Sie sie überhaupt nicht angeben, bestimmt Hollywood automatisch eine passende Bildschirmgröße.

**Depth:** Dieser Tag kann verwendet werden, um die Farbtiefe des Bildschirms festzulegen. Es kann nur verwendet werden, wenn Hollywood im Vollbildmodus

geöffnet wurde. Wenn Sie diesen Tag nicht angeben, wird Hollywood die Farbtiefe des Desktopbildschirms für den neuen Bildschirm verwenden. Normalerweise sollten Sie diesen Tag nicht verwenden und die Wahl Hollywood überlassen, eine geeignete Tiefe für das aktuelle System zu erfassen.

Alternativ können die Einstellungen für den Bildschirmmodus auch mit den **Konsolenargumente** konfiguriert werden. Wenn Sie diese Möglichkeit deaktivieren möchten, sollten Sie Ihre Skripte mit dem Konsolenargument **-locksettings** kompilieren. Das Umschalten zwischen Vollbild- und Fenstermodus zur Laufzeit ist mit dem Befehl **ChangeDisplayMode()** möglich.

## EINGABEN

**table**      Tabelle, welche den Bildschirmmodus für das Skript deklariert

## BEISPIEL

```
@SCREEN {Mode = "FullScreen", Width = 800, Height = 600}
```

Dieses Beispiel wird einen 800x600 Vollbildschirm für das Skript einrichten.

## 24.19 SelectDisplay

### BEZEICHNUNG

SelectDisplay – wählt ein Display als Ausgabeziel (V4.5)

### ÜBERSICHT

```
SelectDisplay(id[, noactivate])
```

### BESCHREIBUNG

Dieser Befehl macht das in **id** angegebene Display zum aktuellen Ausgabeziel. Das aktuelle Display ist das Display, auf welches nun alle Befehle von Hollywoods Grafikbibliothek Einfluss nehmen. Darüber hinaus arbeiten mehrere andere Befehle mit dem aktuellen Display, zum Beispiel wird **SetPointer()** die Maus des aktuell aktiven Display ändern.

Dieses optionale Argument **NoActivate** ist standardmäßig auf **False** gesetzt, womit das angegebene Display zum aktuellen Display und gleichzeitig aktiviert wird.

Stellen Sie sicher, dass Sie den Unterschied zwischen dem aktuellen und dem aktiven Display verstehen: Das aktuelle Display muss nicht unbedingt das aktive Display sein und umgekehrt. Das aktive Display hat nichts mit Hollywood selbst zu tun. Es bedeutet nur, dass der Window-Manager des Host-Betriebssystem das Display als aktiv markiert. Aktive Fenster werden auch die Tastaturfokus erhalten und auf einigen Betriebssystemen sind aktive Fenster immer nach vorne gesetzt.

Auf der anderen Seite ist das aktuelle Display das Display, in welches Befehle von Hollywood ihre Grafiken zeichnen können. So ist es auch möglich, ein inaktives Display zum aktuellen Display zu erklären.

Daher sollten Sie im Auge behalten, dass dieser Befehl das angegebene Display zum aktuellen Display macht. Optional ist es aber möglich, dieses Display auch zu aktivieren. Wenn Sie aber das Display nur aktivieren möchten, ohne dass es das aktuelle wird, benutzen Sie stattdessen den Befehl **ActivateDisplay()**.

Beachten Sie auch, dass Hollywood immer ein aktuelles Display erfordert. Es ist nicht möglich, ein Hollywoodskript zu erstellen, ohne dass ein aktuelles Display vorhanden wäre. Sie können jedoch die Illusion eines Hollywoodskripts ohne aktuelles Display erzeugen, indem Sie mit dem Befehl `CloseDisplay()` alle Displays schließen (oder mit dem `Hidden`-Tag der `@DISPLAY`-Anweisung alle Fenster verbergen). Es wird dann so erscheinen, als ob es kein aktuelles Display hat. Das ist aber eine falsche Annahme, das das aktuelle Display einfach nicht sichtbar ist. Selbst wenn alle Displays geschlossen sind, werden Sie noch in der Lage sein, Befehle wie `DisplayBrush()` aufzurufen. Sie können immer noch ihre Grafiken auf das Display zeichnen, auch wenn es momentan nicht sichtbar ist. Deshalb können Sie `SelectDisplay()` auch auf geschlossene Displays anwenden.

Stellen Sie sicher, dass Sie `SelectDisplay()` nicht mit den ähnlich genannten Befehlen wie `SelectBrush()`, `SelectBGPic()`, `SelectAnim()`, `SelectMask()`, und `SelectAlphaChannel()` verwechseln. All diese Befehle benötigen den Aufruf von `EndSelect()`, wenn Sie mit ihnen fertig sind, aber `SelectDisplay()` hat diese Anforderung nicht. `SelectDisplay()` funktioniert in der Tat auf eine ganz andere Art und Weise, so dass Sie nie `EndSelect()` aufrufen müssen. Wenn Sie auf das zuvor aktuelle Display zurückkehren wollen, müssen Sie `SelectDisplay()` erneut aufrufen.

Wenn Sie `SelectDisplay()` aufrufen, während `SelectBrush()` (oder eines der anderen `Select...` Befehle) aktiv ist, wird Hollywood intern `EndSelect()` aufrufen, um zunächst `SelectBrush()` zu beenden. Danach wird `SelectDisplay()` das aktuelle Display wechseln.

Doppelgepufferte Modi werden durch `SelectDisplay()` nicht aufgehoben. Sie können den Fokus sicher von einem doppeltgepufferten Display nehmen, indem Sie `SelectDisplay()` verwenden, einige Änderungen in dem anderen Display erledigen und dann wieder auf das doppeltgepufferte Display zurück kehren. Doppelgepufferte Modi werden vollständig von `SelectDisplay()` erhalten.

Beachten Sie auch, dass der Befehl `OpenDisplay()` das angegebene Display immer automatisch zum aktuellen Display erklärt, außer Sie benutzen das optionale Argument `NoSelect`. In der Regel müssen Sie `SelectDisplay()` überhaupt nicht aufrufen, wenn Sie `OpenDisplay()` benutzen. `SelectDisplay()` wird nur benötigt, wenn ein Display bereits geöffnet ist.

## EINGABEN

<code>id</code>	Identifikator des Displays, welches zum aktuellen wird
<code>noactivate</code>	optional: Auf <code>True</code> gesetzt wird das Display nicht aktiviert (voreingestellt ist <code>False</code> )

## 24.20 SetDisplayAttributes

### BEZEICHNUNG

`SetDisplayAttributes` – ändert die Attribute des aktuellen Displays (V4.5)

### ÜBERSICHT

`SetDisplayAttributes(table)`

## BESCHREIBUNG

Dieser Befehl kann verwendet werden, um ein oder mehrere Attribute des aktuellen Displays mit einem einzigen Aufruf zu ändern. Es ist eine sehr leistungsfähiger Befehl, der Ihnen ein Höchstmaß an Flexibilität für den Umgang mit Ihren Displays gibt. Fast alle Attribute des Displays können im laufenden Betrieb mit diesem Befehl geändert werden. Zum Beispiel können Sie ein Display größenveränderbar oder randlos machen.

Sie übergeben diesem Befehl eine Tabelle mit Attributen, die Sie ändern möchten. Hier ist eine Liste dieser Displayattribute:

**BGPic:** Mit diesem Tag können Sie das BGPic auszutauschen. Dies ist im Grunde das gleiche, als würden Sie `DisplayBGPic()` aufrufen. Der Vorteil des Austauschs von BGPics mit `SetDisplayAttributes()` ist, dass Sie zur gleichen Zeit noch andere Displayattribute ändern können. Um das BGPic für dieses Display auszutauschen, übergeben Sie in diesem Tag einfach die ID des neuen BGPic.

**Width, Height:**

Hier können Sie die Dimensionen des aktuellen Displays ändern. Dies ist das gleiche wie der Aufruf von `ChangeDisplaySize()`. Sie können entweder einen direkten Wert oder eine Zeichenkette mit einem Prozentsatz (z.B. "200%") angeben. Ab Hollywood 7.0 können Sie diese Tags auch auf die spezielle Konstante `#NATIVE` setzen. In diesem Fall wird Hollywood die Dimensionen des Host-Geräts des Displays verwenden.

**X, Y:** Hier können Sie die Position des aktuellen Displays ändern. Dies ist das gleiche wie der Aufruf `MoveDisplay()`. Sie können entweder einen direkten Wert oder eine von Hollywoods speziellen **Positionkonstanten** angeben.

**Title:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), [Seite 370](#), für Details.

**Borderless:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), [Seite 370](#), für Details.

**Sizeable:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), [Seite 370](#), für Details.

**Fixed:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), [Seite 370](#), für Details.

**NoHide:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), [Seite 370](#), für Details.

**NoModeSwitch:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), [Seite 370](#), für Details.

**NoClose:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), [Seite 370](#), für Details.

**HidePointer:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), [Seite 370](#), für Details.

**ScaleMode:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), [Seite 370](#), für Details.

**ScaleWidth, ScaleHeight:**

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), [Seite 370](#), für Details.

- SmoothScale:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.
- DragRegion:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.
- SizeRegion:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.
- PubScreen:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.2)
- FillStyle:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.2)
- Color:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.2)
- TextureBrush:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.2)
- TextureX, TextureY:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.2)
- GradientStyle:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.2)
- GradientAngle:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.2)
- GradientStartColor, GradientEndColor:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.2)
- GradientCenterX, GradientCenterY:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.2)
- GradientBalance:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.2)
- GradientBorder:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.2)
- GradientColors:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.2)
- HideFromTaskbar:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.3)
- HideOptionsMenu:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.3)
- Orientation:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V5.3)
- Menu:** Dieser Tag kann verwendet werden, um eine Menüleiste dem aktuellen Display zuzuweisen. Sie müssen die ID einer Menüleiste übergeben, welche entweder mit der Präprozessor-Anweisung `@MENU` oder dem Befehl `CreateMenu()` erstellt wurde. Wenn Sie den besonderen Wert -1 übergeben,



wird die derzeit aktuelle Menüleiste aus dem Display entfernt. Siehe [Abschnitt 37.8 \[MENU\]](#), Seite 754, für Details. (V6.0)

**Monitor:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V6.0)

**XServer:** Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V6.0)

**ScreenTitle:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V6.0)

**ScreenName:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V6.0)

**RememberPosition:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V7.0)

**HideTitleBar:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V9.0)

**ImmersiveMode:**  
Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details. (V9.0)

Weitere Bildschirmbefehle sind [GetFrontScreen\(\)](#), [GetPubScreens\(\)](#), [HideScreen\(\)](#), [ShowScreen\(\)](#) und [SetScreenTitle\(\)](#).

## EINGABEN

**table** Tabelle, die eine Auflistung von Attributen enthält. Siehe Liste oben

## BEISPIEL

```
SetDisplayAttributes({Borderless = True, Sizeable = True})
```

Dieser Code ändert das Display auf einen randlosen Rahmen und es wird in der Größe veränderbar.

## 24.21 SetSubtitle

### BEZEICHNUNG

**SetSubtitle** – setzt den Untertitel des aktuellen Displays (V8.0)

### ÜBERSICHT

```
SetSubtitle(subtitle$)
```

### PLATTFORMEN

Nur Android

### BESCHREIBUNG

Dieser Befehl ändert den Untertitel des derzeit aktiven Displays in den neuen Untertitel, der in `subtitle$` angegeben ist. Der Untertitel wird in der Aktionsleiste unterhalb des Displaytitels angezeigt, der mit [SetTitle\(\)](#) festgelegt werden kann. Um den Untertitel zu entfernen, übergeben Sie eine leere Zeichenkette in `subtitle$`. Standardmäßig gibt es überhaupt keinen Untertitel.

Alternativ können Sie den Untertitel auch in der Präprozessor-Anweisung [@DISPLAY](#) oder beim Erstellen des Displays mit [CreateDisplay\(\)](#) einstellen.



**EINGABEN**

```
subtitle$  
        neuer Untertitel für das Display
```

**BEISPIEL**

```
SetTitle("Written by Andreas Falkenhahn")
```

Der obige Code ändert den Untertitel in "Written by Andreas Falkenhahn".

## 24.22 SetTitle

**BEZEICHNUNG**

SetTitle – ändert den Titel des aktuellen Displays

**ÜBERSICHT**

```
SetTitle(title$)
```

**BESCHREIBUNG**

Dieser Befehl ändert den Titel des momentan aktiven Displays auf den in `title$` angegebenen neuen Titel. Dies ist nur nützlich, während Ihr Skript läuft. Wenn Sie Ihrer Anwendung einen globalen Titel geben möchten, verwenden Sie einfach die Präprozessor-Anweisung `@DISPLAY`.

Möchten Sie den Screen-/Bildschirmtitel ändern, verwenden Sie `SetScreenTitle()`.

**EINGABEN**

```
title$    neuer Displaytitel
```

**BEISPIEL**

```
SetTitle("My cool program")
```

Dieser Code ändert den Fenstertitel in "My cool program".

## 24.23 ShowDisplay

**BEZEICHNUNG**

ShowDisplay – holt ein minimiertes Display zurück (V3.0)

**ÜBERSICHT**

```
ShowDisplay()
```

**BESCHREIBUNG**

Mit diesem Befehl holen Sie das momentan minimierte Display an die Front zurück. Das Display muss vorher mit dem Befehl `HideDisplay()` oder durch den Benutzer mit der Minimierschaltfläche des Fensters minimiert worden sein.

Beachten Sie, dass Sie diesen Befehl nicht für geschlossene Displays verwenden können. Wurde ein Display geschlossen, müssen Sie `OpenDisplay()` verwenden.

**EINGABEN**

Keine

**BEISPIEL**

Siehe [Abschnitt 24.12 \[HideDisplay\]](#), Seite 389.

**24.24 Skalierungssysteme**

Beginnend mit Hollywood 4.0 stehen zwei Skalierungssysteme zur Verfügung. Beide können Ihr Skript dazu zwingen, in einer anderen Auflösung zu laufen, als es eigentlich entworfen wurde. Zum Beispiel schreibt man ein kleines Spiel in 320x240, so dass es schnell genug auf klassische Amigas läuft. Auf modernen Amigas wird jedoch 320x240 als ziemlich kleines Fenster angezeigt. So können Sie jetzt Hollywoods Skalierungssystem verwenden, um Ihr Skript ohne eine einzige Zeile Code zu ändern, in 640x480 oder 800x600 erscheinen zu lassen! Sie müssen nur eine der beiden Skalierungssystemen aktivieren.

Wenn ein Skalierungssystem aktiv ist, ist das Skript immer noch der Meinung, dass es in seiner ursprünglichen Auflösung ausgeführt wird. Dies bedeutet zum Beispiel, dass die Aufrufe wie

```
width = GetAttribute(#DISPLAY, 0, #ATTRWIDTH)
height = GetAttribute(#DISPLAY, 0, #ATTRHEIGHT)
```

320x240 zurückgeben wird, obwohl das Skript jetzt in einer ganz anderen Auflösung läuft. Das Skript wird nicht bemerken, dass eine Skalierungssystem überhaupt aktiv ist! Das Skalierungssystem wird komplett transparent am Anfang des Skripts installiert.

Skalierungssysteme können entweder über die Konsole oder dem Skript aktiviert werden. Die beiden Konsolenargumente heißen `-autoscale` und `-layerscale`. Im Skript geschieht dies über die Präprozessor-Anweisung `@DISPLAY` und dem Parameter `ScaleMode` oder durch den Aufruf vom Befehl `SetDisplayAttributes()`.

Hollywood besitzt zwei Skalierungssysteme:

1. Autoskalierung: Sie können dieses Skalierungssystem aktivieren, indem Sie das `-autoscale` Argument benutzen, den Parameter `#SCALEMODE_AUTO` mit der Präprozessor-Anweisung `@DISPLAY` zusammen verwenden oder den Befehl `SetDisplayAttributes()` aufrufen. Dieses Skalierungssystem ist ein Lowlevel Skalierungsmodul, das einfach alle grafischen Ausgaben von Hollywood auf die neue Dimensionen skaliert. Deswegen wird Autoskalierung mit allen Hollywood-Skripten ohne Ausnahmen funktionieren. Der Nachteil des Autoskalierung-Systems ist a), dass es ziemlich CPU-lastig werden kann, falls das Hostsystem keine hardwarebeschleunigte Skalierung unterstützt. b) dass es Vektorgrafiken im Bitmap-Modus skaliert, was beispielsweise bedeutet, dass z.B. TrueType-Schriften, Grafikgrundelemente oder Vektorpinsel sich in der Qualität verringern und c) das Zeichnen ist langsamer, da das gesamte Display aktualisiert werden muss, selbst wenn sich nur ein Pixel geändert hat. Sie können die Leistung der automatischen Skalierung verbessern, indem Sie einen Doppelpuffer zum Zeichnen verwenden. Kapseln Sie alle Zeichnungsbefehle in einem `BeginRefresh()` und `EndRefresh()` Abschnitt. Siehe [Abschnitt 28.4 \[BeginRefresh\]](#), Seite 569, für Details.

Eine weitere Möglichkeit, die Leistung der Autoskalierung stark zu verbessern, besteht darin, ein Plugin mit hardwarebeschleunigte Skalierung zu verwenden, z.B. GL Galore oder RebelSDL. Plugins mit hardwarebeschleunigter Skalierung können die Autoska-

lierung sehr schnell anwenden. Siehe [Abschnitt 5.4 \[Vorhandene Plugins\]](#), Seite 68, für Details.

2. Ebenenskalierung: Die Ebenenskalierung ist ein anspruchsvolles, High-Level Skalierungsmodul, welches Sie durch die Angabe des `-layerscale` Arguments, mit dem `#SCALEMODE_LAYER` Parameter mit der Präprozessor-Anweisung `@DISPLAY` oder dem Befehl `SetDisplayAttributes()` aktivieren. Diese Skalierungsmethode a) ist oftmals viel schneller als die Autoskalierung, da die Ebenen nur einmal skaliert werden b) bietet eine höhere Ausgabequalität für Vektorgrafiken (d.h. Grafikgrundelemente, Vektorpinsel, TrueType-Schriften), da sie ohne Qualitätsverlust skaliert werden können c) zeichnet schneller, weil nur Teile des Displays aktualisiert werden müssen. Der Nachteil der Ebenenskalierung ist, dass es nur funktioniert, wenn Hollywood im `Ebenenmodus` ist und es braucht mehr Speicher. Außerdem können sie aber auch nicht den Befehl `DisableLayers()` verwenden, wenn die Ebenenskalierung aktiv ist. Wenn Sie also die Ebenenskalierung verwenden wollen, muss Ihr ganzes Skript im Ebenenmodus laufen.

Wie es aus diesen Beschreibungen klingen mag, wäre Option (2) die ideale Skalierungsmethode. Das ist aber nicht unbedingt wahr. In der Tat ist die Autoskalierung in den meisten Fällen ziemlich ausreichend. Die Ebenenskalierung ist nur für Projekte wie Präsentationen wichtig, die zu einer HDTV-Auflösung oder ähnlichem verwendet werden. In diesem Fall ist es wichtig, dass die Vektorgrafiken im Vektormodus skaliert werden, so dass ein scharfes Ergebnis erzielt wird. Unter normalen Umständen sollte jedoch die Autoskalierung für die meisten Projekte ausreichen. Und lassen Sie sich nicht durch die Tatsache entmutigen, dass sie langsamer ist als die Ebenenskalierung. Auf modernen Systemen werden Sie wahrscheinlich nichts von dieser Verlangsamung bemerken!

Wenn Sie ein Skalierungssystem aktivieren, müssen Sie auch die Skalierungsauflösung angeben. Dies kann auf zwei verschiedene Arten erfolgen: Sie können die Skalierungsauflösung entweder mit den Konsolenargumenten `-scalewidth` und `-scaleheight` oder deren Gegenstücken `ScaleWidth` und `ScaleHeight` einstellen, wenn Sie `@DISPLAY` oder `SetDisplayAttributes()` verwenden. Ansonsten können Sie einen globalen Skalierungskoeffizienten mit den Konsolenargumenten `-scalefactor` oder `-systemscales` festlegen oder deren Laufzeitgegenstücke `@DISPLAY` oder `SetDisplayAttributes()` benutzen.

Beachten Sie, dass das Verhalten zwischen der Einstellung der Skalierungsauflösung mit `ScaleWidth` sowie `ScaleHeight` und der mit `ScaleFactor` sowie `SystemScale` unterscheidet. Wenn Sie `ScaleWidth` und `ScaleHeight` verwenden, wird die an diese Tags übergebene Größe streng erzwungen und Ihr Display behält diese Größe immer bei, selbst wenn das Skript eine Änderung durch den Befehl `ChangeDisplaySize()` vornimmt oder indem Sie ein Hintergrundbild mit einer anderen Größe mit dem Befehl `DisplayBGPic()` anzeigen. Die Displaygröße kann nur geändert werden, wenn zum Einstellen der Skalierungsauflösung `ScaleWidth` und `ScaleHeight` verwendet wurde und der Benutzer das Fenster ändert. Andernfalls ändert das Display niemals seine Abmessungen.

Bei der Verwendung von `ScaleFactor` oder `SystemScale` wird dagegen ein Skalierungskoeffizient auf die aktuelle Displaygröße angewendet. Dies bedeutet, dass die Skalierungsgröße nicht statisch ist wie bei `ScaleWidth` und `ScaleHeight` (siehe oben). Sie ist jedoch dynamisch, da sie relativ zur aktuellen Displaygröße ist, z.B. wenn ein Skalierungskoeffizient von 2.0 angewendet wird und ein Display zuerst 640x480 Pixel und später 800x600 Pixel groß ist, wird das Display zuerst auf 1280x960 Pixel und dann auf 1600x1200 Pixel erhöht.

Dies macht **ScaleFactor** ideal für die Skalierung eines Skripts für ein Display mit hoher Auflösung, da sichergestellt ist, dass sich das Skript genau gleich verhält, aber nur größer erscheint (oder kleiner, wenn Sie möchten!).

Es besteht auch der Unterschied, dass **SizeWindow**-Ereignisse nur dann an Ihr Display gesendet werden, wenn ein Skalierungssystem aktiv ist und Sie entweder **ScaleFactor** oder **SystemScale** verwenden, um die Auflösung der Skalierung festzulegen. Wenn Sie **ScaleWidth** und **ScaleHeight** verwenden, werden keine **SizeWindow**-Ereignisse an Ihr Display gesendet, da Änderungen der Fenstergröße vom Skalierungssystem verarbeitet werden.

Wenn Sie ein Skalierungssystem aktivieren, aber weder **-scalewidth**, **-scaleheight** noch **-scalefactor** oder **-systemscale** angeben, dann wird ein Skalierungssystem erst aktiviert, wenn der Benutzer die Fenstergröße ändert. Sobald er dies tut, wird die neue Fenstergröße als Skalierungsgröße festgelegt.

Schließlich können Sie das Argument **-smoothscale** angeben, um die interpolierte Antialias-Skalierung für Bitmap-Grafiken zu aktivieren. Dies sieht zwar besser aus, kann aber langsamer sein, wenn keine hardwarebeschleunigte Skalierung verfügbar ist.

## 25 Ebenenbibliothek

### 25.1 Übersicht

Hollywood bietet Ihnen ein leistungsstarkes und dennoch einfaches Ebenensystem, das in der Lage sein sollte, alles, was Sie für Ihre Anwendung brauchen, zu realisieren. Die Ebenen sind Kinder eines Hintergrundbildes. Jedes Hintergrundbild hat seine eigenen angebrachten Ebenen. Hollywoods Ebenensystem wird beim Start nicht aktiviert. Sie müssen es durch den manuellen Aufruf vom Befehl `EnableLayers()` aktivieren. Alternativ können Sie den Tag `Layers` in `@DISPLAY` oder `CreateDisplay()` verwenden. Sobald Ebenen aktiviert sind, wird jedes Objekt auf einer eigenen Ebene auf dem Bildschirm angezeigt.

Beachten Sie, dass Ebenen aktiviert/deaktiviert auf einer pro-Display-Basis basieren. Somit ist es durchaus möglich, Displays mit und ohne Ebenen zu mischen. Wenn Sie zum Beispiel zwei Displays haben, könnte Display 1 Ebenen verwenden und Display 2 nicht. Das ist durchaus möglich.

Was Sie vermeiden sollten, ist Ebenen in einem Display zu deaktivieren, wenn sie vorher aktiviert wurden. Dies kann man zwar tun, aber es sollte in jedem Fall vermieden werden, da das Arbeiten mit und ohne Ebenen deutlich unterschiedlich sind.

Lassen Sie uns jetzt einen Blick auf ein kurzes Beispiel werfen:

```
EnableLayers()
DisplayBGPic(2)
DisplayBrush(1, #CENTER, #CENTER)
Plot(100, 100, #RED)
Print("Hello World!")
Box(50, 50, 100, 100, #BLUE)
```

Der obige Code zeigt 4 verschiedene Objekttypen und legt zugleich vier Ebenen auf dem Hintergrundbild Nummer 2 an, da Ebenen aktiviert wurde. Jedes angezeigte Objekt erhält eine eigene Ebene und deshalb haben wir jetzt die folgenden Ebenen beim Hintergrundbild 2:

```
Ebene id 1: Pinsel 1 bei den Koordinaten #CENTER : #CENTER
Ebene id 2: Ein roter Pixel bei 100 : 100
Ebene id 3: Text "Hello World!"
Ebene id 4: Ein blaues Rechteck bei 50 : 50 mit der Größe 100 : 100
```

Jetzt können Sie alles tun, was man mit diesen Ebenen machen könnte, wie z.B. sie bewegen, Vordergrund Prioritäten tauschen, verstecken oder sie entfernen. Hollywood bietet viele Befehle, die Ebenen verarbeiten können.

Bitte beachten Sie, dass Ebenen-IDs dynamisch sind. Zum Beispiel wenn im oben genannten Code den Befehl

```
RemoveLayer(2)
```

aufrufen würde, dann würden die Ebenen-IDs geändert. Nach diesem Befehl haben wir die folgenden Ebenen für das Hintergrundbild 2:

```
Layer id 1: Pinsel 1 bei den Koordinaten #CENTER : #CENTER
Layer id 2: Text "Hello World!"
Layer id 3: Ein blaues Rechteck bei 50 : 50 mit der Größe 100 : 100
```

Sie sehen, dass der Text "Hello World!" jetzt die ID 2 hat und das Rechteck nun auf der Ebene 3 ist.

Ab Hollywood 2.0 gibt es einen neuen Befehl: `SetLayerName()`. Sie können ihn verwenden, um Ihren Ebenen einen eindeutigen Namen zu geben, so dass Sie die Ebene einfach durch ihren Namen ansprechen, statt über dessen ID-Adresse. Dies ist sehr nützlich, wenn Sie viele Ebenen haben und Sie sich nicht an ihre IDs erinnern möchten. Alle Befehle, die mit Ebenen arbeiten, akzeptieren nun zusätzlich zur numerischen ID auch einen Namen als Zeichenfolge. Hier ist wieder unser Beispiel:

```
Layer id 1: Pinsel 1 bei den Koordinaten #CENTER : #CENTER
Layer id 2: Ein roter Pixel bei 100 : 100
Layer id 3: Text "Hello World!"
Layer id 4: Ein blaues Rechteck bei 50 : 50 mit der Größe 100 : 100
```

Nun tun wir folgendes:

```
SetLayerName(1, "brush: 1")
SetLayerName(2, "red pixel")
SetLayerName(3, "text: hello world")
SetLayerName(4, "blue box")
```

Nun löschen wir die Ebene 2 mit dem Befehl

```
RemoveLayer("red pixel")
```

Wir müssen uns nicht um die Tatsache kümmern, dass die Ebenen IDs jetzt verändert wurden, da alle Ebenen Namen haben und so können wir sie leicht ansprechen.

Bitte beachten Sie, dass Ebenen vom Hintergrundbild immer privat sind. Zum Beispiel wenn Sie jetzt

```
DisplayBGPic(3)
```

aufrufen, können Sie auf keine Ebenen zugreifen. Wenn Sie jetzt nochmals

```
DisplayBGPic(2)
```

aufrufen, wird Hollywood Ihr Hintergrundbild 2 zusammen mit allen Ebenen anzuzeigen. So können Sie zwischen Hintergrundbilder wechseln und Sie müssen nicht wieder alle Ihre Daten anzuzeigen lassen. Wenn Sie Ebenen aktiviert haben, wird Hollywood alle Ebenen auf einem Hintergrundbild mit dem Befehl `DisplayBGPic()` automatisch anzuzeigen.

Um Speicherplatz zu sparen ist es jedoch ratsam, `FreeLayers()` aufzurufen, wenn Sie die Ebenen nicht mehr brauchen.

Bitte beachten Sie auch, dass Sie `EnableLayers()` aufrufen, bevor Sie auf die Objekte der Ebenen zugreifen können. Zum Beispiel wird der folgende Code nicht funktioniert:

```
DisplayBrush(1, #CENTER, #CENTER)
EnableLayers()
Undo(#BRUSH, 1)
```

Jeder Grafikbefehl überprüft, ob Ebenen aktiviert sind. Ist dies der Fall, wird eine Ebene hinzugefügt. Daher kann das obige Beispiel nicht funktionieren, da Ebenen nach `DisplayBrush()` aktiviert wurde. So müssen Sie den folgenden Code verwenden:

```
EnableLayers()
DisplayBrush(1, #CENTER, #CENTER)
Undo(#BRUSH, 1)
```

Dies funktioniert dann gut.

Wenn Sie Ebenen in Ihrer gesamten Anwendung verwenden, wird empfohlen, `EnableLayers()` gleich zu Beginn des Codes aufzurufen. Dadurch wird sichergestellt, dass Ebenen immer aktiviert sind.

Sobald eine Ebene auf dem Display ist, können Sie das Aussehen sehr leicht ändern. Hollywood bietet eine breite Palette von Befehlen für Ebenen an. Die stärkste von ihnen ist `SetLayerStyle()`, die verwendet werden kann, um fast alle Ebenenattribute mit nur einem einzigen Aufruf zu ändern. Es kann sogar die Attribute mehrerer Ebenen auf einmal ändern! Darüber hinaus können Sie eine Ebene mit `RotateLayer()` drehen und mit `ScaleLayer()` skalieren. Es ist auch möglich, Ebenen mit einem von Hollywoods Vielzahl von Übergangseffekten mit den Befehlen `ShowLayerFX()` und `HideLayerFX()` ein- und auszublenken.

## 25.2 AddMove

### BEZEICHNUNG

AddMove – fügt der Bewegungsliste ein Objekt hinzu (V1.5)

### ÜBERSICHT

```
[id] = AddMove(id, type, sourceid[, par1, par2, par3])
```

### BESCHREIBUNG

Dieser Befehl fügt ein Objekt in die von `id` angegebene Bewegungsliste hinzu. Wenn die Bewegungsliste noch nicht vorhanden ist, wird sie durch diesen Befehl erstellt. Sie können auch `Nil` als ID angeben, womit `AddMove()` eine neue Bewegungsliste erstellt und die automatisch ausgewählte ID zurückgibt. Bewegungslisten werden für ein optimiertes Zeichnen mit `DoMove()` verwendet. Die optionalen Parameter `par1`, `par2` und `par3` geben verschiedene Dinge an, abhängig davon, welchen Objekttyp Sie übergeben haben.

Die folgenden Typen werden momentan von `AddMove()` unterstützt:

**#BRUSH** fügt den Pinsel mit dem Identifikator `sourceid` zu der Liste hinzu; `par1` gibt die X-Position des Pinsels an und `par2` die Y-Position; `par3` ist unbenutzt.

**#HIDEBRUSH** versteckt den durch `sourceid` bestimmten Pinsel; optionale Parameter werden nicht benutzt

**#HIDELAYER** versteckt den durch `sourceid` bestimmte Ebene; optionale Parameter werden nicht benutzt

**#INSERTBRUSH** fügt den durch `sourceid` bestimmten Pinsel an der durch `par3` bestimmten Ebenenposition ein; `par1` gibt die X-Position des Pinsels an und `par2` die Y-Position. Siehe [Abschnitt 25.19 \[InsertLayer\]](#), [Seite 422](#), um mehr Informationen über das Einfügen von Ebenen zu erhalten.

**#LAYER** fügt die von `sourceid` angegebene Ebene in die Bewegungsliste ein; `par1` gibt die X-Position für die Ebene, `par2` die Y-Position; neu in Hollywood 4.0: mit `par3` kann ein Sichtbarkeitsmodus angegeben werden. 0 bedeutet

"Zeige die Ebene", 1 bedeutet "verstecke die Ebene" und 2 bedeutet "aktuelle Sichtbarkeit beibehalten" (das heißt, Ebene bleibt verborgen, wenn sie gerade versteckt ist); `par3` ist standardmäßig 0, womit immer die Ebene gezeigt wird, auch wenn sie zur Zeit versteckt ist

#### #NEXTFRAME

zeigt ein neues Einzelbild einer Animationsebene an; `par1` gibt die neue X-Position für die Ebene, `par2` die Y-Position; mit `par3` wird das Einzelbild angegeben; geben Sie 0 ein, um das nächste Bild anzuzeigen, -1 für das letzte Bild der Animation (V2.0)

#### #NEXTFRAME2

gleich wie `#NEXTFRAME`, nimmt aber als `sourceid` die ID einer Ebene; dies ermöglicht es, Animationsebene direkt zu adressieren; `par1` gibt die neue X-Position für die Ebene, `par2` die Y-Position; `par3` gibt das Einzelbild an; geben Sie 0 ein, um das nächste Bild anzuzeigen, -1 für das letzte Bild der Animation (V2.5)

#### #REMOVELAYER

entfernt die durch `sourceid` angegebenen Ebenen aus dem Ebenencache des Hintergrundbildes; optionale Parameter werden nicht benutzt

#### #TEXTOBJECT

fügt das Textobjekt mit dem Identifikator `sourceid` zur Liste hinzu; `par1` gibt die X-Position des Pinsels an und `par2` die Y-Position; `par3` ist unbenutzt

#### #UNDO

fügt eine `Undo()` Operation zur Liste hinzu; `sourceid` gibt den Typ des zu entfernenden Objekts an, `par1` gibt den Identifikator (ID) des zu entfernenden Objekts an, `par2` gibt die Rücknahmetiefe an; `par3` ist unbenutzt; Siehe [Abschnitt 25.60 \[Undo\]](#), [Seite 471](#), für Details.

Nachdem Sie die Bewegungsliste mit Objekten gefüllt haben, können Sie `DoMove()` aufrufen, um das neue Display zu zeichnen.

Bitte beachten Sie: Es ist nicht möglich, mehrere Objekte desselben Typs und derselben ID in Ihrer Bewegungsliste zu haben. Zum Beispiel können Sie das folgende nicht tun:

```
DisplayBrush(1, #LEFT, #TOP)
DisplayBrush(1, #RIGHT, #BOTTOM)

/* Das wird nicht funktionieren */
AddMove(1, #BRUSH, 1, #CENTER, #CENTER)
AddMove(1, #BRUSH, 1, #LEFTOUT, #TOPOUT)
/* Das wird nicht funktionieren */
```

```
DoMove(1)
```

Obenstehender Code würde nicht funktionieren, weil Sie Pinsel 1 zweimal in derselben Liste verwenden. Hollywood weiß dann nicht, welcher Pinsel zu verwenden ist, was zu unvorhersehbaren Ergebnissen führen würde.

Siehe [Abschnitt 25.7 \[DoMove\]](#), [Seite 412](#), für Details.



**EINGABEN**

<b>id</b>	ID der Bewegungsliste oder <b>Nil</b> fürs Erstellen einer neuen Bewegungsliste
<b>type</b>	Typ des Objekts, das hinzugefügt werden soll (siehe Liste oben)
<b>sourceid</b>	hängt vom angegebenen Typ ab (siehe Liste oben)
<b>par1</b>	hängt vom angegebenen Typ ab (siehe Liste oben)
<b>par2</b>	hängt vom angegebenen Typ ab (siehe Liste oben)
<b>par3</b>	hängt vom angegebenen Typ ab (siehe Liste oben)

**RÜCKGABEWERTE**

<b>id</b>	optional: Identifikator der Bewegungsliste; wird nur zurückgegeben, wenn Sie <b>Nil</b> als Argument 1 angegeben haben (siehe oben)
-----------	---

**BEISPIEL**

Siehe [Abschnitt 25.7 \[DoMove\]](#), Seite 412.

## 25.3 ClearMove

**BEZEICHNUNG**

ClearMove - löscht die Bewegungsliste (V1.5)

**ÜBERSICHT**

ClearMove(id)

**BESCHREIBUNG**

Dieser Befehl löscht alle Objekte, die sich in der durch **id** angegebenen Liste befinden. Nachdem Sie dieses Kommando aufgerufen haben ist Ihre Liste wieder leer und kann mit neuen Objekten befüllt werden.

Bitte lesen Sie die Dokumentation von [DoMove\(\)](#) und [AddMove\(\)](#), um mehr über Hollywoods Bewegungslisten zu erfahren.

**EINGABEN**

<b>id</b>	ID der zu löschenden Bewegungsliste
-----------	-------------------------------------

**BEISPIEL**

Siehe [Abschnitt 25.7 \[DoMove\]](#), Seite 412.

## 25.4 CopyLayer

**BEZEICHNUNG**

CopyLayer – kopiert eine Ebene (V9.1)

**ÜBERSICHT**

CopyLayer(id, pos[, t])

**BESCHREIBUNG**

Dieser Befehl kopiert die durch **id** angegebene Ebene und fügt die Kopie an der durch **pos** angegebenen Ebenenposition ein. In **pos** kann der Sonderwert 0 übergeben werden,

um die kopierte Ebene als letzte einzufügen. `CopyLayer()` kopiert alle Ebenenattribute außer dem Namen der Ebene, da dieser einmalig sein muss. Sie können das optionale Tabellenargument `t` verwenden, um einen Namen für die kopierte Ebene anzugeben.

Das optionale Argument `t` unterstützt diese Tags:

**Name:** Wenn Sie der neuen Ebene einen Namen zuweisen möchten, setzen Sie diesen Tag auf eine Zeichenkette, die den gewünschten Namen enthält. Standardmäßig wird der neuen Ebene kein Name zugewiesen.

**Hidden:** Dieser Tag kann auf `True` gesetzt werden, um die neue Ebene nach der Erstellung automatisch auszublenden. Voreingestellt ist `False`.

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl verwenden können. Siehe **Abschnitt 25.1 [Einführung in die Ebenentechnik], Seite 405**, für Details.

## EINGABEN

`id` Identifikator oder Name der zu kopierenden Ebene  
`pos` gewünschte Position für die neue Ebene oder 0 für die letzte Ebene  
`t` optional: Tabelle mit weiteren Parametern

## BEISPIEL

```
EnableLayers
SetFillStyle(#FILLCOLOR)
Box(0, 0, 320, 480, #RED)
CopyLayer(1, 2, {Hidden = True})
SetLayerStyle(2, {X = 320, Color = #BLUE, Hidden = False})
```

Der obige Code erstellt eine gefüllte rote Rechteckebene, kopiert sie, ändert die Farbe der kopierten Ebene in Blau und positioniert sie neben der roten Ebene.

## 25.5 CreateLayer

### BEZEICHNUNG

`CreateLayer` – erstellt eine neue Ebene (V4.7)

### ÜBERSICHT

`CreateLayer(x, y, width, height[, table])`

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um eine neue Ebene dem aktuellen BGPic einzufügen. Die Ebene wird die Dimensionen erhalten, welche in `width/height` (Breite/Höhe) angegeben wurde und sie wird an der angegebenen Position `x/y` angezeigt. Dieser Befehl wird entweder eine Ebene vom Typ `#BRUSH` oder vom Typ `#ANIM` erstellen. Wenn Sie eine Animationsebene erstellen möchten, müssen Sie die gewünschte Anzahl der Einzelbilder im Tag `Frames` der optionalen Tabelle `table` übergeben.

Die optionale Tabelle erkennt die folgenden Tags:

**Frames** Gibt die Anzahl der Einzelbilder für diese Ebene an. Wenn diese Option auf 1 gesetzt ist, wird `CreateLayer()` eine Pinselebene erstellen. Andernfalls wird eine Animationsebene erstellt, welche die angegebene Anzahl von

	Einzelbildern enthalten wird. Der Standardwert ist 1 (was bedeutet, dass <code>CreateLayer()</code> eine Pinselebene erstellt).
Color	Stellt die anfängliche <b>RGB-Farbe</b> der Ebene\$ ein. Der Standardwert ist \$000000 (das ist Schwarz).
Mask	Setzen Sie diesen Tag auf <b>True</b> , wenn <code>CreateLayer()</code> eine Maske an der neuen Ebene anbringen soll. Wenn dies <b>True</b> ist, muss <b>AlphaChannel</b> <b>False</b> sein. Der Standardwert ist <b>False</b> .
AlphaChannel	Setzen Sie diesen Tag auf <b>True</b> , wenn <code>CreateLayer()</code> einen Alphakanal an der neuen Ebene anbringen soll. Wenn dies <b>True</b> ist, muss <b>Mask</b> <b>False</b> sein. Der Standardwert ist <b>False</b> .
Clear	Dieser Tag wird nur dann berücksichtigt, wenn entweder <b>AlphaChannel</b> oder <b>Mask</b> auf <b>True</b> gesetzt wurde. Wenn das der Fall ist, gibt <b>Clear</b> an, ob die Maske oder der Alphakanal gelöscht werden sollen (das heißt vollständig transparent) oder nicht (das heißt undurchsichtig). Der Standardwert ist <b>False</b> , was bedeutet, dass die neue Maske oder der Alphakanal undurchsichtig sind.

Darüber hinaus können Sie eine oder mehrere der **Standard-Tags zum Zeichnen** im optionalen Tabellenargument übergeben. Mit diesen Tags können Sie zum Beispiel die Position der Ebene angeben, einen Namen zuweisen und den Ankerpunkt dieser Ebene ändern. Siehe **Abschnitt 29.17 [Standard-Tags zum Zeichnen]**, **Seite 613**, für Details.

`CreateLayer()` ist der bevorzugte Weg, um eine leere Ebene zu erstellen, die Sie später mit dem Befehl `SelectLayer()` ändern können. Natürlich könnte man auch einen leeren Pinsel mit `CreateBrush()` erstellen und dann mit `DisplayBrush()` oder `InsertLayer()` als Ebene einfügen, aber das ist nicht so effektiv wie die der neue `CreateLayer()` Befehl. Denn wenn man dann `SelectLayer()` mit einer Ebene benutzt, die aus einer bestehenden Pinselquelle erstellt wurde, muss Hollywood zunächst eine Kopie der Ebene erstellen, weil `SelectLayer()` nur den Inhalt der Ebene und nicht den Inhalt des Pinsels ändern.

Dies ist mit Pinselebenen nicht sehr kritisch, aber mit vielen Einzelbildern einer Animationsebene hingegen schon. `SelectLayer()` auf eine solche Animationsebene anzuwenden, würde einige Zeit dauern. In diesen Fällen ist `CreateLayer()` wirklich viel effektiver.

## EINGABEN

x	gewünschte X-Position für die neue Ebene
y	gewünschte Y-Position für die neue Ebene
width	gewünschte Breite der Ebene
height	gewünschte Höhe der Ebene
table	optional: weitere Optionen; kann ein oder mehrere Tags von oben oder die <b>Standard-Tags zum Zeichnen</b> beinhalten

## BEISPIEL

```
CreateLayer(#CENTER, #CENTER, 100, 100, {Color = #RED})
SelectLayer(1)
```

```
Circle(0, 0, 50, #WHITE)
EndSelect
```

Der obige Code erstellt eine neue 100x100 rote Ebene und zeichnet dann einen weißen Kreis darauf.

## 25.6 DisableLayers

### BEZEICHNUNG

DisableLayers – schaltet das Ebenensystem für das aktuelle Display ab (V1.5)

### ÜBERSICHT

DisableLayers()

### BESCHREIBUNG

Dieser Befehl schaltet das eingebaute Ebenensystem des aktuellen Displays ab.

Bitte beachten Sie, dass dieser Befehl keine Ebene löscht, die an irgendwelche Hintergrundbilder zugewiesen sind. Diese werden behalten, bis Sie sie freigeben oder Hollywood beendet wird. Also können Sie Ebenen auch zeitweise abstellen und später wieder einschalten, ohne eine Ebene zu verlieren.

Bitte beachten Sie aber, dass es in der Regel nicht ratsam ist, zwischen ein- und ausgeschalteten Ebenen zu wechseln, weil beide Modi nicht wirklich miteinander kompatibel sind. Somit ist es die beste Idee, gleich beim Erstellen des Displays zu entscheiden, ob die Ebenentechnik angewendet wird oder nicht (z.B. beim Aufruf von **@DISPLAY** oder **CreateDisplay()**) und dann bleiben Sie bei dieser Entscheidung. Das Mischen der Modi im gleichen Display ist wirklich nur zu empfehlen, wenn Sie genau wissen, was Sie tun.

Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), [Seite 405](#), für Details.

### EINGABEN

keine

## 25.7 DoMove

### BEZEICHNUNG

DoMove – zeichnet die Bewegungsliste (V1.5)

### ÜBERSICHT

DoMove(id)

### BESCHREIBUNG

Dieser Befehl zeichnet alle Objekte auf einmal, die zu der in `id` angegebenen Liste hinzugefügt wurden (durch **AddMove()**). Das ist sehr nützlich, wenn Sie animierte Grafiken mit verschiedenen Objekten darstellen wollen. Wenn Sie jedes Objekt mit **DisplayBrush()** zeichnen würden, würde Ihr Display mit Sicherheit sehr stark flackern. Das kann verhindert werden, indem das Display mit einer einzigen Zeichnungsoperation erneuert wird. **DoMove()** realisiert das: Sie fügen alle Objekte, die gezeichnet werden sollen, zu einer Liste hinzu (mit **AddMove()**) und rufen dann **DoMove()** auf, was Ihre ganze Liste mit einer einzigen Zeichnungsoperation zeichnen wird.

Die Implementation von `DoMove()` ist, dass es die angegebene Liste nach den Objekten durchsucht, die gezeichnet werden sollen. Bei jedem Objekt in der Liste wird Hollywood prüfen, ob es sich bereits auf dem Bildschirm befindet. Wenn ja, wird es von Hollywood an seine neue Position bewegt. Wenn nein, wird es auf den Schirm gezeichnet. Deshalb werden, wenn sich alle Objekte schon auf dem Bildschirm befinden und nur mit `DoMove()` bewegt werden müssen, alle Ebenenpositionen beibehalten. Befinden sich Objekte in der Liste, die nicht bereits auf dem Bildschirm sind, werden sie neu gezeichnet und erhalten die obersten Ebenenpositionen zugewiesen.

Nachdem `DoMove()` beendet ist, sollten Sie `ClearMove()` aufrufen. Das wird die angegebene Liste leeren, so dass Sie sie wieder mit neuen Objektpositionen verwenden können.

Dieser Befehl setzt **aktivierte Ebenen** voraus.

### EINGABEN

`id`            Identifikator der zu zeichnenden Bewegungsliste

### BEISPIEL

```
EnableLayers()
For x = 0 To 400
AddMove(1, #BRUSH, 1, x, 0)
AddMove(1, #BRUSH, 2, x, 100)
AddMove(1, #BRUSH, 3, x, 200)
AddMove(1, #BRUSH, 4, x, 300)
DoMove(1)
ClearMove(1)
Next
```

Dieser Code bewegt Pinsel 1 bis 4 von 0 nach 400. Sie werden kein Flackern bemerken, da die `DoMove`-Liste-Technologie verwendet wird.

## 25.8 DumpLayers

### BEZEICHNUNG

`DumpLayers` – gibt interne Informationen über Ebenen aus (V2.0)

### ÜBERSICHT

`DumpLayers([what])`

### BESCHREIBUNG

Wurde erst ab Hollywood V7.0 dokumentiert.

Dieser Befehl gibt interne Informationen über die Ebenen des aktuellen BGPics an das Debug-Gerät aus. Dies ist vor allem für Debuggingzwecke nützlich. Die Informationen enthalten Positions- und Größenangaben, das Sichtbarkeitsflag der Ebene sowie die interne Speichergröße einer Hollywood-Ebene.

Das Argument `what` kann verwendet werden, um die auszugebende Informationen zu bestimmen. Intern können Hollywood-Ebenen bis zu drei verschiedene Darstellungsarten haben. Eine normale Darstellung ohne irgendwelche Transformationen, eine transformierte Darstellung und eine ebenenskalierte-transformierte Darstellung. Die transformierte Darstellung wird durch Befehle wie `RotateLayer()` und `ScaleLayer()` erzeugt, während

die ebenenskalierte-transformierte Darstellung einer Ebene entweder die normale oder die transformierte Ebene mit zusätzlichen Transformationen darstellt, welche durch das Skalierungssystem hinzugefügt wurde.

Die folgenden Werte werden derzeit vom Argument **what** akzeptiert:

- 0:           Gibt Informatinen über die normale Darstellung der Ebene aus. Dies ist die Voreinstellung.
- 1:           Gibt Informatinen über die transformierte Darstellung der Ebene aus.
- 2:           Gibt Informatinen über die ebenenskalierte-transformierte Darstellung der Ebene aus.

Beachten Sie, dass auch wenn keine ebenenskalierte-transformierte Darstellung aktuell aktiv ist, Wert 2 immer das physische Erscheinungsbild einer Ebene darstellt. Also, wenn Sie Details über das physische Erscheinungsbild einer Ebene wissen müssen, geben Sie immer 2 beim Argument **what** an, auch wenn derzeit keine Transformation aktiv ist.

Beachten Sie auch, dass dieser Befehl die Positions- und Größeninformationen, vom Skript und die tatsächliche physische Position und Größe ausgibt. Die richtige physikalische Position und Größeninformationen werden in Klammern ausgegeben.

All diese Informationen sind jedoch für normale Programmierer wohl nicht sehr sinnvoll. Dieser Befehl dient hier hauptsächlich für Debuggingzwecke. Wenn Sie Ebenenattribute für Ihr Skript abfragen müssen, verwenden Sie den Befehl **GetLayerStyle()** oder **GetAttribute()** mit dem Objekttyp **#LAYER**. Siehe [Abschnitt 25.15 \[GetLayerStyle\]](#), Seite 418, für Details. Siehe [Abschnitt 40.4 \[GetAttribute\]](#), Seite 808, für Details.

Sie müssen **Ebenen aktivieren** bevor Sie diesen Befehl nutzen können. Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), Seite 405, für Details.

## EINGABEN

**what**           optional: bestimmt, welche Informationen ausgegeben werden (siehe oben); voreingestellt ist 0

## 25.9 EnableLayers

### BEZEICHNUNG

EnableLayers – schaltet das Ebenensystem für das aktuelle Display ein (V1.5)

### ÜBERSICHT

EnableLayers()

### BESCHREIBUNG

Dieser Befehl schaltet das eingebaute Ebenensystem des aktuellen Displays ein. Um eine der Ebenenbefehle in diesem Display zu verwenden, müssen Sie zuerst diesen Befehl ausführen. Alternativ können Sie die Ebenen bereits bei der Erstellung des Displays unter Verwendung des Tags **Layers** mit den Befehlen **@DISPLAY** oder **CreateDisplay()** aktivieren.

Bitte beachten Sie aber, dass es in der Regel nicht ratsam ist, zwischen ein- und ausgeschalteten Ebenen zu wechseln, weil beide Modi nicht wirklich miteinander kompatibel

sind. Somit ist es die beste Idee, gleich beim erstellen des Displays zu entscheiden, ob die Ebenentechnik angewendet wird oder nicht (z.B. beim Aufruf von `@DISPLAY` oder `CreateDisplay()`) und dann bleiben Sie bei dieser Entscheidung. Das Mischen der Modi im gleichen Display ist wirklich nur zu empfehlen, wenn Sie genau wissen, was Sie tun.

Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), Seite 405, für Details.

#### EINGABEN

keine

## 25.10 FreeLayers

#### BEZEICHNUNG

FreeLayers – löscht die Ebenen eines Hintergrundbildes (V1.5)

#### ÜBERSICHT

`FreeLayers([keep])`

#### BESCHREIBUNG

Dieser Befehl löscht alle Ebenen, die mit dem aktuellen Hintergrundbild verbunden sind. Sie sollten diesen Befehl aufrufen, wenn Sie mit Ebenen auf einem Hintergrundbild fertig sind, da es schon einigen Arbeitsspeicher freigibt.

Standardmäßig werden bei `FreeLayers()` alle Ebenen gelöscht, aber vorher auch in das Hintergrundbild gezeichnet. Das bedeutet, dass nach dem Aufruf von `FreeLayers()` keine sichtbaren Änderungen angezeigt werden. Die Ebenen werden weg sein, aber es sieht so aus, als wären sie noch da, weil sie in das Hintergrundbild gezeichnet werden. Wenn Sie das nicht möchten, setzen Sie den Parameter `keep` auf `False`. In diesem Fall werden die Ebenen gelöscht und nicht in den Hintergrund gezeichnet. Dies ist dasselbe wie das Aufrufen von `RemoveLayers()`.

Bitte beachten Sie: Ebenen werden nicht automatisch ihren Speicher freigeben, wenn Sie ein neues Hintergrundbild anzeigen. Sie müssen den Arbeitsspeicher dieser Ebenen selbst von Hand freigeben.

Sie müssen [Ebenen aktivieren](#), bevor Sie diesen Befehl benutzen können. Um mehr über Hollywoods Ebenen zu erfahren, lesen Sie bitte die [Einführung in die Ebenentechnik](#).

#### EINGABEN

`keep` optional: ob die Ebenen in das Hintergrundbild gezeichnet werden sollen oder nicht (Standardeinstellung ist `True`) (V4.0)

## 25.11 GetLayerAtPos

#### BEZEICHNUNG

GetLayerAtPos – gibt die oberste Ebene an der bestimmten Position zurück (V4.7)

#### ÜBERSICHT

`id, name$ = GetLayerAtPos(x, y)`

**BESCHREIBUNG**

Dieser Befehl gibt die oberste Ebene an der angegebenen Position zurück. Dies ist nützlich, wenn sie eine interaktive Benutzeroberfläche erstellen, wo Ebenen mit der Maus verschoben werden können oder die Maus über eine Ebene bewegt wird und der Stil dieser Ebene geändert wird. Die Position, welche diesem Befehl übergeben wird, ist relativ zu der oberen linken Ecke des Displays. Somit ist eine Position von (0.0) die linke obere Ecke.

`GetLayerAtPos()` gibt die ID der obersten Ebene an der angegebenen Position, sowie den Namen dieser Ebene zurück. Wenn die Ebene keinen Namen hat, wird eine leere Zeichenkette in `name$` zurückgegeben. Wenn sich keine Ebene an der angegebenen Position befindet, wird 0 in `id` und eine leere Zeichenfolge in `name$` zurückgegeben.

**EINGABEN**

`x`            Position x, die abgefragt wird  
`y`            Position y, die abgefragt wird

**RÜCKGABEWERTE**

`id`            Identifikator der obersten Ebene an dieser Position oder 0, wenn es keine Ebene an dieser Position hat  
`name$`        Name der obersten Ebene oder eine leere Zeichenkette (""), wenn die Ebene keinen Namen hat oder keine Ebene gefunden wurde

## 25.12 GetLayerGroupMembers

**BEZEICHNUNG**

`GetLayerGroupMembers` – gibt alle Ebenen der Ebenengruppe zurück (V10.0)

**ÜBERSICHT**

```
t = GetLayerGroupMembers(group$)
```

**BESCHREIBUNG**

Dieser Befehl findet alle Ebenen in der durch `group$` angegebenen Ebenengruppe und gibt sie in einer Tabelle zurück. Die Reihenfolge, in der die Ebenen in der Gruppe dieser Tabelle zurückgegeben werden, ist beliebig.

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Siehe **Abschnitt 25.1 [Einführung in die Ebenentechnik]**, Seite 405, für Details.

**EINGABEN**

`group$`       Name der zu verwendenden Ebenengruppe

**RÜCKGABEWERTE**

`t`            Tabelle mit allen Ebenen der Gruppe

**BEISPIEL**

```
t = GetLayerGroupMembers("mygroup")
For Local k = 0 To ListItems(t) - 1 Do DebugPrint(t[k])
```

Der obige Code ruft alle Ebenen der Ebenengruppe "mygroup" ab und gibt sie aus.



## 25.13 GetLayerGroups

### BEZEICHNUNG

GetLayerGroups – gibt alle Ebenengruppen im aktuellen BGPic zurück (V10.0)

### ÜBERSICHT

```
t = GetLayerGroups()
```

### BESCHREIBUNG

Dieser Befehl sammelt alle Ebenengruppen im aktuellen BGPic und gibt sie in einer Tabelle zurück. Die Reihenfolge, in der die Gruppen in der Tabelle zurückgegeben werden, ist beliebig.

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), [Seite 405](#), für Details.

### EINGABEN

keine

### RÜCKGABEWERTE

t            Tabelle mit allen Ebenengruppen

### BEISPIEL

```
t = GetLayerGroups()  
For Local k = 0 To ListItems(t) - 1 Do DebugPrint(t[k])
```

Der obige Code ermittelt alle Ebenengruppen und gibt sie aus.

## 25.14 GetLayerPen

### BEZEICHNUNG

GetLayerPen – gibt die Stiftfarbe aus der Palette der Ebene zurück (V9.0)

### ÜBERSICHT

```
color = GetLayerPen(id, pen)
```

### BESCHREIBUNG

Dieser Befehl ruft die Farbe des durch **pen** angegebenen Stifts aus der Palette in der durch **id** angegebenen Ebene ab. Die Farbe wird als **RGB color** zurückgegeben.

Sie müssen **Ebenen aktivieren** bevor Sie diesen Befehl nutzen können. Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), [Seite 405](#), für Details.

### EINGABEN

id            Identifikator der Ebene

pen           Stift, dessen Farbe Sie ermitteln wollen (beginnend bei 0)

### RÜCKGABEWERTE

color        Farbe des Stifts, angegeben als **RGB-Farbe**

### BEISPIEL

```
color = GetLayerPen(1, 0)
```

Der Code gibt die Farbe des ersten Stifts der Ebene 1 zurück.

## 25.15 GetLayerStyle

### BEZEICHNUNG

GetLayerStyle – gibt den Stil einer Ebene zurück (V4.0)

### ÜBERSICHT

```
t = GetLayerStyle(id)
```

### BESCHREIBUNG

Dieser Befehl liefert alle Stilattribute der angegebenen Ebene. Die verschiedenen Attribute werden in einer Tabelle zurückgegeben, die Sie dann untersuchen können. Der Inhalt der Stiltabelle von diesem Befehl hängt von der Art der Ebene ab, die Sie angegeben haben. Eine vollständige Übersicht über alle Stilelemente, die von diesem Befehl zurückgegeben werden, finden Sie unter dem Befehl [SetLayerStyle\(\)](#), der eine Liste der Ebenenstilelemente enthält und bei welchen Ebenentypen sie gelten. Siehe [Abschnitt 25.48 \[SetLayerStyle\]](#), [Seite 449](#), für Details.

Bitte beachten Sie, dass dieser Befehl immer alle Attribute prüft, so dass es manchmal ein wenig dauern kann. Wenn Sie nur einige grundlegende Informationen über eine Ebene benötigen, könnte es stattdessen mit dem Befehl [GetAttribute\(\)](#) schneller gehen.

### EINGABEN

id            ID der zu prüfenden Ebene

### RÜCKGABEWERTE

t            eine Tabelle mit allen Attributen dieser Ebene

### BEISPIEL

```
t = GetLayerStyle(1)
Print("This layer is at position", t.x, ":", t.y, "!")
```

Der obige Code fragt die Art der Ebene 1 ab und zeigt dann deren Position.

## 25.16 GroupLayer

### BEZEICHNUNG

GroupLayer – fügt Ebene(n) zur Ebenengruppe hinzu (V10.0)

### ÜBERSICHT

```
GroupLayer(group$, layer1[, layer2, ...])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um eine oder mehrere Ebenen zu der durch **group\$** angegebenen Ebenengruppe hinzuzufügen. Wenn die durch **group\$** angegebene Ebenengruppe noch nicht existiert, wird sie automatisch von **GroupLayer()** erstellt. Ebenengruppen werden einfach durch eine Namenszeichenkette referenziert, die beliebige Zeichen enthalten kann, solange der Name der Gruppe nicht bereits von einer Ebene verwendet wird. Die Ebene(n), die der Gruppe hinzugefügt werden sollen, müssen durch ihre ID in den Parametern nach **group\$** angegeben werden. Sie können diesem Befehl eine unbegrenzte Anzahl von Ebenen übergeben.

Wenn Sie Ihre Ebenen gruppiert haben, können Sie den Namen der Gruppe an die meisten Befehle übergeben, die sich mit Ebenen befassen, z.B. könnten Sie eine Gruppe

von Ebenen anzeigen, indem Sie einfach den Namen Ihrer Ebenengruppe an `ShowLayer()` übergeben. Sie können auch alle Ebenen einer Ebenengruppe auf einmal verschieben, indem Sie die Ebenengruppe an `MoveLayer()` übergeben usw.

Beachten Sie, dass beim Übergeben von Gruppen anstelle von einzelnen Ebenen an Befehlen der Ebenenbibliothek, diese Befehle die Ebenengruppe nicht als eigene Entität behandeln, sondern einfach die entsprechende Operation auf alle Ebenen anwenden, die Teil der Gruppe sind. Wenn Sie beispielsweise `MoveLayer()` für eine Ebenengruppe aufrufen und 100:100 als neue Position übergeben, verschiebt Hollywood nicht die Gruppe als Ganzes auf Position 100:100, sondern alle Ebenen der Gruppe einzeln auf 100:100, so dass nach dem Aufruf alle Ebenen, die Teil der Gruppe sind, bei der Position 100:100 erscheinen, d.h. sie alle an der gleichen Position sind, was möglicherweise nicht das ist, was Sie erwartet haben. Wenn Sie Ebenen verschieben möchten, die Teil einer Gruppe sind, und ihre individuelle Position innerhalb der Gruppe beibehalten möchten, müssen Sie stattdessen `TranslateLayer()` aufrufen, da dies das Verschieben von Ebenen relativ zu ihrer aktuellen Position ermöglicht. Siehe [Abschnitt 25.59 \[TranslateLayer\]](#), [Seite 470](#), für Details.

Ebenen können auch direkt bei ihrer Erstellung zu einer Gruppe hinzugefügt werden, indem der Name der Gruppe im Tag `Group` der [Standard-Tags zum Zeichnen](#) übergeben wird, die von allen Hollywood-Befehlen verwendet werden, die eine Ebene hinzufügen. Siehe [Abschnitt 29.17 \[Standard-Tags zum Zeichnen\]](#), [Seite 613](#), für Details.

Um eine Ebene aus einer Gruppe zu entfernen, verwenden Sie den Befehl `UngroupLayer()`. Siehe [Abschnitt 25.62 \[UngroupLayer\]](#), [Seite 474](#), für Details. Sobald einer Gruppe keine Ebene mehr zugeordnet ist, wird sie automatisch gelöscht.

Eine weitere Möglichkeit, Ebenen zu gruppieren, besteht darin, sie zusammenzuführen. Im Vergleich zum Gruppieren von Ebenen bedeutet das Zusammenführen von Ebenen, dass sie zu einer einzigen Ebene werden. Ein Vorteil zusammengeführter Ebenen besteht darin, dass sie als Ganzes behandelt werden, beispielsweise wenn sie mithilfe von Übergangseffekten ein- oder ausgeblendet werden. Bei gruppierten Ebenen hingegen werden Übergangseffekte für jede Ebene in der Gruppe einzeln angezeigt. Siehe [Abschnitt 25.24 \[MergeLayers\]](#), [Seite 425](#), für Details.

Sie müssen [Ebenen aktivieren](#) bevor Sie diesen Befehl nutzen können. Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), [Seite 405](#), für Details.

## EINGABEN

<code>group\$</code>	Name der Gruppe, zu der die Ebene(n) hinzugefügt werden soll(en)
<code>layer1</code>	erste Ebene, die der Gruppe hinzugefügt wird
<code>...</code>	weitere Ebenen, die der Gruppe hinzugefügt werden können

## BEISPIEL

```
EnableLayers
SetFillStyle(#FILLCOLOR)
Box(0, 0, 100, 100, #RED, {Hidden = True})
Box(100, 0, 100, 100, #GREEN, {Hidden = True})
Box(200, 0, 100, 100, #BLUE, {Hidden = True})
GroupLayer("mygroup", 1, 2, 3)
TranslateLayer("mygroup", 170, 190)
```

```
ShowLayerFX("mygroup", #SCROLLSOUTH)
```

Der obige Code erstellt drei ausgeblendete 100x100-Rechtecke, gruppiert sie und verschiebt dann die Gruppe in die Mitte der 640x480-Displays und scrollt sie von Süden her ein.

## 25.17 HideLayer

### BEZEICHNUNG

HideLayer – verbirgt eine Ebene (V1.5)

### ÜBERSICHT

```
HideLayer(id)
```

### BESCHREIBUNG

Dieser Befehl verbirgt die durch `id` angegebene Ebene oder Ebenengruppe, dadurch ist sie nicht mehr auf dem Display sichtbar. Die Ebene wird nicht gelöscht, sondern nur verborgen. Sie können sie mit `ShowLayer()` wieder zur Anzeige bringen. Wenn Sie sie komplett löschen wollen, benutzen Sie `RemoveLayer()` oder `Undo()`.

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Um mehr über Hollywoods Ebenen zu erfahren, lesen Sie bitte die **Einführung in die Ebenentechnik**.

### EINGABEN

`id`            Identifikator der Ebene oder Ebenengruppe, die verborgen werden soll

### BEISPIEL

```
EnableLayers()
NPrint("Hello World!")
WaitLeftMouse
HideLayer(1)
WaitLeftMouse
ShowLayer(1)
```

Dieses Beispiel schreibt "Hello World!" auf das Display, verbirgt den Text dann und zeigt ihn wieder an.

## 25.18 HideLayerFX

### BEZEICHNUNG

HideLayerFX – verbirgt eine Ebene mit Effekt (V1.9)

### ÜBERSICHT

```
[handle] = HideLayerFX(id[, table])
```

### BESCHREIBUNG

Dieser Befehl ist eine erweiterte Version des `HideLayer()` Befehls. Er versteckt den durch die `id` angegebene Ebene oder Ebenengruppe und benutzt dazu einen der vielen Übergangseffekte, die Hollywood besitzt. Sie können ebenfalls die Übergangsgeschwindigkeit angeben.

Ab Hollywood 4.0 verwendet dieser Befehl eine neue Syntax mit nur einer einzigen Tabelle als optionales Argument. Die alte Syntax wird weiterhin aus Kompatibilitätsgründen unterstützt. Das optionale Tabellenargument kann den Übergangseffekt konfigurieren. Folgende Optionen sind möglich:

**Type**        Legt den gewünschten Effekt für den Übergang fest. Siehe [Abschnitt 30.11 \[DisplayTransitionFX\]](#), [Seite 630](#), für eine Liste aller unterstützten Übergangseffekte. (voreingestellt ist `#RANOMEFFECT`)

**Speed**        Legt die gewünschte Geschwindigkeit für den Übergang fest. Je höher der Wert, desto schneller wird der Effekt gezeigt. (voreingestellt ist `#NORMALSPEED`)

**Parameter**    Einige Übergangseffekte akzeptieren einen zusätzlichen Parameter. Dieser kann hier angegeben werden. (voreingestellt ist `#RANDOMPARAMETER`)

**Async:**        Sie können dieses Feld verwenden, um ein asynchrones Zeichnungsobjekt für diesen Übergang zu erstellen. Wenn Sie hier `True` angeben, wird `HideLayerFX()` sofort verlassen und es wird ein Handler für das asynchrone Zeichnungsobjekt kreiert, den Sie dann mit dem Befehl `AsyncDrawFrame()` verwenden können. Siehe [Abschnitt 19.1 \[AsyncDrawFrame\]](#), [Seite 231](#), für weitere Informationen über asynchrone Zeichnungsobjekte.

**NoBorderFade**    Wenn die zu verbergende Ebene einen Rahmen hat, wird er nicht allmählich verblassen, sondern am Ende des Übergangseffekts in einem Rutsch entfernt. (V5.0)

**BorderFX:**        Wenn die Ebene, die ausgeblendet werden soll, einen Rahmen hat, wendet Hollywood den Übergangseffekt nur dann auf den Rahmen an, wenn es sich um eine transparente Ebene mit Text- oder Pixelgrafiken handelt. Für nicht-transparente Ebenen und Vektorgrafikebenen wird stattdessen ein allgemeiner Überblendungseffekt verwendet, da es sonst aufgrund von Unterschieden in den Randalgorithmen zu visuellen Störungen zwischen dem vorletzten und letzten Effekteinzelbild kommen würde. Wenn Ihnen dieser Fehler egal ist und Sie Hollywood zwingen möchten, den Übergangseffekt immer auf den Rahmen anzuwenden, setzen Sie diesen Tag auf `True`. Um Hollywood zu zwingen, immer den generischen Überblendungsmodus zu verwenden, setzen Sie diesen Tag auf `False`. (V9.0)

Bevor Sie diesen Befehl verwenden können, müssen Sie zuerst die [Ebenen aktivieren](#). Siehe [Abschnitt 25.1 \[Ebenen Einführung\]](#), [Seite 405](#), für Details.

## EINGABEN

**id**            Identifikator oder Name der zu verbergenden Ebene oder Ebenengruppe

**table**        optional: Tabelle, um den Übergangseffekt zu definieren

## RÜCKGABEWERTE

**handle**        optional: Handler auf ein asynchrones Zeichnungsobjekt; wird nur zurückgegeben, wenn `Async` auf `True` gesetzt wurde (siehe oben)

**BEISPIEL**

```
HideLayerFX(5, {Type = #CROSSFADE}) ; neue Syntax
```

OR

```
HideLayerFX(5, #CROSSFADE) ; alte Syntax
```

Dieser Code verbirgt die Ebene 5 mit einem schönen Effekt.

## 25.19 InsertLayer

**BEZEICHNUNG**

InsertLayer – fügt eine neue Ebene ein (V1.5)

**ÜBERSICHT**

```
InsertLayer(pos, type, id, x, y[, hidden])
```

**BESCHREIBUNG**

Dieser Befehl fügt eine neue Ebene des in **type** angegebenen Typs und dem Identifikator **id** an der Ebenenposition **pos** ein. Alle folgenden Ebenen werden verschoben und erhalten deshalb eine neue Kennung. Die neue Ebene wird außerdem an der durch **x** und **y** bestimmten Position angezeigt. Geben Sie in **pos** 0 als Position an, wird sie als letzte Ebene angehängt.

Folgende Objekttypen werden zur Zeit unterstützt:

**#BRUSH**      Fügt den in **id** angegebenen Pinsel bei **x/y** ein

**#TEXTOBJECT**  
Fügt das in **id** angegebene Textobjekt bei **x/y** ein

**#ANIM**      Fügt die in **id** angegebene Animation bei **x/y** ein (V2.0)

**#VIDEO**      Fügt das in **id** angegebene Video bei **x/y** ein (V6.0)

Seit Hollywood 1.9 können Sie das optionale Argument **hidden** angeben, welches eine verborgene Ebene einsetzen wird, die Sie mit dem Befehl **ShowLayer()** oder **ShowLayerFX()** anzeigen können.

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Um mehr über Hollywoods Ebenen zu erfahren, lesen Sie bitte die **Einführung in die Ebenentechnik**.

**EINGABEN**

<b>pos</b>	gewünschte Position für die Ebene oder 0 als letzte Ebene
<b>type</b>	Objekttyp, den Sie einfügen möchten (siehe Liste oben)
<b>id</b>	Identifikator des Objektes, das Sie einfügen möchten
<b>x</b>	x-Position der neuen Ebene
<b>y</b>	y-Position der neuen Ebene
<b>hidden</b>	optional: <b>True</b> , wenn die neue Ebene verborgen sein soll (voreingestellt ist <b>False</b> ) (V1.9)

**BEISPIEL**

```

EnableLayers()
SetFillStyle(#FILLCOLOR)
Box(0, 0, 100, 100, #RED)
Circle(#CENTER, #CENTER, 50, #BLUE)
TextOut(#RIGHT, #BOTTOM, "Hello World")
InsertLayer(1, #BRUSH, 1, #CENTER, #CENTER)

```

Der obige Code fügt Pinsel 1 als die erste Ebenen ein. Dies bedeutet, dass alle anderen Ebenen neu positioniert werden. Das rote Rechteck erhält jetzt Ebenenposition 2 (war vorher Ebene 1), der blaue Kreis ist jetzt Ebene 3 (war zuvor Ebene 2) und die "Hello World" Textebene wird 4 (war Ebene 3).

## 25.20 LayerExists

**BEZEICHNUNG**

LayerExists – überprüft, ob die angegebene Ebene existiert (V4.6)

**ÜBERSICHT**

```
ret = LayerExists(layer$)
```

**BESCHREIBUNG**

Dieser Befehl prüft einfach, ob die angegebene Ebene vorhanden ist. Sie müssen hier einen Ebenennamen angeben und keine ID der Ebene, da Ebenen-IDs per se vorhanden sind.

**EINGABEN**

layer\$      der Name der Ebene, die überprüft wird

**RÜCKGABEWERTE**

ret            True, falls die Ebene existiert, andernfalls False

## 25.21 LayerGroupExists

**BEZEICHNUNG**

LayerGroupExists – prüft, ob die Gruppe existiert (V10.0)

**ÜBERSICHT**

```
ok = LayerGroupExists(group$)
```

**BESCHREIBUNG**

Dieser Befehl prüft, ob die durch **group\$** angegebene Ebenengruppe existiert. Wenn dies der Fall ist, wird **True** zurückgegeben, andernfalls **False**.

Sie müssen **Ebenen aktivieren** bevor Sie diesen Befehl nutzen können. Siehe **Abschnitt 25.1 [Einführung in die Ebenentechnik]**, Seite 405, für Details.

**EINGABEN**

group\$      Name der zu verwendenden Ebenengruppe

**RÜCKGABEWERTE**

ok            **True**, wenn die Gruppe existiert, **False**, wenn sie nicht existiert

**25.22 LayerToBack****BEZEICHNUNG**

LayerToBack - verschiebt die Ebene an die hinterste Position (V5.0)

**ÜBERSICHT**

LayerToBack(layer[, swap])

**BESCHREIBUNG**

Dieser Befehl verschiebt die angegebene Ebene den ganzen Weg in den Hintergrund. LayerToBack() ist ein Komfortbefehl. Das gleiche könnte mit Hilfe von **SwapLayers()** oder **SetLayerZPos()** erreicht werden.

Wenn das optionale Argument **swap** auf **False** gesetzt ist, wird die Ebene durch einfaches Verschieben nach hinten gebracht. Dies unterscheidet sich von dem Standardverhalten, bei dem lediglich die Positionen der hintersten Ebene und der in **layer** angegebenen Ebene vertauscht werden. Wenn **swap** auf **False** gesetzt ist, kann **layer** auch der Name einer Ebenengruppe sein.

**EINGABEN**

**layer**        Ebene, die an die hinterste Position verschoben werden soll

**swap**        wird dies auf **False** gesetzt, werden die Ebenen nicht vertauscht, sondern die angegebene Ebene wird nur nach hinten verschoben (voreingestellt ist **True**) (V7.1)

**25.23 LayerToFront****BEZEICHNUNG**

LayerToFront - verschiebt die Ebene an die vorderste Position (V5.0)

**ÜBERSICHT**

LayerToFront(layer[, swap])

**BESCHREIBUNG**

Dieser Befehl verschiebt die angegebene Ebene den ganzen Weg nach Vorne. LayerToFront() ist ein Komfortbefehl. Das gleiche könnte mit Hilfe von **SwapLayers()** oder **SetLayerZPos()** erreicht werden.

Wenn das optionale Argument **swap** auf **False** gesetzt ist, wird die Ebene durch einfaches Verschieben nach vorne gebracht. Dies unterscheidet sich von dem Standardverhalten, bei dem lediglich die Positionen der vordersten Ebene und der in **layer** angegebenen Ebene vertauscht werden. Wenn **swap** auf **False** gesetzt ist, kann **layer** auch der Name einer Ebenengruppe sein.

**EINGABEN**

**layer**        Ebene, die an die vorderste Position verschoben werden soll



**swap** wird dies auf **False** gesetzt, werden die Ebenen nicht vertauscht, sondern die angegebene Ebene wird nur ganz nach vorne verschoben (voreingestellt ist **True**) (V7.1)

## 25.24 MergeLayers

### BEZEICHNUNG

MergeLayers – führt Ebenen in eine neue Ebene zusammen (V10.0)

### ÜBERSICHT

MergeLayers(layer1[, layer2, ..., t])

### BESCHREIBUNG

Dieser Befehl fügt die durch **layer1**, **layer2** usw. angegebenen Ebenen zu einer neuen Ebene zusammen, wobei die Quellebenen erhalten bleiben. Standardmäßig blendet **MergeLayers()** die Quellebenen automatisch aus, aber dieses Verhalten kann geändert werden, indem das Tag **AutoHide** im optionalen Tabellenargument auf **False** gesetzt wird. Statt einzelner Ebenen können Sie diesem Befehl auch Ebenengruppen übergeben, die zusammengeführt werden sollen.

Die neue Ebene, die von **MergeLayers()** erstellt wird, verwendet den speziellen Ebenentyp **#MERGED**. Ebenen vom Typ **#MERGED** können nicht transformiert werden (außer durch die Ebenenskalierung) und enthalten normalerweise alle Einstellungen ihrer untergeordneten Ebenen, z.B. Schatten, Rahmen, Filter, Transparenzeinstellungen usw., die in die Ebene gerendert werden, obwohl dies mit bestimmten Tags im optionalen Tabellenargument geändert werden kann. Im Vergleich zu Ebenengruppen, die mit **GroupLayer()** erstellt wurden, besteht ein Vorteil zusammengeführter Ebenen darin, dass sie bei der Anzeige mit Übergangseffekten als Ganzes behandelt werden, während bei der Darstellung von Ebenen, die Teil einer von **GroupLayer()** erstellten Gruppe sind, würde die Verwendung von Übergangseffekten die Übergänge auf jede Ebene der Gruppe einzeln anwenden, was möglicherweise nicht immer so gut aussieht, wie wenn die Übergänge auf die Ebenen als Ganzes angewendet werden. Diese Einschränkung von **GroupLayer()** kann also durch die Verwendung von **MergeLayers()** überwunden werden.

Das optionale Tabellenargument **t** kann die folgenden Tags enthalten:

#### AutoHide:

Gibt an, ob die Quellebenen automatisch von **MergeLayers()** ausgeblendet werden sollen oder nicht. Standardmäßig blendet **MergeLayers()** automatisch die Ebenen aus, die zu einer neuen zusammengeführt werden. Wenn Sie das nicht möchten, setzen Sie diesen Tag auf **False**. Voreingestellt ist **True**.

#### MergeShadow:

Legt fest, ob ein möglicher Ebenenschatten ebenfalls in die neue Ebene eingefügt werden soll oder nicht. Dies ist standardmäßig **True**. Wenn Sie dies auf **False** setzen, wird kein Schatteneffekt von einer der Quellebenen mit der neuen Ebene zusammengeführt, so dass die neue Ebene ohne Schatten erscheint. Natürlich ist es möglich, der neuen Ebene mit **SetLayerShadow()** oder den **Standard-Tags zum Zeichnen** einen Schatten hinzuzufügen.

**MergeBorder:**

Legt fest, ob eventuelle Randeffect der Ebene ebenfalls in die neue Ebene eingebunden werden sollen. Dies ist standardmäßig **True**. Wenn Sie dies auf **False** setzen, wird kein Rahmeneffekt von einer der Quellebenen mit der neuen Ebene zusammengeführt, so dass die neue Ebene ohne Rahmeneffekt erscheint. Natürlich ist es möglich, der neuen Ebene mit **SetLayerBorder()** oder den **Standard-Tags zum Zeichnen** einen Randeffect hinzuzufügen.

**MergeFilter:**

Gibt an, ob eventuelle Ebenenfilter auch in der neuen Ebene eingebunden werden sollen. Dies ist standardmäßig **True**. Wenn Sie dies auf **False** setzen, wird kein Filter aus einer der Quellebenen mit der neuen Ebene zusammengeführt, so dass die neue Ebene ohne Filter angezeigt wird. Natürlich ist es möglich, der neuen Ebene Filter hinzuzufügen, indem Sie **SetLayerFilter()** oder die **Standard-Tags zum Zeichnen** verwenden.

**MergeTransparency:**

Legt fest, ob eventuell vorhandene Ebenentransparenz ebenfalls in die neue Ebene eingebunden werden soll. Dies ist standardmäßig **True**. Wenn Sie dies auf **False** setzen, wird keine Transparenz von einer der Quellebenen mit der neuen Ebene zusammengeführt, so dass die neue Ebene ohne Transparenzeinstellung angezeigt wird. Natürlich ist es möglich, die Transparenz der neuen Ebene mit **SetLayerTransparency()** oder den **Standard-Tags zum Zeichnen** einzustellen.

**MergeFX:** Legt fest, ob eventuelle Ebenenübergangseffekte ebenfalls in die neue Ebene eingefügt werden sollen oder nicht. Dies ist standardmäßig **True**. Wenn Sie dies auf **False** setzen, wird kein Übergangseffekt von einer der Quellebenen mit der neuen Ebene zusammengeführt, so dass die neue Ebene ohne Übergangseffekte angezeigt wird.

Darüber hinaus unterstützt das optionale Tabellenargument auch Hollywoods **Standard-Tags zum Zeichnen**. Siehe **Abschnitt 29.17 [Standard-Tags zum Zeichnen]**, **Seite 613**, für weitere Informationen über die Standard-Tags, die fast alle Hollywood-Zeichenbefehle unterstützen.

Beachten Sie, dass zusammengeführte Ebenen nicht automatisch aktualisiert werden, wenn ihre Quellebenen ihre Grafik ändern. Sie müssen den Befehl **RefreshLayer()** verwenden, um eine Aktualisierung einer zusammengeführten Ebene zu erzwingen. Siehe **Abschnitt 25.30 [RefreshLayer]**, **Seite 430**, für Details. Beachten Sie auch, dass nur sichtbare Ebenen zusammengeführt werden. Ausgeblendete Ebenen werden von **MergeLayers()** ignoriert.

Sie müssen **Ebenen aktivieren** bevor Sie diesen Befehl nutzen können. Siehe **Abschnitt 25.1 [Einführung in die Ebenentechnik]**, **Seite 405**, für Details.

**EINGABEN**

<b>layer1</b>	erste Ebene oder Ebenengruppe, welche zusammengeführt werden soll
<b>...</b>	weitere Ebenen oder Ebenengruppen zum Zusammenführen
<b>t</b>	optional: Tabelle mit weiteren Optionen (siehe oben)

**BEISPIEL**

```

EnableLayers
SetFillStyle(#FILLCOLOR)
SelectBGPic(1)
Box(0, 0, 100, 100, #RED)
Box(100, 0, 100, 100, #GREEN)
Box(200, 0, 100, 100, #BLUE)
MergeLayers(1, 2, 3, {Name = "newlayer", Hidden = True})
MoveLayer("newlayer", #CENTER, #CENTER)
EndSelect
ShowLayerFX("newlayer", #ZOOMCENTER)

```

Der obige Code erstellt drei ausgeblendete 100x100-Rechtecke, fügt sie zu einer neuen Ebene zusammen, verschiebt diese neue Ebene in die Mitte und zeigt dann die zusammengeführte Ebene mit einem Übergangseffekt. Beachten Sie, dass wir `SelectBGPic()` verwenden, um sicherzustellen, dass vor unserem Aufruf von `ShowLayerFX()` nichts gezeichnet wird.

**25.25 ModifyLayerFrames****BEZEICHNUNG**

`ModifyLayerFrames` – ändert die Anzahl Einzelbilder einer Animationsebene (V4.7)

**ÜBERSICHT**

```
ModifyLayerFrames(id, frames[, pos])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die Einzelbilder einer Animationsebene zu erweitern oder zu verkleinern. Wenn Sie einen positiven Wert in `frames` angeben, dann wird die Animationsebene durch diese Anzahl von Einzelbildern vergrößert. Wenn Sie hingegen einen negativen Wert angeben, gibt das die Anzahl der Einzelbilder an, welche aus der Animationsebene entfernt werden.

Das optionale Argument `pos` kann verwendet werden, um festzulegen, wo die neuen Einzelbilder eingefügt oder entfernt werden sollen. Wenn Sie das optionale Argument nicht angeben oder auf 0 setzen, werden die Einzelbilder am Ende der Animationsebene hinzugefügt oder am Ende entfernt.

Dieser Befehl funktioniert nur mit Animationsebenen, die sich vollständig im Arbeitsspeicher befinden. Sie können ihn nicht für Animationsebenen verwenden, die ihre Einzelbilder dynamisch von der Festplatte laden.

**EINGABEN**

<code>id</code>	Identifikator der Animationsebene
<code>frames</code>	Anzahl der einzufügenden Einzelbilder (wenn der Wert positiv ist) oder Anzahl der zu entfernenden Einzelbilder (falls der Wert negativ wäre)
<code>pos</code>	optional: wo werden die Einzelbilder eingefügt oder entfernt (voreingestellt ist 0: Es wird am Ende eingefügt oder entfernt)

**BEISPIEL**

```
ModifyLayerFrames(1, -5, 1)
```

Dieser Code entfernt die fünf ersten Einzelbilder der Animationsebene 1.

**25.26 MoveLayer****BEZEICHNUNG**

MoveLayer – verschiebt die Ebene an eine neue Position (V1.9)

**ÜBERSICHT**

```
MoveLayer(id, xa, ya, xb, yb[, table])
```

```
MoveLayer(id, x, y) (V9.1)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um entweder die durch `id` angegebene Ebene an eine neue Position zu scrollen oder sie einfach ohne Scrollen an eine neue Position zu verschieben.

Wenn Sie die Argumente `xa`, `ya`, `xb` und `yb` übergeben, scrollt `MoveLayer()` die durch `id` angegebene Ebene sanft von der durch `xa` und `ya` angegebenen zu der durch `xb` und `yb` angegebenen Position. Weitere Konfigurationsmöglichkeiten sind mit dem optionalen Argument `table` möglich. Sie können die Bewegungsgeschwindigkeit, den Spezialeffekt und ob die Bewegung asynchron sein soll oder nicht, festlegen. Siehe [Abschnitt 43.46 \[MoveBrush\]](#), [Seite 948](#), für weitere Informationen zum optionalen Tabellenargument.

Wenn Sie nur die Argumente `x` und `y` übergeben, verschiebt `MoveLayer()` die Ebene einfach an die durch `x` und `y` angegebene Position. In diesem Fall kann `id` auch der Name einer Ebenengruppe sein.

Für alle Koordinaten können Sie die spezielle Konstante `#USELAYERPOSITION` angeben. Hollywood verwendet dann die aktuelle Position der Ebene.

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), [Seite 405](#), für Details.

**EINGABEN**

<code>id</code>	ID oder Name der Ebene
<code>xa</code>	Startposition x
<code>ya</code>	Startposition y
<code>xb</code>	Zielposition x
<code>yb</code>	Zielposition y
<code>table</code>	optional: Argument für weitere Optionen (siehe oben)

**BEISPIEL**

```
MoveLayer(5, #LEFTOUT, #CENTER, #RIGHTOUT, #CENTER)
```

Scrollt die Ebene 5 von außen links nach außen rechts vom Display und zentriert es vertikal.

```
MoveLayer(4, #USELAYERPOSITION, #USELAYERPOSITION, #LEFTOUT, #CENTER)
```

Scrollt die Ebene 4 von der aktuellen Position aus dem Bildschirm heraus.

```
MoveLayer(5, #CENTER, #CENTER)
```

Verschiebt die Ebene 5 in die Bildschirmmitte.

## 25.27 NextFrame

### BEZEICHNUNG

NextFrame – zeigt ein neues Einzelbild einer Animationsebene an (V2.0)

### ÜBERSICHT

```
NextFrame(id[, x, y, frame])
```

### BESCHREIBUNG

Dieser Befehl zeigt ein neues Einzelbild einer Animationsebene an. Wenn Sie beim optionalen Argument **frame** nichts angeben oder auf 0 setzen, wird **NextFrame()** das nächste Einzelbild der Animationsebene zeigen. Wenn Sie -1 im Argument **frame** übergeben, wird das letzte Einzelbild angezeigt. Die Argumente **x** und **y** können verwendet werden, um die Ebene an eine neue Position zu bewegen, während das Einzelbild gewechselt wird. Wenn Sie das nicht benötigen, übergeben Sie **#USELAYERPOSITION** und die Ebene bleibt, wo sie ist.

Ab Hollywood 9.0 kann dieser Befehl auch mit Textebenen verwendet werden, die sich im Listenmodus befinden, um die nächsten Listenelemente anzuzeigen. Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für Details.

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), [Seite 405](#), für Details.

### EINGABEN

<b>id</b>	Identifikator oder Name der Animationsebene
<b>x</b>	optional: neue x-Position der Ebene (voreingestellt ist <b>#USELAYERPOSITION</b> )
<b>y</b>	optional: neue y-Position der Ebene (voreingestellt ist <b>#USELAYERPOSITION</b> )
<b>frame</b>	optional: welches Einzelbild angezeigt werden soll (voreingestellt ist 0, womit das nächste Einzelbild angezeigt wird)

### BEISPIEL

```
EnableLayers
InsertLayer(1, #ANIM, 1, 0, #CENTER)
For k = 0 To 400 Step 3
    NextFrame(1, k, #USELAYERPOSITION)
    Wait(5)
Next
```

Spielt die Animationsebene Nummer 1 ab, während es sich von x-Position 0 bis 400 bewegt.

## 25.28 PauseLayer

### BEZEICHNUNG

PauseLayer – hält eine laufende Video-Ebene an (V6.0)

### ÜBERSICHT

PauseLayer(id)

### BESCHREIBUNG

Dieser Befehl pausiert die in `id` angegebene Video-Ebene. Diese Video-Ebene muss gerade das Video abspielen, wenn Sie diesen Befehl aufrufen. Sie können später die Wiedergabe mit dem Befehl `ResumeLayer()` fortsetzen.

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), [Seite 405](#), für Details.

### EINGABEN

`id`            Identifikator oder Name der Video-Ebene, welche pausieren soll

## 25.29 PlayLayer

### BEZEICHNUNG

PlayLayer – spielt eine derzeit gestoppte Video-Ebene ab (V6.0)

### ÜBERSICHT

PlayLayer(id)

### BESCHREIBUNG

Dieser Befehl startet die Wiedergabe der in `id` angegebenen Video-Ebene von Anfang an. Sie können die Wiedergabe stoppen, indem Sie den Befehl `StopLayer()` verwenden.

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), [Seite 405](#), für Details.

### EINGABEN

`id`            Identifikator oder Name der Video-Ebene, die von Anfang an gestartet wird

## 25.30 RefreshLayer

### BEZEICHNUNG

RefreshLayer – aktualisiert eine Ebene (V10.0)

### ÜBERSICHT

RefreshLayer(id)

### BESCHREIBUNG

Dieser Befehl aktualisiert die durch `id` angegebene Ebene oder Ebenengruppe. Dies ist normalerweise nicht nötig, da Hollywood Ebenen bei Bedarf automatisch aktualisiert. Es gibt jedoch eine Ausnahme: Aus Leistungsgründen werden zusammengeführte Ebenen, die mit `MergeLayers()` erstellt wurden, nicht automatisch aktualisiert, wenn sich die Grafik einer ihrer Quellebenen ändert. Daher müssen Sie zusammengeführte Ebenen

manuell aktualisieren, indem Sie `RefreshLayer()` aufrufen, wann immer Sie möchten, dass sie ihre Grafiken aktualisieren.

Sie müssen **Ebenen aktivieren** bevor Sie diesen Befehl nutzen können. Siehe **Abschnitt 25.1 [Einführung in die Ebenentechnik]**, Seite 405, für Details.

#### EINGABEN

`id`            Identifikator der zu aktualisierenden Ebene oder Ebenengruppe

## 25.31 RemoveLayer

#### BEZEICHNUNG

`RemoveLayer` – löscht eine Ebene (V1.5)

#### ÜBERSICHT

`RemoveLayer(id)`

#### BESCHREIBUNG

Dieser Befehl löscht die `id` angegebene Ebene oder Ebenengruppe. Das ist prinzipiell das gleiche wie der `Undo()` Befehl mit der Ausnahme, dass dieser Befehl EbenenIDs direkt akzeptiert. Mit `Undo()` müssten Sie einen Typ angeben, und vielleicht noch eine Rücknahmeebene. Jetzt können Sie einfach die Ebenen-ID angeben, was viel bequemer sein dürfte.

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Um mehr über Hollywoods Ebenen zu erfahren, lesen Sie bitte die **Einführung zu Ebenen**.

#### EINGABEN

`id`            Identifikator der zu löschenden Ebene oder Ebenengruppe

## 25.32 RemoveLayerFX

#### BEZEICHNUNG

`RemoveLayerFX` – löscht eine Ebene mit Effekt (V3.0)

#### ÜBERSICHT

`[handle] = RemoveLayerFX(id[, table])`

#### BESCHREIBUNG

Dieser Befehl ist eine erweiterte Version des `RemoveLayer()` Befehls. Er entfernt die in `id` angegebene Ebene oder Ebenengruppe und verwendet eine der vielen Übergangseffekte von Hollywood. Sie können auch die Geschwindigkeit für den Übergang und ein optionales Argument angeben.

Ab Hollywood 4.0 verwendet dieser Befehl eine neue Syntax mit nur einer einzigen Tabelle als optionales Argument. Die alte Syntax wird weiterhin aus Kompatibilitätsgründen unterstützt. Das optionale Tabellenargument kann den Übergangseffekt konfigurieren. Folgende Optionen sind möglich:

**Type**            Spezifiziert den gewünschten Effekt für den Übergang. Siehe **Abschnitt 30.11 [DisplayTransitionFX]**, Seite 630, für eine Liste aller unterstützten Übergangseffekte. (voreingestellt ist `#RANDEFFECT`)

**Speed**      Legt die gewünschte Geschwindigkeit für den Übergang fest. Je höher der Wert, desto schneller wird der Effekt gezeigt. (voreingestellt ist #NORMALSPEED)

**Parameter**      Einige Übergangseffekte akzeptieren einen zusätzlichen Parameter. Dieser kann hier angegeben werden. (voreingestellt ist #RANDOMPARAMETER)

**Async:**      Sie können dieses Feld verwenden, um ein asynchrones Zeichnungsobjekt für diesen Übergang zu erstellen. Wenn Sie hier **True** angeben, wird `HideLayerFX()` sofort verlassen und es wird ein Handler für das asynchrone Zeichnungsobjekt kreiert, den Sie dann mit dem Befehl `AsyncDrawFrame()` verwenden können. Siehe [Abschnitt 19.1 \[AsyncDrawFrame\]](#), [Seite 231](#), für weitere Informationen über asynchrone Zeichnungsobjekte.

**NoBorderFade**      Wenn die zu verbergende Ebene einen Rahmen hat, wird er nicht allmählich verblassen, sondern am Ende des Übergangseffekts in einem Rutsch entfernt. (V5.0)

**BorderFX:**      Wenn die Ebene, die entfernt werden soll, einen Rahmen hat, wendet Hollywood den Übergangseffekt nur dann auf den Rahmen an, wenn es sich um eine transparente Ebene mit Text- oder Pixelgrafiken handelt. Für nicht-transparente Ebenen und Vektorgrafikebenen wird stattdessen ein allgemeiner Überblendungseffekt verwendet, da es sonst aufgrund von Unterschieden in den Randalgorithmen zu visuellen Störungen zwischen dem vorletzten und letzten Effekteinzelbild kommen würde. Wenn Ihnen dieser Fehler egal ist und Sie Hollywood zwingen möchten, den Übergangseffekt immer auf den Rahmen anzuwenden, setzen Sie diesen Tag auf **True**. Um Hollywood zu zwingen, immer den generischen Überblendungsmodus zu verwenden, setzen Sie diesen Tag auf **False**. (V9.0)

Bevor Sie diesen Befehl verwenden können, müssen Sie zuerst die [Ebenen aktivieren](#). Siehe [Abschnitt 25.1 \[Ebenen Einführung\]](#), [Seite 405](#), für Details.

## EINGABEN

**id**      Identifikator der Ebene oder Ebenengruppe, die gelöscht wird

**table**      optional: Tabelle, um den Übergangseffekt zu definieren

## RÜCKGABEWERTE

**handle**      optional: Handler auf ein asynchrones Zeichnungsobjekt; wird nur zurückgegeben, wenn **Async** auf **True** gesetzt wurde (siehe oben)

## BEISPIEL

```
RemoveLayerFX(5, #CROSSFADE)                      ; alte Syntax
```

OR

```
RemoveLayerFX(5, {Type = #CROSSFADE})           ; neue Syntax
```



Dieser Code löscht die Ebene 5 mit einem schönen Effekt.

### 25.33 RemoveLayers

#### BEZEICHNUNG

RemoveLayers – entfernt alle Ebenen im aktuellen Hintergrundbild (V8.0)

#### ÜBERSICHT

RemoveLayers()

#### BESCHREIBUNG

Dieser Befehl entfernt alle Ebenen im aktuellen Hintergrundbild. Wenn dieser Befehl beendet ist, sind sie nicht mehr sichtbar. Wenn Sie alle Ebenen entfernen möchten, die Grafik jedoch weiterhin auf dem Bildschirm angezeigt werden soll, verwenden Sie stattdessen [FreeLayers\(\)](#).

Sie müssen [EnableLayers](#) aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 25.1 \[Ebenen Einführung\]](#), [Seite 405](#), für Details.

#### EINGABEN

keine

### 25.34 RenderLayer

#### BEZEICHNUNG

RenderLayer – konvertiert die Ebene in eine Pinselebene (V10.0)

#### ÜBERSICHT

RenderLayer(id)

#### BESCHREIBUNG

Dieser Befehl konvertiert die durch `id` angegebene Ebene in eine Pinselebene. Dies bedeutet in der Regel Qualitätseinbußen, da Pinsel gerastert sind und daher nicht ohne Qualitätsverlust skaliert oder transformiert werden können, weshalb dieser Befehl wahrscheinlich nicht viel nützt.

Sie müssen [EnableLayers](#) aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 25.1 \[Ebenen Einführung\]](#), [Seite 405](#), für Details.

#### EINGABEN

`id`            Identifikator der Ebene, welche konvertiert werden soll

### 25.35 ResumeLayer

#### BEZEICHNUNG

ResumeLayer – setzt eine angehaltene Video-Ebene fort (V6.0)

#### ÜBERSICHT

ResumeLayer(id)

**BESCHREIBUNG**

Dieser Befehl setzt die Wiedergabe der in `id` angehaltenen Video-Ebene fort. Sie können die Wiedergabe einer Video-Ebene mit dem Befehl `PauseLayer()` anhalten.

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Um mehr über Hollywoods Ebenen zu erfahren, lesen Sie bitte die **Einführung zu Ebenen**.

**EINGABEN**

`id`            Identifikator oder Name der Video-Ebene, welche fortgesetzt werden soll

## 25.36 RotateLayer

**BEZEICHNUNG**

RotateLayer – dreht eine Ebene (V4.0)

**ÜBERSICHT**

`RotateLayer(id, angle[, smooth])`

**BESCHREIBUNG**

Dieser Befehl dreht die in `id` angegebene Ebene oder Ebenengruppe um den in `angle` angegebenen Winkel (in Grad). Ein positiver Winkel dreht gegen den Uhrzeigersinn, ein negativer im Uhrzeigersinn. Darüber hinaus können Sie mit setzen von `smooth` auf `True` wählen, ob die Grafik unter Verwendung von Antialiasing gedreht wird. Das sieht zwar besser aus, dessen Berechnung dauert aber auch länger.

Wenn die festgelegte Ebene eine Vektorebene ist (zum Beispiel Kreis, Vieleck, Text mit TrueType oder ein Rechteck), wird Hollywood die Ebene ohne jeglichen Qualitätsverlust drehen können. Somit hat das Argument `smooth` bei einer Vektorebene keine Funktion. Wenn die Ebene hingegen ein Pixelbild enthält, wird das `smooth` Argument berücksichtigt.

Im Gegensatz zu den rotierenden Pinsel, welche mit `RotateBrush()` gedreht werden, behalten die Ebenen immer ihre ursprünglichen Daten. Somit gibt es keine Qualitätsverluste, wenn Sie eine Ebene um einige Grad und dann wieder in die ursprüngliche Position zurück drehen. Dies ist also durchaus möglich und erzeugt keine Qualitätsverluste mit `RotateLayer()`.

**EINGABEN**

`id`            Ebene oder Ebenengruppe, die gedreht wird

`angle`        Drehwinkel in Grad

`smooth`      optional: Drehung mit oder ohne Antialiasing (wird nur berücksichtigt, wenn die Ebene keine Vektorebene ist)

## 25.37 ScaleLayer

**BEZEICHNUNG**

ScaleLayer – skaliert eine Ebene (V4.0)

**ÜBERSICHT**

```
ScaleLayer(id, width, height[, smooth])
```

**BESCHREIBUNG**

Dieser Befehl skaliert die in `id` angegebene Ebene oder Ebenengruppe auf die Breite `width` und Höhe `height`. Optional können Sie die Grafik unter Verwendung von Antialiasing skalieren, indem Sie dem Argument `smooth True` zuweisen.

Wenn die festgelegte Ebene eine Vektorebene ist (zum Beispiel Kreis, Vieleck, Text mit `TrueType` oder ein Rechteck), wird Hollywood die Ebene ohne jeglichen Qualitätsverlust drehen können. Somit hat das Argument `smooth` bei einer Vektorebene keine Funktion. Wenn die Ebene jedoch Rastergrafiken verwendet, wird die normale rasterbasierte Drehung verwendet.

Im Gegensatz zu den skalierten Pinseln, welche mit `ScaleBrush()` in der Größe geändert wurden, behalten die Ebenen immer ihre ursprünglichen Daten. Somit gibt es keine Qualitätsverluste, wenn Sie eine Ebene zuerst auf (20,15) verkleinern und dann wieder auf (640,480) vergrößern. Dies ist also durchaus möglich und erzeugt keine Qualitätsverluste mit `ScaleLayer()`.

Den Argumenten `width` und `height` können auch eine prozentige Angabe in Form einer Zeichenkette übergeben werden, z.B. "50%".

Wenn Sie es vorziehen, anstelle von absoluten Pixelwerten lieber mit relativen Skalierungsfaktoren zu arbeiten, dann sollten Sie stattdessen die Tags `ScaleX` und `ScaleY` des Befehls `SetLayerStyle()` verwenden.

**EINGABEN**

<code>id</code>	ID der Ebene, die skaliert wird
<code>width</code>	gewünschte neue Breite für die Ebene
<code>height</code>	gewünschte neue Höhe der Ebene
<code>smooth</code>	optional: Skalierung mit oder ohne Antialiasing (wird nur berücksichtigt, wenn die Ebene keine Vektorebene ist)

**BEISPIEL**

```
ScaleLayer(1,640,480)
```

Skaliert die Ebene 1 auf die Größe von 640x480.

## 25.38 SeekLayer

**BEZEICHNUNG**

`SeekLayer` – springt an eine bestimmte Position in einer Video-Ebene (V6.0)

**ÜBERSICHT**

```
SeekLayer(id, pos)
```

**BESCHREIBUNG**

Sie können diesen Befehl verwenden, um die angegebene Position `pos` in der Video-Ebene `id` zu springen. Die Video-Ebene muss nicht gerade abgespielt werden. Falls die Video-Ebene läuft und Sie den Befehl `SeekLayer()` aufrufen, wird es sofort an die angegebene

Position springen. Die Position wird in Millisekunden angegeben. Wenn Sie also in die Position 03.24 springen möchten, würden Sie den Wert 204000 ( $3 * 60 * 1000 + 24 * 1000 = 204000$ ) in `pos` übergeben.

Bitte beachten Sie, dass das Springen an eine bestimmte Position im Video ein komplexer Vorgang ist. Es gibt Videoformate, die keine Positionstabellen haben, so dass Hollywood sich zunächst an die Position annähert und dann einige Feinabstimmungen und Einzelbildersuche durchführt, so dass die endgültige Position ein wenig abseits von der in `SeekLayer()` angegebenen Position liegen kann. Es kann auch vorkommen, dass Hollywood nicht direkt zu einem Einzelbild springt, so könnte es am linken Rand des Bildschirms (Screen) Artefakte vom vorherigen Einzelbild haben.

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Um mehr über Hollywoods Ebenen zu erfahren, lesen Sie bitte die **Einführung zu Ebenen**.

## EINGABEN

<code>id</code>	Identifikator oder Name der Video-Ebene
<code>pos</code>	neue Position im Video (in Millisekunden)

## 25.39 SelectLayer

### BEZEICHNUNG

SelectLayer – wählt eine Ebene als Ausgabeziel (V4.7)

### ÜBERSICHT

SelectLayer(id, [, mode, frame, combomode])

### BESCHREIBUNG

Dieser Befehl wählt die angegebene Ebene als aktuelles Ausgabegerät. Dies bedeutet, dass alle Grafikdaten, die nun Hollywood erstellt, an diese Ebene übertragen werden. Wenn `EndSelect()` aufgerufen wird, zeichnet die Ebene automatisch die Änderungen. Sie übergeben im ersten Argument die ID der Ebene. Wenn diese Ebene eine Animations-ebene ist, werden Sie auch das Einzelbild (Frame) angeben müssen.

Das optionale Argument `mode` ist auf `#SELMODE_NORMAL` voreingestellt, womit nur die Farbkanäle der Ebene geändert werden. Der Transparenzkanal der Ebene (ist entweder eine Maske oder ein Alphakanal) wird dann nicht verändert werden. Sie können dieses Verhalten ändern, indem Sie `#SELMODE_COMBO` im optionalen `mode` Argument übergeben. Wenn Sie diesen Modus verwenden, werden alle Hollywoodgrafiken nach dem Aufruf von `SelectLayer()` in die Farb- und Transparenzkanäle der Ebene zeichnen. Wenn die Ebene keinen Transparenzkanal hat, verhält sich `#SELMODE_COMBO` gleich wie `#SELMODE_NORMAL`.

Ab Hollywood 5.0 können Sie das optionale Argument `combomode` verwenden, um festzulegen, wie sich `#SELMODE_COMBO` verhalten soll. Wenn `combomode` auf 0 gesetzt ist, werden die Farb- und Transparenzinformation aller Pixel in dem Quellenbild in jedem Fall zu dem Zielbild kopiert, selbst wenn die Pixel unsichtbar sind. Dies ist auch voreingestellt. Wenn `combomode` hingegen auf 1 gesetzt ist, werden nur die sichtbaren Pixel in das Zielbild kopiert. Dies bedeutet, dass, wenn der Alphawert eines Pixels in dem Quellenbild 0 ist (unsichtbar), wird es nicht in das Zielbild kopiert werden.

Ab Hollywood 6.0 steht die neue `combomode 2` zur Verfügung. Wenn Sie 2 in `combomode` übergeben, wird Hollywood die Farbkanäle und Alphakanäle des Quellbildes mit der Farbe und Alpha der Zielbildkanäle mischen. Wenn Sie später das Zielbild zeichnen, wird es so aussehen, als ob ein Bild hintereinander auf das jeweils andere gezeichnet wurde. Bitte beachten Sie, dass das Argument `combomode` nur zusammen mit `#SELMODE_COMBO` unterstützt wird. Es hat keine Wirkung, wenn sie es mit den anderen Modi verwenden.

Ein alternativer Weg in die Transparenzkanäle einer Animation zu zeichnen, ist dies separat mit den Befehlen `SelectMask()` oder `SelectAlphaChannel()` zu erledigen. Diese beiden Befehle werden jedoch die Schreibdaten nur auf den Transparenzkanal anwenden. Sie werden den Farbkanal nicht ändern. Also, wenn Sie beide Kanäle (Farbe und Transparenz) ändern wollen, müssen Sie `SelectLayer()` mit `mode #SELMODE_COMBO` verwenden.

Wenn Sie mit dem ändern der Ebene fertig sind und möchten, dass Ihr Display wieder das Ausgabeziel wird, rufen Sie einfach den Befehl `EndSelect()` auf. Wenn Ihre Ebene sichtbar ist, wird Hollywood sie automatisch mit den Änderungen aktualisieren. Es ist wichtig zu berücksichtigen, dass die Änderungen nicht sichtbar sind, bevor Sie den Befehl `EndSelect()` aufgerufen haben.

Beachten Sie, dass Sie alle Befehle, die die Ebene ändern, nicht aufrufen, während sie als Ausgabeziel ausgewählt ist (z.B. `SetLayerStyle()` oder `RemoveLayer()`).

Es können nur Befehle von Hollywood verwendet werden, welche Grafiken direkt zeichnen (wenn `SelectLayer()` aktiv ist). Sie können keine animierten Befehle wie `MoveAnim()` oder `DisplayBrushFX()` aufrufen, während `SelectLayer()` aktiv ist.

Bitte beachten Sie, dass wenn Sie diesen Befehl für eine Vektorebene verwenden (zum Beispiel ein Vieleck oder Textebene), die Ebene automatisch gerastert wird. Das bedeutet, dass effektiv aus der Vektor- eine Pinsel-Ebene wird. Der Unterschied zwischen den beiden ist nur sichtbar, wenn es zur Umwandlung der Ebene kommt: Eine Vektorebene kann ohne Verluste in der Qualität frei umgewandelt werden. Gerasterte Pinselebenen auf der anderen Seite werden immer Verluste in der Qualität haben.

Siehe auch `EndSelect()`, `SelectAlphaChannel()`, `SelectBrush()`, `SelectMask()`, `SelectBGPic()` und `SelectAnim()`.

## EINGABEN

<code>id</code>	Ebene, welche als Ausgabeziel ausgewählt wird
<code>mode</code>	optional: Darstellungsmodus (siehe oben); dies kann entweder <code>#SELMODE_NORMAL</code> oder <code>#SELMODE_COMBO</code> sein; voreingestellt ist <code>#SELMODE_NORMAL</code>
<code>frame</code>	optional: nur wenn es eine Animationsebene ist; dieses Argument gibt das Einzelbild an (erstes Bild: <code>frame=1</code> )
<code>combomode</code>	optional: Wenn <code>#SELMODE_COMBO</code> aktiv ist, kann 0,1 oder 2 eingesetzt werden (siehe oben); voreingestellt ist 0

## BEISPIEL

```
SelectLayer(1)
SetFillStyle(#FILLCOLOR)
Box(0, 0, 320, 256, #RED)
```

```
EndSelect()
```

Der obige Code zeichnet ein 320x256 großes Rechteck auf Ebene 1.

## 25.40 SetLayerAnchor

### BEZEICHNUNG

SetLayerAnchor – ändert den Ankerpunkt der Ebene (V4.5)

### ÜBERSICHT

```
SetLayerAnchor(id, ax, ay)
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Ankerpunkt einer Ebene zu ändern. Der Ankerpunkt ist ein Punkt innerhalb der Ebene, die als Ursprung für alle Ebenentransformationen verwendet wird (Skalieren, Drehen) und auch die Position einer Ebene auf dem Ankerpunkt ist immer relativ. Manchmal wird der Ankerpunkt auch als "Hot Spot" einer Ebene bezeichnet.

Der Ankerpunkt kann jeder Punkt innerhalb der Ebene im Bereich von 0.0/0.0 (obere linke Ecke der Ebene) auf 1.0/1.0 (untere rechte Ecke der Ebene) sein. Das Zentrum der Ebene würde durch einen Ankerpunkt von 0.5/0.5 definiert werden.

Wenn Sie zum Beispiel eine Ebene haben wollen, die um seinen Mittelpunkt gedreht werden soll, dann müssen Sie den Ankerpunkt dieser Ebene auf 0.5/0.5 setzen. Wenn Sie sie um die linke obere Ecke drehen wollen, müssen Sie 0.0/0.0 als Ankerpunkt verwenden. Um die Ebene um die rechte untere Ecke zu drehen, benutzen Sie 1.0/1.0 als Ankerpunkt. Die übliche Einstellung ist um das Zentrum zu drehen, so dass Sie in der Regel den Ankerpunkt auf 0.5/0.5 einstellen.

Wenn Sie einen anderen Ankerpunkt als 0.0/0.0 verwenden, bedenken Sie, dass alle Angaben von Positionen nun relativ zum Ankerpunkt sein werden. Das heißt, dass eine Position von 0:0 auf der Ebene nicht zwangsläufig in der oberen linken Ecke des Displays ist. Zum Beispiel, wenn Sie eine Ebene mit einem Ankerpunkt von 1.0/1.0 haben und sie auf die Position 0:0 bewegen möchten (obere linke Ecke des Bildschirms) würde die Ebene so ziemlich unsichtbar machen, weil sein Ankerpunkt unten rechts eingestellt ist. Wenn Sie also eine Ebene mit einem unteren rechten Ankerpunkt auf Position 0:0 bewegen, bedeutet dies, dass die untere rechte Ecke der Ebene tatsächlich bei 0:0 erscheint. Somit ist nur ein einziges Pixel der Ebene sichtbar. Der Rest wird außerhalb des Bildschirmes sein.

Voreingestellt ist, dass alle Ebenen einen Ankerpunkt von 0.0/0.0 besitzen.

Ab Hollywood 10.0 kann dieser Befehl auch mit Ebenengruppen arbeiten, so dass Sie auch den Namen einer Ebenengruppe an diesen Befehl übergeben können.

### EINGABEN

<b>id</b>	Identifikator der Ebene
<b>ax</b>	Koordinate x des Ankerpunktes; muss zwischen 0.0 und 1.0 sein
<b>ay</b>	Koordinate y des Ankerpunktes; muss sich auch zwischen 0.0 und 1.0 befinden

**BEISPIEL**

```

EnableLayers
SetFillStyle(#FILLCOLOR)
Box(300, 200, 300, 200, #RED)
WaitLeftMouse
SetLayerAnchor(1, 0.5, 0.5)
WaitLeftMouse
SetLayerAnchor(1, 1.0, 1.0)
WaitLeftMouse

```

Der Code zeigt drei verschiedene Ankerpunkte: Erster bei 0.0/0.0, dann bei 0.5/0.5 und zuletzt bei 1.0/1.0. Sie können sehen, dass die Ebene mit jedem Aufruf von `SetLayerAnchor()` bewegt wird. Das ist, weil die Lage einer Ebene mit ihrem Verankerungspunkt immer relativ ist. Somit wird die Ebene bewegt, obwohl seine Position immer 300:200 sein wird.

**25.41 SetLayerBorder****BEZEICHNUNG**

`SetLayerBorder` – aktiviert/deaktiviert den Rahmen einer Ebene (V5.0)

**ÜBERSICHT**

```
SetLayerBorder(layer, enable[, color, size])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um einen Rahmen für die angegebene Ebene oder Ebenengruppe zu aktivieren oder deaktivieren, je nachdem, ob das Argument **enable** auf **True** oder **False** gesetzt ist. Im dritten Argument können Sie die Farbe des Rahmens als **ARGB-Farbwert** definieren. Das optionale Argument **size** kann verwendet werden, um die Dicke des Rahmens festzulegen. Dieser Wert setzt die gewünschte Rahmendicke auf jeder Seite der Ebene.

Sie können auch den Befehl `SetLayerStyle()` verwenden, um den Rahmen einer Ebene zu aktivieren/deaktivieren oder die Rahmenparameter zu ändern.

**EINGABEN**

<b>layer</b>	die zu verwendende Ebene oder Ebenengruppe
<b>enable</b>	aktiviert/deaktiviert den Rahmen der Ebene ( <b>True</b> bedeutet aktiviert, <b>False</b> deaktiviert)
<b>color</b>	optional: Farbe als <b>ARGB-Wert</b> , die vom Rahmen verwendet werden soll (Voreinstellung ist <b>#BLACK</b> )
<b>size</b>	optional: Dicke des Rahmens jeder Seite (Voreingestellt ist 2)

**BEISPIEL**

```

EnableLayers
SetFillStyle(#FILLCOLOR)
Box(#CENTER, #CENTER, 320, 240, #RED)
SetLayerBorder(1, True, #WHITE, 10)

```

Der obige Code zeichnet ein rotes Rechteck in die Mitte des Displays und fügt dann einen 10 Pixel dicken weißen Rahmen um ihn.

## 25.42 SetLayerDepth

### BEZEICHNUNG

SetLayerDepth – stellt die Farbtiefe der Ebenenpalette ein (V9.0)

### ÜBERSICHT

SetLayerDepth(id, depth[, t])

### BESCHREIBUNG

Dieser Befehl setzt die Farbtiefe der Palette in der durch `id` angegebenen Ebene auf die in `depth` angegebene Farbtiefe. Die Farbtiefe in `depth` muss ein Bit zwischen 1 (= 2 Farben) und 8 (= 256 Farben) haben. Siehe [Abschnitt 41.1 \[Palettenübersicht\]](#), [Seite 841](#), für Details. Beachten Sie, dass die Pixeldaten neu zugeordnet werden, wenn die angegebene Farbtiefe geringer ist als die der an die Palette angehängten Pixeldaten. Die folgenden Tags werden durch das optionale Tabellenargument `t` unterstützt:

**Frame:** Wenn es sich bei der Ebene um eine Animationsebene handelt, können Sie mit diesem Tag das Einzelbild angeben, dessen Farbtiefe festgelegt werden soll. Einzelbilder werden ab 1 gezählt. Standardmäßig wird das aktuelle Einzelbild der Animationsebene verwendet.

**Remap:** Wenn dieser Tag auf `False` gesetzt ist, werden außerhalb des Bereichs befindliche Stifte nicht auf vorhandene Stifte neu zugeordnet, sondern einfach auf den im Tag `ClipPen` (siehe unten) angegebenen Stift gesetzt, d.h. es findet keine Neuordnung statt. Beachten Sie, dass **Remap** nur beim Reduzieren von Farben wirksam ist. Wenn die neue Farbtiefe mehr Stifte hat als die alte, wird **Remap** nichts bewirken. (V10.0)

**ClipPen:** Dies wird nur verwendet, wenn der Tag **Remap** auf `False` gesetzt ist (siehe oben). In diesem Fall werden Stifte außerhalb des Bereichs nicht bestehenden Stiften neu zugeordnet, sondern einfach auf den im Tag `ClipPen` angegebenen Stift gesetzt, d.h. es findet keine Neuordnung statt. Beachten Sie, dass **ClipPen** nur beim Reduzieren von Farben wirksam ist. Wenn die neue Farbtiefe mehr Stifte hat als die alte, wird **ClipPen** nichts tun. (V10.0)

Sie müssen [Ebenen aktivieren](#) bevor Sie diesen Befehl nutzen können. Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), [Seite 405](#), für Details.

### EINGABEN

<code>id</code>	Identifikator der Ebene
<code>depth</code>	gewünschte neue Farbtiefe der Palette (im Bereich von 1 bis 8)
<code>t</code>	optional: Tabellenargument mit weiteren Optionen (siehe oben)

### BEISPIEL

```
SetLayerDepth(1, 8)
```

Der obige Code ändert die Farbtiefe der Palette von Ebene 1 auf 8 (= 256 Farben).



## 25.43 SetLayerFilter

### BEZEICHNUNG

SetLayerFilter – aktiviert/deaktiviert Filter für Ebenen (V5.0)

### ÜBERSICHT

SetLayerFilter(layer, table)

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um zu steuern, welche Filter und in welcher Reihenfolge auf die Ebene angewendet werden. Sie müssen diesem Befehl eine Tabelle übergeben, welche eine Anzahl von Untertabellen enthält, von denen jeder Parameter für einen einzelnen Ebenenfilter enthält. Für jede Untertabelle werden die folgenden Tags unterstützt:

<b>Name</b>	Enthält den Namen des Filters, das untergeordnete Tabellenelemente konfigurieren soll. Dieser Tag ist obligatorisch und muss immer in jeder Untertabelle angegeben werden, damit <b>SetLayerFilter()</b> den Filter kennt, welcher die Untertabellen adressiert. Bitte sehen Sie unten für eine Liste der unterstützten Filtertypen.
<b>Args</b>	Enthält eine Reihe von Argumenten für den Filter, welcher im Tag <b>Name</b> festgelegt wurde. Je nach Filter werden hier die entsprechenden Werte angegeben. Einige Filter wie <b>XFlip</b> oder <b>Gray</b> benötigen überhaupt keine Argumente. In diesen Fällen müssen Sie nicht die Tabelle <b>Args</b> übergeben. Bitte siehe unten, welche Argumente von den einzelnen Filter erforderlich sind. Beachten Sie auch, dass <b>SetLayerFilter()</b> Standardwerte für jeden Filter hat. So können Sie auch Argumente weglassen. In diesem Fall wird <b>SetLayerFilter()</b> die Voreinstellungen für den jeweiligen Filter verwenden.
<b>Disable</b>	Dieser Tag kann verwendet werden, um einen Filter zu aktivieren oder zu deaktivieren. Übergeben Sie hier <b>False</b> , um den Filter zu aktivieren oder <b>True</b> , um ihn zu deaktivieren. Dieser Tag ist optional. Wenn nichts angegeben ist, wird der Filter standardmäßig aktiviert.
<b>Priority</b>	Mit diesem Tag können Sie eine Prioritätsstufe für den aktuellen Filter angeben. Eine Prioritätsstufe ist einfach ein numerischer Wert, der dann durch <b>SetLayerFilter()</b> verwendet wird um herauszufinden, in welcher Reihenfolge die Filter angewendet werden sollen. Die Prioritätsstufe muss zwischen 0 (=niedrigste Priorität) und 255 (=höchste Priorität) liegen. Als Beispiel wird vor dem Farbtonfilter der Unschärfefilter angewendet werden, wenn Sie dem Filter <b>Blur</b> eine Priorität von 10 und dem Filter <b>Tint</b> eine Priorität von 9 zuweisen. Dieser Tag ist auf 0 voreingestellt, falls nichts angegeben wird.

Eine Liste der unterstützten Ebenenfilter folgt weiter unten. Bitte beachten Sie, dass die Argumente nicht dem Filter als Namentabellen-Tag, sondern der Reihe nach in dem **Args**-Feld übergeben werden. Das heißt, für den Filter **Modulate**, würden Sie die Helligkeitseinstellung im Feld-Element 0 setzen, im Feld-Element 1 die Sättigung und die Farbtonseinstellung im Element 2 die Reihenfolge, in der die einzelnen Argumente unten

aufgeführt werden, entspricht der Reihenfolge, wie sie in der **Args**-Tabelle zu erwarten sind. Hier ist die Liste:

<b>Blur</b>	Dieser Filter verändert das Aussehen der Ebene über einen Gauss'schen Weichzeichner. Das folgende Argument muss übergeben werden:  <b>Radius:</b> Legt den Effektradius fest. Je höher der hier angegebene Wert ist, desto länger dauert die Berechnung des Effekts.
<b>Charcoal</b>	Dieser Filter verändert das Aussehen der Ebene über einen Holzkohle-Effekt. Das folgende Argument muss übergeben werden:  <b>Radius:</b> Legt den Effektradius fest. Je höher der hier angegebene Wert ist, desto länger dauert die Berechnung des Effekts.
<b>Contrast</b>	Dieser Filter verbessert oder reduziert den Farbkontrast der Ebene. Die folgenden Argumente müssen übergeben werden:  <b>Inc:</b> Wenn das Argument <b>Inc</b> auf <b>True</b> gesetzt ist, wird der Kontrast erhöht. bei <b>False</b> hingegen verringert.  <b>Repeat:</b> Legt fest wie oft der Effekt wiederholt werden soll. Dies ist nützlich um den Effekt zu verstärken. Standardmäßig ist diese Option auf 1 gesetzt, was bedeutet, dass der Effekt nur einmal angewendet wird. Wenn Sie zwei Durchgänge haben möchten, geben Sie hier 2 an usw. Denken Sie daran, je größer die hier angegebene Zahl ist, desto länger dauert die Berechnung des Effekts.
<b>Edge</b>	Dieser Filter betont die Kanten einer Ebene. Das folgende Argument muss übergeben werden:  <b>Radius:</b> Spezifiziert den Effektradius. Je größer der Radius, desto länger braucht dieser Filter, den Kantenbetonungseffekt anzuwenden.
<b>Emboss</b>	Dieser Filter legt einen Reliefeffekt auf die Ebene (wird auch Stanzen genannt). Das folgende Argument muss übergeben werden:  <b>Radius:</b> Legt den Effektradius fest. Je höher der hier angegebene Wert ist, desto länger dauert die Berechnung des Effekts.
<b>Gamma</b>	Dieser Filter kann verwendet werden, um die Gammafarbkanäle der Ebene zu ändern. Die folgenden Argumente müssen übergeben werden:  <b>Red:</b> Gammakorrektur für den roten Kanal <b>Green:</b> Gammakorrektur für den grünen Kanal <b>Blue:</b> Gammakorrektur für den blauen Kanal  Für jeden Farbkanal müssen Sie einen Fließkommawert übergeben, der die gewünschte Gammakorrektur angibt. Ein Wert von 1.0 bedeutet keine Veränderung, ein Wert kleiner als 1.0 dunkelt den Kanal ab und ein Wert von mehr als 1.0 wird den Kanal aufhellen. Siehe <a href="#">Abschnitt 43.34 [GammaBrush]</a> , <a href="#">Seite 938</a> , für Details.

**Grayscale**

Dieser Filter wandelt die Ebene in Graustufen um. Für diesen Filter existieren keine Argumente.

**Invert**

Dieser Filter invertiert die Farben der Ebene. Auch für diesen Filter gibt es keine Argumente.

**Modulate**

Dieser Filter kann verwendet werden, um die Einstellungen der Helligkeit, Sättigung und des Farbtons einer Ebene zu ändern. Die folgenden Argumente müssen übergeben werden:

**Brightness:**

Gewünschte Helligkeit.

**Saturation:**

Gewünschte Sättigung.

**Hue:**

Gewünschter Farbton.

Für jede Einstellung übergeben Sie einen Fließkommawert, der die gewünschte Änderung beschreibt. Ein Wert von 1.0 bedeutet keine Veränderung, ein Wert kleiner als 1.0 verringert die Helligkeit/Sättigung/Farbton, während ein Wert von mehr als 1.0 sie erhöht. Siehe [Abschnitt 43.45 \[Pinsel modulieren\]](#), [Seite 948](#), für Details.

**Monochrome**

Dieser Filter kann verwendet werden, um die Ebene in Schwarz/Weiß abzubilden. Das folgende Argument muss übergeben werden:

**Dither:** Wenn **Dither** auf **True** gesetzt ist, wird Dithering angewendet, mit **False** nicht. Dithering ist langsamer, aber erzeugt besser aussehende Grafiken.

**OilPaint**

Dieser Filter wendet einen Ölfarbeneffekt auf die Ebene an. Das folgende Argument muss übergeben werden:

**Radius:** Legt den Effektradius fest. Je größer der Radius, desto länger braucht dieser Filter, den Ölfarbeneffekt anzuwenden.

**Pixelate**

Dieser Filter vergrößert/vergrößert die Pixel der Ebene. Das folgende Argument muss übergeben werden:

**CellSize:**

Legt die gewünschte Zellgröße für jedes einzelne Pixel fest. Jedes Pixel der Ebene wird auf diese Größe gezoomt. Die Vergrößerung startet in der oberen linken Ecke des Pinsels.

Siehe [Abschnitt 43.50 \[Pixelierter Pinsel\]](#), [Seite 953](#), für Details.

**Quantize**

Dieser Filter reduziert die Anzahl Farben der Ebene. Die folgenden Argumente müssen übergeben werden:

Standardmäßig wird dieser Befehl die Farben auf 256 reduzieren und das Dithering ist aktiviert. Sie können dies ändern, indem Sie die optionalen Parameter angeben. Beachten Sie, dass der **colors** Parameter einen Wert zwischen 1 und 256 haben muss.

**Colors:** Gewünschte Anzahl der Farben. Der Bereich geht von 1 bis 256.

**Dither:** Wenn **Dither** auf **True** gesetzt ist, wird Dithering angewendet, mit **False** nicht. Dithering ist langsamer, aber erzeugt besser aussehende Bilder.

Siehe [Abschnitt 43.52 \[Pinsel Quantisieren\]](#), Seite 954, für Details. (V6.0)

### **SepiaTone**

Dieser Filter wendet den Sepiatoneffekt auf die Ebene an. Das folgende Argument muss übergeben werden:

Der **SepiaTone**-Befehl versucht das Aussehen von alten Fotografien zu simulieren. Das zweite Argument **level** steuert die Intensität der Sepiatönung und kann sich von 0 bis 255 bewegen. Sie können auch einen Prozentsatz als Zeichenkette angeben ("50%"). Üblicherweise wird ein Wert von etwa 204 verwendet (= 80%), der am besten aussieht.

**Level:** Gewünschte Intensität des Sepiatons. Sie liegt im Bereich von 0 bis 255. Sie können auch einen Prozentsatz als Zeichenkette angeben ("50%"). Üblicherweise wird ein Wert von etwa 204 verwendet (= 80%), der am besten aussieht.

Siehe [Abschnitt 43.66 \[SepiaTone Pinsel\]](#), Seite 971, für Details.

**Sharpen** Hiermit wird das Aussehen der Ebene geschärft. Das folgende Argument muss übergeben werden:

**Radius:** Legt den Effektradius fest. Je größer der Radius, desto länger braucht dieser Filter, die Ebene zu schärfen.

**Solarize** Dieser Filter wendet einen Solarisationseffekt auf die Ebene an. Das folgende Argument muss übergeben werden:

**Level:** Gewünschte Intensität des Solarisationseffekt (Bereich geht von 0 bis 255).

Siehe [Abschnitt 43.75 \[Pinsel solarisieren\]](#), Seite 978, für Details.

**Swirl** Dieser Filter legt eine Wirbel-/Strudelwirkung um den Mittelpunkt der Ebene. Das folgende Argument muss übergeben werden:

**Degrees:** Legt die gewünschte Verwirbelung in Grad fest. Dies kann von 0 für keine Verwirbelung bis zu 360 für den dramatischsten Wirbeleffekt reichen.

**Tint** Dieser Filter färbt die Ebene ein. Die folgenden Argumente müssen übergeben werden:

**Color:** Legt die **RGB-Farbe** fürs einfärben fest.

**Ratio:** Stellt die Färbungsintensität ein. Der Wert muss im Bereich von 0 (= keine Färbung) bis 255 (= volle Färbung) liegen. Sie können auch eine Prozentangabe als Zeichenkette übergeben (50% entsprechen dem Wert 128).

**WaterRipple**

Dieser Filter wendet einen kreisförmigen Wasserwelleneffekt (Wasserkräuseln) auf die Ebene an. Die folgenden Argumente müssen übergeben werden:

**Wavelength:**

Gewünschte Wellenlänge für den Effekt.

**Amplitude:**

Gewünschte Wellenweite.

**Phase:** Gewünschte Wellen-Phase.

**CX:** X Mittelpunkt der Wasserwellen.

**CY:** Y Mittelpunkt der Wasserwellen.

Siehe [Abschnitt 43.81 \[Wasserregenpinsel\]](#), Seite 982, für Details.

**XFlip** Dieser Filter spiegelt die Ebene in der X-Achse. Für diesen Filter existieren keine Argumente.

**YFlip** Dieser Filter spiegelt die Ebene in der Y-Achse. Auch für diesen Filter existieren keine Argumente.

Um alle Ebenenfilter zu deaktivieren, können Sie im zweiten Argument den besonderen Wert 0 an Stelle einer Tabelle übergeben. **SetLayerFilter()** hebt dann alle Filter auf, die derzeit aktiv auf der angegebenen Ebene sind.

Bitte beachten Sie, dass **SetLayerFilter()** keine vorhandenen Filtereinstellungen zurücksetzen wird, wenn der Befehl aufgerufen wird. Stattdessen werden alle vorhandenen Filtereinstellungen behalten und die neuen Einstellungen werden mit den alten zusammengeführt werden. Wenn Sie also eine Ebene haben, auf die mehrere Filter wirken und Sie wollen nur die Konfiguration eines dieser Filter ändern, reicht es aus, nur eine untergeordnete Tabelle für diesen einzelnen Filter in **SetLayerFilter()** anzugeben. Es ist nicht notwendig, alle anderen Filter erneut an **SetLayerFilter()** zu übergeben.

Beachten Sie auch, dass die Ebenenfilter die CPU intensiv belasten können, vor allem dann, wenn für eine Ebene Übergangseffekte gebraucht werden, auf welche einige Filter angewendet wurden. In diesem Fall müssen die Filter für jedes neue Einzelbild des Übergangseffekts neu berechnet werden. Je nach Komplexität des Filters kann dies einige Zeit dauern.

Sie können auch den **SetLayerStyle()** Befehl verwenden, um die Konfiguration eines oder mehrerer Ebenenfilter zu ändern.

Ab Hollywood 10.0 kann dieser Befehl auch mit Ebenengruppen arbeiten, so dass Sie auch den Namen einer Ebenengruppe übergeben können.

**EINGABEN**

<b>layer</b>	Ebene oder Ebenengruppe, die verwendet werden soll
<b>table</b>	eine Tabelle, die eine oder mehrere Untertabellen für die Beschreibung von Filtern enthält, um sie auf eine Ebene anzuwenden oder zu entfernen; siehe oben für weitere Informationen; um alle Filter aus einer Ebene zu entfernen, übergeben Sie hier 0 anstelle einer Tabelle

**BEISPIEL**

```
table = {
  {Name = "YFlip"},
  {Name = "Modulate", Args = {1.0, 2.0, 1.0}, Priority = 10},
  {Name = "Swirl", Args = {128}, Priority = 9} }
SetLayerFilter(1, table)
```

Der obige Code erhöht die Sättigung der Ebene 1 um 200%, wirbelt die Ebene um 180 Grad und spiegelt sie dann auf der y-Achse.

```
SetLayerFilter(1, {{Name = "YFlip", Disable = True}})
```

Dieser Code entfernt den Filter YFlip von der Ebene 1, jedoch werden die beiden anderen Filter (Modulieren und Wirbel) behalten.

## 25.44 SetLayerName

**BEZEICHNUNG**

SetLayerName – weist einer Ebene einen Namen zu (V2.0)

**ÜBERSICHT**

```
SetLayerName(id, name$)
```

**BESCHREIBUNG**

Sie können diesen Befehl verwenden, um einen Namen für die in `id` angegebene Ebene zuzuweisen. Dies ist sehr nützlich, wenn Sie mehrere Ebenen benutzen, deren IDs ständig ändern (zum Beispiel, weil Sie häufig Ebenen einfügen und/oder entfernen). Wenn Sie Ihren Ebenen einen Namen geben, müssen Sie sich keine Sorgen mehr machen, an welcher Position die Ebene zur Zeit ist. Sie können leicht auf die Ebene zugreifen, indem Sie einfach ihren Namen verwenden. Alle Befehle, die Ebenen-IDs annehmen, werden auch die Namen akzeptieren.

Bitte beachten Sie, dass der Name für die Ebene innerhalb des aktuellen Ebenenspeichers des Hintergrundbildes eindeutig sein muss. Bei Ebenennamen werden Groß- und Kleinschreibung nicht unterschieden, das heißt "layer1" ist die gleiche Ebene wie "LAYER1".

Um herauszufinden, welche ID eine Ebene zur Zeit belegt, können Sie das Attribut `#ATTRLAYERID` mit dem Befehl `GetAttribute()` verwenden.

Wenn Sie einen Namen der neuesten Ebene zuordnen wollen, übergeben Sie einfach 0 und Hollywood wird automatisch die oberste Ebene verwenden. Um eine Ebenennamen zu entfernen, übergeben Sie eine leere Zeichenkette ("") in `name$`.

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), Seite 405, für Details.

**EINGABEN**

<code>id</code>	Identifikator der Ebene oder 0 für die neuste Ebene
<code>name\$</code>	gewünschter Name für die Ebene

**BEISPIEL**

```
EnableLayers()
```

```

SetFillStyle(#FILLCOLOR)

Box(0, 0, 100, 100, #RED)      ; erstellt Ebene 1
Box(50, 50, 100, 100, #GREEN) ; erstellt Ebene 2

SetLayerName(1, "redbox")      ; weist Ebene 1 den Namen zu
SetLayerName(2, "greenbox")    ; weist Ebene 2 den Namen zu

SwapLayers("redbox", "greenbox") ; tauscht sie! Nun ist "greenbox"
Ebene 1; und "redbox" Ebene 2!

ShowLayer("redbox", #RIGHT, #BOTTOM) ; verschiebt Ebene 2 nach
unten rechts
ShowLayer("greenbox", #LEFT, #TOP)    ; verschiebt Ebene 1 nach
oben links

Sie sehen, dass es viel einfacher ist für die Ebenen, wenn man anstelle von Ebenen-IDs
mit Zeichenkettennamen arbeitet, da IDs auf die Ebenenposition relativ sind.

```

## 25.45 SetLayerPalette

### BEZEICHNUNG

SetLayerPalette – wechselt die Palette der Ebene (V9.0)

### ÜBERSICHT

```
SetLayerPalette(id, palid[, t])
```

### BESCHREIBUNG

Dieser Befehl ersetzt die Palette der durch `id` angegebenen Ebene durch die in `palid` angegebene Palette. Mit dem optionalen Tabellenargument `t` können Sie einige weitere Optionen angeben. Die folgenden Tags werden derzeit vom optionalen Tabellenargument `t` unterstützt:

**Remap:** Wenn dies auf `True` gesetzt ist, werden die Pixel der Ebene neu zugeordnet, um den Farben der neuen Palette so genau wie möglich zu entsprechen. Standardmäßig erfolgt keine Neuordnung und die tatsächlichen Pixeldaten der Ebene bleiben unberührt. Wenn Sie eine Neuordnung wünschen, setzen Sie diesen Tag auf `True`. Beachten Sie jedoch, dass die Neuordnung aller Pixel natürlich viel länger dauert als das Festlegen einer neuen Palette ohne Neuordnung. Voreingestellt ist `False`.

**Dither:** Wenn Sie den Tag **Remap** (siehe oben) auf `True` gesetzt haben, können Sie mit dem Tag **Dither** angeben, ob Dithering verwendet werden soll oder nicht. Voreingestellt ist `True`, was bedeutet, dass Dithering verwendet wird.

**CopyCycleTable:**

Paletten können eine Tabelle mit Farbwechselinformationen enthalten. Wenn Sie diesen Tag auf `True` setzen, wird diese Wechseltabelle ebenfalls in den Pinsel kopiert. Der Standardwert ist `False`.

Sie müssen **Ebenen aktivieren** bevor Sie diesen Befehl nutzen können. Siehe **Abschnitt 25.1 [Einführung in die Ebenentechnik], Seite 405**, für Details.

#### EINGABEN

<code>id</code>	Identifikator der Ebene
<code>palid</code>	Identifikator der Palette, der in die Ebene kopiert werden soll
<code>t</code>	optional: Tabelle zum Festlegen weiterer Optionen (siehe oben)

## 25.46 SetLayerPen

#### BEZEICHNUNG

SetLayerPen – wechselt den Palettenstift der Ebene (V9.0)

#### ÜBERSICHT

SetLayerPen(`id`, `pen`, `color`)

#### BESCHREIBUNG

Dieser Befehl setzt die Farbe des in `pen` angegebenen Stifts auf die durch `color` angegebene Farbe in der Palette der Ebene, welche durch `id` angegeben wird.

Sie müssen **Ebenen aktivieren** bevor Sie diesen Befehl nutzen können. Siehe **Abschnitt 25.1 [Einführung in die Ebenentechnik], Seite 405**, für Details.

#### EINGABEN

<code>id</code>	Identifikator der Ebene
<code>pen</code>	Stift, den Sie ändern möchten (beginnend bei 0)
<code>color</code>	neue Farbe für den Stift, muss als <b>RGB-Farbe</b> angegeben werden

#### BEISPIEL

```
SetLayerPen(1, 0, #RED)
```

Der obige Code setzt den Stift 0 in der Palette der Ebene 1 auf Rot.

## 25.47 SetLayerShadow

#### BEZEICHNUNG

SetLayerShadow – aktiviert/deaktiviert den Schattenbereich für eine Ebene (V5.0)

#### ÜBERSICHT

SetLayerShadow(`layer`, `enable`[, `color`, `radius`, `size`, `dir`])

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen Schatteneffekt für die angegebene Ebene oder Ebenengruppe zu aktivieren oder deaktivieren, je nachdem, ob das Argument `enable` `True` oder `False` gesetzt wird. Im dritten Argument können Sie die Farbe des Schattens angeben. Dies wird in der Regel `#BLACK` sein, welches mit einem Transparenzwert kombiniert wird, weil einfach ein undurchsichtiges Schwarz als Schatten nicht



allzu gut aussieht. Sie können `ARGB()` verwenden, um einen Transparenzwert und eine Farbe in eine **ARGB-Farbe** zu kombinieren. Die optionalen Argumente `radius` und `size` können benutzt werden, um beim Schatten die Glätte und die Größe zu steuern. Normalerweise werden beide Werte auf etwa den gleichen Wert gesetzt. schließlich kann das `dir`-Argument verwendet werden, um die Richtung des Schattens zu steuern. Dieses Argument muss mit einer **Richtungskonstanten** von Hollywood eingestellt werden. Siehe **Abschnitt 29.13 [Richtungskonstanten]**, Seite 609, für Details.

Bitte beachten Sie, dass Schatten die CPU ziemlich in Anspruch nehmen kann, weil Hollywood sie neu berechnen muss, wenn der Inhalt der Ebene ändert. Normalerweise wird dies nicht oft passieren, obwohl es eine Ausnahme gibt: Wenn Sie einen Übergangseffekt auf eine Ebene ausführen, die über einen Schlagschatten verfügt. In diesem Fall muss Hollywood den Schlagschatten für jedes neue Einzelbild des Übergangseffekts neu berechnen. Auf langsameren Systemen kann dies möglicherweise die Show total vermiesen, so dass Sie Schlagschatten besser ausschalten, bevor ein Übergangseffekt auf eine Ebene ausgeführt wird.

Sie können auch den `SetLayerStyle()` Befehl verwenden, um den Schlagschatten einer Ebene zu aktivieren/deaktivieren oder die Parameter des Schattens ändern.

## EINGABEN

<code>layer</code>	Identifikator der Ebene oder Ebenengruppe
<code>enable</code>	aktiviert ( <code>True</code> ) oder deaktiviert ( <code>False</code> ) den Schlagschatten der Ebene
<code>color</code>	optional: Farbe für den Schlagschatten im <b>ARGB-Format</b> (voreingestellt ist <code>\$80000000</code> , also Schwarz mit 50% Transparenz)
<code>radius</code>	optional: Radius für die Glätte des Schattens (voreingestellt ist 5)
<code>size</code>	optional: Größe der Schattenverschiebung der Ebene (Standard ist 4)
<code>dir</code>	optional: Lichtrichtung des Schlagschattens (voreingestellt ist <code>#SHDWSOUTHEAST</code> )

## BEISPIEL

```
EnableLayers
SetFillStyle(#FILLCOLOR)
Box(#CENTER, #CENTER, 320, 240, #RED)
SetLayerShadow(1, True)
```

Der obige Code malt ein rotes Rechteck in die Mitte des Displays und zeichnet dann einen Schlagschatten.

## 25.48 SetLayerStyle

### BEZEICHNUNG

`SetLayerStyle` – ändert den Stil einer oder mehreren Ebenen (V4.0)

### ÜBERSICHT

```
SetLayerStyle(id1, style1, ...)
```

## BESCHREIBUNG

Dieser Befehl kann verwendet werden, um nahezu alle Attribute einer, mehreren bestehenden Ebenen oder Ebenengruppen mit einem einzigen Aufruf zu ändern. Es ist ein sehr mächtiger Befehl, der komplexe Animationsmechanismen in einer sehr einfachen und geradlinigen Art und Weise realisiert. Für jede Ebene oder Ebenengruppe, deren Stil Sie ändern möchten, müssen Sie immer seine Ebenen(gruppe)-ID gefolgt von einer Tabelle mit den Attributen übergeben, die Sie ändern möchten. Sie können diese Muster so oft wiederholen, wie Sie es brauchen.

Die Konfiguration der Stiltabelle hängt von der Art der Ebene ab. Allerdings können einige Stilelemente mit allen Ebenentypen verwendet werden. Diese allgemeinen Ebenenstile werden zuerst abgedeckt. Die von Ebenen abhängigen und spezifischen Ebenenstile werden weiter unten behandelt.

Die folgenden Stilelemente sind generisch und können somit mit jeder Ebene oder Ebenengruppe benutzt werden:

**X,Y**            Legt die Position der Ebene fest. Wenn nichts angegeben wird, behält die Ebene ihre aktuelle Position.

### Width,Height

Mit diesen beiden Elementen kann die Ebene auf eine neue Dimension skaliert werden. Dies kann entweder ein numerischer Wert für eine neue Pixelgröße oder einer Zeichenkette mit einer prozentuellen Angabe (zum Beispiel "50%") sein. Siehe [Abschnitt 25.37 \[ScaleLayer\]](#), [Seite 434](#), für Details.

**Rotate**        Dieses Stilelement kann verwendet werden, um die Ebenenrotation zu steuern. Sie müssen einen Wert in Grad angeben. Ein positiver Wert bedeutet Drehung gegen den Uhrzeigersinn, negative Werte im Uhrzeigersinn. Siehe [Abschnitt 25.36 \[RotateLayer\]](#), [Seite 434](#), für Details..

### SmoothScale

Hier können Sie angeben, ob Hollywood bei der Skalierung und Rotation der Ebene Antialiasing verwenden soll oder nicht. Dies gilt nur für Ebenen, welche nicht im Vektorformat sind und natürlich macht es nur Sinn, wenn die Ebene skaliert oder gedreht wird.

### Transparency

Verwenden Sie dieses Stilelement, um die Transparenz einer Ebene zu verändern. Dies kann entweder ein Wert im Bereich von 0 (= keine Transparenz) bis 255 (= volle Transparenz) oder sie können auch eine Prozentangabe als Zeichenkette übergeben (50% entspricht halbtransparent). Siehe [Abschnitt 25.50 \[SetLayerTransparency\]](#), [Seite 464](#), für Details.

### Tint

Verwenden Sie dieses Stilelement, um den Farbton einer Ebene zu verändern. Dies kann entweder ein Wert im Bereich von 0 (= keine Tönung) bis 255 (= Vollfarbtönung) oder Sie können auch hier eine Prozentangabe als Zeichenfolge übergeben (z.B. "50%" bedeutet Halbtönung). Wenn dieses Element nicht auf Null gesetzt ist, wird die Ebene mit der Farbe in `TintColor` getönt. Siehe [Abschnitt 25.49 \[SetLayerTint\]](#), [Seite 463](#), für Details.

- TintColor** Definiert die Farbe für die Tönung. Kann nur in Verbindung mit dem Stilelement **Tint** verwendet werden.
- Hidden** Sie können dieses Stilelement verwenden, um die angegebene Ebene ein- oder auszublenden. Übergeben Sie **True**, um sie zu verstecken oder **False**, um sie zu zeigen. Siehe [Abschnitt 25.17 \[HideLayer\]](#), [Seite 420](#), für Details.
- Type** Mit diesem Stilelement können Sie den Ebenentyp ändern. Bitte beachten Sie, dass wenn Sie den Typ einer Ebene ändern, werden Sie höchstwahrscheinlich weitere Informationen **SetLayerStyle()** zur Verfügung stellen müssen. Zum Beispiel wenn Sie eine **#ELLIPSE** Ebene in eine **#BRUSH** Ebene umwandeln, ist es zwingend erforderlich, dass Sie auch das Stilelement **ID** angeben, um Hollywood mitzuteilen, welcher Pinsel die Ellipse-Ebene ersetzen soll. Hollywood wird versuchen, allgemeine Stilelemente wie Farbe, Position, Größe vom vorherigen Ebenentyp zu erben, aber für bestimmte Typkonvertierungen müssen Sie zusätzliche Elemente angeben. Falls Sie **Type** angegeben, werden nur Stilelemente, die vom neuen Ebenentyp unterstützt werden, übernommen werden. (V4.5)
- ClipRegion** Verwenden Sie diesen Stil, um die Clip-Region dieser Ebene zu ändern. Jede Ebene kann eine private Clip-Region haben. Siehe [Abschnitt 28.31 \[SetClipRegion\]](#), [Seite 592](#), für Details.
- ScaleX, ScaleY** Dies ist eine alternative Möglichkeit, die Ebene zu skalieren. Sie übergeben hier einen Fließkommawert, der einen Skalierungsfaktor angibt. Zum Beispiel bedeutet 0.5 halb so groß, 2.0 hingegen bedeutet doppelt so groß. Dies ist besonders praktisch, wenn man die Proportionen der Ebene beibehalten möchte. Wenn Sie also den gleichen Faktor für **ScaleX** und **ScaleY** verwenden, werden die Proportionen der Ebene intakt bleiben. Bitte beachten Sie, dass **ScaleX/ScaleY** und **Width/Height** sich gegenseitig ausschließen. Sie müssen nicht beide Elementstilgruppen mischen. Entweder verwenden Sie **ScaleX/ScaleY** oder **Width/Height**. (V4.5)
- Transform** Mit diesem Tag können Sie eine 2x2-Transformations-Matrix für diese Ebene verwenden. Transformationsmatrizen sind nützlich, wenn Sie Skalierung und Rotation gleichzeitig anwenden möchten, oder wenn Sie eine Ebene spiegeln wollen. Sie übergeben eine Tabelle in **Transform**. Die Tabelle muss die vier Bestandteile einer 2x2-Transformations-Matrix in der folgenden Reihenfolge enthalten: **sx, rx, ry, sy**. Siehe [Abschnitt 43.78 \[TransformBrush\]](#), [Seite 980](#), für mehr Informationen über Transformationsmatrizen. Bitte beachten Sie, dass das Stilelement **Transform** nicht mit folgenden Tags gemischt werden darf, da sie sich gegenseitig ausschließen: **Width/Height/ScaleX/ScaleY/Rotate**. (V4.5)
- AnchorX, AnchorY** Sie können diese beiden Tags verwenden, um den Ankerpunkt dieser Ebene zu ändern. Der Ankerpunkt kann jeder Punkt zwischen 0.0/0.0 (oben links

der Ebene) und 1.0/1.0 (Ecke unten rechts der Ebene) sein. Das Zentrum der Ebene würde durch einen Ankerpunkt von 0.5/0.5 definiert werden. Alle Transformationen (Skalierung, Drehung, usw.) werden relativ auf den Ankerpunkt angewendet werden. Auch ist die Lage einer Ebene immer relativ zu seinem Verankerungspunkt. Siehe [Abschnitt 25.40 \[SetLayerAnchor\]](#), [Seite 438](#), für Details. (V4.5)

#### NoClipTransform

Dieser Tag kann verwendet werden, um die automatische Transformation der Clip-Region zu deaktivieren. Wenn Sie eine Ebene transformieren, wird standardmäßig auch dessen Clip-Region im selben Verhältnis transformiert werden. Wenn Sie dies nicht wollen, setzen Sie `NoClipTransform` auf `True`.

#### TextureX, TextureY

Diese Tags funktionieren nur mit den Grafikgrundelementen, die den Stil `#FILLTEXTURE` verwenden. Wenn das der Fall ist, können Sie diese Tags verwenden, um den Startversatz in der Pinseltextur zu steuern. Siehe [Abschnitt 29.14 \[SetFillStyle\]](#), [Seite 610](#), für Details. (V4.6)

#### Z

Dieser Tag kann verwendet werden, um die z-Position dieser Ebene zu ändern. Die z-Position einer Ebene ist seine Position in der Hierarchie von Ebenen. Die erste (d.h. hinterste Ebene) hat eine z-Position von 1, die letzte (d.h. vorderste) Ebene eine z-Position der zu der Anzahl der vorhandenen Ebenen. Sie müssen die neue gewünschte z-Position für die angegebene Ebene in diesem Tag übergeben. Die Ebene wird dann genau diese z-Position einnehmen und die bestehenden Ebenen, welche an oder nach dieser z-Position sind, werden nach vorne verschoben. Um eine Ebene an vorderster Stelle zu bewegen (das heißt höchste z-Position), können Sie den besonderen Wert 0 angeben. Um eine Ebene ganz nach hinten zu verschieben, verwenden Sie den Wert 1. Siehe [Abschnitt 25.53 \[SetLayerZPos\]](#), [Seite 466](#), für Details. (V4.7)

#### TranslateX, TranslateY

Diese beiden Tags können verwendet werden, um eine Ebene mit einem angegebenen Delta-X und Delta-Y zu verschieben. Das bedeutet, dass eine Ebene durch den angegebenen Deltaversatz relativ zu seiner aktuellen Position bewegt wird. Somit würde ein Versatz von (1,1) die Ebene ein Pixel nach rechts und ein Pixel nach unten bewegen. Siehe [Abschnitt 25.59 \[TranslateLayer\]](#), [Seite 470](#), für Details. Bitte beachten Sie: Diese beiden Tags und `X/Y` schließen sich gegenseitig aus. Sie können sie nicht zusammen verwenden. (V4.7)

#### Shadow

Dieser Tag kann verwendet werden, um den Schlagschatten einer Ebene ein- und auszuschalten. Wenn dieser Tag auf `True` gesetzt ist, wird der Schlagschatten eingeschaltet, sonst wird er ausgeschaltet. Wenn Sie `Shadow` auf `True` setzen, können Sie das Aussehen des Schattens mit den Tags `ShadowDir`, `ShadowSize`, `ShadowColor` und `ShadowRadius` konfigurieren, Siehe unten für weitere Informationen. Siehe [Abschnitt 25.47 \[SetLayerShadow\]](#), [Seite 448](#), für Details. (V5.0)

- ShadowDir**     Ändert die Richtung des Schattens für diese Ebene. Hier muss eine der **Richtungskonstanten** von Hollywood übergeben werden. Dieser Tag wird nur dann ausgeführt, wenn die Schatten mit **Shadow** eingeschaltet wurden. (V5.0)
- ShadowColor**     Ändert die Farbe des Schattens für diese Ebene. Dies muss ein **ARGB-Wert** sein, die eine Transparenz enthält. Dieser Tag wird nur dann ausgeführt, wenn die Schatten mit **Shadow** eingeschaltet wurden. (V5.0)
- ShadowPen**     Legt den Stift fest, der zum Zeichnen des Schattens der Ebene verwendet werden soll, wenn die Palette **#PALETTEMODE\_PEN** ist. Siehe **Abschnitt 41.35 [SetShadowPen]**, **Seite 872**, für Details. (V9.0)
- ShadowSize**     Ändert die Größe des Schattens für diese Ebene. Dieser Tag wird nur dann ausgeführt, wenn die Schatten mit **Shadow** eingeschaltet wurden. (V5.0)
- ShadowRadius**     Mit diesem Tag kann der Schattenradius für diese Ebene geändert werden. Dieser Tag wird nur dann ausgeführt, wenn die Schatten mit **Shadow** eingeschaltet wurden. (V5.0)
- Border**     Dieser Tag kann verwendet werden, um den Rahmen einer Ebene ein- und auszuschalten. Wenn Sie diesen Tag auf **True** setzen, wird der Rahmen eingeschaltet, sonst wird er ausgeschaltet. Wenn Sie Rahmen eingeschaltet haben, können Sie das Aussehen des Rahmens mit den Tags **BorderSize** und **Border** konfigurieren. Siehe unten für weitere Informationen. Siehe **Abschnitt 25.41 [SetLayerBorder]**, **Seite 439**, für Details. (V5.0)
- BorderColor**     Hiermit können Sie die Farbe des Rahmens für diese Ebene ändern. Dies muss ein **ARGB-Wert** sein, der eine Transparenz enthält. Dieser Tag wird nur dann ausgeführt, wenn **Border** derzeit eingeschaltet ist. (V5.0)
- BorderPen**     Legt den Stift fest, der zum Zeichnen des Ebenenrahmens verwendet wird, wenn der Palettenmodus **#PALETTEMODE\_PEN** ist. Siehe **Abschnitt 41.22 [SetBorderPen]**, **Seite 860**, für Details. (V9.0)
- BorderSize**     Gibt die Dicke des Rahmens dieser Ebene an. Dieser Tag wird nur dann ausgeführt, wenn **Border** derzeit eingeschaltet ist. (V5.0)
- Filters**     Dieser Tag kann verwendet werden, um einen Filter auf die Ebene anzuwenden, einen Filter zu entfernen oder die Parameter von bereits angelegten Filter zu ändern. Sie übergeben diesem Tag eine Tabelle, die die gewünschte Konfiguration der einzelnen Filter beschreibt. Siehe **Abschnitt 25.43 [SetLayerFilter]**, **Seite 441**, für Details. (V5.0)

**ClearFilters**

Dieser Tag kann verwendet werden, um alle Filter aus einer Ebene zu entfernen. Geben Sie hier einfach **True** ein und **SetLayerStyle()** wird alle Filter entfernen, die bei dieser Ebene derzeit aktiv sind.

**PaletteMode**

Gibt den Palettenmodus an, der für die Ebene verwendet werden soll, wenn sich Hollywood im Palettenmodus befindet. Dies muss einer der Palettenmodi sein, die von **SetPaletteMode()** unterstützt werden. Siehe [Abschnitt 41.31 \[SetPaletteMode\]](#), [Seite 868](#), für Details. (V9.0)

**DitherMode**

Gibt den Dither-Modus für die Ebene an, wenn Hollywood im Palettenmodus ist. Dies muss einer der Dither-Modi sein, die von **SetDitherMode()** unterstützt werden. Siehe [Abschnitt 41.26 \[SetDitherMode\]](#), [Seite 863](#), für Details. (V9.0)

**IgnoreAnchor**

Wenn dieser Tag auf **True** gesetzt ist, ignoriert Hollywood den Ankerpunkt der Ebene beim Ändern ihrer Position oder ihres Stils. Das bedeutet, dass der Ankerpunkt als 0/0 behandelt wird. Dies kann z.B. nützlich sein, wenn Sie den Text einer Ebene ändern möchten, deren Ankerpunkt nicht 0/0 ist. In solchen Fällen würde das Ändern des Textes auch die Ebene neu positionieren. Dies kann verhindert werden, indem **IgnoreAnchor** auf **True** gesetzt wird. **IgnoreAnchor** kann auch nützlich sein, wenn Sie Ebenen, deren Ankerpunkt nicht 0/0 ist, relativ zu ihrer oberen linken Ecke positionieren möchten. (V9.1)

**Refresh** Erzwingt eine Aktualisierung der Ebene. Dies ist nur für zusammengeführte Ebenen nützlich, die von **MergeLayers()** erstellt wurden, da sie nicht automatisch aktualisiert werden, wenn die Grafiken einer oder mehrerer ihrer Quellevbenen geändert werden. Siehe [Abschnitt 25.24 \[MergeLayers\]](#), [Seite 425](#), für Details. (V10.0)

Die folgenden Stilelemente sind abhängig von einem bestimmten Ebenentyp:

Ebenen vom Typ **#ANIM** können die folgenden Elemente verwenden:

- ID** Dieses Tabellenelement kann verwendet werden, um eine neue Animation der angegebenen Ebene zuzuweisen. Die alte Animation wird dann durch die in ID angegebene Animation ersetzt werden.
- Frame** Sie können dieses Stilelement verwenden, um ein bestimmtes Einzelbild der Animation anzuzeigen. Die Einzelbilder werden von 1 an gezählt. Sie können den speziellen Wert 0 übergeben, um das nächste Bild in der Animation anzuzeigen. Siehe [Abschnitt 25.27 \[NextFrame\]](#), [Seite 429](#), für Details.

Die Ebenentypen **#BRUSH**, **#BRUSHPART** und **#BGPICPART** erkennen folgende Elemente:

- ID** Dieses Tabellenelement kann verwendet werden, um einen neuen Pinsel oder ein neues Hintergrundbild der angegebenen Ebene zuzuweisen. Der alte Pin-

sel/das alte Hintergrundbild wird dann durch die in `ID` angegebenen Pinsel/angegebenes Hintergrundbild ersetzt werden. Dies ist zum Beispiel für eine Diashow nützlich, wenn alle `n` Sekunden ein neues Bild angezeigt werden soll.

#### **PartX, PartY, PartWidth, PartHeight**

Diese vier Elemente können Sie verwenden, um den sichtbaren Teil des Pinsels oder Hintergrundbildes zu konfigurieren. In **PartX** und **PartY** geben Sie die x- und y-Koordinaten des Pinsels/des Hintergrundbildes an und in **PartWidth** sowie **PartHeight** die Größe, die sichtbar sein soll. Dies ist nützlich, um nur einen Teil eines Pinsels oder Hintergrundbildes anzuzeigen. Für weitere Informationen werfen Sie einen Blick auf die Befehle **DisplayBrushPart()** und **DisplayBGPicPart()**. Bitte beachten Sie, dass diese Elemente nicht nur bei Ebenentypen **#BRUSHPART** und **#BGPICPART** verwendet werden können, sondern auch mit dem Ebenentyp **#BRUSH** funktionieren. Wenn Sie eine der **PartXXX** Elemente auf eine Ebene vom Typ **#BRUSH** verwenden, wird diese Ebene automatisch in eine vom Typ **#BRUSHPART** umgewandelt werden.

Folgende Elemente sind typisch für die Ebenentypen **#ARC**, **#BOX**, **#CIRCLE**, **#ELLIPSE**, **#POLYGON** und **#VECTORPATH**:

**Color**      Ändert die Farbe der Ebene. Übergeben wird ein **ARGB-Wert**.

**DrawPen:**   Wenn der Palettenmodus **#PALETTEMODE\_PEN** ist, gibt **DrawPen** den Stift an, der zum Zeichnen dieser Ebene verwendet werden soll. Siehe **Abschnitt 41.27 [SetDrawPen]**, **Seite 864**, für Details. (V9.0)

#### **FormStyle**

Auf diese Weise können Sie den Zeichnungsstil der Ebene verändern. Sie können hier eine oder mehrere Arten wählen. Wenn Sie mehrere Zeichnungsstile übergeben, müssen Sie den Bit-Operator (**|**) verwenden. Siehe **Abschnitt 29.15 [SetFormStyle]**, **Seite 611**, für mögliche Kombinationen. Wenn Sie einen Zeichnungsstil aus einer Ebene entfernen möchten, müssen Sie den Tag **FormStyleClear** verwenden. Hinweis: Ab Hollywood 5.0 können Sie **FormStyle** noch für das Setzen von **#ANTIALIAS** gebrauchen, da Schatten und Randeinstellungen einfacher mit den getrennten Tags zu kontrollieren sind (siehe oben).

#### **FormStyleClear**

Alle Formstile, die Sie in diesem Element setzen, werden aus der Ebene entfernt werden. Mehrere Formstile werden mit dem Bit-Operator (**|**) getrennt. Das ist das Pendant zum "Formstyle" Element. Hinweis: Ab Hollywood 5.0 können Sie **FormStyleClear** noch für das Entfernen von **#ANTIALIAS** gebrauchen, da Schatten und Randeinstellungen einfacher mit den getrennten Tags zu kontrollieren sind (siehe oben).



**FillStyle**

Sie können dieses Stilelement verwenden, um die Hintergrundfüllung für diese Ebene zu ändern. Siehe [Abschnitt 29.14 \[SetFillStyle\]](#), [Seite 610](#), für Details.

**GradientStyle**

Damit ändern Sie den Stil des Farbverlaufs, wenn für die Hintergrundfüllung `#FILLGRADIENT` eingestellt ist. Dies kann `#LINEAR`, `#RADIAL` oder `#CONICAL` sein.

**GradientAngle**

Ändert den Winkel der Hintergrundfüllung, wenn der Stil `#FILLGRADIENT` eingestellt ist. Der Winkel wird in Grad ausgedrückt. Nur für Füllungsart `#LINEAR` und `#CONICAL`.

**GradientStartColor, GradientEndColor**

Verwenden Sie diese beiden Farben, um den Farbverlauf der Hintergrundfüllung zu konfigurieren, wenn als Stil `#FILLGRADIENT` gesetzt wurde.

**GradientCenterX, GradientCenterY**

Legt den Mittelpunkt für den Farbverlauf des Typs `#RADIAL` oder `#CONICAL` an. Muss ein Fließkommawert zwischen 0.0 und 1.0 liegen. Siehe [Abschnitt 30.6 \[CreateGradientBGPic\]](#), [Seite 624](#), für Details. (V5.0)

**GradientBalance**

Dieser Tag steuert den Gleichgewichtspunkt für Farbverläufe des Typs `#CONICAL`. Muss ein Fließkommawert zwischen 0.0 und 1.0 sein. Siehe [Abschnitt 30.6 \[CreateGradientBGPic\]](#), [Seite 624](#), für Details. (V5.0)

**GradientBorder**

Dieser Tag steuert die Rahmengröße für den Farbverlauf des Typs `#RADIAL`. Muss ein Fließkommawert zwischen 0.0 und 1.0 sein. Siehe [Abschnitt 30.6 \[CreateGradientBGPic\]](#), [Seite 624](#), für Details. (V5.0)

**GradientColors**

Dieser Tag kann verwendet werden, um einen Farbverlauf zwischen mehr als zwei Farben zu erzeugen. Dies muss mit einer Tabelle gesetzt werden, die Sequenzen von abwechselnden Farben und Stoppwerte enthält. Siehe [Abschnitt 30.6 \[CreateGradientBGPic\]](#), [Seite 624](#), für Details. Wenn dieser Tag gesetzt ist, werden die Tags `GradientStartColor` und `GradientEndColor` ignoriert werden. (V5.0)

**OutlineThickness**

Wenn der Füllstil auf `#FILLNONE` festgelegt ist, ändert dieser Wert die Dicke der Kontur. Siehe [Abschnitt 29.14 \[SetFillStyle\]](#), [Seite 610](#), für Details.

**TextureBrush**

Wenn der Füllstil auf `#FILLTEXTURE` eingestellt ist, können Sie die aktuell verwendete Textur mit diesem Stilelement ändern. Übergeben Sie einfach die ID eines Pinsels in diesem Stile, um auf die neue Textur zu verwenden.



Zusätzlich zu den oben genannten Tags akzeptiert der Ebenentyp **#ARC** noch folgende Stilelemente:

**RadiusA, RadiusB**

Mit diesen beiden Werten können Sie die x und y Radien der Teilellipse ändern.

**StartAngle, EndAngle**

Mit diesen beiden Werten können Sie den Start- und Endwinkel einer Teilellipse ändern. Siehe [Abschnitt 29.1 \[Arc\]](#), [Seite 597](#), für Details.

**Clockwise**

Mit diesem Tag können Sie die Richtung einer Teilellipse ändern. Siehe [Abschnitt 29.1 \[Arc\]](#), [Seite 597](#), für Details. (V4.5)

Ebenen vom Typ **#BOX** akzeptieren noch folgende zusätzlichen Stilelemente:

**SizeX, SizeY**

Sie können diese beiden Werte verwenden, um die Abmessungen des Rechtecks zu ändern. **SizeX** ist die Breite des Rechtecks und **SizeY** ist somit die Höhe.

**RoundLevel**

Ändert die Rundungen der vier Ecken. Ein Wert von 0 bedeutet, dass die Ecken keine Rundung haben, ein Wert von 100 bedeutet vollständig runde Ecken. Siehe [Abschnitt 29.2 \[Box\]](#), [Seite 598](#), für Details.

**CornerA, CornerB, CornerC, CornerD**

Mit diesen vier Tags können Sie die Feinabstimmung der Eckenabrundung des Rechtecks vornehmen. Sie können eine Rundungsstufe (0 bis 100) für jede Ecke des Rechtecks angeben, so dass Sie ein Rechteck erstellen, in dem nicht alle Ecken abgerundet sind, oder wo die verschiedenen Ecken unterschiedliche Rundungen verwenden. Diese Elemente setzen alle Einstellungen des Tags **RoundLevel** außer Kraft. (V5.0)

Ebenen vom Typ **#CIRCLE** verwenden noch folgende zusätzlichen Stilelemente:

**Radius**     Ändert den Radius eines Kreises. Siehe [Abschnitt 29.3 \[Circle\]](#), [Seite 600](#), für Details.

Ebenen vom Typ **#ELLIPSE** benutzen noch diese zusätzlichen Stilelemente:

**RadiusA, RadiusB**

Diese beiden Werte ändern die x und y Radien einer Ellipse. Siehe [Abschnitt 29.5 \[Ellipse\]](#), [Seite 601](#), für Details.

Ebenen vom Typ **#LINE** können noch folgende Stilelemente haben:

**Thickness**

Ändert die Dicke der Linie. Siehe [Abschnitt 29.9 \[Line\]](#), [Seite 605](#), für Details.

**X1,Y1,X2,Y2**

Verwenden Sie diese Tags, um die Linienausrichtung zu ändern. Bitte beachten Sie, dass diese Tags und die generischen **X** und **Y** Tags sich gegenseitig ausschließen. Wenn Sie diese vier Tags verwenden, müssen Sie nicht die generischen Tags benutzen und umgekehrt. (V4.6)

**Arrowhead**

Mit diesem Tag können Sie die Linie in einen Pfeil umwandeln. Er kann auf eines der folgenden Tags gesetzt werden:

**#ARROWHEAD\_NONE**

Keine Pfeilspitze. Dies ist die Voreinstellung.

**#ARROWHEAD\_SINGLE**

Fügt eine Pfeilspitze ans Ende der Linie hinzu.

**#ARROWHEAD\_DOUBLE**

Fügt eine Pfeilspitze am Anfang und am Ende der Linie hinzu.

(V9.1)

Ebenen vom Typ **#POLYGON** akzeptieren die folgenden zusätzlichen Stilelemente:

**Vertices** Dieses Stilelement kann verwendet werden, um das Aussehen des Vielecks zu ändern, indem Sie einen neuen Satz von Eckpunkten definieren. Das Argument **Vertices** ist eine Tabelle mit x/y Punkte, die verwendet werden, um das Vieleck zu zeichnen. Somit benutzt dieser Tag das gleiche Format wie der Befehl **Polygon()**, außer dass Sie nicht die Anzahl der Ecken in der Tabelle angeben. **SetLayerStyle()** wird dies automatisch bestimmen.

Ebenen der Typen **#PRINT** und **#TEXTOUT** verwenden noch folgende zusätzliche Stilelemente:

**Color** Ändert die Farbe des Textes mit einem **ARGB-Wert**.

**DrawPen:** Wenn der Palettenmodus **#PALETTEMODE\_PEN** ist, gibt **DrawPen** den Stift an, der zum Zeichnen des Textes verwendet werden soll. Siehe **Abschnitt 41.27 [SetDrawPen]**, **Seite 864**, für Details. (V9.0)

**FontStyle**

Auf diese Weise können Sie den Schriftstil der Ebene verändern. Sie können hier eine oder mehrere Stile übergeben. Wenn Sie mehrere Schriftstile übergeben, müssen Sie den Bit-Operator (**|**) verwenden. Siehe **Abschnitt 52.34 [SetFontStyle]**, **Seite 1166**, für mögliche Kombinationen. Wenn Sie einen Schriftstil aus einer Ebene entfernen möchten, benutzen Sie das **FontStyleClear** Element. Hinweis: Ab Hollywood 5.0 wird dieser Tag nicht mehr verwendet, um Schatten und Kantenbetonung zu setzen. Für diese beiden Stile verwenden Sie nun die neuen Tags (siehe oben).

**FontStyleClear**

Alle Schriftstile, die Sie in diesem Element setzen, werden aus der Ebene entfernt. Mehrere Schriftstile trennen Sie mit dem Bit-Operator (**|**). Dies ist

das Gegenstück zum Element **Fontstyle**. Hinweis: Ab Hollywood 5.0 wird dieser Tag nicht mehr zum Entfernen der Stile Schatten und Kantenbetonung verwendet. Für diese Stile benutzen Sie nun die neuen Tags (siehe oben).

**Font** Sie können dieses Stilelement benutzen, um die Schriftart der Ebene zu wechseln. Siehe [Abschnitt 52.32 \[SetFont\]](#), [Seite 1163](#), für Details. Alternativ können Sie die neue Schriftart auch durch Setzen des Tags ID angeben (siehe unten).

**FontSize** Dieses Stilelement können Sie verwenden, um die Größe der Schriftart auf der Ebene zu ändern. Siehe [Abschnitt 52.32 \[SetFont\]](#), [Seite 1163](#), für Details.

**ID** Mit diesem Stilelement können Sie die Schriftart der Textebene ändern. Setzen Sie diesen Tag einfach auf die ID der neuen Schriftart und die Schriftart wird geändert. Siehe [Abschnitt 52.42 \[UseFont\]](#), [Seite 1181](#), für Details. Alternativ können Sie die neue Schriftart auch namentlich angeben, indem Sie den Tag **Font** setzen (siehe oben). (V10.0)

**Text** Dieses Stilelement ermöglicht es Ihnen, den Inhalt der Textebene zu ändern. Sie können den gesamten alten Inhalt der Ebene mit dem neuen Text ersetzen.

**Align** Hier können Sie die Textausrichtung nach einer neuen Zeile aktivieren. Mögliche Werte sind **#LEFT**, **#RIGHT**, **#CENTER** und **#JUSTIFIED**. Die Standardausrichtung ist **#CENTER**.

**LeftMargin, RightMargin**

Hier können Sie die Randeinstellungen der aktuellen Textebene ändern. **LeftMargin** kann nur für die Ebenen vom Typ **#PRINT** verwendet werden, aber **RightMargin** kann auch für Textobjekte und Ebenen vom Typ **#TEXTOUT** benutzt werden. Siehe [Abschnitt 52.35 \[SetMargins\]](#), [Seite 1168](#), für Details.

**CursorX, CursorY**

Hier können Sie die Cursorposition dieser Ebene ändern. Dies ist nur mit Ebenen vom Typ **#PRINT** möglich. Beachten Sie auch, dass wenn Sie **CursorX/CursorY** angeben, müssen Sie **X/Y** nicht benutzen. **CursorX/CursorY** schließen sich gegenseitig aus und müssen daher nicht zusammen verwendet werden. (V4.5)

**Tabs** Hier können Sie die Tabulatorpositionen für diese Ebene modifizieren. Dies ist nur mit Ebenen vom Typ **#PRINT** möglich. In **Tabs** übergeben Sie eine Tabelle von Tabulatorpositionen. Siehe [Abschnitt 52.3 \[AddTab\]](#), [Seite 1138](#), für Details. (V4.5)

**Encoding** Hier können Sie die Zeichencodierung dieser Textebene ändern. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details. (V4.7)

**Linespacing:**

Hier können Sie den Abstand zwischen den Zeilen einstellen. Sie können dies auf einen positiven oder negativen Wert setzen. Ein positiver Wert vergrößert den Zeilenabstand, ein negativer Wert verringert ihn. (V9.0)

**Charspacing:**

Dies ermöglicht Ihnen den Abstand zwischen den Zeichen anzupassen. Sie können diesen auf einen positiven oder negativen Wert setzen. Ein positiver Wert vergrößert den Abstand zwischen den Zeichen, ein negativer Wert verringert ihn. (V10.0)

**Tabs:** Hiermit können Sie die Tabulatoren für diese Ebene ändern. Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für Details. (V9.0)

**ListMode:**

Ermöglicht es Ihnen, den Listenmodus für diese Textebene zu aktivieren oder zu deaktivieren. Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für Details. (V9.0)

**DefListBullet:**

Verwenden Sie diesen Tag, um das Standardaufzählungszeichen festzulegen, das verwendet wird, wenn sich die Textebene im Listenmodus befindet. Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für weitere Informationen zu Aufzählungslisten. (V9.0)

**ListBullet:**

Verwenden Sie diesen Tag, um einen benutzerdefinierten Satz von Aufzählungszeichen anzugeben, der verwendet werden soll, wenn sich die Textebene im Listenmodus befindet. Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für Details. (V9.0)

**DefListIndent:**

Mit diesem Tag kann die Anzahl der Leerzeichen angegeben werden, die für die Einrückung von Listenelementen verwendet werden sollen, wenn sich die Textebene im Listenmodus befindet. Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für Details. (V9.0)

**ListIndent:**

Verwenden Sie diesen Tag, um einen benutzerdefinierten Satz von Einzugs-ebenen anzugeben, die verwendet werden sollen, wenn sich die Textebene im Listenmodus befindet. Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für Details. (V9.0)

**DefListOffset:**

Wenn sich die Textebene im Listenmodus befindet und einen nummerierten Aufzählungstyp verwendet, können Sie diesen Tag verwenden, um einen Anfangsversatz für die Nummerierung anzugeben. Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für Details. (V9.0)

**ListOffset:**

Wenn sich die Textebene im Listenmodus befindet und einen nummerierten Aufzählungstyp verwendet, können Sie diesen Tag verwenden, um einen benutzerdefinierten Satz von Startversätzen für die individuelle Listennummerierung anzugeben. Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für Details. (V9.0)

**DefListSpacing:**

Dieser Tag kann verwendet werden, um den Zeilenabstand zwischen den Listenelementen anzugeben, wenn sich die Textebene im Listenmodus befindet. Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für Details. (V9.1)

**ListSpacing:**

Verwenden Sie diesen Tag, um einen benutzerdefinierten Satz von Zeilenabständen festzulegen, die verwendet werden sollen, wenn sich die Textebene im Listenmodus befindet. Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für Details. (V9.1)

**Frame:**

Wenn sich die Textebene im Listenmodus befindet, können Sie mit diesem Tag einen bestimmten Teil der Liste anzeigen. Diese Teile werden ab 1 gezählt. Um den nächsten Teil anzuzeigen, können Sie den Sonderwert 0 an **Frame** übergeben. Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für Details. (V9.0)

**FrameMode:**

Wenn sich die Textebene im Listenmodus befindet, können Sie mit diesem Tag den Modus konfigurieren, den die Liste verwenden soll. Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für Details. (V9.0)

**BulletColor:**

Wenn sich die Textebene im Listenmodus befindet, können Sie mit diesem Tag die Farbe der Aufzählungszeichen ändern. Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für Details. (V9.0)

**BulletPen:**

Wenn sich die Textebene im Listenmodus befindet, können Sie mit diesem Tag das Aufzählungszeichen ändern. Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für Details. (V9.0)

**Wordwrap:**

Dieser Tag kann verwendet werden, um eine Wortumbruchbreite für die Textebene festzulegen. Immer wenn ein Wort die angegebene Breite überschreitet, wird es in die nächste Zeile umgebrochen. Setzen Sie **Wordwrap** auf 0, um den Wortumbruch zu deaktivieren. (V9.0)

Ebenen vom Typ **#TEXTOBJECT** benutzen zusätzlich folgende Stilelemente:

**ID** Dieses Stilelement kann verwendet werden, um ein neues Textobjekt dieser Ebene zuzuordnen. Übergeben Sie hier einfach die ID des gewünschten Textobjekts und es wird das aktuelle Textobjekt dieser Ebene ersetzt. (V4.5)

Ebenen vom Typ **#VECTORPATH** können noch folgende Stilelemente haben:

**ID** Dieses Stilelement kann verwendet werden, um ein neues Vektorlinienobjekt dieser Ebene zuzuordnen. Übergeben Sie hier einfach die ID der gewünschten Vektorlinie und es wird die aktuelle Linie dieser Ebene ersetzt. (V5.0)

**LineJoin** Hier können Sie den Verbindungsstil der Vektorlinie dieser Ebene ändern. Siehe [Abschnitt 53.33 \[SetLineJoin\]](#), [Seite 1204](#), für Details. (V5.0)

- LineCap** Hier können Sie den Stil der Vektorlinienenden dieser Ebene ändern. Siehe [Abschnitt 53.32 \[SetLineCap\]](#), [Seite 1204](#), für Details. (V5.0)
- FillRule** Hier können Sie den Füllstil dieser Ebene ändern. Siehe [Abschnitt 53.31 \[SetFillRule\]](#), [Seite 1203](#), für Details. (V5.0)
- Dashes** Dieses Stilelement kann dazu verwendet werden, um das Strichmuster der Umrissvektorzeichnungen zu ändern. Sie übergeben hier eine Tabelle, welche ein Strichmuster im gleichen Format enthält wie beim Befehl [SetDash\(\)](#) beschrieben ist. Wenn Sie den Tag **Dashes** verwenden, sollten Sie auch den Tag **DashOffset** (siehe unten) übergeben, womit Sie den Startpunkt des Strichmuster angeben. (V5.0)
- DashOffset**  
Mit diesem Stilelement können Sie den Startpunkt für das Strichmuster dieser Ebene ändern. Dieser Tag wird in der Regel mit dem Tag **Dashes** zusammen angegeben (siehe oben).
- VectorEngine:**  
Dieser Tag kann verwendet werden, um das Plugin zu setzen, welches für das Zeichnen dieser Vektorebene verwendet werden sollte. Siehe [Abschnitt 53.35 \[SetVectorEngine\]](#), [Seite 1205](#), für Details. (V6.0)

Ebenen vom Typ **#VIDEO** akzeptieren die folgenden zusätzlichen Stilelemente:

- ID** Dieser Tag kann nur mit dem Befehl [GetLayerStyle\(\)](#) abgefragt werden. Er enthält die ID des Videos, das dieser Ebene zugeordnet ist. Sie können zur Zeit kein neues Video der Ebene mit [SetLayerStyle\(\)](#) zuweisen. (V6.0)
- PartX, PartY, PartWidth, PartHeight**  
Mit diesen vier Elementen können Sie den sichtbaren Teil der Video-Ebene konfigurieren. Bei **PartX** und **PartY** geben Sie die x- und y-Koordinaten innerhalb der Video-Ebene an und mit **PartWidth** und **PartHeight** geben Sie die Größe des Bereichs an, der sichtbar sein soll. Dies ist nützlich, um nur einen Teilbereich eines Videos zu zeigen. (V6.0)

## EINGABEN

- id1** ID der Ebene oder Ebenengruppe, dessen Stil Sie ändern wollen
- style1** Tabelle mit einem oder mehreren Stilelementen aus den oberen Listen
- ...** optional: Sie können so oft die ID/Stil-Sequenz wiederholen, wie Sie benötigen; so können Sie die Stile aus vielen Ebenen mit nur einem einzigen Aufruf des Befehls ändern

## BEISPIEL

```
SetLayerStyle(1, {x = #LEFT, y = #TOP}, 4, {x = #CENTER, y = #CENTER},
5, {x = #RIGHT, y = #BOTTOM}, "mylayer", {x = 100, y = 100})
```

Dieser Aufruf ändert die Position mehrerer Ebenen. Ebene 1 wird zur linken oberen Ecke, Ebene 4 in die Mitte, Ebene 5 in die linke untere Ecke und die Ebene "mylayer" an die Koordinaten 100:100 verschoben.

```
Box(0, 0, 100, 100, #BLUE)
WaitLeftMouse
SetLayerStyle(1, {Color = #RED})
```

Der obige Code zeichnet ein blaues Rechteck, wartet auf die linke Maustaste und dann ändert sich die Farbe des Rechtecks auf Rot.

```
SetLayerStyle(1, {Frame = 0})
```

Dieser Code zeigt das nächste Einzelbild der Ebene 1. Die Ebene muss vom Typ `#ANIM` sein.

```
Polygon(#CENTER, #CENTER, {0, 0, 319, 0, 319, 159, 0, 159}, 4, #RED)
WaitLeftMouse
SetLayerStyle(1, {Vertices = {0, 159, 160, 0, 319, 159}, Color = #YELLOW})
```

Der obige Code zeichnet ein rotes rechteckiges Vieleck, wartet auf die linke Maustaste und dann ändert sich das rechteckige Vieleck in ein gelbes dreieckiges Vieleck.

```
Box(0, 0, 100, 100, #RED)
WaitLeftMouse
SetLayerStyle(1, {Type = #BRUSH, ID = 1})
```

Der obige Code zeichnet ein rotes Rechteck auf dem Bildschirm, wartet auf die linke Maustaste und ersetzt dann das rote Rechteck mit dem Pinsel Nummer 1. Der Ebenentyp `#BOX` wurde zum Typ `#BRUSH` geändert.

## 25.49 SetLayerTint

### BEZEICHNUNG

SetLayerTint – setzt den Farbton einer Ebene (V2.0)

### FRÜHERER NAME

SetLayerLight (V1.5)

### ÜBERSICHT

```
SetLayerTint(id, tintcolor, tintlevel)
```

### BESCHREIBUNG

Dieser Befehl kann benutzt werden, um die Farbtönung einer Ebene oder Ebenengruppe zu korrigieren. Das bedeutet, dass Sie eine **RGB-Farbe** angeben, die in die Ebene oder Ebenengruppe mit der angegebenen Intensität gemischt wird. Das ist z.B. nützlich, wenn Sie die Ebene aufhellen wollen (benutzen Sie dazu `#WHITE` als Farbe) oder wenn sie abdunkeln wollen (benutzen Sie dazu `#BLACK` als Farbe). Natürlich können Sie auch andere Farben benutzen. Die Intensität der Farbtönung reicht von 0 bis 255, wobei 0 keine Tönung bedeutet (Standardeinstellung einer Ebene) und 255 für totale Tönung steht, was die Ebene dann komplett in der angegebenen Farbe erscheinen lässt.

Ab Hollywood 2.0 kann auch eine prozentige Angabe in Form einer Zeichenkette verwendet werden, z.B. "50%".

Ab Hollywood 5.0 wird dieser Befehl einfach einen Filter vom Typ **Tint** (Farbton) in der angegebenen Ebene installieren. Siehe [Abschnitt 25.43 \[SetLayerFilter\]](#), Seite 441, für Details.

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), Seite 405, für Details.

#### EINGABEN

**id** ID der Ebene oder Ebenengruppe

**tintcolor** **RGB-Farbe** für die Tönung

**tintlevel** Intensität der Farbtönung (0 bis 255 oder Prozentangabe)

#### BEISPIEL

```
EnableLayers()
DisplayBrush(1, #CENTER, #CENTER)
SetLayerTint(1, #BLACK, 128)
```

Der obige Code verdunkelt die Ebene 1 (= Pinsel 1) mit einem Verhältnis von 50% (= 128).

## 25.50 SetLayerTransparency

#### BEZEICHNUNG

SetLayerTransparency – setzt die Transparenz einer Ebene (V1.5)

#### ÜBERSICHT

```
SetLayerTransparency(id, level)
```

#### BESCHREIBUNG

Mit diesem Befehl kann der Transparenzgrad einer Ebene oder Ebenengruppe eingestellt werden. Die Transparenzstufe muss zwischen 0 und 255 liegen, wobei 0 keine Transparenz (Standardeinstellung der Ebene) und 255 volle Transparenz bedeutet, womit Sie die Ebene nicht mehr sehen werden (in diesem Fall ist es natürlich effizienter, die Ebene einfach mit **HideLayer()** auszublenden. Bitte beachten Sie, dass dies genau umgekehrt zum befehl **SetAlphaIntensity()** ist, wo 255 keine Transparenz und 0 volle Transparenz bedeutet.

Ab Hollywood 2.0 kann auch eine prozentige Angabe in Form einer Zeichenkette verwendet werden, z.B. "50%".

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), Seite 405, für Details.

#### EINGABEN

**id** ID der Ebene oder Ebenengruppe

**level** Intensität der Transparenz (0 bis 255 oder Prozentangabe)

#### BEISPIEL

```
EnableLayers()
```



```
;den Text nicht anzeigen, nur die Ebene hinzufügen
SelectBGPic(1)
TextOut(#RIGHT, #BOTTOM, "Hello World")
EndSelect
```

```
; nun wird der Text gezeigt!
SetLayerTransparency(1, 128)
```

Der obige Code erzeugt die Ebene 1 (Text "Hello World") und zeigt sie mit einer Transparenz von 50% an (= 128).

## 25.51 SetLayerTransparentPen

### BEZEICHNUNG

SetLayerTransparentPen – setzt den transparenten Stift der Ebenenpalette (V9.0)

### ÜBERSICHT

```
SetLayerTransparentPen(id, pen)
```

### BESCHREIBUNG

Dieser Befehl setzt den transparenten Stift der Palette der durch `id` angegebenen Ebene auf den in `pen` angegebenen Stift. Stifte werden ab 0 gezählt.

Sie müssen **Ebenen aktivieren** bevor Sie diesen Befehl nutzen können. Siehe **Abschnitt 25.1 [Einführung in die Ebenentechnik]**, Seite 405, für Details.

### EINGABEN

<code>id</code>	Identifikator der Ebene
<code>pen</code>	gewünschter transparenter Stift (beginnend bei 0)

### BEISPIEL

```
SetLayerTransparentPen(1, 4)
```

Der Code macht den Stift 4 in der Palette der Ebene 1 transparent.

## 25.52 SetLayerVolume

### BEZEICHNUNG

SetLayerVolume – ändert die Lautstärke einer Video-Ebene (V6.0)

### ÜBERSICHT

```
SetLayerVolume(id, volume)
```

### BESCHREIBUNG

Dieser Befehl ändert die Lautstärke der in `id` angegebene Video-Ebene. Wenn die Video-Ebene gerade läuft, ändert dieser Befehl die Lautstärke on-the-fly, womit Sie dies für Ausblendungen des Sounds verwendet werden können usw. Das Argument `volume` kann auch eine prozentuale Angabe als Zeichenfolge sein, z.B. "50%".

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Um mehr über Hollywoods Ebenen zu erfahren, lesen Sie bitte die **Einführung zu Ebenen**.

**EINGABEN**

<code>id</code>	Identifikator oder Name der Video-Ebene
<code>volume</code>	neue Lautstärke für das Video (Bereich geht von 0=still bis 64=volle Lautstärke oder prozentuale Angabe)

**25.53 SetLayerZPos****BEZEICHNUNG**

SetLayerZPos – wechselt die z-Position einer Ebene (V4.6)

**ÜBERSICHT**

SetLayerZPos(layer, zpos)

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um eine z-Position der Ebene zu ändern. Die z-Position der Ebene ist seine Position in der Hierarchie aller Ebenen. Die erste (d.h. hinterste Ebene) eine z-Position von 1, die letzte (d.h. vorderste) Ebene hat die z-Position, die gleich der Anzahl aller vorhandenen Ebenen ist. Sie müssen die neue gewünschte z-Position für die angegebene Ebene diesem Befehl übergeben. Die Ebene wird dann genau diese z-Position einnehmen und die bestehenden Ebenen, welche an oder nach dieser z-Position sind, werden nach vorne verschoben. Um eine Ebene an vorderster Stelle zu bewegen (das heißt höchste z-Position), können Sie den besonderen Wert 0 in **zpos** angeben. Um eine Ebene ganz nach hinten zu verschieben, übergeben Sie **zpos** den Wert 1.

Sie können auch einen Ebenennamen im Argument **zpos** übergeben. In diesem Fall wird die Ebene in dem ersten Argument die z-Position der Ebene im zweiten Argument einnehmen.

**EINGABEN**

<code>layer</code>	Ebene, dessen z-Position geändert werden soll
<code>zpos</code>	neue z-Position für die Ebene oder 0, um die Ebene an vorderster Stelle zu schieben und 1, um die Ebene an hinterster Stelle zu bringen

**25.54 ShowLayer****BEZEICHNUNG**

ShowLayer – zeigt oder verschiebt eine Ebene (V1.5)

**ÜBERSICHT**

ShowLayer(id[, x, y])

**BESCHREIBUNG**

Dieser Befehl zeigt die durch `id` angegebene verborgene Ebene oder Ebenengruppe an. Sie können Ebenen oder Ebenengruppen verbergen, indem Sie den Befehl **HideLayer()** aufrufen.

Sie müssen **Ebenen aktivieren**, bevor Sie diesen Befehl benutzen können. Um mehr über Hollywoods Ebenen zu erfahren, lesen Sie bitte die **Einführung zu Ebenen**.

Ab Hollywood 2.0 gibt es die zwei optionalen Argumente `x` und `y`. Sie können diese Argumente verwenden, um die angegebene Ebene neu zu positionieren. So können Sie eine Ebene mit diesem Befehl an eine neue Position verschieben. Beide Argumente sind auf `#USELAYERPOSITION` voreingestellt, so wird die Ebene nicht verschoben, wenn Sie diesen beiden Argumenten nichts übergeben.

#### EINGABEN

<code>id</code>	Identifikator der Ebene oder Ebenengruppe, die gezeigt werden soll
<code>x</code>	optional: neue x-Position für die Ebene (voreingestellt ist <code>#USELAYERPOSITION</code> )
<code>y</code>	optional: neue y-Position für die Ebene (voreingestellt ist <code>#USELAYERPOSITION</code> )

#### BEISPIEL

Siehe **Abschnitt 25.17 [HideLayer]**, Seite 420.

## 25.55 ShowLayerFX

#### BEZEICHNUNG

ShowLayerFX – zeigt eine versteckte Ebene mit einem Übergangseffekt an (V1.9)

#### ÜBERSICHT

```
[handle] = ShowLayerFX(id[, table])
```

#### BESCHREIBUNG

Dieser Befehl ist eine erweiterte Version des **ShowLayer()** Befehls. Sie zeigt die durch `id` angegebene, versteckte Ebene oder Ebenengruppe an und benutzt dazu einen der vielen Übergangseffekte, die Hollywood besitzt (angegeben in **Type** weiter unten). Sie können ebenfalls die Übergangsgeschwindigkeit angeben.

Ab Hollywood 4.0 verwendet dieser Befehl eine neue Syntax mit nur einer einzigen Tabelle als optionales Argument. Die alte Syntax wird weiterhin aus Kompatibilitätsgründen unterstützt. Die optionale Tabelle kann den Übergangseffekt konfigurieren. Folgende Optionen sind möglich:

<b>Type</b>	Stellt den gewünschten Übergangseffekt ein. Siehe <b>Abschnitt 30.11 [Display-TransitionFX]</b> , Seite 630, für eine Liste von allen unterstützten Transition-Effekten. (voreingestellt ist <code>#RANDEFFECT</code> )
<b>Speed</b>	Legt die gewünschte Geschwindigkeit für den Übergang fest. Je höher der Wert, den Sie hier angeben, desto schneller wird der Effekt angezeigt werden. (Standardeinstellung ist <code>#NORMALSPEED</code> )
<b>Parameter</b>	Einige Übergangseffekte akzeptieren einen zusätzlichen Parameter, der hier angegeben werden kann. (Standardeinstellung ist <code>#RANDOMPARAMETER</code> )

**Async** Sie können dieses Feld verwenden, um ein asynchrones Zeichnungsobjekt für diesen Übergang zu erstellen. Wenn Sie hier **True** angeben, wird **ShowLayerFX()** sofort verlassen, und es wird ein Handler für das asynchrone Zeichnungsobjekt kreiert, den Sie dann mit dem Befehl **AsyncDrawFrame()** verwenden können. Ein Beispielskript finden Sie unter dem Befehl **AsyncDrawFrame()**. Siehe [Abschnitt 19.1 \[AsyncDrawFrame\]](#), [Seite 231](#), für weitere Informationen über asynchrone Zeichnungsobjekte.

**NoBorderFade**

Wenn die Ebene, welche gezeigt wird, einen Rahmen hat, wird er nicht allmählich erscheinen, sondern am Ende des Übergangseffekts in einem Rutsch angezeigt. (V5.0)

**BorderFX:**

Wenn die anzuzeigende Ebene einen Rahmen hat, wendet Hollywood den Übergangseffekt nur dann auf den Rahmen an, wenn es sich bei der Ebene um eine transparente Ebene mit Text- oder Pixelgrafiken handelt. Für nicht-transparente und Vektorgrafikebenen wird stattdessen ein allgemeiner Überblendungseffekt verwendet, da es sonst aufgrund von Unterschieden in den Rahmenalgorithmen zu visuellen Störungen zwischen dem vorletzten und letzten Effekteinzelbild kommen würde. Wenn Ihnen dieser Fehler egal ist und Sie Hollywood zwingen möchten, den Übergangseffekt immer auf den Rahmen anzuwenden, setzen Sie diesen Tag auf **True**. Um Hollywood zu zwingen, immer den generischen Überblendungsmodus zu verwenden, setzen Sie diesen Tag auf **False**. (V9.0)

Bevor Sie diesen Befehl verwenden können, müssen Sie zuerst die [Ebenen aktivieren](#). Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), [Seite 405](#), für Details.

**EINGABEN**

**id** Identifikator der Ebene oder Ebenengruppe, die gezeigt werden soll  
**table** optional: Tabelle, welche den Übergangseffekt angibt

**RÜCKGABEWERTE**

**handle** optional: Handler auf ein asynchrones Zeichnungsobjekt; wird nur zurückgegeben, wenn **Async** auf **True** gesetzt wurde (siehe oben)

**BEISPIEL**

`ShowLayerFX(5, #CROSSFADE)` ; alte Syntax

OR

`ShowLayerFX(5, {Type = #CROSSFADE})` ; neue Syntax

Dieser Code löscht die Ebene 5 mit einem schönen Effekt.

## 25.56 StopLayer

**BEZEICHNUNG**

StopLayer – stoppt eine gerade laufende Video-Ebene (V6.0)

**ÜBERSICHT**

`StopLayer(id)`

**BESCHREIBUNG**

Dieser Befehl stoppt die in `id` angegebene Video-Ebene. Sie können das Abspielen mit dem Befehl `PlayLayer()` wieder von Anfang an starten.

Bevor Sie diesen Befehl verwenden können, müssen Sie zuerst die **Ebenen aktivieren**. Siehe **Abschnitt 25.1 [Einführung in die Ebenentechnik]**, **Seite 405**, für Details.

**EINGABEN**

`id`            Identifikator oder Name der Video-Ebene, die gestoppt werden soll

## 25.57 SwapLayers

**BEZEICHNUNG**

`SwapLayers` – vertauscht zwei Ebenen (V1.5)

**ÜBERSICHT**

`SwapLayers(a, b)`

**BESCHREIBUNG**

Dieser Befehl vertauscht die Positionen der Ebenen `a` und `b`. Das kann sehr nützlich sein, wenn Sie Ihre Ebenen neu positionieren müssen.

Sie müssen **Ebene aktivieren**, bevor Sie diesen Befehl benutzen können. Um mehr über Hollywoods Ebenen zu erfahren, lesen Sie bitte die **Einführung zu Ebenen**.

**EINGABEN**

`a`            Nummer oder Name der Ebene 1

`b`            Nummer oder Name der Ebene 2

**BEISPIEL**

```
EnableLayers()
SetFillStyle(#FILLCOLOR)
Box(0, 0, 150, 150, #RED)
Box(0, 0, 100, 100, #GREEN)
WaitLeftMouse
SwapLayers(1, 2)
```

Oben stehender Code zeichnet ein rot- und ein grüngefülltes Rechteck und vertauscht dann die Ebenen, was bedeutet, dass das kleinere grüne Rechteck plötzlich nicht mehr sichtbar ist. Vor dem vertauschen war das rote Rechteck Ebene 1 und das grüne Ebene 2. Nach dem Vertauschen ist das grüne Rechteck Ebene 1 und das rote ist Ebene 2.

## 25.58 TransformLayer

**BEZEICHNUNG**

`TransformLayer` – wendet eine affine Transformation auf die Ebene an (V4.5)

**ÜBERSICHT**

```
TransformLayer(id, sx, rx, ry, sy[, smooth])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um eine affine Transformation auf eine Ebene oder Ebenengruppe anzuwenden. Sie müssen eine 2x2 Transformationsmatrix diesem Befehl übergeben, die definiert, wie jedes Pixel in dem Pinsel umgewandelt wird. Dieser Befehl ist nützlich, wenn Sie Rotation und Skalierung gleichzeitig anwenden möchten. Optional können Sie das Argument `smooth` auf `True` setzen, um Antialiasing beim Transformieren zu benutzen, welches zu einem glatteren Aussehen führt, aber dessen Berechnung länger dauert.

Wenn die angegebene Ebene eine Vektorebene ist (zum Beispiel Kreis, Vieleck, TrueType-Text oder ein Rechteck), wird Hollywood die Ebene ohne Qualitätsverlust transformieren können, weil Vektorgrafiken frei umgewandelt werden können. Somit wird das Argument `smooth` keine Funktion haben, wenn die angegebene Ebene eine Vektor-Ebene ist. Wenn die Ebene jedoch Rastergrafiken verwendet, wird die normale rasterbasierte Drehung verwendet.

Im Gegensatz zu den mit dem Befehl `TransformBrush()` transformierten Pinseln behalten Ebenen immer ihre ursprünglichen Daten. So wird es keinen Qualitätsverlust geben, wenn Sie eine Ebene zuerst auf die Größe (20,15) und dann wieder zurück auf (640.480) transformieren. Das ist mit Ebenen durchaus machbar.

Siehe [Abschnitt 43.78 \[TransformBrush\]](#), [Seite 980](#), für weitere Information über das Verwenden der Transformationsmatrix.

**EINGABEN**

<code>id</code>	Identifikator der Ebene oder Ebenengruppe, die transformiert wird
<code>rx</code>	Rotierungsfaktor x
<code>ry</code>	Rotierungsfaktor y
<code>sx</code>	Skalierungsfaktor x; darf nie 0 sein
<code>sy</code>	Skalierungsfaktor y; darf nie 0 sein
<code>smooth</code>	optional: Mit <code>True</code> wird Antialiasing bei der affine Transformation verwendet (Standard ist <code>False</code> )

**BEISPIEL**

```
angle = Rad(45)      ; wandelt Grad in Radiant um
TransformLayer(1, Cos(angle), Sin(angle), -Sin(angle), Cos(angle))
Der obige Code dreht die Ebene 1 um 45 Grad.
```

**25.59 TranslateLayer****BEZEICHNUNG**

TranslateLayer – versetzt/verschiebt eine Ebene (V4.7)

**ÜBERSICHT**

```
TranslateLayer(id, dx, dy)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um eine Ebene oder Ebenengruppe mit den in `dx` und `dy` angegebenen Deltawerten zu verschieben. Das bedeutet, dass eine Ebene durch die angegebenen Deltaversatz relativ zu seiner aktuellen Position bewegt wird. Somit würde ein Versatz von (100,-100) die Ebene um 100 Pixel nach rechts und 100 Pixel nach oben bewegen. Ein Versatz von (0,0) würde die Ebene nicht bewegen. Verschiebungen sind sehr nützlich, um Ebenen unabhängig von ihrer aktuellen Position zu verschieben.

Als Alternative können Sie auch die beiden Tags `TranslateX/TranslateY` mit dem Befehl `SetLayerStyle()` verwenden.

**EINGABEN**

<code>id</code>	ID der Ebene oder Ebenengruppe, die verschoben wird
<code>dx</code>	Deltaversatz x (0 bedeutet keine Verschiebung)
<code>dy</code>	Deltaversatz y (0 bedeutet keine Verschiebung)

**BEISPIEL**

```
TranslateLayer(1, -50, -50)
```

Der obige Code verschiebt die erste Ebene um 50 Pixel nach oben und nach links.

## 25.60 Undo

**BEZEICHNUNG**

Undo – nimmt eine grafische Operation zurück (V1.5)

**ÜBERSICHT**

```
Undo(type[, id, level, quiet])
```

**BESCHREIBUNG**

Dieser Befehl nimmt eine grafische Operation zurück, die durch `type` und - optional - `id` angegeben ist. Sie müssen **Ebene aktivieren**, um diesen Befehl benutzen zu können. Hollywood merkt sich in einem internen Puffer alle grafischen Operationen, die es ausführt, z.B. Pinsel 2 darstellen. Wenn Sie nun Pinsel 2 vom Display entfernen möchten, rufen Sie einfach `Undo(#BRUSH,2)` auf. Die folgenden Typen sind möglich:

<code>#ANIM</code>	Entfernt die durch <code>id</code> angegebene Animation vom Display
<code>#ARC</code>	Entfernt ein mit <code>Arc()</code> gezeichneten Kreis
<code>#BGPICPART</code>	Entfernt eine mit <code>DisplayBGPicPart()</code> dargestellte Grafik
<code>#BOX</code>	Entfernt ein mit <code>Box()</code> gezeichnetes Rechteck
<code>#BRUSH</code>	Entfernt den durch <code>id</code> angegebenen Pinsel vom Display
<code>#BRUSHPART</code>	Entfernt eine mit <code>DisplayBrushPart()</code> dargestellte Grafik
<code>#CIRCLE</code>	Entfernt einen mit <code>Circle()</code> gezeichneten Kreis
<code>#ELLIPSE</code>	Entfernt eine mit <code>Ellipse()</code> gezeichnete Ellipse

#LINE	Entfernt eine mit <code>Line()</code> gezeichnete Linie
#MERGED:	Entfernt eine mit <code>MergeLayers()</code> erstellte zusammengeführte Ebene.
#PLOT	Entfernt einen mit <code>Plot()</code> dargestellten Pixel
#POLYGON	Entfernt ein mit <code>Polygon()</code> gezeichnetes Vieleck
#PRINT	Entfernt einen mit <code>Print()</code> oder <code>NPrint()</code> gedruckten Text
#TEXTOBJECT	Entfernt das durch <code>id</code> angegebene Textobjekt vom Display
#TEXTOUT	Nimmt den letzten <code>TextOut()</code> Befehl zurück; <code>id</code> ist nicht notwendig
#VECTORPATH	Nimmt den letzten <code>DrawPath()</code> Befehl zurück; <code>id</code> ist nicht notwendig
#VIDEO	Entfernt das mit <code>id</code> angegebene Video vom Display

Das optionale Argument `id` wird nur bei Typen verlangt, die einen Identifikator benötigen (`#ANIM`, `#BGPICPART`, `#BRUSH`, `#BRUSHPART`, `#TEXTOBJECT`, `#VIDEO`). Für die anderen Typen ist das Argument `id` nicht erforderlich. Bitte setzen Sie `id` auf 0.

Das Argument `level` gibt die Tiefe der Rücknahmeebene an, die benutzt werden soll. Das Argument ist optional und auf 1 voreingestellt. Die Rücknahmestufe definiert, in welcher Tiefe sich das Objekt befindet, das entfernt werden soll. Wenn Sie zum Beispiel Pinsel 3 viermal darstellen und jetzt den ersten aller "Pinsel 3" entfernen wollen, müssen Sie eine 4 für `level` angeben. Um den letzten wieder zu entfernen, setzen Sie `level` auf 1, was auch der Voreinstellung entspricht. Wird `level` also nicht explizit angegeben oder auf 1 gesetzt, nimmt Hollywood das neueste Objekt zurück, das vom angegebenen Typ dargestellt wurde.

Das Argument `quiet` ist ebenfalls optional. Wenn Sie es auf `True` setzen, entfernt Hollywood nur den Eintrag des Objekts aus der internen Objektliste, lässt es aber auf dem Display sichtbar. Setzen Sie es auf `False`, entfernt Hollywood es ebenfalls vom Display.

## EINGABEN

<code>type</code>	eine der Typkonstanten (siehe Liste oben)
<code>id</code>	optional: nur anzugeben für Typen, die eine zugehörige ID verlangen (voreingestellt ist 0)
<code>level</code>	optional: Tiefe der Rücknahmeebene (voreingestellt ist 1)
<code>quiet</code>	optional: <code>True</code> , wenn das Objekt nur aus der internen Liste entfernt werden soll, nicht aber vom Display (voreingestellt ist <code>False</code> )

## BEISPIEL

```
EnableLayers()
DisplayBrush(1, #CENTER, #CENTER)
WaitLeftMouse
Undo(#BRUSH, 1)
```

Obiger Code stellt Pinsel 1 im Zentrum des Displays dar, wartet auf einen Mausklick und entfernt ihn wieder.



```

EnableLayers()
Print("Hello ")
Print("This ")
Print("Is ")
Print("An ")
Print("Undo ")
Print("Test!")
WaitLeftMouse
Undo(#PRINT, 0, 6)
Undo(#PRINT, 0, 5)
Undo(#PRINT, 0, 4)

```

Dieser Code schreibt "Hello This Is An Undo Test!" auf das Display, wartet auf einen Mausklick und entfernt dann die Texte "Hello", "This" und "Is" durch Verwendung des optionalen Arguments `level` des `Undo()` Befehls.

## 25.61 UndoFX

### BEZEICHNUNG

UndoFX – nimmt eine grafische Operation mit einem Übergangseffekt zurück (V1.5)

### ÜBERSICHT

```
[handle] = UndoFX(type, id[, table])
```

### BESCHREIBUNG

Dieser Befehl ist dem `Undo()` Befehl ähnlich, benutzt aber einen Übergangseffekt, um die Operation rückgängig zu machen. Siehe [Abschnitt 25.60 \[Undo\]](#), [Seite 471](#), für Details.

Denken Sie daran, dass Sie die Ebenen eingeschaltet haben müssen, wenn Sie diesen Befehl benutzen!

Ab Hollywood 4.0 verwendet dieser Befehl eine neue Syntax mit nur einer einzigen Tabelle als optionales Argument. Die alte Syntax wird weiterhin aus Kompatibilitätsgründen unterstützt. Die optionale Tabelle kann den Übergangseffekt konfigurieren. Folgende Optionen sind möglich:

<b>Type</b>	Legt den gewünschten Übergangseffekt fest. Siehe <a href="#">Abschnitt 30.11 [Display-TransitionFX]</a> , <a href="#">Seite 630</a> , für eine Liste aller unterstützten Übergangseffekte. (voreingestellt ist <code>#RANOMEFFECT</code> )
<b>Speed</b>	Legt die gewünschte Geschwindigkeit für den Übergang fest. Je höher der Wert, den Sie hier angeben, desto schneller wird der Effekt angezeigt werden. (Standardeinstellung ist <code>#NORMALSPEED</code> )
<b>Parameter</b>	Einige Übergangseffekte akzeptieren einen zusätzlichen Parameter, der hier angegeben werden kann. (Standardeinstellung ist <code>#RANDOMPARAMETER</code> )
<b>Async</b>	Sie können dieses Feld verwenden, um ein asynchrones Zeichnungsobjekt für diesen Übergang zu erstellen. Wenn Sie hier <code>True</code> angeben, wird <code>UndoFX()</code> sofort verlassen, und es wird ein Handler für das asynchrone Zeichnungsobjekt kreiert, den Sie dann mit dem Befehl <code>AsyncDrawFrame()</code> verwenden

können. Ein Beispielskript finden Sie unter dem Befehl `AsyncDrawFrame()`. Siehe [Abschnitt 19.1 \[AsyncDrawFrame\]](#), [Seite 231](#), für weitere Informationen über asynchrone Zeichnungsobjekte.

#### UndoLevel

Definiert die Undostufe für diesen Vorgang. Siehe [Abschnitt 25.60 \[Undo\]](#), [Seite 471](#), für Informationen über die Undostufen.

#### NoBorderFade

Wenn die Ebene, welche entfernt wird, einen Rahmen hat, wird er sich nicht allmählich auflösen, sondern am Ende des Übergangseffekts in einem Rutsch verschwinden. (V5.0)

#### BorderFX:

Wenn die zu entfernende Ebene einen Rahmen hat, wendet Hollywood den Übergangseffekt nur dann auf den Rahmen an, wenn es sich bei der Ebene um eine transparente Ebene mit Text- oder Pixelgrafiken handelt. Für nicht-transparenten und Vektorgrafikebenen wird stattdessen ein allgemeiner Überblendungseffekt verwendet, da es sonst aufgrund von Unterschieden in den Rahmenalgorithmen zu visuellen Störungen zwischen dem vorletzten und letzten Effekteinzelbild kommen würde. Wenn Ihnen dieser Fehler egal ist und Sie Hollywood zwingen möchten, den Übergangseffekt immer auf den Rahmen anzuwenden, setzen Sie diesen Tag auf **True**. Um Hollywood zu zwingen, immer den generischen Überblendungsmodus zu verwenden, setzen Sie diesen Tag auf **False**. (V9.0)

### EINGABEN

<b>type</b>	eine der Typkonstanten (Siehe <a href="#">Abschnitt 25.60 [Undo]</a> , <a href="#">Seite 471</a> , für Details.)
<b>id</b>	ID des Objekts
<b>table</b>	optional: Konfiguriert den Übergangseffekt

### RÜCKGABEWERTE

<b>handle</b>	optional: Handler auf ein asynchrones Zeichnungsobjekt; wird nur zurückgegeben, wenn <b>Async</b> auf <b>True</b> gesetzt wurde (siehe oben)
---------------	--

## 25.62 UngroupLayer

### BEZEICHNUNG

UngroupLayer – entfernt Ebenen aus der Gruppe (V10.0)

### ÜBERSICHT

`UngroupLayer(layer1[, layer2, ...])`

### BESCHREIBUNG

Dieser Befehl entfernt die angegebene(n) Ebene(n) aus der Ebenengruppe. Die Ebenen, die Sie an diesen Befehl übergeben, müssen nicht zur selben Gruppe gehören. Sie werden aus der Gruppe, zu der sie gehören, herausgelöst. Sie können beliebig viele Ebenen

übergeben. Beachten Sie, dass eine Gruppe automatisch gelöscht wird, sobald ihr keine Ebenen mehr zugeordnet sind.

Um Ebenen zu einer Gruppe hinzuzufügen, verwenden Sie den Befehl `GroupLayer()`. Siehe [Abschnitt 25.16 \[GroupLayer\]](#), [Seite 418](#), für Details.

Sie müssen **Ebenen aktivieren** bevor Sie diesen Befehl nutzen können. Siehe [Abschnitt 25.1 \[Einführung in die Ebenentechnik\]](#), [Seite 405](#), für Details.

#### EINGABEN

<code>layer1</code>	erste Ebene, welche entfernt werden soll
<code>...</code>	weitere zu entfernenden Ebenen



## 26 Ereignisbibliothek

### 26.1 BreakEventHandler

#### BEZEICHNUNG

BreakEventHandler – unterbricht aktueller Zyklus des Ereignis-Handlers (V5.2)

#### ÜBERSICHT

BreakEventHandler()

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den internen Hollywood-Zyklus vom Ereignis-Handler zu unterbrechen. Dies ist ein Lowlevel-Befehl und Sie werden ihn in der Regel nicht verwenden müssen. Er ist für bestimmte Notsituationen und Debug-Zwecke gedacht.

#### EINGABEN

Keine

### 26.2 ChangeInterval

#### BEZEICHNUNG

ChangeInterval – wechselt die Intervallfrequenz (V2.0)

#### ÜBERSICHT

ChangeInterval(id, ms)

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die Frequenz eines laufenden Intervall zu ändern. Geben Sie einfach die ID des Intervalls und die neue Frequenz an. Siehe [Abschnitt 26.26 \[SetInterval\]](#), [Seite 515](#), für alles, was Sie über Intervalle wissen müssen. Weiterer Intervall-Befehl: [ClearInterval\(\)](#)

#### EINGABEN

id	Identifikator des Intervalls
ms	neue Intervallfrequenz in Millisekunden

### 26.3 CheckEvent

#### BEZEICHNUNG

CheckEvent – überprüft ein Ereignis ohne zu blockieren (V1.9)

#### ÜBERSICHT

info = CheckEvent()

#### BESCHREIBUNG

Dieser Befehl prüft, ob sich ein Ereignis in der Warteschlange befindet. Wenn ja, wird `CheckEvent()` das Ereignis aus der Warteschlange entfernen und die Callback-Funktion

ausführen. Wenn sich kein Ereignis in der Warteschlange befindet, wird das Skript sofort nach `CheckEvent()` weiter geführt.

`CheckEvent()` gibt eine Tabelle in `info` zurück, die Informationen darüber enthält, ob es eine Callback-Funktion ausgeführt hat. Die folgenden Felder werden in dieser Tabelle initialisiert:

**Action:** Enthält den Namen des Ereignisses, das die Callback-Ausführung verursacht hat (z.B. `OnMouseDown`). Wenn `CheckEvent()` keine Callback-Funktion ausgeführt hat, wird dieses Feld auf eine leere Zeichenkette gesetzt.

**ID:** Enthält die ID des Objekts, welches die Callback-Ausführung (zum Beispiel ein Display-Identifikator) verursacht hat. ID kann auch Null sein, falls ein Ereignis verursacht wurde, das keine ID hat.

**Triggered:** Wird auf `True` gesetzt, wenn `CheckEvent()` eine Callback-Funktion ausgeführt hat.

**NResults:** Enthält die Anzahl der Werte, die der Benutzer-Callback zurückgegeben hat (z.B. 1). Diese wird 0 sein, wenn der Benutzer-Callback keine Werte zurückgegeben hat oder überhaupt kein Benutzer-Callback lief.

**Results:** Wenn `NResults` größer ist als 0, wird diese Tabelle alle Werte enthalten, die der Benutzer-Callback zurückgibt. Sonst wird diese Tabelle überhaupt nicht vorhanden sein. Diese Tabelle können Sie leicht für zusätzliche Informationen verwenden, die Sie von Ihrem Callback an das Hauptprogramm übergeben.

`CheckEvent()` ist ähnlich wie der Befehl `WaitEvent()` mit dem Unterschied, dass `WaitEvent()` die Ausführung des Skripts blockiert, bis ein Ereignis eintritt, während `CheckEvent()` sofort beendet wird, wenn kein Ereignis vorhanden ist. Mit diesem Befehl können Sie etwas erledigen, während auf ein Ereignis gewartet wird. Mit `WaitEvent()` wäre dies nicht möglich.

Beachten Sie, dass `CheckEvent()` nur ein einzelnes Ereignis aus der Ereigniswarteschlange verarbeitet. Wenn Sie alle Ereignisse behandeln möchten, die derzeit in der Ereigniswarteschlange sind, müssen Sie `CheckEvents()` verwenden. Siehe [Abschnitt 26.4 \[CheckEvents\]](#), [Seite 479](#), für Details.

Bitte beachten Sie auch, dass Sie im Allgemeinen `CheckEvent()` nur dann verwenden sollten, wenn Sie ihn wirklich brauchen. `WaitEvent()` ist in der Regel die bessere Idee als `CheckEvent()`.

## EINGABEN

Keine

## RÜCKGABEWERTE

**info** Tabelle mit Informationen darüber, ob ein Ereignis aufgetreten ist oder nicht, und der Rückgabewert des Benutzer-Callback, falls er aufgerufen wurde.

## 26.4 CheckEvents

### BEZEICHNUNG

CheckEvents – überprüft mehrere Ereignisse ohne zu blockieren (V6.1)

### ÜBERSICHT

CheckEvents()

### BESCHREIBUNG

Dieser Befehl macht das gleiche wie `CheckEvent()`, aber er kümmert sich um alle Ereignisse in der Warteschlange. `CheckEvent()` andererseits verarbeitet nur ein einzelnes Ereignis aus der Warteschlange.

Ein weiterer Unterschied besteht darin, dass `CheckEvents()` keine Informationen über die Ereignisse zurückgeben wird. Wenn Sie diese Informationen benötigen, müssen Sie `CheckEvent()` verwenden. Siehe [Abschnitt 26.3 \[CheckEvent\]](#), [Seite 477](#), für Details.

Bitte beachten Sie, dass Sie `CheckEvents()` nur dann verwenden sollten, wenn Sie ihn wirklich brauchen. `WaitEvent()` ist in der Regel die bessere Idee als `CheckEvents()`.

### EINGABEN

Keine

## 26.5 ClearInterval

### BEZEICHNUNG

ClearInterval – entfernt eine Intervallfunktion (V2.0)

### ÜBERSICHT

ClearInterval(id)

### BESCHREIBUNG

Dieser Befehl entfernt die in `id` angegebene Intervallfunktion. Siehe [Abschnitt 26.26 \[SetInterval\]](#), [Seite 515](#), für alles, was Sie über Intervalle wissen müssen. Weiterer Intervall-Befehl: `ChangeInterval()`

### EINGABEN

`id`            Identifikator der zu entfernende Intervallfunktion

### BEISPIEL

Siehe [Abschnitt 26.26 \[SetInterval\]](#), [Seite 515](#).

## 26.6 ClearTimeout

### BEZEICHNUNG

ClearTimeout – entfernt eine Timeout-Funktion (V2.0)

### ÜBERSICHT

ClearTimeout(id)

### BESCHREIBUNG

Dieser Befehl entfernt die in `id` angegebene Timeout-Funktion. Siehe [Abschnitt 26.27 \[SetTimeout\]](#), [Seite 517](#), für alles, was Sie über Timeout-Funktionen wissen möchten.

**EINGABEN**

`id`            Identifikator der Timeout-Funktion, die Sie löschen wollen

**BEISPIEL**

Siehe [Abschnitt 26.27 \[SetTimeout\]](#), Seite 517.

## 26.7 CtrlCQuit

**BEZEICHNUNG**

CtrlCQuit – aktiviert/deaktiviert Programm verlassen mit Control-C (V2.0)

**ÜBERSICHT**

`CtrlCQuit(enable)`

**BESCHREIBUNG**

Standardmäßig können alle Hollywood-Skripte jederzeit durch drücken der Tastenkombination CTRL-C unterbrochen werden. Wenn Sie dies nicht wünschen, verwenden Sie diesen Befehl, um diese Funktionalität zu deaktivieren.

Bitte beachten Sie: Denken Sie zweimal nach, bevor Sie CTRL-C sperren. Zum Beispiel, wenn Ihr Skript in einem rahmenlosen Fenster oder Vollbildmodus ausgeführt wird, fehlt das sonst im Fensterrahmen integrierte Schließsymbol. So ist es ganz praktisch, das Skript mit CTRL-C zu beenden.

Siehe auch [EscapeQuit\(\)](#) und [LeftMouseQuit\(\)](#).

**EINGABEN**

`enable`        `True` aktiviert CTRL-C Verlassen, `False` schaltet es aus

**BEISPIEL**

```
CtrlCQuit(TRUE)
Repeat
  Wait(10)
Forever
```

Obiger Codeteil läuft in eine Endlosschleife, die normalerweise Ihr Programm blockieren würde. Die Verwendung von `CtrlCQuit(True)` erlaubt dem Benutzer jedoch, sie abbrechen.

## 26.8 DeleteButton

**BEZEICHNUNG**

DeleteButton – löscht eine Schaltfläche (V2.0)

**ÜBERSICHT**

`DeleteButton(id)`

**BESCHREIBUNG**

Dieser Befehl löscht die Schaltfläche mit der angegebenen ID aus dem aktuellen Hintergrundbild.

Siehe auch [DisableButton\(\)](#), [EnableButton\(\)](#) und [MakeButton\(\)](#).



**EINGABEN**

`id`            Identifikator der zu löschenden Schaltfläche

## 26.9 DisableButton

**BEZEICHNUNG**

DisableButton – deaktiviert eine Schaltfläche (V2.0)

**ÜBERSICHT**

`DisableButton(id)`

**BESCHREIBUNG**

Dieser Befehl deaktiviert vorübergehend die Schaltfläche mit der Nummer `id`. Sie können sie später wieder mit dem Befehl `EnableButton()` aktivieren. Wenn Sie eine Schaltfläche vollständig entfernen möchten, verwenden Sie den Befehl `DeleteButton()`.

Siehe auch `MakeButton()`.

**EINGABEN**

`id`            Identifikator der Schaltfläche, die deaktiviert wird.

## 26.10 EnableButton

**BEZEICHNUNG**

EnableButton – aktiviert eine Schaltfläche (V2.0)

**ÜBERSICHT**

`EnableButton(id)`

**BESCHREIBUNG**

Mit diesem Befehl können Sie die Schaltfläche mit der Nummer `id` wieder aktivieren. Dies ist nur erforderlich, wenn Sie sie zuvor mit `DisableButton()` deaktiviert haben.

Siehe auch `DeleteButton()` und `MakeButton()`.

**EINGABEN**

`id`            Identifikator der Schaltfläche, die aktiviert wird

## 26.11 EscapeQuit

**BEZEICHNUNG**

EscapeQuit – erlaubt/verbietet das Beenden mit der Escape-Taste (V1.5)

**ÜBERSICHT**

`EscapeQuit(enable)`

**BESCHREIBUNG**

Wenn Sie `enable` auf `True` setzen, wird das Drücken der Escape-Taste Ihr Skript sofort beenden.

Siehe auch `CtrlCQuit()` und `LeftMouseQuit()`.

**EINGABEN**

**enable**      **True**, um beenden mit der Escape-Taste zu erlauben, **False**, um es zu verbieten

**BEISPIEL**

```
EscapeQuit(TRUE)
Repeat
    Wait(10)
Forever
```

Obiger Codeteil läuft in einer Endlosschleife, die normalerweise Ihr Programm blockieren würde. Die Benutzung von `EscapeQuit(True)` erlaubt dem Benutzer jedoch, es abzubrechen.

**26.12 InKeyStr****BEZEICHNUNG**

`InKeyStr` – fragt die Benutzereingabe ab (V1.5)

**ÜBERSICHT**

```
input$ = InKeyStr(type[, maxlen, password, cursor])
```

**BESCHREIBUNG**

Dieser Befehl erlaubt Ihnen, auf einfache Weise Eingaben von der Tastatur zu lesen. **type** gibt die Art der Zeichen an, die als Eingabe erlaubt sind. **maxlen** kann benutzt werden, um die maximale Länge zu begrenzen, die der Benutzer eingeben kann (0 ist voreingestellt und bedeutet keine Begrenzung). Falls **password** auf **True** gesetzt ist, zeigt Hollywood für jedes eingegebene Zeichen ein (\*).

Momentan können folgende Typen angegeben werden:

**#ALL**            Akzeptiert alle sichtbaren Zeichen.

**#ALPHABETICAL**

Akzeptiert nur alphabetische Zeichen; das ist nicht notwendigerweise begrenzt auf die Zeichen A-Z. Der Benutzer kann auch spezielle Zeichen eintippen, die es nur in seiner Landessprache gibt.

**#ALPHANUMERICAL**

Akzeptiert alphabetische und numerische Zeichen.

**#HEXNUMERICAL**

Akzeptiert hexadezimale Zeichen (0-9 und a-f).

**#NUMERICAL**

Akzeptiert 0-9.

Wenn Sie Ebenen aktiviert haben, während Sie diesen Befehl verwenden, werden Sie eine neue Ebene vom Typ **#PRINT** erhalten, die die vom Benutzer eingegebene Zeichenfolge enthält (seit Hollywood 2.0, in früheren Versionen wurde für jedes Zeichen eine neue Ebene hinzugefügt).

Ab Hollywood 8.0 gibt es ein neues optionales Argument namens **cursor**. Wenn dies auf **True** gesetzt ist, zeigt `InKeyStr()` einen Cursor während der Benutzereingabe an. In

diesem Fall können Sie auch mit den Cursortasten vorwärts und rückwärts navigieren. Mit der ENTF-Taste (Del oder Delete) können Sie auch Zeichen löschen. Der Cursor wird in derselben Farbe wie der Text gezeichnet.

Hollywood 8.0 fügt `InKeyStr()` außerdem die Unterstützung fürs Einfügen hinzu. Drücken Sie einfach `STRG + V` (unter Windows) oder `CMD + V` (auf allen anderen Systemen), um Text aus der Zwischenablage an der aktuellen Einfügeposition einzufügen.

## EINGABEN

<code>type</code>	gibt den Typ der erlaubten Zeichen an
<code>maxlen</code>	optional: Wenn Sie dieses Argument angeben, kann der Benutzer nur die angegebene Anzahl Zeichen eingeben; andernfalls kann er so viele Zeichen eintippen, wie er möchte und mit der RETURN-Taste beenden (voreingestellt ist 0, was unbegrenzte Anzahl Zeichen bedeutet)
<code>password</code>	optional: Falls auf <code>True</code> gesetzt, wird Hollywood ein (*) anstelle des momentan getippten Zeichens zeigen (voreingestellt ist <code>False</code> )
<code>cursor</code>	optional: Wenn er auf <code>True</code> gesetzt ist, wird ein Cursor angezeigt, der die aktuelle Einfüge- und Löschposition angibt (Standardeinstellung ist <code>False</code> ) (V8.0)

## RÜCKGABEWERTE

<code>input\$</code>	Die eingegebene Zeichenkette
----------------------	------------------------------

## BEISPIEL

```
Print("What is your name? ")
name$ = InKeyStr(#ALPHABETICAL)
Print("Hello", name$, "!")
```

Dieses Programmstück fragt den Benutzer nach seinem Namen und gibt den eingegebenen Namen wieder auf dem Bildschirm aus.

## 26.13 InstallEventHandler

### BEZEICHNUNG

`InstallEventHandler` – installiert/entfernt ein neuer Ereignis-Handler (V2.0)

### ÜBERSICHT

```
InstallEventHandler(table[, userdata])
```

### BESCHREIBUNG

Sie können diesen Befehl benutzen, um Ihre eigenen Ereignis-Handler für Standardereignisse zu installieren. Sie übergeben diesem Befehl eine Tabelle, mit welcher Hollywood den Ereignis-Handler installiert oder entfernt. Um einen neuen Handler zu installieren, müssen Sie die entsprechenden Tabellenfelder mit Ihrer eigenen Funktion initialisieren. Wenn Sie einen Ereignis-Handler entfernen möchten, stellen Sie das entsprechende Tabellenfeld auf 0.

Folgenden Tabellenfelder werden durch diesen Befehl erkannt:

**OnKeyDown:**

Die Funktion, die Sie hier angeben, wird jedes Mal aufgerufen werden, wenn der Benutzer eine Kontroll-, eine Nummern- oder eine englische Alphabettaste drückt. An die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Wird **OnKeyDown** beinhalten.

**Key:** Wird die Taste beinhalten, die gedrückt wurde.

**ID:** Wird auf die ID des Display gesetzt, welches den Tastenanschlag empfangen hat.

Beachten Sie, dass **OnKeyDown** nur Bearbeitungs- (wie ESC, Rücklöschaste (Backspace)) Cursor-, Nummern- und englische Alphabettasten unterstützt. Um nicht-englische Tasten zu überwachen, verwenden Sie stattdessen den Ereignis-Handler **VanillaKey**. **VanillaKey** unterstützt die komplette Unicode-Reihe von Tasten. Beachten Sie auch, dass **OnKeyDown** bestimmte nicht-englische Tasten auf einigen Plattformen unterstützt, aber das ist inoffizielles Verhalten und Sie sollten sich nicht darauf verlassen. Wenn Sie die Umschalttasten wie Shift, Alt, Control usw. überwachen möchten, verwenden Sie stattdessen den Ereignis-Handler **OnRawKeyDown** (siehe unten).

**OnKeyUp:** Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Benutzer eine Bearbeitungs-, eine Cursortaste, eine numerische Taste oder eine englische alphabetische Taste loslässt. Die Funktion erhält eine Nachricht als Parameter 1 mit folgenden Feldern:

**Action:** Wird **OnKeyUp** beinhalten.

**Key:** Wird auf die Taste gesetzt, die losgelassen wurde. Siehe [Abschnitt 26.23 \[Raw keys\]](#), [Seite 512](#), für eine Liste von den rohen Tasten, die von Hollywood unterstützt werden.

**ID:** Wird auf den Identifikator des Display gesetzt, welches das Loslassen der Taste empfangen hat.

Beachten Sie, dass **OnKeyUp** nur Steuer- (wie ESC, Rücklöschaste (Backspace), Cursortasten usw.), Nummern- und englische Alphabettasten unterstützt. Um nicht-englische Tasten zu überwachen, verwenden Sie stattdessen den Ereignis-Handler **VanillaKey**. **VanillaKey** unterstützt die komplette Unicode-Reihe von Tasten. Beachten Sie auch, dass **OnKeyUp** bestimmte nicht-englische Tasten auf einigen Plattformen unterstützt, aber das ist inoffizielles Verhalten und Sie sollten sich nicht darauf verlassen. Wenn Sie die Umschalttasten wie Shift, Alt, Control usw. überwachen möchten, verwenden Sie stattdessen den Ereignis-Handler **OnRawKeyUp** (siehe unten).

**OnRawKeyDown:**

Dieser Ereignis-Handler kann verwendet werden, um rohe Tastaturereignisse zu überwachen. Der Unterschied zwischen rohen und normalen Tastatur-

ereignissen besteht darin, dass rohe Tastaturereignisse immer die Grundtaste liefert, ohne dass potenzielle Umschalttasten wie Shift, Alt, Control, etc. darauf einfluss nehmen, die ebenfalls gedrückt sein könnten. Wenn Sie z.B. die Shift-Taste und die "1"-Taste auf einer englischen oder deutschen Tastatur drücken, meldet `OnKeyDown`, dass die Taste "!" gedrückt wurde, während `OnRawKeyDown` zwei Tastaturereignisse meldet: Zuerst meldet der Ereignis-Handler, dass die Shift-Taste und dann die Taste "1" gedrückt wurde. Im Gegensatz zu `OnKeyDown` wird `OnRawKeyDown` niemals die Shift- und die "1"-Tastenkombination als "!"-Taste melden. Außerdem wird `OnKeyDown` niemals ausgelöst, wenn eine Umschalttaste wie Shift, Alt, Control, etc. alleine gedrückt wurde. `OnRawKeyDown` wird aber auch ausgelöst, wenn der Benutzer eine Umschalttaste drückt. So ist `OnRawKeyDown` sehr nützlich, um Umschalttasten oder Kombinationen von Zeichen- und Umschalttasten zu überwachen. Sie können z.B. diesen Ereignis-Handler verwenden, um herauszufinden, ob die rechte Alt- und eine Zeichentaste gleichzeitig gedrückt wurden. Die Funktion, die Sie an diesen Ereignis-Handler übergeben, erhält eine Meldung als Parameter 1 mit den folgenden Feldern:

**Action:** Wird `OnRawKeyDown` beinhalten.

**Key:** Wird auf die gedrückte Taste gesetzt. Siehe [Abschnitt 26.23 \[Rohe Tasten\]](#), [Seite 512](#), für eine Liste der von Hollywood unterstützten rohen Tasten.

**Modifiers:**

Wird auf eine Kombination von Umschalttasten gesetzt, die momentan gedrückt sind. Die folgenden Umschalttasten-Flags können gesetzt werden:

`#MODLSHIFT`

Linke Shift-Taste

`#MODRS SHIFT`

Rechte Shift-Taste

`#MODLALT` Linke Alt-Taste

`#MODRAL T` Rechte Alt-Taste

`#MODLCOMMAND`

linke Steuer-Taste (z.B. Windows Start-Taste, linke Amiga-Taste)

`#MODRCOMMAND`

Rechte Steuer-Taste (z.B. Windows Menü-Taste, rechte Amiga-Taste)

`#MODLCONTROL`

Linke Control-Taste

`#MODRCONTROL`

Rechte Control-Taste

**ID:** Wird auf die ID des Display gesetzt, welches die gedrückte Taste empfangen hat.

(V7.1)

#### **OnRawKeyUp:**

Dieser Ereignis-Handler wird immer dann ausgelöst, wenn eine rohe Taste losgelassen wurde. Bitte lesen Sie die Beschreibung von **OnRawKeyDown** weiter oben, um mehr über den Unterschied zwischen rohen und normalen Tastaturereignissen zu erfahren. Die Funktion, die Sie an diesen Ereignis-Handler übergeben, erhält eine Meldung als Parameter 1 mit den folgenden Feldern:

**Action:** Wird **OnRawKeyUp** beinhalten.

**Key:** Wird auf die losgelassene Taste gesetzt. Siehe [Abschnitt 26.23 \[Rohe Tasten\]](#), [Seite 512](#), für eine Liste der von Hollywood unterstützten rohen Tasten.

**Modifiers:**

Wird auf eine Kombination von Umschalttasten gesetzt, die momentan gedrückt sind. Siehe oben bei **OnRawKeyDown** für eine Liste der Umschalttasten-Flags.

**ID:** Wird auf die ID des Display gesetzt, welches die losgelassene Taste empfangen hat.

(V7.1)

#### **VanillaKey:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Benutzer eine Taste oder eine Tastenkombination drückt, die zu einem Zeichen mit grafischer Darstellung einschließlich dem Leerzeichen führt. **VanillaKey** unterstützt die ganze Unicode-Reihe und kann auch Zeichen behandeln, die durch mehrere gedrückte Tasten erzeugt werden, wie z.B. Diakritische Zeichen. Diesen Ereignis-Handler können Sie verwenden, wenn Ihre Anwendung auch nicht-englische Zeichen berücksichtigen muss. Die Funktion erhält eine Meldung als Parameter 1 mit folgenden Feldern:

**Action:** Wird **VanillaKey** beinhalten.

**Key:** Wird die Taste beinhalten, die gedrückt wurde.

**ID:** Wird auf die ID des Display gesetzt, welches die Taste empfangen hat.

Beachten Sie, dass **VanillaKey** nur Taste-Drücken- (Key-Down) und Taste-Halten-Ereignisse (Key-Repeat) meldet. Es kann keine Taste-Loslassen-Ereignisse (Key-Up) melden, da dies für Zeichen mit mehreren Tastenanschläge nicht möglich ist. Darüber hinaus werden nur druckbare Zeichen (einschließlich des Leer-/Zwischenraum-Zeichens (SPACE)) gemeldet. Wenn Sie auf Steuertasten wie ESC, Rücklösch Taste (Backspace), Cursorstasten usw. zugreifen müssen, verwenden Sie die **OnKeyDown-** und **OnKeyUp-Ereignis-Handler**.

(V7.0)

**OnMouseMove:**

Die Funktion, die Sie hier angeben, wird jedes Mal aufgerufen werden, wenn der Benutzer die Maus bewegt. An die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Wird **OnMouseMove** beinhalten.

**X,Y:** Werden die aktuellen Mauskoordinaten enthalten. Wenn sie negativ sind, befindet sich der Mauszeiger außerhalb der Grenzen des Displays.

**ID:** Wird auf die ID des Display gesetzt, welches die Bewegung der Maus empfangen hat.

**OnMouseDown:**

Die Funktion, die Sie hier angeben, wird jedes Mal aufgerufen werden, wenn der Benutzer die linke Maustaste drückt. An die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Wird **OnMouseDown** beinhalten.

**ID:** Wird auf die ID des Display gesetzt, welches die linke Maustaste empfangen hat.

(V3.1)

**OnMouseUp:**

Die Funktion, die Sie hier angeben, wird jedes Mal aufgerufen werden, wenn der Benutzer die linke Maustaste loslässt. An die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Wird **OnMouseUp** beinhalten.

**ID:** Wird auf die ID des Display gesetzt, welches die linke Maustaste empfangen hat.

(V3.1)

**OnRightMouseDown**

Die Funktion, die Sie hier angeben, wird jedes Mal aufgerufen werden, wenn der Benutzer die rechte Maustaste drückt. An die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **OnRightMouseDown**.

**ID:** Wird auf die ID des Display gesetzt, welches die rechte Maustaste empfangen hat.

(V3.1)

**OnRightMouseUp:**

Die Funktion, die Sie hier angeben, wird jedes Mal aufgerufen werden, wenn der Benutzer die rechte Maustaste loslässt. An die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **OnRightMouseUp**.

**ID:** Wird auf die ID des Display gesetzt, welches die rechte Maustaste empfangen hat.

(V3.1)

**OnWheelDown:**

Die Funktion, die Sie hier angeben, wird jedes Mal aufgerufen werden, wenn der Benutzer das Mausrad runter dreht. An die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **OnWheelDown**.

**ID:** Wird auf die ID des Display gesetzt, welches das Mausraddrehen empfangen hat.

(V4.0)

**OnWheelUp:**

Die Funktion, die Sie hier angeben, wird jedes Mal aufgerufen werden, wenn der Benutzer das Mausrad rauf dreht. An die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **OnWheelUp**.

**ID:** Wird auf die ID des Display gesetzt, welches das Mausraddrehen empfangen hat.

(V4.0)

**OnMusicEnd:**

Die Funktion, die Sie hier angeben, wird jedes Mal aufgerufen, wenn eine Musikwiedergabe beendet wurde. An die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **OnMusicEnd**.

**ID:** Identifikator des Objekts, welches beendet wurde.

Bitte beachten Sie, dass dieses Ereignis nur ausgelöst wird, wenn die Wiedergabe von Musik beendet ist. Es wird nicht ausgelöst, wenn Sie **StopMusic()** aufrufen.

**OnSampleLoop:**

Die hier angegebene Funktion wird jedes Mal aufgerufen werden, wenn ein Sample gestartet wird. Wenn das Sample nur einmal spielt, wird die Funktion auch nur einmal aufgerufen. Wird das Sample im Schleifenmodus gespielt, können Sie der Funktion angeben, dass bei jeder Wiederholung des Samples die Funktion aufgerufen wird. An die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Felder übergeben:

**Action:** Beinhaltet **OnSampleLoop**.

**ID:** Identifikator des Samples, der gerade gestartet oder wiederholt wurde.

**Time:** Die Zeit in Millisekunden, die das Sample jetzt gespielt hat.



**Starts:** Die Anzahl der Starts von diesem Sample. Dieses Feld beginnt bei 1 und wird bei jeder Hollywood-Schleife Ihres Samples erhöht werden.

Achtung! Verwenden Sie diesen Ereignis-Handler mit Vorsicht. Wenn Sie ein kurzes Sample unendlich schleifen, wird Ihre Callback-Funktion wieder und wieder aufgerufen, die Ihre Skriptleistung massiv reduzieren wird. Dieser Ereignis-Handler ermöglicht es Ihnen, exaktes Timing mit Sample-Wiedergabe zu erzielen.

**OnSampleEnd:**

Die Funktion, die Sie hier angeben, wird jedes Mal aufgerufen, wenn ein Sample beendet wurde. An die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **OnSampleEnd**.

**ID:** Identifikator des Samples, der gerade aufgehört hat.

**Time:** Die Zeit in Millisekunden, die das Sample jetzt gespielt hat.

**Starts:** Die Anzahl der Starts von diesem Sample. Dieses Feld beginnt bei 1 und wird bei jeder Hollywood-Schleife Ihres Samples erhöht werden.

Bitte beachten Sie, dass dieses Ereignis nur ausgelöst wird, wenn die Wiedergabe des Samples beendet ist. Es wird nicht ausgelöst, wenn Sie **StopSample()** aufrufen.

**OnARexx:** Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn eine ARexx-Nachricht an den REXX-Port (dieser Port wurde mit **CreateRexxPort()** erstellt) ankommt. Die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **OnARexx**.

**Command:** Eine Zeichenkette mit dem Befehl, der ausgeführt werden soll.

**Args:** Eine vorab von Hollywood analysierte Zeichenfolge, die alle Argumente (getrennt durch Null-Zeichen ("\0")) für diesen Befehl enthält. Sie können den **SplitStr()** Befehl verwenden, um die einzelnen Argumente zu erhalten.

**ArgC:** Die Anzahl der Argumente in der Zeichenkette **Args**.

**RawArgs:** Die nicht analysierte Zeichenfolge, die alle Argumente enthält.

Bitte beachten Sie, dass Hollywood nicht immer die Argumente in der Art und Weise trennt, wie Sie sie haben wollen. In diesem Fall können Sie das Feld **RawArgs** verwenden, die für den Zugriff auf die Argumente in ihrem ursprünglichen Format verwendet werden kann. Die Felder **Args** und **ArgC** sind nur für Ihre Bequemlichkeit enthalten, aber einige erfahrene Benutzer könnten manchmal **RawArgs** bevorzugen.

Siehe **Abschnitt 16.3 [CreateRexxPort]**, Seite 174, für ein Beispiel von diesem Ereignis-Handler.

(V2.5)

**SizeWindow:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Benutzer die Fenstergröße ändert. Die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **SizeWindow**.  
**Width:** Die neue Breite des Fensters.  
**Height:** Die neue Höhe des Fensters.  
**ID:** Identifikator des grössenveränderten Fensters.

**MoveWindow:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Benutzer das Fenster bewegt. Die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **MoveWindow**.  
**X,Y:** Die neue Position des Fensters auf dem Bildschirm.  
**ID:** Identifikator des verschobenen Fensters.

**CloseWindow:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Benutzer das Schließsymbol des Fensters drückt. Sie können dies verwenden, um ein Dialogfenster erscheinen zu lassen, das den Benutzer fragt, ob er wirklich beenden will. Nachrichtentext:

**Action:** Beinhaltet **CloseWindow**.  
**ID:** Identifikator des Fensters, bei welchem das Schließsymbol gedrückt wurde.

**ActiveWindow:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn das Hollywood-Fenster aktiv wird. Nachrichtentext:

**Action:** Beinhaltet **ActiveWindow**.  
**ID:** Identifikator des Fensters, das aktiviert wurde.

**InactiveWindow:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn das Hollywood-Fenster inaktiv wurde. Nachrichtentext:

**Action:** Beinhaltet **InactiveWindow**.  
**ID:** Identifikator des Fensters, welches nun gerade inaktiv wurde/den Focus verloren hat.

**ShowWindow:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Benutzer ein verstecktes (minimisiertes) Fenster wieder sichtbar auf den Bildschirm zurück holt. Nachrichtentext:

**Action:** Beinhaltet **ShowWindow**.

**ID:** Identifikator des Fensters, welches aus dem minimierten Modus gebracht wurde.

(V3.0)

**HideWindow:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Benutzer ein Hollywood-Fenster versteckt (minimisiert). Nachrichtentext:

**Action:** Beinhaltet HideWindow.

**ID:** Identifikator des Fensters, welches minimiert wurde.

(V3.0)

**ModeSwitch:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Benutzer mit der Tastenkombination CMD+RETURN (LALT+RETURN auf Windows) vom Fenster- in den Vollbildmodus und umgekehrt wechselt. Nachrichtentext:

**Action:** Beinhaltet ModeSwitch.

**Mode:** Beinhaltet den Modus, in den gerade gewechselt wurde. Kann entweder #DISPMODE\_WINDOWED für Fenstermodus oder #DISPMODE\_FULLSCREEN für den Vollbildmodus sein.

**ID:** Identifikator des Fensters, für das die Tastenkombination gedrückt wurde.

**Width:** Display Breite. (V6.0)

**Height:** Display Höhe. (V6.0)

(V4.5)

**OnDropFile:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Benutzer ein oder mehrere Piktogramme auf ein Display legt. Nachrichtentext:

**Action:** Beinhaltet OnDropFile.

**ID:** Identifikator des Fensters, über dem die Piktogramme fallen gelassen wurden.

**NumDropFiles:**

Die Anzahl der Dateien, die der Benutzer über dem Display fallen gelassen hat. Dies ist in der Regel 1.

**DropFiles:**

Diese Tabelle enthält die Liste der Dateien, die auf das Display fallen gelassen wurde. Diese Tabelle wird genau NumDropFiles Einträge enthalten.

**X,Y:** Enthält die relative Position zu der oberen linken Ecke des Displays, über den die Dateien fallen gelassen wurden.

(V4.5)

**ClipboardChange:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Inhalt der Zwischenablage geändert wurde. Dies ist nützlich, um in Ihrem Skript die Einfüge-Funktionalität zu aktivieren oder zu deaktivieren. Nachrichtentext:

**Action:** Beinhaltet **ClipboardChange**.

**ID:** Identifikator des Fensters, welches dieses Ereignis empfangen hat.

(V4.5)

**OnMouseDown:**

Die Funktion, die Sie hier angeben, wird jedes Mal aufgerufen werden, wenn der Benutzer die mittlere Maustaste drückt. An die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **OnMouseDown**.

**ID:** Wird auf die ID des Display gesetzt, welches die mittlere Maustaste empfangen hat.

(V4.5)

**OnMouseUp:**

Die Funktion, die Sie hier angeben, wird jedes Mal aufgerufen werden, wenn der Benutzer die mittlere Maustaste loslässt. An die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **OnMouseUp**.

**ID:** Wird auf die ID des Display gesetzt, welches die mittlere Maustaste empfangen hat.

(V4.5)

**OnConnect:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn sich ein neuer Client mit einem Server (erstellt mit **CreateServer()**) verbindet. Die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **OnConnect**.

**ClientID:**

Identifikator des Clienten, der sich mit dem Server verbunden hat.

**ServerID:**

Identifikator des Servers, mit dem sich der Client verbunden hat.

Die **ClientID** ist wichtig und man sollte sie irgendwo speichern, weil Sie sie brauchen werden, um mit dem Clienten zu kommunizieren. Sie können auch diese ID verwenden, um die IP-Adresse und Port-Nummer des Clienten herauszufinden. Dies ist mit den Befehlen **GetConnectionIP()** und **GetConnectionPort()** möglich. Sie können auch Daten an den Clienten senden, indem Sie den Befehl **SendData()** benutzen.

(V5.0)

**OnDisconnect:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn sich ein Client von einem Server (erstellt mit `CreateServer()`) trennt. Die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet `OnDisconnect`.

**ID:** Identifikator des Clienten, der vom Server getrennt wurde. Sie müssen `CloseConnection()` mit dieser ID des Clienten verwenden, um ihn vom Server zu entfernen.

Wenn Sie dieses Ereignis erhalten, vergessen Sie nicht, `CloseConnection()` mit der Client-ID zu verwenden, um den Client in vollem Umfang vom Server zu trennen. Dies ist sehr wichtig.

(V5.0)

**OnReceiveData:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn neue Daten über eine bestehende Verbindung empfangen werden. Die Funktion wird eine Nachricht als Parameter 1 mit folgenden Feldern übergeben:

**Action:** Beinhaltet `OnReceiveData`.

**ID:** Identifikator der Verbindung, welche die neuen Daten enthält.

Wenn Sie diese Meldung erhalten, bedeutet es, dass in der Verbindung von ID neuen Daten verfügbar sind und dass Sie mit dem Befehl `ReceiveData()` nun diese Daten aus dem Netzwerk-Puffer lesen können.

(V5.0)

**OnReceiveUDPData:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn neue Daten von einer vorhandenen UDP-Verbindung empfangen wurden. Die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet `OnReceiveUDPData`.

**ID:** Identifikator der UDP-Verbindung, welche neue Daten enthält.

Wenn Sie diese Meldung erhalten, bedeutet es, dass in der Verbindung von ID neuen Daten verfügbar sind und dass Sie mit dem Befehl `ReceiveUDPData()` nun diese Daten aus dem Netzwerk-Puffer lesen können.

(V5.0)

**OnVideoEnd:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn ein Video beendet wurde. An die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet `OnVideoEnd`.

**ID:** Identifikator des Videos, welches beendet wurde.

Bitte beachten Sie, dass dieses Ereignis nur dann ausgelöst wird, wenn das Video fertig abgespielt wurde. Es wird nicht ausgelöst, wenn Sie `StopVideo()` verwenden.

(V5.0)

#### **FillMusicBuffer:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Sound-Server mehr Audiodaten benötigt, wenn eine dynamische Musik (mit `CreateMusic()` erstellt) spielt. Ihre Callback-Funktion muss dann `FillMusicBuffer()` aufrufen, um das Gerät (Device) mit mehr Audiodaten zu versorgen. Nachrichtentext:

**Action:** Beinhaltet `FillMusicBuffer`.

**ID:** Identifikator der Music, welche dieses Ereignis ausgelöst hat.

**Samples:** Enthält die Anzahl der PCM-Daten, welche Hollywood von Ihrer Callback-Funktion anfordern wird. Ihre Callback-Funktion muss diese Menge an PCM-Daten an Hollywood übergeben, so dass es diese Audiodaten an das System des Audio-Gerät weiterleiten kann. Siehe [Abschnitt 47.7 \[FillMusicBuffer\]](#), [Seite 1026](#), für Details.

**Count:** Enthält einen globalen Zähler, wie viele PCM-Daten bereits an das Audio-Gerät gesendet wurde. Nützlich um herauszufinden, wie viele Sekunden die Musik bereits gespielt hat.

(V5.0)

#### **OrientationChange:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Benutzer sein Mobilphone gedreht hat. Nachrichtentext:

**Action:** Beinhaltet `OrientationChange`.

**ID:** Identifikator des Displays.

**Orientation:**

Konstante unter Angabe der neuen Ausrichtung des Geräts. Dies wird eine der folgenden Konstanten sein:

`#ORIENTATION_PORTRAIT`

`#ORIENTATION_LANDSCAPE`

`#ORIENTATION_PORTRAITREV`

`#ORIENTATION_LANDSCAPEREV`

Bitte beachten Sie, dass dieser Ereignis-Handler nur in der mobilen Version von Hollywood unterstützt wird.

(V5.0)

#### **ShowKeyboard:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn die Software-Tastatur auf mobilen Geräten sichtbar wird. Die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet `ShowKeyboard`.

**ID:** Identifikator des Displays.

Bitte beachten Sie, dass dieser Ereignis-Handler nur in der mobilen Version von Hollywood unterstützt wird.

(V5.0)

**HideKeyboard:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn die Software-Tastatur auf mobilen Geräten versteckt wird. Die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **HideKeyboard**.

**ID:** Identifikator des Displays.

Bitte beachten Sie, dass dieser Ereignis-Handler nur in der mobilen Version von Hollywood unterstützt wird.

(V5.0)

**OnUserMessage:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn eine neue Benutzernachricht beim Nachrichten-Port (mit **CreatePort()** erstellt) ankommt. Die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **OnUserMessage**.

**Command:** Eine Zeichenkette mit dem Befehl, der ausgeführt werden soll.

**Args:** Eine vorab von Hollywood analysierte Zeichenfolge, die alle Argumente (getrennt durch Null-Zeichen ("\0")) für diesen Befehl enthält. Sie können den Befehl **SplitStr()** verwenden, um die einzelnen Argumente zu erhalten.

**ArgC:** Die Anzahl der Argumente in der Zeichenkette **Args**.

**RawArgs:** Die nicht analysierte Zeichenfolge, die alle Argumente enthält.

Bitte beachten Sie, dass Hollywood nicht immer die Argumente in der Art und Weise trennt, wie Sie sie haben wollen. In diesem Fall können Sie das **RawArgs**-Feld verwenden, die für den Zugriff auf die Argumente in ihrem ursprünglichen Format verwendet werden kann. Die Felder **Args** und **ArgC** sind nur für Ihre Bequemlichkeit enthalten, aber einige erfahrene Benutzer könnten manchmal **RawArgs** bevorzugen.

Siehe **Abschnitt 31.1 [CreatePort]**, **Seite 643**, für ein Beispiel von diesem Ereignis-Handler.

(V5.0)

**Hotkey:** Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Benutzer die im Argument **-cxkey** definierte Tastenkombination drückt. Die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **Hotkey**.

**ID:** Identifikator des Displays.

Bitte beachten Sie, dass dieser Ereignis-Handler nur auf AmigaOS und Kompatiblen von Hollywood funktioniert.

(V5.2)

#### TrayIcon:

Wenn Sie mit dem Befehl **SetTrayIcon()** ein Piktogramm in der Taskleiste installieren, wird jedes Mal der Benutzer beim Klicken auf dieses Piktogramm diesen Ereignis-Handler aufgerufen. Nachrichtentext:

**Action:** Beinhaltet **TrayIcon**.

**ID:** Identifikator des Displays.

Bitte beachten Sie, dass dieser Ereignis-Handler nur in der Microsoft Windows-Version von Hollywood unterstützt wird.

(V5.2)

**OnTouch:** Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn eine Interaktion mit dem Benutzer über den Touchscreen stattfindet. Es wird erkannt, wie ein neuer Finger auf den Touchscreen gesetzt wird, ein Finger sich hebt oder bewegt. Dieses Ereignis wird nur erforderlich sein, wenn Sie eine fein abgestimmte Kontrolle über alle Berührungseignisse benötigen, wie z.B. Multi-Touch-Ereignisse oder Gesten. Wenn Sie nur eine einfache Kontrolle über den Touch-Screen benötigen, ist es leichter, die Ereignis-Handler **OnMouseDown**, **OnMouseUp** und **OnMouseMove** zu verwenden. Hollywood wird die primäre Finger auf dem Touchscreen immer der linken Maustaste zuordnen. Nachrichtentext:

**Action:** Beinhaltet **OnTouch**.

**ID:** Identifikator des Fensters, welches dieses Ereignis auslöst hat.

**Type:** In einer Zeichenkette wird der Typ des Touch-Ereignisses beschrieben. Dies wird entweder **Down**, **Up** oder **Move** sein.

**Finger:** Die Fingernummer, auf die sich dieses Ereignis bezieht.

**X:** Die aktuelle X-Koordinaten des Fingers auf dem Touchscreen.

**Y:** Die aktuelle Y-Koordinaten des Fingers auf dem Touchscreen.

**Pressure:** Enthält den aktuellen Druck des Fingers. Der allgemeinen Bereich geht von 0 (überhaupt kein Druck) bis 1 (Normaldruck). Werte größer als 1 sind ebenfalls möglich, ist aber abhängig von der Kalibrierung des Touchscreen.

**DownTime:** Enthält die Uhrzeit, wann die Finger auf dem Touchscreen gelegt wurden.

**EventTime:** Enthält die Uhrzeit, wann das Ereignis generiert wurde.

**Size:** Enthält den skalierten Wert der ungefähren Größe der aktuellen Finger.



**TouchMajor:**

Enthält die aktuelle Länge der Hauptachse einer Ellipse der Berührungsfläche für den aktuellen Finger.

**TouchMinor:**

Enthält die aktuelle Länge der Nebenachse einer Ellipse der Berührungsfläche für den aktuellen Finger.

**ToolMajor:**

Enthält die aktuelle Länge der Hauptachse einer Ellipse des Werkzeugbereichs für den aktuellen Finger.

**ToolMinor:**

Enthält die aktuelle Länge der Nebenachse einer Ellipse des Werkzeugbereichs für den aktuellen Finger.

**Orientation:**

Enthält die aktuelle Ausrichtung des Berührungs- und Werkzeugbereich in Radiant, welcher im Uhrzeigersinn von der Vertikalen für den aktuellen Finger läuft. Der Bereich ist von  $-\pi/2$  Radiant (Finger ist ganz links) und  $\pi/2$  Radiant (Finger ist ganz rechts).

Bitte beachten Sie, dass dieser Ereignis-Handler nur in der mobilen Version von Hollywood unterstützt wird.

(V5.3)

**OnApplicationMessage:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn eine neue Nachricht in AmigaOS 4 über die application.library Messaging-System eintrifft. Wenn Sie eine application.library Nachricht empfangen wollen, müssen Sie zuerst den Tag **RegisterApplication** in **@OPTIONS** auf **True** gesetzt haben. Nachrichtentext:

**Action:** Beinhaltet **OnApplicationMessage**.

**Sender:** Der Name der Anwendung, die diese Nachricht gesendet hat.

**Message:** Die aktuelle Nachricht.

Bitte beachten Sie, dass dieser Ereignis-Handler nur auf AmigaOS 4.x unterstützt wird. (V6.0)

**OnDockyClick:**

Wenn Sie den Tag **RegisterApplication** in der Präprozessor-Anweisung **@OPTIONS** auf **True** gesetzt haben, wird die hier angegebene Funktion jedes Mal aufgerufen, wenn der Benutzer auf das Piktogramm Ihrer Anwendung im AmiDock klickt. Nachrichtentext:

**Action:** Beinhaltet **OnDockyClick**.

**ID:** Identifikator des Displays.

Bitte beachten Sie, dass dieser Ereignis-Handler nur auf AmigaOS 4.x unterstützt wird. (V6.0)

**OnMenuSelect:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Benutzer einen Menüpunkt auswählt. Die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **OnMenuSelect**.

**ID:** Identifikator des Displays, welches die Menüleiste enthält.

**Item:** Die ID des Menüs, welches ausgewählt wurde. Siehe [Abschnitt 37.8 \[MENU\]](#), [Seite 754](#), für Details.

**Selected:**

Wenn der ausgewählte Menüpunkt ein Toggle-Menüpunkt (das heißt ein Menüpunkt, der zwei unterschiedliche Zustände haben kann) ist, wird dieses Feld den aktuellen Umschaltstatus enthalten. Siehe [Abschnitt 37.8 \[MENU\]](#), [Seite 754](#), für Details.

(V6.0)

**RunFinished:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn ein Programm asynchron mit dem Befehl **Run()** ausgeführt wurde und der Befehl nun beendet ist. Die Funktion wird eine Nachricht als Parameter 1 mit den folgenden Feldern übergeben:

**Action:** Beinhaltet **RunFinished**.

**Program:** Hier wird eine Zeichenkette mit dem Namen des Programmes sein, die mit dem Befehl **Run()** ausgeführt wurde.

**Args:** Diese Zeichenkette enthält die Argumente, die an das Programm mit dem Befehl **Run()** übergeben wurden.

**RunUserData:**

Wenn benutzerdefinierte Benutzerdaten in dem Aufruf von **Run()** angegeben wurden, wird es in diesem Nachrichtenfeld an Ihre Callback-Funktion übergeben werden. Wenn Sie keine benutzerdefinierten Benutzerdaten übergeben, wird dieses Feld nicht initialisiert werden.

**ReturnCode:**

Dieser Tag wird nur gesetzt, wenn das gleichnamige Tag beim Aufruf von **Run()** auf **True** gesetzt wurde. In diesem Fall enthält **ReturnCode** den Rückkehrcode des Programms, wenn es beendet wird. (V9.0)

(V6.1)

**DirectoryChanged:**

Wenn Sie derzeit Verzeichnisse mit dem Befehl **MonitorDirectory()** überwachen, wird die hier angegebene Funktion immer dann aufgerufen, wenn eine Änderung in einem überwachten Verzeichnis auftritt. Die Funktion erhält eine Nachricht als Parameter 1 mit den folgenden Feldern:

**Action:** Beinhaltet **DirectoryChanged**.

**ID:** Identifikator des Verzeichnisobjekts, in dem die Änderung aufgetreten ist.

**Directory:** Dies wird auf eine Zeichenkette festgelegt, die einen vollständig qualifizierten Pfad des Verzeichnisses enthält, in dem die Änderung aufgetreten ist.

**MonitorUserData:** Wenn beim Aufruf von `MonitorDirectory()` benutzerdefinierte Benutzerdaten angegeben wurden, werden diese in diesem Meldungsfeld an Ihre Callback-Funktion übergeben. Wenn Sie keine benutzerdefinierten Benutzerdaten übergeben, wird dieses Feld überhaupt nicht initialisiert.

**Type:** Die Art der Änderung. Dies wird nur gesetzt, wenn beim Aufruf von `MonitorDirectory()` der Tag `ReportChanges` auf `True` gesetzt wurde. In diesem Fall handelt es sich um einen der folgenden Typen:

**#DIRMONITOR\_ADD:**

Die Datei oder das Verzeichnis im Tag `Name` wurde dem Verzeichnis hinzugefügt.

**#DIRMONITOR\_REMOVE:**

Die Datei oder das Verzeichnis im Tag `Name` wurde aus dem Verzeichnis entfernt.

**#DIRMONITOR\_CHANGE:**

Die Datei oder das Verzeichnis im Tag `Name` wurde im Verzeichnis geändert.

(V9.0)

**Name:** Dieser enthält den Namen der Datei oder des Verzeichnisses, die je nach Wert im Tag `Type` (siehe oben) hinzugefügt, entfernt oder geändert wurde. Beachten Sie, dass `Name` nur gesetzt wird, wenn der Tag `ReportChanges` beim Aufruf von `MonitorDirectory()` auf `True` gesetzt wurde. (V9.0)

(V8.0)

**OnAccelerometer:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Sensor des Geräts neue Werte für den Beschleunigungssensor meldet. Beachten Sie, dass dies normalerweise immer geschieht. Wenn Sie dieses Ereignis überwachen, werden Sie viele Ereignisse erhalten, die sich auf die Leistung Ihres Skripts auswirken können. Die Funktion erhält eine Nachricht als Parameter 1 mit folgenden Feldern:

**Action:** Beinhaltet `OnAccelerometer`.

**ID:** Wird auf den Identifikator des Displays gesetzt, das dieses Ereignis empfangen hat.

**X:** X-Beschleunigungssensorwert vom Sensor.  
**Y:** Y-Beschleunigungssensorwert vom Sensor.  
**Z:** Z-Beschleunigungssensorwert vom Sensor.

Bitte beachten Sie, dass dieser Ereignis-Handler nur in der Android-Version von Hollywood unterstützt wird. Informationen zum Interpretieren der von diesem Ereignis-Handler bereitgestellten Werte für X, Y und Z finden Sie in der Android-Dokumentation vom **SensorEvent**.

(V8.0)

#### **OnGyroscope:**

Die hier angegebene Funktion wird jedes Mal aufgerufen, wenn der Sensor des Geräts neue Gyroskopwerte meldet. Beachten Sie, dass dies normalerweise immer geschieht. Wenn Sie dieses Ereignis überwachen, werden Sie viele Ereignisse erhalten, die sich auf die Leistung Ihres Skripts auswirken können. Die Funktion erhält eine Nachricht als Parameter 1 mit folgenden Feldern:

**Action:** Beinhaltet **OnGyroscope**.  
**ID:** Wird auf den Identifikator des Displays gesetzt, das dieses Ereignis empfangen hat.  
**X:** X-Gyroskopwert vom Sensor.  
**Y:** Y-Gyroskopwert vom Sensor.  
**Z:** Z-Gyroskopwert vom Sensor.

Bitte beachten Sie, dass dieser Ereignis-Handler nur in der Android-Version von Hollywood unterstützt wird. Informationen zum Interpretieren der von diesem Ereignis-Handler bereitgestellten Werte für X, Y und Z finden Sie in der Android-Dokumentation vom **SensorEvent**.

(V8.0)

#### **RunOutput:**

Die hier angegebene Funktion wird immer dann aufgerufen, wenn ein asynchron ausgeführtes Programm mit dem Befehl **Run()** Daten in die Konsole schreibt. Diese Daten werden an Ihr Programm umgeleitet und können mit diesem Ereignis-Handler verarbeitet werden. Dadurch ist es möglich, die Ausgabe eines Programms zu erfassen. Die Funktion erhält als Parameter 1 eine Nachricht mit den folgenden initialisierten Feldern:

**Action:** Beinhaltet **RunOutput**.  
**Program:** Dies wird auf eine Zeichenkette gesetzt, die den Namen des Programms enthält, welches mit **Run()** gestartet wurde.  
**Args:** Dies wird auf eine Zeichenkette gesetzt, die die Argumente enthält, die an das mit **Run()** gestartete Programm übergeben wurden.  
**Output:** Dies wird auf eine Zeichenkette gesetzt, die die Ausgabe des Programms enthält. Beachten Sie, dass diese Zeichenkette eine beliebige Länge haben kann. Gehen Sie nicht davon aus,

dass `Output` immer eine vollständige Zeile der Programmausgabe ist, es kann auch eine halbe Zeile sein, wobei die andere Hälfte beim nächsten Auslösen des Ereignis-Handlers geliefert wird. Abhängig von der Art und Weise, wie das Programm die Ausgabe an die Konsole schreibt, kann es Ihnen sogar Zeichen für Zeichen übergeben. Also machen Sie keine Annahmen über das tatsächliche Format von `Output`. Die in `Output` übergebene Zeichenkette hat standardmäßig das UTF-8-Format. Sehen Sie unten, wie dies geändert werden kann.

#### **RunUserData:**

Wenn beim Aufruf von `Run()` benutzerdefinierte Benutzerdaten angegeben werden, werden diese in diesem Nachrichtefeld an Ihre Callback-Funktion weitergegeben. Wenn Sie keine benutzerdefinierten Benutzerdaten übergeben, wird dieses Feld überhaupt nicht initialisiert.

Beachten Sie, dass der Ereignis-Handler `RunOutput` standardmäßig erwartet, dass Programme nur Text ausgeben. Aus diesem Grund stellt `RunOutput` sicher, dass nur korrekt UTF-8-codierter Text an Ihre Callback-Funktion übergeben wird. Wenn `RunOutput` den Text nicht als UTF-8 formatieren soll, müssen Sie beim Aufruf von `Run()` das Argument `RawMode` auf `True` setzen. In diesem Fall führt `RunOutput` keine Vorformatierung durch und leitet nur die Rohausgabe des Programms an Sie weiter. Das bedeutet, dass Ihre Callback-Funktion bereit sein muss, auch Binärdaten zu verarbeiten.

Beachten Sie auch, dass die Ausgabe normalerweise nicht in Echtzeit erfolgt, da die Konsolenausgabe normalerweise gepuffert wird. Wenn das externe Programm jedoch ständig Text ausgibt, wird dieser ziemlich sofort bei Ihrem `RunOutput-Callback-Funktion` ankommen.

(V9.0)

#### **ShowSystemBars:**

Die hier angegebene Callback-Funktion wird jedes Mal aufgerufen, wenn die Systemleisten wieder sichtbar werden, wenn sich ein Display im Immersive-Modus befindet. Die Callback-Funktion erhält als Parameter 1 eine Nachricht mit folgenden Feldern:

**Action:** Beinhaltet `ShowSystemBars`.

**ID:** Wird auf den Identifikator des Displays gesetzt, das dieses Ereignis empfangen hat.

Bitte beachten Sie, dass dieser Ereignis-Handler nur in der Android-Version von Hollywood unterstützt wird. (V9.0)

#### **HideSystemBars:**

Die hier angegebene Callback-Funktion wird jedes Mal aufgerufen, wenn die Systemleisten ausgeblendet werden, wenn sich ein Display im immersiven Modus befindet. Die Callback-Funktion erhält als Parameter 1 eine Nachricht mit folgenden Feldern:

**Action:** Beinhaltet `HideSystemBars`.

**ID:** Wird auf den Identifikator des Displays gesetzt, das dieses Ereignis empfangen hat.

Bitte beachten Sie, dass dieser Ereignis-Handler nur in der Android-Version von Hollywood unterstützt wird. (V9.0)

Wenn Sie einen Ereignis-Handler entfernen möchten, müssen Sie einfach das entsprechende Feld in der Tabelle auf 0 setzen, anstatt eine Funktion zu übergeben.

Ab Hollywood 3.1 gibt es ein optionales Argument namens **userdata** (Benutzerdaten). Den hier angegebenen Wert wird der Callback-Funktion übergeben, wenn sie aufgerufen wird. Dies ist nützlich, wenn Sie die Arbeit mit globalen Variablen vermeiden möchten. Mit Hilfe des Argument **userdata** können Sie ganz einfach Daten an Ihrer Callback-Funktion übergeben. Sie können einen Wert eines beliebigen Typs (Zahlen, Zeichenketten, Tabellen, Funktionen) verwenden.

Ab Hollywood 7.0 werden alle Ereignismeldungen ein zusätzliches Feld mit dem Namen **Timestamp** enthalten. Dieses Feld enthält den Zeitstempel, wann das Ereignis erzeugt wurde. Dieser Zeitstempel wird in Sekunden als Bruchzahl übergeben. Er ist relativ zu der Zeit, in der Hollywood gestartet wurde. Siehe [Abschnitt 21.8 \[GetTimestamp\]](#), [Seite 326](#), für Details.

Bitte beachten Sie, dass die Ereignis-Handler-Funktionen nur dann ausgeführt werden, während das Skript in einer **WaitEvent()** Schleife ist. Darum müssen Sie **WaitEvent()** für diese Callback-Funktionen verwenden!

## EINGABEN

<b>table</b>	Tabelle, welche Informationen über die Handler installieren oder entfernen enthält
<b>userdata</b>	optional: Benutzerdefinierte Daten, welche der Callback-Funktion übergeben werden (V3.1)

## BEISPIEL

```
Function p_HandlerFunc(msg)
  Switch(msg.action)
  Case "ActiveWindow":
    DebugPrint("Window has become active again!")
  Case "InactiveWindow":
    DebugPrint("Window has become inactive!")
  Case "MoveWindow":
    DebugPrint("User has moved the window to", msg.x, msg.y)
  Case "OnKeyDown":
    If msg.key = "ESC" Then End
  EndSwitch
EndFunction
```

```
InstallEventHandler({ActiveWindow = p_HandlerFunc,
  InactiveWindow = p_HandlerFunc,
  MoveWindow = p_HandlerFunc,
  OnKeyDown = p_HandlerFunc})
```

```
Repeat
  WaitEvent
Forever
```

Der obige Code installiert vier Ereignis-Handler für `ActiveWindow`, `InactiveWindow`, `MoveWindow` und `OnKeyDown`. Drückt der Benutzer die ESC-Taste, wird das Programm beendet. Wenn Sie zum Beispiel den Ereignis-Handler `MoveWindow` entfernen möchten, rufen Sie einfach `InstallEventHandler()` erneut mit dem Parameter `MoveWindow=0` auf.

## 26.14 IsKeyDown

### BEZEICHNUNG

`IsKeyDown` – prüft, ob eine Taste gedrückt ist (V1.5)

### ÜBERSICHT

```
state = IsKeyDown(key$[, rawkey])
```

### BESCHREIBUNG

Dieser Befehl prüft, ob die durch `key$` angegebene Taste momentan gedrückt ist. Wenn ja, gibt dieser Befehl `True` (WAHR) zurück, andernfalls `False` (FALSCH).

`key$` ist eine Zeichenfolge einer Taste auf der Tastatur. Dies kann eine der folgenden Steuertasten sein:

UP	Pfeil nach oben
DOWN	Pfeil nach unten
RIGHT	Pfeil nach rechts
LEFT	Pfeil nach links
HELP	Help-Taste
DEL	Delete-Taste
BACKSPACE	Rückschritt-Taste
TAB	Tabulator-Taste
RETURN	Return-Taste
ENTER	Enter-Taste
ESC	Escape-Taste
SPACE	Leertaste
F1 – F16	Funktionstasten
INSERT	Einfüge-Taste
HOME	Anfangs-Taste
END	Ende-Taste

PAGEUP	Seite rauf
PAGEDOWN	Seite runter
PRINT	Drucken-Taste
PAUSE	Pause-Taste

Alternativ kann `key$` auch ein Zeichen aus dem englischen Alphabet, z.B. "A" oder eine Zeichenfolge mit einer Zahl von 0 bis 9 sein. Beachten Sie, dass `IsKeyDown()` keine Unicode-Zeichen unterstützt.

Ab Hollywood 4.0 können Sie auch den Status der Umschalttasten überprüfen. Die folgenden Zusatztasten können mit `IsKeyDown()` überprüft werden:

LSHIFT	Linke Shift-Taste
RSHIFT	Rechte Shift-Taste
LALT	Linke Alt-Taste
RALT	Rechte Alt-Taste
LCOMMAND	Linke Steuer-Taste (z.B. Windows Start-Taste, linke Amiga-Taste)
RCOMMAND	Rechte Steuer-Taste (z.B. Windows Menü-Taste, rechte Amiga-Taste)
LCONTROL	Linke Control-Taste
RCONTROL	Rechte Control-Taste

Ab Hollywood 6.1 können Sie die spezielle Zeichenfolge `ANY` in `key$` überprüfen, ob eine beliebige Taste gedrückt wurde.

Ab Hollywood 7.1 gibt es das optionale Argument `rawkey`. Wenn dieses Argument auf `True` gesetzt ist, wird `IsKeyDown()` `key$` als rohe Taste behandeln und prüfen, ob sie gedrückt ist. In diesem Fall muss `key$` einer der von Hollywood definierten rohen Tasten sein. Siehe [Abschnitt 26.23 \[Rohe Tasten\]](#), [Seite 512](#), für Details. Der Unterschied zwischen normalen und rohen Tasten ist in der Dokumentation des Ereignis-Handlers `OnRawKeyDown` beschrieben. Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), [Seite 483](#), für Details.

Siehe auch [IsLeftMouse\(\)](#), [IsMidMouse\(\)](#) und [IsRightMouse\(\)](#).

## EINGABEN

<code>key\$</code>	zu überwachende Taste
<code>rawkey</code>	optional: <code>True</code> , wenn <code>key\$</code> als rohen Taste behandelt werden soll (voreingestellt ist <code>False</code> ) (V7.1)

## RÜCKGABEWERTE

<code>state</code>	<code>True</code> , wenn die Taste gedrückt ist, andernfalls <code>False</code>
--------------------	---

## BEISPIEL

```
Print("Press F1 please.")
Repeat
  VWait
Until IsKeyDown("F1") = True
```



Dieses Programmstück wartet, bis Taste F1 gedrückt wird. (Sie können das einfacher haben, wenn Sie `WaitKeyDown()` benutzen; dieses Beispiel hier ist nur sinnvoll, wenn Sie etwas tun wollen solange die Taste nicht gedrückt ist.)

## 26.15 IsLeftMouse

### BEZEICHNUNG

IsLeftMouse – prüft, ob die linke Maustaste gedrückt ist

### ÜBERSICHT

```
pressed = IsLeftMouse()
```

### BESCHREIBUNG

Dieser Befehl gibt den Wert `True` (WAHR) zurück, wenn momentan die linke Maustaste gedrückt ist, andernfalls `False` (FALSCH).

Siehe auch `IsKeyDown()`, `IsMidMouse()` und `IsRightMouse()`.

### EINGABEN

Keine

### RÜCKGABEWERTE

`pressed`    `True` für gedrückt, `False` für nicht gedrückt

### BEISPIEL

```
Repeat
  Wait(2)
Until IsLeftMouse() = True
```

Obiger Code wartet bis die linke Maustaste gedrückt wird. (Sie können das auch einfacher mit `WaitLeftMouse()` haben; dieses Beispiel hier ist nur sinnvoll, wenn Sie etwas tun möchten, solange die Maustaste nicht gedrückt ist).

## 26.16 IsMidMouse

### BEZEICHNUNG

IsMidMouse – prüft, ob die mittlere Maustaste gedrückt ist (V4.5)

### ÜBERSICHT

```
pressed = IsMidMouse()
```

### BESCHREIBUNG

Dieser Befehl gibt den Wert `True` (WAHR) zurück, wenn momentan die mittlere Maustaste gedrückt ist, andernfalls `False` (FALSCH).

Siehe auch `IsKeyDown()`, `IsLeftMouse()` und `IsRightMouse()`.

### EINGABEN

Keine

### RÜCKGABEWERTE

`pressed`    `True` für gedrückt, `False` für nicht gedrückt

**BEISPIEL**

```
Repeat
  Wait(1)
Until IsMidMouse() = True
```

Obiger Code wartet bis die mittlere Maustaste gedrückt wird. (Sie können das auch einfacher mit `WaitMidMouse()` haben; dieses Beispiel hier ist nur sinnvoll, wenn Sie etwas tun möchten, solange die Maustaste nicht gedrückt ist).

## 26.17 IsRightMouse

**BEZEICHNUNG**

IsRightMouse – prüft, ob die rechte Maustaste gedrückt ist (V1.5)

**ÜBERSICHT**

```
pressed = IsRightMouse()
```

**BESCHREIBUNG**

Dieser Befehl gibt den Wert `True` (WAHR) zurück, wenn momentan die rechte Maustaste gedrückt ist, andernfalls `False` (FALSCH).

Siehe auch `IsKeyDown()`, `IsLeftMouse()` und `IsMidMouse()`.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`pressed`    `True` für gedrückt, `False` für nicht gedrückt

**BEISPIEL**

```
Repeat
  Wait(2)
Until IsRightMouse() = True
```

Dieser Code wartet, bis die rechte Maustaste gedrückt wird. (Sie können das auch einfacher mit `WaitRightMouse()` haben; dieses Beispiel hier ist nur sinnvoll, wenn Sie etwas tun möchten, solange die Maustaste nicht gedrückt ist.)

## 26.18 LeftMouseQuit

**BEZEICHNUNG**

LeftMouseQuit – erlaubt/verbietet das Beenden mit der linken Maustaste

**ÜBERSICHT**

```
LeftMouseQuit(enable)
```

**BESCHREIBUNG**

Wenn Sie `enable` auf `True` setzen, wird Ihr Skript beendet, sobald der Benutzer die linke Maustaste drückt.

Siehe auch `CtrlCQuit()` und `EscapeQuit()`.

**EINGABEN**

**enable**      **True**, um beenden mit der linken Maustaste zu erlauben, **False**, um es zu verbieten

**BEISPIEL**

```
LeftMouseQuit(TRUE)
Repeat
  Wait(10)
Forever
```

Obiger Codeteil läuft in einer Endlosschleife, die normalerweise Ihr Programm blockieren würde. Die Benutzung von `LeftMouseQuit(TRUE)` erlaubt dem Benutzer jedoch, sie abzubrechen.

## 26.19 MakeButton

**BEZEICHNUNG**

`MakeButton` – erstellt eine neue Schaltfläche (V2.0)

**ÜBERSICHT**

```
[id] = MakeButton(id, #LAYERBUTTON, layerid, t[, userdata])
[id] = MakeButton(id, #SIMPLEBUTTON, x, y, width, height, t[, userdata])
```

**FRÜHERE SYNTAX**

```
[id] = MakeButton(id, #LAYERBUTTON, layerid, exactcoll, noautohide, t
[, userdata]) (V2.5)
```

**BESCHREIBUNG**

Dieser Befehl erstellt eine neue Schaltfläche und fügt sie an das aktuelle BGPic an. Die Schaltfläche erhält den durch `id` angegebener Identifikator, oder wenn Sie `Nil` als `id` übergeben, wählt `MakeButton()` automatisch einen Identifikator für Sie aus. Das zweite Argument gibt den Typ der Schaltfläche an. Derzeit werden die folgenden Typen unterstützt:

**#SIMPLEBUTTON:**

Erstellt eine Standard-Schaltfläche. Beachten Sie, dass diese Schaltfläche nicht sichtbar ist. Sie müssen Grafiken in Ihrem BGPic verwenden, um den Benutzer zu zeigen, wo sich Ihre Schaltfläche befindet. Der Typ `#SIMPLEBUTTON` erfordert, dass Sie die Position und die Dimensionen für die Schaltfläche in Argument 3 bis 6 angeben. Das siebte Argument ist ein Tabellenargument, mit dem Sie die Callback-Funktionen angeben können, die aufgerufen werden sollen, wenn die Schaltfläche ein Ereignis empfängt (siehe unten).

**#LAYERBUTTON:**

Dieser Typ ist neu in Hollywood 2.5. Es wird eine dynamische Schaltfläche erstellt, die immer die Position und Größe der durch `layerid` angegebenen Ebene hat. Beachten Sie, dass Ebenenschaltflächen eng mit ihrer Ebene verbunden sind. Wenn die Ebene gelöscht wird, wird also auch die Schaltfläche gelöscht. Ebenenschaltflächen unterstützen auch einige zusätzliche Optionen

im Tabellenargument, das von `MakeButton()` akzeptiert wird (siehe unten für Details).

Das Argument `t` muss eine Tabelle sein, die die Funktionen angibt, die aufgerufen werden sollen, wenn ein bestimmtes Ereignis eintritt. Es kann auch verwendet werden, um einige erweiterte Schaltflächenoptionen zu konfigurieren. Die Tabelle kann folgende Felder enthalten:

**OnMouseOver:**

Die hier angegebene Funktion wird aufgerufen, wenn der Benutzer die Maus über den Bereich der Schaltfläche bewegt.

**OnMouseOut:**

Die hier angegebene Funktion wird aufgerufen, wenn der Mauszeiger den Bereich der Schaltfläche verlässt.

**OnMouseDown:**

Die hier angegebene Funktion wird aufgerufen, wenn der Benutzer die linke Maustaste drückt, während sich der Mauszeiger über der Schaltfläche befindet.

**OnMouseUp:**

Die hier angegebene Funktion wird aufgerufen, wenn der Benutzer die linke Maustaste loslässt, während sich der Mauszeiger über der Schaltfläche befindet. Dieses Ereignis wird nur ausgelöst, wenn der Benutzer auch die linke Maustaste gedrückt hat, während sich der Mauszeiger über der Schaltfläche befand.

**OnRightMouseDown:**

Dasselbe wie `OnMouseDown`, aber mit der rechten Maustaste.

**OnRightMouseUp:**

Dasselbe wie `OnMouseUp`, aber mit der rechten Maustaste.

**OnMidMouseDown:**

Dasselbe wie `OnMouseDown`, aber mit der mittleren Maustaste. (V4.5)

**OnMidMouseUp:**

Dasselbe wie `OnMouseUp`, aber mit der mittleren Maustaste. (V4.5)

**PixelExact:**

Dies wird nur für `#LAYERBUTTON` unterstützt. Es gibt an, ob Ihre Schaltfläche pixelgenaue oder rechteckige Kollisionserkennung verwenden soll oder nicht. Wenn Sie hier `True` übergeben, werden Ereignisse nur bei einer pixelgenauen Kollision ausgelöst. Dies kann bei unregelmäßig geformten Schaltflächen nützlich sein. Die Voreinstellung ist `False`.

**NoAutoHide:**

Dies wird nur für `#LAYERBUTTON` unterstützt. Sie legt fest, ob die Schaltfläche automatisch ausgeblendet werden soll, wenn die Ebene ausgeblendet wird. Wenn Sie hier `True` angeben, wird die Schaltfläche nicht automatisch ausgeblendet, wenn Sie die Ebene ausblenden, mit der sie verbunden ist. Wenn `NoAutoHide False` ist, verschwindet die Schaltfläche, sobald Sie die Ebene ausblenden. Die Voreinstellung ist `False`.

**ZOrder:** Dies wird nur für **#LAYERBUTTON** unterstützt. Es gibt an, ob die Z-Reihenfolge der Ebenen bei der Behandlung von Ereignissen überlappender Ebenenschaltflächen eingehalten werden soll oder nicht. In der Vergangenheit wurden Ereignisse von sich überlappenden Ebenenschaltflächen in der Reihenfolge ihrer Erstellung behandelt, d.h. wenn eine Ebenenschaltfläche vor einer anderen Ebenenschaltfläche erstellt wurde und sich beide Ebenen überlappten, erhielt die zuvor erstellte Ebenenschaltfläche die Ereignisse auch dann, wenn sie sich unter der anderen Ebenenschaltfläche befand. Indem Sie den Tag **ZOrder** auf **True** setzen, können Sie Hollywood zwingen, Ereignisse von überlappenden Ebenenschaltflächen in ihrer Stapelreihenfolge zu behandeln, d.h. Schaltflächen, die über einer anderen Ebene liegen, erhalten ihre Ereignisse zuerst. Aus Kompatibilitätsgründen wird dieser Tag standardmäßig auf **False** gesetzt. (V9.0)

Wenn Sie nur auf einen Mausklick auf einer Schaltfläche reagieren wollen, reicht es eine Callback-Funktion für den Ereignistyp **OnMouseUp** zur Verfügung zu stellen. **OnMouseDown** ist nur erforderlich, wenn Sie die Schaltfläche in irgendeiner Weise hervorgehoben werden soll, während der Benutzer darauf klickt.

Ab Hollywood 3.1 gibt es ein optionales Argument namens **userdata**. Der Wert, den Sie hier angeben, wird der Callback-Funktion übergeben, wenn sie aufgerufen wird. Dies ist nützlich, wenn Sie die Arbeit mit globalen Variablen vermeiden. Mit Hilfe des **userdata** Argument können Sie ganz einfach Daten an Ihre Callback-Funktion übergeben. Sie können einen Wert eines beliebigen Typs wie Zahlen, Zeichenketten, Tabellen und sogar Funktionen angeben.

Die Callback-Funktionen, die Sie in der Ereignistabelle angegeben, wird von Hollywood mit einem Parameter aufgerufen. Dieser Parameter ist eine Nachrichten-Tabelle, die einige Informationen über das Ereignis beinhaltet. Folgende Felder stehen zur Verfügung:

**Action:** Enthält den Namen des ausgelösten Ereignisses wie z.B. **OnMouseUp**, **OnMouseOver** oder **OnMouseOut**. Dieses Feld ist eine Zeichenkette!

**ID:** Enthält den Identifikator der Schaltfläche, die dieses Ereignis ausgelöst hat. Dieses Feld ist eine Nummer!

**X, Y, Width, Height:**

Diese Felder tragen die Abmessungen der Schaltfläche, die dieses Ereignis ausgelöst hat. **Width** beinhaltet die Breite und **Height** die Höhe.

**MouseDown:**

Dieses Feld wird auf **True** gesetzt, wenn die linke Maustaste gerade gedrückt wird. Nützlich in Verbindung mit dem Ereignis **OnMouseOver**, so dass Sie ein hervorgehobene Version der Schaltfläche anzeigen können, wenn der Benutzer den Mauszeiger über die Schaltfläche bewegt, während die linke Maustaste gedrückt wird.

**RightMouseDown:**

Dasselbe wie **MouseDown**, aber mit der rechten Maustaste.

**MidMouseDown:**

Dasselbe wie **MouseDown**, aber mit der mittleren Maustaste. (V4.5)

**Layer:** Der Identifikator der Ebene, an die diese Schaltfläche gebunden ist. Dies ist nur für Schaltflächen vom Typ **#LAYERBUTTON** gesetzt. (V4.5)

**LayerName:** Der Name der Ebene, an die diese Schaltfläche gebunden ist. Dies ist nur für Schaltflächen vom Typ **#LAYERBUTTON** gesetzt. (V4.5)

**UserData:** Enthält den Wert, den Sie dem Argument **userdata** beim Erstellen der Schaltfläche übergeben haben.

**Timestamp:** Enthält eine Zeitangabe, die angibt, wann das Ereignis aufgetreten ist. Siehe [Abschnitt 21.8 \[GetTimestamp\]](#), [Seite 326](#), für Details. (V7.0)

Der Vorteil dieser Nachricht ist, dass Sie ein und die selbe Funktion für alle Schaltflächen und Ereignisse nutzen können. Mit den Feldern **Action** und **ID** finden Sie heraus, welche Schaltfläche das Ereignis ausgelöst hat.

Bitte beachten Sie: Die Schaltflächen sind immer an BGPics gebunden. Also wenn Sie **MakeButton()** aufrufen, während BGPic 1 auf dem Display angezeigt wird, wird die Schaltfläche an BGPic 1 gebunden. Wenn Sie BGPic 2 anzeigen, dann werden die Schaltflächen entfernt. Sobald Sie BGPic 1 wieder zurück auf das Display holen, werden die Schaltflächen wieder erscheinen.

Beachten Sie auch, dass Sie nur rechteckig geformte Schaltflächen mit **#SIMPLEBUTTON** erstellen können. Wenn Sie unregelmäßig geformte Schaltflächen haben möchten, erstellen Sie eine Ebene mit einer Transparenzeinstellung (maskierte oder alphakanalisierte Transparenz) und verwenden Sie dann **#LAYERBUTTON**, um eine Schaltfläche an diese Ebene zu binden.

Weitere Schaltflächenbefehle: [DeleteButton\(\)](#), [DisableButton\(\)](#) und [EnableButton\(\)](#).

## EINGABEN

<b>id</b>	Identifikator von der neuen Schaltfläche oder <b>Nil</b> für die <a href="#">automatische ID-Auswahl</a>
<b>type</b>	Typ der neuen Schaltfläche
<b>x</b>	x-Position der neuen Schaltfläche
<b>y</b>	y-Position der neuen Schaltfläche
<b>width</b>	Breite der neuen Schaltfläche
<b>height</b>	Höhe der neuen Schaltfläche
<b>t</b>	Tabelle, die Callback-Funktionen enthält, die Hollywood aufrufen wird, wenn ein bestimmtes Ereignis eintritt und zusätzliche Argumente (siehe oben)
<b>userdata</b>	optional: Benutzerspezifische Daten, die an die Callback-Funktion übergeben werden (V3.1)

## RÜCKGABEWERTE

<b>id</b>	optional: Identifikator der Schaltfläche; Wird nur zurückgegeben werden, wenn Sie <b>Nil</b> als Argument 1 angegeben haben (siehe oben)
-----------	--

**BEISPIEL**

```

Function p_MyFunc(msg)
  Switch msg.action
  Case "OnMouseUp":
    DebugPrint("User left-clicked button", msg.id)
  Case "OnMouseOver":
    DebugPrint("User moved mouse over button", msg.id)
  Case "OnRightMouseUp":
    DebugPrint("User right-clicked button", msg.id)
  EndSwitch
EndFunction

; zeichnet die Schaltfläche
Box(0, 0, 100, 100, #RED)
Box(200, 200, 100, 100, #BLUE)

; erstellt sie
evtmach = {OnMouseUp = p_MyFunc, OnMouseOver = p_MyFunc,
          OnRightMouseUp = p_MyFunc}
MakeButton(1, #SIMPLEBUTTON, 0, 0, 100, 100, evtmach)
MakeButton(2, #SIMPLEBUTTON, 200, 200, 100, 100, evtmach)

Repeat
  WaitEvent
Forever

```

**26.20 MouseX****BEZEICHNUNG**

MouseX – gibt die X-Position der Maus zurück (V1.5)

**ÜBERSICHT**

```
pos = MouseX()
```

**BESCHREIBUNG**

Dieser Befehl gibt die aktuelle X-Position des Mauszeigers zurück.

Siehe auch **MouseY()**.

**EINGABEN**

keine

**RÜCKGABEWERTE**

pos            X-Position des Mauszeigers

## 26.21 MouseY

### BEZEICHNUNG

MouseY – gibt die Y-Position der Maus zurück (V1.5)

### ÜBERSICHT

```
pos = MouseY()
```

### BESCHREIBUNG

Dieser Befehl gibt die aktuelle Y-Position des Mauszeigers zurück.

Siehe auch **MouseX()**.

### EINGABEN

keine

### RÜCKGABEWERTE

pos            Y-Position des Mauszeigers

## 26.22 ResetKeyStates

### BEZEICHNUNG

ResetKeyStates – setzt den internen Tastatur- und Mausstatus zurück (V4.6)

### ÜBERSICHT

```
ResetKeyStates()
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, Hollywoods interne Tastatur- und Mauszustände zurück zu setzen. Dies ist ein Lowlevel-Befehl und Sie werden ihn in der Regel nicht verwenden müssen. Er ist für bestimmte Notsituationen vorhanden.

Bitte beachten Sie, dass die Tastaturzustände pro Display zwischengespeichert werden. **ResetKeyStates()** wird die wichtigsten Zustände des aktuell aktiven Displays zurücksetzen.

### EINGABEN

Keine

## 26.23 Rohe Tasten (Raw Keys)

Ab Version 7.1 unterstützt Hollywood rohe Tastenereignisse. Die folgenden rohen Tasten werden derzeit von Hollywood unterstützt:

A-Z	Alphabetische Tasten. Beachten Sie, dass diese immer als Großbuchstaben geschrieben werden, da es sich um rohe Tasten handelt und keine Umschalttasten auf rohe Tasten angewendet werden.
0-9	Nummerntasten
UP	Pfeil nach oben
DOWN	Pfeil nach unten



RIGHT	Pfeil nach rechts
LEFT	Pfeil nach links
HELP	Help-Taste
DEL	Delete-Taste
BACKSPACE	Rückschritt-Taste
TAB	Tabulator-Taste
RETURN	Return-Taste
ENTER	Enter-Taste
ESC	Escape-Taste
SPACE	Leertaste
F1-F16	Funktionstasten
INSERT	Einfüge-Taste
HOME	Anfangs-Taste
END	Ende-Taste
PAGEUP	Seite rauf
PAGEDOWN	Seite runter
PRINT	Drucken-Taste
PAUSE	Pause-Taste
NP0-NP9	Ziffernblock-Numerische-Tasten
NPDEC	Ziffernblock-Dezimal-Taste (Punkt)
NPADD	Ziffernblock-Additions-Taste (+)
NPSUB	Ziffernblock-Subtraktion-Taste (-)
NPMUL	Ziffernblock-Multiplikations-Taste (*)
NPDIV	Ziffernblock-Divisions-Taste (/)
LSHIFT	Linke Shift-Taste
RSHIFT	Rechte Shift-Taste
LALT	Linke Alt-Taste
RALT	Rechte Alt-Taste
LCOMMAND	Linke Steuer-Taste (z.B. Windows Start-Taste, linke Amiga-Taste)
RCOMMAND	Rechte Steuer-Taste (z.B. Windows Menü-Taste, rechte Amiga-Taste)
LCONTROL	Linke Control-Taste
RCONTROL	Rechte Control-Taste

## 26.24 RunCallback

### BEZEICHNUNG

RunCallback – schiebt eine Callback-Funktion in die Ereigniswarteschlange (V9.0)

### ÜBERSICHT

RunCallback(func[, userdata])

### BESCHREIBUNG

Dieser Befehl fügt die in **func** angegebene Callback-Funktion der Ereigniswarteschlange hinzu. Die Funktion wird ausgeführt, wenn Hollywood das nächste Mal die Ereigniswarteschlange überprüft, d.h. wenn **CheckEvent()** oder **WaitEvent()** aufgerufen wird.

Mit dem optionalen Argument **userdata** können Sie zusätzliche Daten angeben, die an Ihre Callback-Funktion übergeben werden, wenn diese aufgerufen wird. Dies ist nützlich, wenn Sie globale Variablen vermeiden möchten. Sie können in **userdata** einen Wert eines beliebigen Typs angeben: Zahlen, Zeichenketten, Tabellen und sogar Funktionen können als Benutzerdaten übergeben werden.

Ihre Callback-Funktion wird von Hollywood mit einem einzigen Parameter aufgerufen. Der Parameter ist eine Nachrichtentabelle, die die folgenden Felder enthält:

**Action:** Wird immer auf **RunCallback** gesetzt. Dieses Feld ist eine Zeichenkette!

**UserData:**

Wird beim Aufruf von **RunCallback()** auf das gesetzt, was Sie im Argument **userdata** angegeben haben. Beachten Sie, dass dieses Feld nur vorhanden ist, wenn Sie beim Aufrufen von **RunCallback()** tatsächlich einen Wert in **userdata** übergeben haben.

### EINGABEN

**function** Funktion, die der Ereigniswarteschlange hinzugefügt werden soll

**userdata** optional: Benutzerdaten, die bei der Ausführung an die Funktion übergeben werden sollen

### BEISPIEL

```
RunCallback(Function(msg) NPrint(msg.userdata) EndFunction, "Hello 2!")
NPrint("Hello 1!")
Repeat
    WaitEvent
Forever
```

Dadurch wird zuerst "Hello 1!" und dann "Hello 2!" ausgegeben, weil die Funktion, die "Hello 2!" ausgibt, erst aufgerufen wird, wenn Hollywood die Ereigniswarteschlange leert, was mit **WaitEvent()** geschieht.

## 26.25 SetEventTimeout

### BEZEICHNUNG

SetEventTimeout – legt die Ereignisaufbewahrung fest (V1.9)

**ÜBERSICHT**

`SetEventTimeout(duration)`

**BESCHREIBUNG**

Sie können diesen Befehl benutzen, um Hollywood zu sagen, wie lang es eingetretene Ereignisse aufbewahren soll. Der Standardwert beläuft sich hier momentan auf 60000, was bedeutet, dass Hollywood eingetretene Ereignisse maximal 60 Sekunden aufbewahrt und dann verwirft. Dieser Befehl ist z.B. nützlich, wenn Sie eine massive Berechnung durchführen, welche länger als 60 Sekunden dauert. Dann werden alle Ereignisse, die vor mehr als 60 Sekunden aufgetreten sind, verloren gehen. Um dies zu verhindern, erhöhen Sie einfach das Ereignis Timeout.

Ab Hollywood 7.0 ist es auch möglich, **duration** auf -1 zu setzen, um Ereignis-Timeouts vollständig zu deaktivieren. In diesem Fall werden die Ereignisse niemals verloren gehen.

**EINGABEN**

**duration** gibt an, wie lang Ereignisse aufbewahrt werden (in Millisekunden)

**BEISPIEL**

`SetEventTimeout(5000)`

Setzt die Aufbewahrungsdauer auf 5 Sekunden (= 5000 Millisekunden).

## 26.26 SetInterval

**BEZEICHNUNG**

`SetInterval` – installiert eine neue Intervallfunktion (V2.0)

**ÜBERSICHT**

`[id] = SetInterval(id, func, ms[, userdata])`

**BESCHREIBUNG**

Dieser Befehl installiert eine neue Intervallfunktion und weist die ID zu. Wenn Sie **Nil** als **id** angeben, wird `SetInterval()` automatisch eine ID wählen und sie Ihnen übergeben. Sie müssen im Argument **func** eine Funktion und in **ms** die Zeit in Millisekunden angeben. Die angegebene Funktion wird dann wieder und wieder und wieder in den Abständen der angegebenen Zeit aufgerufen werden. Wenn Sie zum Beispiel 40 als Intervall angeben, wird Ihre Funktion alle 40 Millisekunden aufgerufen werden, das 25-mal pro Sekunde entspricht ( $25 * 40 \text{ ms} = 1000 \text{ ms} = 1 \text{ Sekunde}$ ). Das reicht für die meisten Spiele, Intros usw.

Die Funktion, die Sie in **func** angeben, wird zum ersten Mal aufgerufen werden, wenn die in **ms** angegebene Zeit verstrichen ist. Danach wird Ihre Funktion wiederholt in Intervallen von **ms** Millisekunden aufgerufen werden.

Sie müssen immer `WaitEvent()` in Verbindung mit diesem Befehl verwenden! Wenn eine Intervallfunktion installiert ist, wird der interne Hollywood-Timer Intervall-Ereignisse auslösen und `WaitEvent()` informieren, dass er Ihre Intervallfunktion aufruft. Intervalle funktionieren nicht ohne `WaitEvent()`! Wenn Sie `WaitEvent()` nicht verwenden, wird Ihre Intervallfunktion nie aufgerufen werden. Intervallfunktionen werden nur so lange aufgerufen, wie Sie in einer `WaitEvent()`-Schleife sind.

Dieser Befehl ist sehr wichtig, weil er Ihnen hilft sicherzustellen, dass Ihr Skript auf jedem System mit der gleichen Geschwindigkeit ausgeführt wird. Siehe [Abschnitt 15.3 \[Zeitsteuerung des Skripts\]](#), Seite 170, für weitere Informationen zu diesem Thema und auch für ein Beispiel.

Sie können beliebig viele Intervalle installieren. Hollywoods interner Scheduler wird sicherstellen, dass alle Intervallfunktionen korrekt aufgerufen werden.

Bitte denken Sie daran, dass Hollywood nicht Multithreading unterstützt. Daher dürfen Ihre Intervallfunktionen das Skript nicht blockieren, andernfalls wird das gesamte Skript gesperrt. Zum Beispiel, wenn Sie zwei Intervallfunktionen installiert haben und eine dieser Funktionen führt den Befehl `Wait(100)` aus, dann wird das gesamte Skript für 2 Sekunden gesperrt.

Sie können den Befehl `ClearInterval()` aufrufen, um eine Intervallfunktion zu stoppen.

Ab Hollywood 3.1 gibt es das optionale Argument `userdata`. Der Wert, den Sie hier angeben, wird der Callback-Funktion übergeben, wenn sie aufgerufen wird. Dies ist nützlich, wenn Sie die Arbeit mit globalen Variablen vermeiden möchten. Mit Hilfe von dem Argument `userdata` können Sie ganz einfach Daten an Ihre Callback-Funktion übergeben. Sie können einen Wert eines beliebigen Typs wie Zahlen, Zeichenketten, Tabellen und Funktionen angeben.

Ihre Intervallfunktion wird von Hollywood mit einem Parameter aufgerufen werden. Dieser Parameter ist eine Nachrichtentabelle, die folgende Felder enthält:

- Action:** Wird immer auf `Interval` gesetzt. Dieses Feld ist eine Zeichenkette!
- ID:** Wird auf die ID des Intervalls festgelegt werden, der Hollywood gerade aufgerufen hat.
- UserData:** Wird auf den Inhalt von `userdata` gesetzt, den Sie im Argument `userdata` angegeben haben, als Sie die Intervallfunktion installiert haben.

Diese Meldung ist nützlich, wenn Sie zwei oder mehr Intervalle in der gleichen Funktion handhaben wollen. Die Meldung sagt Ihnen dann, welches Intervall Hollywood ausführt. Wenn Sie diese Nachricht nicht brauchen, einfach ignorieren.

Last but not least: Sie sollten einen Blick in die Beispiele werfen, die mit Hollywood geliefert wurden. Viele von ihnen verwenden `SetInterval()`, um den Zeitpunkt des Skripts zu verwalten!

Weiterer Intervall-Befehl: `ChangeInterval()`

## EINGABEN

- id** Identifikator für die neue Intervallfunktion oder `Nil` fürs [automatische Auswählen der ID](#). Die ID ist erforderlich, so dass Sie später diese Intervallfunktion mit `ClearInterval()` entfernen können
- func** Funktion, welche mit jedem Intervall aufgerufen wird
- ms** Intervalle, in denen die Funktion aufgerufen wird. z.B. 40 ruft die Funktion 25 Mal in der Sekunde auf, weil  $25 * 40\text{ms} = 1000\text{ms} = 1 \text{ Sekunde}$
- userdata** optional: Benutzerdefinierte Daten, welche der Callback-Funktion übergeben werden (V3.1)

**RÜCKGABEWERTE**

**id** optional: Identifikator des Intervalls; wird nur zurückgegeben, wenn bei Argument 1 **Nil** angegeben wurde (siehe oben)

**BEISPIEL**

Siehe [Abschnitt 15.3 \[Zeitsteuerung des Skripts\]](#), Seite 170.

**26.27 SetTimeout****BEZEICHNUNG**

SetTimeout – installiert eine neue Timeout-Funktion (V2.0)

**ÜBERSICHT**

```
[id] = SetTimeout(id, func, timeout[, userdata])
```

**BESCHREIBUNG**

Dieser Befehl installiert eine neue Timeout-Funktion und ordnet ihr die ID zu. Wenn Sie **Nil** als **id** angeben, wird **SetTimeout()** automatisch eine ID erhalten und zurückgeben. Sie müssen eine Funktion dem Argument **func** und die Zeit in Millisekunden in **timeout** angeben. Nach Ablauf dieser Zeit wird Hollywood Ihre Timeout-Funktion aufrufen. Dies ist nützlich, wenn Sie genaue Zeitpunkte benötigen, zum Beispiel wenn Sie Grafiken mit Musik synchronisieren möchten. Timeout-Funktionen sind dafür perfekt.

Sie müssen immer **WaitEvent()** in Verbindung mit diesem Befehl verwenden! Wenn Sie eine Timeout-Funktion installiert haben, wird der interne Hollywood-Timer-Scheduler Timeout-Ereignisse auslösen und **WaitEvent()** informieren, Ihre Timeout-Funktion aufzurufen. Timeouts funktionieren nicht ohne **WaitEvent()**! Wenn Sie **WaitEvent()** nicht verwenden, wird Ihre Timeout-Funktion nie aufgerufen werden. Timeout-Funktionen werden nur so lange aufgerufen, wie Sie in einer **WaitEvent()**-Schleife sind.

Sie können so viele Timeouts installieren, wie Sie möchten. Hollywoods interner Scheduler wird sicherstellen, dass alle Timeout-Funktionen korrekt aufgerufen werden.

Sie können **ClearTimeout()** verwenden, um eine Timeout-Funktion zu stoppen.

Ab Hollywood 3.1 gibt es das optionales Argument **userdata**. Der Wert, den Sie hier angeben, wird der Callback-Funktion übergeben. Dies ist nützlich, wenn Sie die Arbeit mit globalen Variablen vermeiden möchten. Mit Hilfe des Arguments **userdata** können Sie ganz einfach Daten an Ihre Callback-Funktion übergeben. Sie können einen Wert eines beliebigen Typs wie Zahlen, Zeichenketten, Tabellen und Funktionen angeben.

Ihre Timeout-Funktion wird von Hollywood mit einem Parameter aufgerufen werden. Dieser Parameter ist eine Nachrichtentabelle, die folgende Felder enthält:

**Action:** Wird immer auf **Timeout** gesetzt. Dieses Feld ist eine Zeichenkette!

**ID:** Identifikator der Timeout-Funktion.

**UserData:**

Wird auf den Inhalt von **userdata** gesetzt, die Sie im Argument **userdata** angegeben haben, als Sie die Timeout-Funktion installiert haben.

Diese Meldung ist nützlich, wenn Sie zwei oder mehr Intervalle in der gleichen Funktion handhaben wollen. Die Meldung sagt Ihnen dann, welches Intervall Hollywood ausführt. Wenn Sie diese Nachricht nicht brauchen, einfach ignorieren.

**EINGABEN**

<b>id</b>	Identifikator für die neue Timeout-Funktion oder <b>Nil</b> für <b>automatisches Auswählen der ID</b> . Die ID ist erforderlich, so dass Sie später diese Timeout-Funktion mit <b>ClearTimeout()</b> entfernen können
<b>func</b>	Funktion, welche aufgerufen wird, nachdem die angegebene Zeit verstrichen ist
<b>ms</b>	Die Dauer in Millisekunden, bis die Timeout-Funktion aufgerufen wird.
<b>userdata</b>	optional: Benutzerdefinierte Daten, welche der Callback-Funktion übergeben werden (V3.1)

**RÜCKGABEWERTE**

<b>id</b>	optional: Identifikator des Timeouts; wird nur zurückgegeben, wenn bei Argument 1 <b>Nil</b> angegeben wurde (siehe oben)
-----------	---

**BEISPIEL**

```
Function p_TenSeconds()
  SystemRequest("Hollywood", "Ten seconds are over now!", "OK")
EndFunction

SystemRequest("Hollywood", "I will call the function TenSeconds()\n" ..
  "after 10 seconds have elapsed!\nCheck your watch, then click Go!",
  "Let's go!")

SetTimeout(1, p_TenSeconds, 10000)

Repeat
  WaitEvent
Forever

Der obige Code installiert eine Timeout-Funktion, die nach 10000 Millisekunden aufgerufen wird (= 10 Sekunden).
```

**26.28 WaitEvent****BEZEICHNUNG**

WaitEvent – wartet auf ein Ereignis

**ÜBERSICHT**

```
info = WaitEvent()
```

**BESCHREIBUNG**

Der Befehl **WaitEvent()** setzt Hollywood in den Schlafzustand. Das Programm wird aktiv, wenn ein Ereignis ausgelöst wird. In diesem Fall wird **WaitEvent()** die Funktion ausführen, die Sie für dieses Ereignis installiert haben und kehrt dann wieder zurück. Daher müssen Sie immer **WaitEvent()** in einer Schleife verwenden. Beispielsweise:

```
While quit = False
  WaitEvent
```

Wend

Oder Sie nutzen eine Endlosschleife:

Repeat

WaitEvent

Forever

`WaitEvent()` ist ein Kernbefehl der Hollywoodsprache und Sie sollten ihn in der Hauptschleife in jedem Skript verwenden. `WaitEvent()` hat den Vorteil, dass er schläft, bis ein Ereignis ausgelöst wird. Dies ist wichtig in einer Multitasking-Umgebung, da sie Rechenzeit sparen. Verwenden Sie niemals Abfrageschleifen! Sie verbrauchen alle CPU-Zeit. Wenn Sie Code in Ihre Hauptschleife ständig ausführen müssen, verwenden Sie `SetInterval()`, um eine Intervallfunktion zu installieren, die 25-mal pro Sekunde durch `WaitEvent()` aufgerufen wird.

Die Funktionen, die `WaitEvent()` aufruft, wenn ein Ereignis ausgelöst wird, kann mit den folgenden Befehlen der Hollywood-Ereignisbibliothek installiert werden: `MakeButton()`, `SetInterval()`, `SetTimeout()` und `InstallEventHandler()`.

Beachten Sie, dass `WaitEvent()` nicht von Callback-Funktionen aufgerufen werden kann, die von `WaitEvent()` ausgeführt werden. Im Allgemeinen sollten Sie `WaitEvent()` nur einmal in Ihrem Skript verwenden: In Ihrer Hauptschleife. Wenn Sie wirklich Ereignisse in einem von `WaitEvent()` ausgeführten Callback-Funktion verarbeiten müssen, verwenden Sie stattdessen den Befehl `CheckEvents()`. Dies sollte jedoch normalerweise nicht erforderlich sein.

`WaitEvent()` liefert eine Tabelle, die Informationen über die gerade ausgeführt Callback-Funktion enthält. Die folgenden Felder werden in dieser Tabelle initialisiert werden:

**Action:** Enthält den Namen des Ereignisses, welches die Callback-Ausführung (z.B. `OnMouseDown`) verursacht. Wenn `WaitEvent()` zurückkehrt, ohne eine Callback-Funktion ausgeführt zu haben, wird dieses Feld eine leere Zeichenkette enthalten.

**ID:** Enthält die ID des Objekts, das die Callback-Funktion (zum Beispiel eine Display-ID) aufgerufen hat. ID kann auch Null sein, falls ein Ereignis eintraf, welches mit keiner ID in Verbindung gebracht werden kann.

**Triggered:**

Wird auf `True` gesetzt werden, wenn `WaitEvent()` eine Callback-Funktion ausgeführt hat und dann die die Kontrolle an das Skript zurückgibt. Wenn dies auf `False` gesetzt ist, dann hat dies ein anderes internes Ereignis verursacht und `WaitEvent()` gibt die Kontrolle an das Skript zurück.

**NResults:**

Enthält die Anzahl der Werte, die die Callback-Funktion zurückgibt (z.B. 1). Dies wird 0 sein, wenn keine Werte zurückgegeben wurden oder wenn überhaupt keine Callback-Funktion ausgeführt wurde.

**Results:** Wenn `NResults` größer als 0 ist, wird diese Tabelle alle Werte enthalten, die die Callback-Funktion zurückgibt. Sonst wird diese Tabelle überhaupt nicht vorhanden sein. Diese Tabelle können Sie leicht dazu verwenden, um zusätzliche Informationen von Ihrer Callback-Funktion dem Hauptprogramm zu übergeben.

**EINGABEN**

Keine

**RÜCKGABEWERTE**

**info** Rückgabewerte von einer zuvor ausgeführten Ereignis-Funktion; normalerweise benötigen Sie diese nicht und Sie können sie ignorieren

**BEISPIEL**

Siehe [Abschnitt 26.19 \[MakeButton\]](#), Seite 507.

Siehe [Abschnitt 26.26 \[SetInterval\]](#), Seite 515.

Siehe [Abschnitt 26.27 \[SetTimeout\]](#), Seite 517.

Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), Seite 483.

## 26.29 WaitKeyDown

**BEZEICHNUNG**

WaitKeyDown – wartet, bis der Benutzer eine Taste drückt (V1.5)

**ÜBERSICHT**

WaitKeyDown(key\$[, rawkey])

**BESCHREIBUNG**

Dieser Befehl hält das Programm an, bis der Benutzer die durch **key\$** angegebene Taste drückt. Siehe [Abschnitt 26.14 \[IsKeyDown\]](#), Seite 503, für die Tasten, die Sie in **key\$** angeben können. Bitte beachten Sie, dass **WaitKeyDown()** nicht für Unicode-Zeichen verwendet werden kann. Nur Zeichen aus dem englischen Alphabet werden universell unterstützt.

Ab Hollywood Version 6.1 können Sie die spezielle Zeichenfolge **ANY** in **key\$** angeben, um auf eine beliebige Taste zu warten.

Ab Hollywood 7.1 gibt es das optionale Argument **rawkey**. Wenn dieses Argument auf **True** gesetzt ist, wird **WaitKeyDown()** **key\$** als rohe Taste behandeln und warten, bis sie unten ist. In diesem Fall muss **key\$** einer der von Hollywood definierten rohen Tasten sein. Siehe [Abschnitt 26.23 \[Rohe Tasten\]](#), Seite 512, für Details. Der Unterschied zwischen normalen und rohen Tasten ist in der Dokumentation des Ereignis-Handlers **OnRawKeyDown** beschrieben. Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), Seite 483, für Details. Das Setzen von **rawkey** auf **True** kann nützlich sein, wenn Sie auf eine Umschalttaste wie Shift, Alt, Control, etc. warten müssen.

Siehe auch [WaitLeftMouse\(\)](#), [WaitMidMouse\(\)](#) und [WaitRightMouse\(\)](#).

**EINGABEN**

**key\$** abzuwartende Taste

**rawkey** optional: **True**, wenn **key\$** als rohe Taste behandelt werden soll (voreingestellt ist **False**) (V7.1)

**BEISPIEL**

```
Print("Drücken Sie Return, um fortzufahren.")
WaitKeyDown("Return")
```



Der obige Code wartet, bis der Benutzer die Eingabetaste drückt.

## 26.30 WaitLeftMouse

### BEZEICHNUNG

WaitLeftMouse – wartet darauf, dass der Benutzer die linke Maustaste drückt

### ÜBERSICHT

WaitLeftMouse()

### BESCHREIBUNG

Dieser Befehl hält die Abarbeitung des Skripts an, bis die linke Maustaste gedrückt wird.

Siehe auch [WaitKeyDown\(\)](#), [WaitMidMouse\(\)](#) und [WaitRightMouse\(\)](#).

### EINGABEN

keine

### BEISPIEL

```
Print("Drücken Sie zum Beenden die linke Mausaste.")  
WaitLeftMouse  
End
```

Wartet darauf, dass der Benutzer die linke Mausaste drückt.

## 26.31 WaitMidMouse

### BEZEICHNUNG

WaitMidMouse – wartet darauf, dass der Benutzer die mittlere Maustaste drückt (V4.5)

### ÜBERSICHT

WaitMidMouse()

### BESCHREIBUNG

Dieser Befehl hält die Abarbeitung des Skripts an, bis die mittlere Maustaste gedrückt wird.

Siehe auch [WaitKeyDown\(\)](#), [WaitLeftMouse\(\)](#) und [WaitRightMouse\(\)](#).

### EINGABEN

Keine

### BEISPIEL

```
Print("Zum Beenden drücken Sie die mittlere Maustaste.")  
WaitMidMouse  
End
```

Wartet darauf, dass der Benutzer die mittlere Maustaste drückt.

## 26.32 WaitRightMouse

### BEZEICHNUNG

WaitRightMouse – wartet darauf, dass der Benutzer die rechte Maustaste drückt (V1.5)

### ÜBERSICHT

WaitRightMouse()

### BESCHREIBUNG

Dieser Befehl hält die Abarbeitung des Skripts an, bis die rechte Maustaste gedrückt wird.

Siehe auch `WaitKeyDown()`, `WaitLeftMouse()` und `WaitMidMouse()`.

### EINGABEN

keine

### BEISPIEL

```
PPrint("Drücken Sie zum Beenden die rechte Taste.")  
WaitRightMouse  
End
```

Wartet darauf, dass der Benutzer die rechte Taste drückt.

## 27 Fehlerbehandlungsbibliothek

### 27.1 ERROR

#### BEZEICHNUNG

ERROR – bricht die Kompilation mit einer Fehlermeldung ab (V6.1)

#### ÜBERSICHT

@ERROR msg\$

#### BESCHREIBUNG

Jedes Mal, wenn der Präprozessor diese Anweisung erreicht, wird die Kompilierung sofort abgebrochen und die angegebene Fehlermeldung angezeigt. Dies ist nur für Debugging-zwecke nützlich.

#### EINGABEN

msg\$ Fehlermeldung, die angezeigt wird

### 27.2 Error

#### BEZEICHNUNG

Error – beendet das Skript mit einer benutzerdefinierter Fehlermeldung (V2.0)

#### ÜBERSICHT

Error(msg\$)

#### BESCHREIBUNG

Dieser Befehl beendet das Skript und zeigt die Meldung in msg\$ an. Dies ist nützlich, wenn Ihr Skript abgebrochen wird, weil eine unerwartete Bedingung aufgetreten ist.

#### EINGABEN

msg\$ zeigt die Fehlermeldung an

#### BEISPIEL

```
If Exists("Game.dat") = False Then Error("Cannot read game data!")
```

Der obige Code prüft, ob die Datei "Game.dat" vorhanden ist und endet mit einer Fehlernachricht, wenn sie nicht existiert.

### 27.3 ExitOnError

#### BEZEICHNUNG

ExitOnError – aktiviert/deaktiviert den Hollywood Fehler-Handler

#### ÜBERSICHT

ExitOnError(enable)

#### BESCHREIBUNG

Dieser Befehl aktiviert bzw. deaktiviert Hollywoods Fehler-Handler. Wenn der Fehler-Handler aktiviert ist und ein Fehler auftritt, wird Ihr Skript entweder gestoppt und

Hollywood zeigt die Fehlermeldung an, oder falls Sie einen benutzerdefinierte Fehlerbehandlung unter Verwendung von `RaiseOnError()` installiert haben, wird die benutzerdefinierte Fehlerbehandlung ausgeführt. Siehe [Abschnitt 7.7 \[Fehlerbehandlung\]](#), [Seite 94](#), für Details.

Es kann nützlich sein, den Fehlerhandler für eine sehr kurze Zeit zu deaktivieren, wenn Sie überprüfen müssen, ob ein bestimmter Befehl erfolgreich war oder nicht. Dies kann durch Einschließen des Befehls in einen `ExitOnError()`-Block erfolgen. Siehe [Abschnitt 7.7 \[Fehlerbehandlung\]](#), [Seite 94](#), für Details. Es wird nicht empfohlen, den Fehlerhandler für eine längere Zeit zu deaktivieren, da Fehler leicht auftreten können. Im Allgemeinen sollten Sie den Fehlerhandler immer aktiviert lassen.

In Hollywood 7.1 und höher ist die neue Fragezeichensyntax zum Prüfen von Fehlern der Verwendung vom Befehl `ExitOnError()` vorzuziehen, da sie viel kürzer und weniger anfällig für Fehler ist. Siehe [Abschnitt 7.7 \[Fehlerbehandlung\]](#), [Seite 94](#), für Details.

## EINGABEN

**enable**      `True`, um die Fehlerbehandlung zu ermöglichen; `False`, um sie zu deaktivieren

## 27.4 Fehlercodes

In der Standardeinstellung wird Hollywood immer verlassen, wenn ein Fehler auftritt. Sie können dieses Verhalten ändern, indem Sie den Befehl `ExitOnError()` und `GetLastError()` verwenden oder eine benutzerdefinierte Fehlerbehandlung mit `RaiseOnError()` installieren. Beide Wege werden einen Fehlercode liefern. Hier ist eine Liste aller Fehlercodes, die derzeit in Hollywood definiert sind (einige Fehlercodes werden nicht mehr verwendet, bleiben aber aus Gründen der Kompatibilität erhalten):

```
#ERR_NONE
    Kein Fehler aufgetreten! (0)

#ERR_MEM    Es ist nicht mehr genug Speicher vorhanden! (1000)

#ERR_UNIMPLCMD
    Befehl ist nicht implementiert! (1001)

#ERR_NORETVAL
    Es wurde keine Variable für den Rückgabewert angegeben! (1002)

#ERR_USERABORT
    Abbruch durch den Benutzer! (1003)

#ERR_SCREEN
    Fehler beim Öffnen des Bildschirms! (1004)

#ERR_WRITE
    Konnte nicht alle Zeichen in die Datei schreiben! Prüfen Sie, ob genügend Platz
    frei ist! (1005)

#ERR_UNTERMINTDSTR
    Zeichenkette wurde nicht terminiert! (1006)
```

**#ERR\_UNKNOWNCOND**  
Unbekannte Bedingung! (1007)

**#ERR\_MISSINGSEPARTR**  
Mehrere Befehle in einer Zeile müssen durch Doppelpunkte getrennt werden!  
(1008)

**#ERR\_READ**  
Konnte nicht alle Zeichen aus der Datei lesen! Prüfen Sie, ob es lesegeschützt  
ist! (1009)

**#ERR\_WINDOW**  
Fehler beim Öffnen des Fensters! (1010)

**#ERR\_ELSEWOIF**  
Else() ohne If() ! (1011)

**#ERR\_ENDIFWOIF**  
EndIf() ohne If() ! (1012)

**#ERR\_IFWOENDIF**  
If() ohne Endif() ! (1013)

**#ERR\_MISSINGPARAMTR**  
Nicht genug Argumente übergeben! (1014)

**#ERR\_FORWONEXT**  
For() ohne Next() ! (1015)

**#ERR\_NEXTWOFOR**  
Next() ohne For() ! (1016)

**#ERR\_WHILEWOWEND**  
While() ohne Wend() ! (1017)

**#ERR\_SYNTAXERROR**  
Genereller Syntaxfehler! (1018)

**#ERR\_WRONGDTYPE**  
Falscher Datentyp angegeben! (1019)

**#ERR\_VARSYNTAX**  
Syntaxfehler im Variablennamen! (1020)

**#ERR\_WENDWOWHILE**  
Wend() ohne While() ! (1021)

**#ERR\_UNKNOWNCMD**  
Unbekannter Befehl %s ! (1022)

**#ERR\_MISSINGBRACKET**  
Sie haben zu viele Argumente angegeben oder eine Klammer vergessen! (1023)

**#ERR\_VALUEEXPECTED**  
Es wurde ein Wert erwartet! (1024)

**#ERR\_OPENLIB**  
Kann %s nicht öffnen! (1025)

#ERR_VAREXPECTED	Es wurde eine Variable erwartet! (1026)
#ERR_LABINFOR	Labels innerhalb einer For()-Schleife sind nicht erlaubt! (1027)
#ERR_LABINIF	Labels innerhalb einer If()-Bedingung sind nicht erlaubt! (1028)
#ERR_LABINWHILE	Labels innerhalb einer While()-Schleife sind nicht erlaubt! (1029)
#ERR_WRONGOP	Falscher Operator für diesen Datentyp! (1030)
#ERR_GETDISKOBJ	Kann das Piktogramm des Skripts nicht öffnen! (1031)
#ERR_EVNTEXPECTED	Sie müssen ein Hollywood-Ereignis angeben! (1032)
#ERR_EMPTYOBJ	Kann kein leeres Textobjekt erstellen! (1033)
#ERR_EMPTYSCRIPT	Ihr Skript ist leer! (1034)
#ERR_COMMENTSTRUCT	Unzusammenhängender Kommentar! (1035)
#ERR_ALRDYDECLRD	Diese Variable wurde schon benutzt und initialisiert! (1036)
#ERR_WRONGFLOAT	Nicht erlaubtes Fließkommaformat! (1037)
#ERR_REQUIREFIELD	Sie müssen einen Index im Feld angeben! (1038)
#ERR_OUTOFRANGE	Der angegebene Index ist größer als der Feldumfang! (1039)
#ERR_RETWOGOSUB	Return() ohne Gosub() ! (1040)
#ERR_FINDARRAY	Das angeforderte Objekt wurde nicht gefunden! (1041)
#ERR_FINDCST	Konstante wurde nicht gefunden! (1042)
#ERR_LOCK	Fehler beim Abfragen des Verzeichnisses! (1043)
#ERR_LOADPICTURE	Fehler beim Laden des Bildes %s! Vergewissern Sie sich, dass Sie einen Datatype für das Format haben! (1044)

**#ERR\_READFILE**  
Kann Datei %s nicht öffnen! (1045)

**#ERR\_NOTPROTRACKER**  
Modul ist nicht im Protracker-Format! (1046)

**#ERR\_UNKNOWNSEQ**  
Unbekannter Charakter nach dem Backslash! (1047) (Beachten Sie: Charakter ist ein einzelnes Zeichen)

**#ERR\_DIRLOCK**  
Fehler beim Abfragen des Verzeichnisses %s ! (1048)

**#ERR\_KEYWORD**  
Unbekanntes Schlüsselwort! (1049)

**#ERR\_KICKSTART**  
Sie benötigen mindestens Kickstart 3.0! (1050)

**#ERR\_FREEABGPIC**  
Sie können nicht das momentan angezeigte Hintergrundbild freigeben! (1051)

**#ERR\_WRITEFILE**  
Kann Datei %s nicht schreiben! (1052)

**#ERR\_VERSION**  
Dieses Skript benötigt mindestens Hollywood %s ! (1053)

**#ERR\_NOFUNCTION**  
Dieser Befehl hat keinen Rückgabewert! (1054)

**#ERR\_WRONGUSAGE**  
Falsche Benutzung/Argumente für diesen Befehl! Überprüfen Sie die Dokumentation! (1055)

**#ERR\_SELECTBG**  
Dieser Befehl kann nicht benutzt werden, wenn SelectBGPic() aktiv ist! (1056)

**#ERR\_ARRAYDECLA**  
Das Feld "%s[]" wurde nicht deklariert! (1057)

**#ERR\_CMDASVAR**  
Dieser Variablenname ist schon von einem Befehl belegt! (1058)

**#ERR\_CONFIG**  
Kein Skript übergeben! (1059)

**#ERR\_ARGS**  
Falsche Argumente übergeben! (1060)

**#ERR\_DOUBLEDECLA**  
Diese Kennung wurde schon mal deklariert! (1061) (Beachten Sie: Kennung wird auch ID oder Identifikator genannt)

**#ERR\_EQUALEXPECTED**  
Es wurde ein Gleichheitszeichen erwartet! (1062)

**#ERR\_OPENANIM**

Konnte Animation nicht laden! Vergewissern Sie sich, dass Sie mindestens Version 40 des animation.datatype und der realtime.library installiert haben! Bitte beachten Sie auch, dass MorphOS momentan keinen Datatype für IFF ANIM hat, d.h. Sie müssen selbst einen IFF ANIM Datatype installieren, wenn Sie das Format benutzen möchten! Zum Beispiel können Sie den Datatype von OS3.1 installieren. (1063)

**#ERR\_OPENFONT**

Nicht definiert (1064)

**#ERR\_OPENSOUND**

Kann Sample %s nicht öffnen! Vergewissern Sie sich, dass Sie einen Datatype für dieses Format haben! Wenn Sie versucht haben, ein 16-bit Sample zu laden und diesen Fehler bekommen, dann müssen Sie einen Datatype-Ersatz installieren, weil der OS3.x Datatype nur 8-bit Samples unterstützt. Sie können den Ersatz für den sound.datatype von <http://www.stephan-rupprecht.de/> laden oder von der Hollywood CD-ROM installieren. Sie benötigen diesen Ersatz NICHT unter MorphOS 1.x weil dies schon einen neuen Datatype besitzt, der 16-bit Samples unterstützt! (1065)

**#ERR\_INTERNAL**

Internes Limit erreicht! Kontaktieren Sie den Autor... (1066)

**#ERR\_PUBSCREEN**

Kann den angegebenen Bildschirm nicht finden! (1067)

**#ERR\_BRUSHLINK**

Dieser Befehl kann keine verknüpften Pinsel benutzen! (1068)

**#ERR\_WRONGID**

Bitte benutzen Sie nur positive Ganzzahlen für Ihre Objekte! (1069)

**#ERR\_AHI** Genereller AHI-Fehler! Überprüfen Sie Ihre AHI-Installation und Einstellungen! (1070)

**#ERR\_VARENGTH**

Die Variablennamenslänge ist begrenzt auf 64 Zeichen! (1071)

**#ERR\_LABELDECLA**

Kann das Label "%s" nicht finden! (1072)

**#ERR\_LABELDOUBLE**

Das Label "%s" wurde schon deklariert! (1073)

**#ERR\_NOKEYWORDS**

Schlüsselwörter sind hier nicht mehr erlaubt! (1074)

**#ERR\_NOCONSTANTS**

Konstanten sind hier nicht erlaubt! (1075)

**#ERR\_SEEK**

Ungültige Suchposition angegeben! (1076)



**#ERR\_CSTDDOUBLEDEF**

Die Konstante #%s wurde schon deklariert! (1077)

**#ERR\_NOLAYERS**

Diese Funktion benötigt angeschaltete Layer! (1078) (Beachten Sie: Mit Layer sind Ebenen gemeint)

**#ERR\_LAYERSUPPORT**

Dieser Layertyp wird nicht von GetBrushLink() unterstützt! (1079) (Beachten Sie: Mit Layer sind Ebenen gemeint)

**#ERR\_UNKNOWNATTR**

Unbekanntes Attribut angegeben! (1080)

**#ERR\_LAYERRANGE**

Der angegebene Layer existiert nicht! (1081) (Beachten Sie: Mit Layer sind Ebenen gemeint)

**#ERR\_SELECTBRUSH**

Dieser Befehl kann nicht benutzt werden, wenn SelectBrush() aktiv ist! (1082)

**#ERR\_POINTERFORMAT**

Der Mauszeiger muss in 4 Farben vorliegen und darf nicht breiter als 16 Pixel sein! (1083)

**#ERR\_CREATEDIR**

Fehler beim Erstellen des Verzeichnisses %s ! (1084)

**#ERR\_DISPLAYSIZE**

Kann die Displaygröße nicht auf %s wechseln! (1085)

**#ERR\_DISPLAYDESKTOP**

Sie können kein Hintergrundbild Nummer 1 zusammen mit DISPLAY-DESKTOP angeben! (1086)

**#ERR\_GUIGFX**

Kann die guigfx.library in Version 20 nicht öffnen! Stellen Sie sicher, dass Version 20 installiert ist! (1087)

**#ERR\_RENDER**

Kann die render.library in Version 30 nicht öffnen! Stellen Sie sicher, dass Version 30 installiert ist! (1088)

**#ERR\_ZERODIVISION**

Unerlaubte Division durch Null! (1089)

**#ERR\_WARPOS**

Sie benötigen mindestens WarpUP v5.1 für dieses Programm! (1090)

**#ERR\_UNKNOWN**

Unbekannte Fehlernummer! (1091)

**#ERR\_STRINGCST**

Sie können hier keine Stringkonstanten angeben! (1092) (Beachten Sie: Strings sind Zeichenketten/Zeichenfolgen)

#ERR_LABINFUNC	Labels sind innerhalb von Funktionen nicht erlaubt! (1093)
#ERR_ANIMFRAME	Das angegebene Einzelbild ist nicht im zulässigen Bereich! (1094)
#ERR_REPEATWUNTIL	REPEAT ohne UNTIL! (1095)
#ERR_UNTILWOREPEAT	UNTIL ohne REPEAT! (1096)
#ERR_FUNCWOENDFUNC	FUNCTION ohne ENDFUNCTION! (1097)
#ERR_ENDFUNCWOFUNC	ENDFUNCTION ohne FUNCTION! (1098)
#ERR_UNEXPECTEDSYM	Unerwartetes Zeichen! (1099)
#ERR_FUNCARGS	Es wurde eine Variable, schließende Klammer oder "..." erwartet! (1100)
#ERR_NOLOOP	Keine Schleife zum Unterbrechen gefunden! (1101)
#ERR_TOKENEXPECTED	"%s" wurde erwartet! (1102)
#ERR_BRACKETOPEN	Es wurde eine öffnende Klammer erwartet! (1103)
#ERR_BRACKETCLOSE	Es wurde eine schließende Klammer erwartet! (1104)
#ERR_BRACEOPEN	Es wurde eine öffnende geschweifte Klammer erwartet! (1105)
#ERR_BRACECLOSE	Es wurde eine schließende geschweifte Klammer erwartet! (1106)
#ERR_SQBRACKETOPEN	Es wurde eine öffnende eckige Klammer erwartet! (1107)
#ERR_SQBRACKETCLOSE	Es wurde eine schließende eckige Klammer erwartet! (1108)
#ERR_NOCOMMA	Es wurde ein Komma erwartet! (1109)
#ERR_SYNTAXLEVELS	Zu viele Syntaxebenen! (1110)
#ERR_COMPLEXWHILE	WHILE Bedingung ist zu komplex! (1111)

**#ERR\_NOCHAR**  
Angegebener ASCII-Code ist außerhalb des zulässigen Bereichs! (1112)

**#ERR\_CHRCSTLEN**  
Zeichenkonstante ist zu lang! (1113)

**#ERR\_CHRCSTEMPTY**  
Leere Zeichenkonstanten sind nicht zulässig! (1114)

**#ERR\_HEXPOINT**  
Dezimalpunkt ist in hexadezimaler Notation nicht zulässig! (1115)

**#ERR\_NUMCONCAT**  
Ein Leerzeichen ist zwischen Zahlenverkettungen notwendig! (1116)

**#ERR\_MAXLOCALS**  
Zu viele lokale Variablen! (1117)

**#ERR\_MAXUPVALS**  
Zu viele Upvalues! (1118)

**#ERR\_MAXPARAMS**  
Zu viele Parameter! (1119)

**#ERR\_CONITEMS**  
Zu viele Elemente im Konstruktor! (1120)

**#ERR\_MAXLINES**  
Zu viele Zeilen in diesem Block! (1121)

**#ERR\_COMPLEXEXPR**  
Ausdruck oder Funktion ist zu komplex! (1122)

**#ERR\_CTRLSTRUCT**  
Kontrollstruktur ist zu lang! (1123)

**#ERR\_NOCOLON**  
Es wurde ein Doppelpunkt erwartet! (1124)

**#ERR\_CASECST**  
Case-Ausdrücke müssen konstant sein! (1125)

**#ERR\_SWCHWOENDSWCH**  
SWITCH ohne ENDSWITCH! (1126)

**#ERR\_ENDSWCHWOSWCH**  
ENDSWITCH ohne SWITCH! (1127)

**#ERR\_BLKWOENDBLK**  
BLOCK ohne ENDBLOCK! (1128)

**#ERR\_ENDBLKWOBLK**  
ENDBLOCK ohne BLOCK! (1129)

**#ERR\_NUMSTRCMP**  
Zahlen können nicht mit Strings verglichen werden! (1130) (Beachten Sie: Strings sind Zeichenketten/Zeichenfolgen)

**#ERR\_CONCAT**  
Falsche Datentypen für den Verkettungsoperator! (1131)

**#ERR\_TABLEDECLA**  
Table %s wurde nicht deklariert! (1132) (Beachten Sie: Table ist eine Tabelle)

**#ERR\_FUNCDECLA**  
Funktion %s wurde nicht deklariert! (1133)

**#ERR\_INTERNAL1**  
Interne Grenze erreicht. Fehlercode %s. (1134)

**#ERR\_STACK**  
Stacküberlauf! (1135)

**#ERR\_MEMCODE**  
Codegrößenüberlauf! (1136)

**#ERR\_MEMCST**  
Konstantentabellenüberlauf! (1137)

**#ERR\_NUMEXPECTED**  
Es wurde ein String als Argument %ld erwartet! (1138) (Beachten Sie: Strings sind Zeichenketten/Zeichenfolgen)

**#ERR\_STREXPECTED**  
Es wurde ein String als Argument %ld erwartet! (1139) (Beachten Sie: Strings sind Zeichenketten/Zeichenfolgen)

**#ERR\_TABEXPECTED**  
Es wurde ein Table als Argument %ld erwartet! (1140) (Beachten Sie: Table ist eine Tabelle)

**#ERR\_READONLY**  
Datei wurde im Lesemodus geöffnet! Schreibzugriff nicht zulässig! (1141)

**#ERR\_WRITEONLY**  
Datei wurde im Schreibmodus geöffnet! Lesezugriff nicht zulässig! (1142)

**#ERR\_DELETEFILE**  
Konnte Datei nicht löschen! (1143)

**#ERR\_EXAMINE**  
Konnte %s nicht untersuchen! (1144)

**#ERR\_RENAME**  
Konnte Dateien nicht umbenennen! (1145)

**#ERR\_MEMRANGE**  
Die angegebene Position ist außerhalb des zulässigen Bereichs! (1146)

**#ERR\_SELECTMASK**  
Dieser Befehl kann nicht benutzt werden während SelectMask() aktiv ist! (1147)

**#ERR\_MODIFYABG**  
Aktuelles Hintergrundbild kann nicht verändert werden! (1148)

- #ERR\_MODIFYABR**  
Dieser Pinsel kann momentan nicht verändert werden! (1149)
- #ERR\_FUNCJMP**  
GOTO/GOSUB sind innerhalb Funktionen nicht zulässig! (1150)
- #ERR\_REVDWORD**  
Reserviertes Wort kann hier nicht benutzt werden! (1151)
- #ERR\_LOCKBMAP**  
Konnte Bitmap nicht sperren! (1152)
- #ERR\_PALSCREEN**  
Hollywood unterstützt keine Palettebildschirme mehr! Bitte wechseln Sie in den High- oder Truecolor-Modus! (1153)
- #ERR\_NEGCOORDS**  
Negative Koordinaten sind hier nicht zulässig! (1154)
- #ERR\_NOANMLAYER**  
Die angegebene Ebene ist keine Animeebene! (1155)
- #ERR\_BRUSHSIZE**  
Pinselgröße passt nicht zu den übergebenen Parametern! (1156)
- #ERR\_ENDWITHWOWITH**  
ENDWITH ohne WITH! (1157)
- #ERR\_WITHWOENDWITH**  
WITH ohne ENDWITH! (1158)
- #ERR\_FIELDINIT**  
Tablefeld %s wurde nicht initialisiert! (1159) (Beachten Sie: Table ist eine Tabelle)
- #ERR\_LABMAINBLK**  
Labels sind nur innerhalb des Hauptblocks zulässig! (1160)
- #ERR\_NAMEUSED**  
Dieser Ebenenname wurde schon vergeben! (1161)
- #ERR\_LAYERSOFF**  
Layer müssen abgeschaltet sein, bevor Sie diese Funktion benutzen! (1162) (Beachten Sie: Layer ist eine Ebene)
- #ERR\_LAYERSON**  
Layer können nicht an-/ausgeschaltet werden solange das Ausgabegerät nicht die Anzeige ist! (1163) (Beachten Sie: Layer ist eine Ebene und Anzeige ist das Display)
- #ERR\_NOLOOPCONT**  
Keine Schleife zum Fortführen gefunden! (1164)
- #ERR\_LOOPRANGE**  
Angegebene Schleifenzahl ist außerhalb des zulässigen Bereichs! (1165)

- #ERR\_INTEXPECTED**  
Es wurde eine ganze Zahl erwartet! (1166)
- #ERR\_SELECTALPHACHANNEL**  
Dieser Befehl kann nicht benutzt werden während SelectAlphaChannel() aktiv ist! (1167)
- #ERR\_PIXELRANGE**  
Angegebener Pixel ist außerhalb des zulässigen Bereichs! (1168)
- #ERR\_DATATYPEALPHA**  
Ihr picture.datatype unterstützt keinen Alphakanal! (1169)
- #ERR\_NOALPHA**  
Das Bild "%s" besitzt keinen Alphakanal! (1170)
- #ERR\_PIXELFORMAT**  
Unbekanntes Pixelformat festgestellt! Hollywood kann auf diesem Bildschirm nicht laufen! (1171)
- #ERR\_NOMASKBRUSH**  
Dieser Pinsel hat keine Maske! (1172)
- #ERR\_FOREVERWOREPEAT**  
FOREVER ohne REPEAT! (1173)
- #ERR\_FINDBRUSH**  
Konnte Pinsel %ld nicht finden! (1174)
- #ERR\_FINDTEXTOBJECT**  
Konnte Textobjekt %ld nicht finden! (1175)
- #ERR\_FINDANIM**  
Konnte Animation %ld nicht finden! (1176)
- #ERR\_FINDBGPIC**  
Konnte Hintergrundbild %ld nicht finden! (1177)
- #ERR\_FINDSAMPLE**  
Konnte Sample %ld nicht finden! (1178)
- #ERR\_FINDFILE**  
Konnte Datei %ld nicht finden! (1179)
- #ERR\_FINDMEMBLK**  
Konnte Speicherblock %ld nicht finden! (1180)
- #ERR\_FINDTIMER**  
Konnte Timer %ld nicht finden! (1181)
- #ERR\_FINDMOVE**  
Konnte Bewegungsliste %ld nicht finden! (1182)
- #ERR\_STORNUM**  
Es wurde ein String oder eine Zahl als Argument %ld erwartet! (1183) (Beachten Sie: Mit String ist eine Zeichenkette/Zeichenfolge gemeint)

- #ERR\_PERCENTFORMAT**  
Ungültiges Prozentformat in Argument %ld! (1184)
- #ERR\_FUNCEXPECTED**  
Es wurde eine Funktion als Argument %ld erwartet! (1185)
- #ERR\_UNMPARENTHESSES**  
Unausgeglichene Klammerung! (1186)
- #ERR\_WRONGOPCST**  
Dieser Operator kann hier nicht benutzt werden! (1187)
- #ERR\_FINDBUTTON**  
Konnte Knopf %ld nicht finden! (1188)
- #ERR\_NUMTABLEARG**  
Es wurde eine Zahl im Tableargument "%s" erwartet! (1189) (Beachten Sie: Table ist eine Tabelle)
- #ERR\_NUMCALLBACK**  
Callback-Funktion muss eine Zahl zurückgeben! (1190)
- #ERR\_BGPICBUTTON**  
Das Hintergrundbild muss aktiv sein während Knopffunktionen aufgerufen werden! (1191)
- #ERR\_WRONGHEX**  
Ungültige hexadezimale Angabe! (1192)
- #ERR\_TOOMANYARGS**  
Zu viele Argumente für diese Funktion! (1193)
- #ERR\_FINDINTERVAL**  
Konnte Interval %ld nicht finden! (1194)
- #ERR\_FINDTIMEOUT**  
Konnte Timeout %ld nicht finden! (1195)
- #ERR\_LOADSOUND**  
Fehler beim Laden des Samples! (1196)
- #ERR\_STRINGEXPECTED**  
Es wurde ein String erwartet! (1197) (Beachten Sie: Mit String ist eine Zeichenkette/Zeichenfolge gemeint)
- #ERR\_UNEXPECTEDEOF**  
Unerwartetes Dateiende! (1198)
- #ERR\_VMMISMATCH**  
Virtual Machine Datentyp passt nicht! (1199)
- #ERR\_BADINTEGER**  
Fehlerhafter Integer im Bytecode! (1200) (Beachten Sie: Integer ist eine Ganzzahl)
- #ERR\_BADUPVALUES**  
Fehlerhafte Upvalues im Bytecode! (1201)

**#ERR\_BADCONSTANT**  
Fehlerhafter Konstantentyp im Bytecode!! (1202)

**#ERR\_BADBYTECODE**  
Fehlerhafter Bytecode! (1203)

**#ERR\_BADSIGNATURE**  
Fehlerhafte Bytecodesignatur! (1204)

**#ERR\_UNKNUMFMT**  
Unbekanntes Zahlenformat im Bytecode! (1205)

**#ERR\_INVNEXTKEY**  
Ungültiger Schlüssel für das nächste Tableelement! (1206) (Beachten Sie: Table ist eine Tabelle)

**#ERR\_TABLEOVERFLOW**  
Tableüberlauf! (1207) (Beachten Sie: Table ist eine Tabelle)

**#ERR\_TABLEINDEX**  
Tableindex ist keine Zahl! (1208) (Beachten Sie: Table ist eine Tabelle)

**#ERR\_APPLETVERSION**  
Dieses Applet benötigt mindestens Hollywood %s! (1209)

**#ERR\_UNKNOWNSEC**  
Unbekannte Sektion im Applet! (1210)

**#ERR\_NOAPPLET**  
%s ist kein Hollywood-Applet! (1211)

**#ERR\_PLAYERCOMP**  
Kompilieren ist nicht möglich mit dem HollywoodPlayer! (1212)

**#ERR\_FILEEXIST**  
Datei %s existiert nicht! (1213)

**#ERR\_MAGICKEY**  
Kann Einsprungspunkt im Player nicht finden! (1214)

**#ERR\_FINDCLIPREGION**  
Konnte Clipregion %ld nicht finden! (1215)

**#ERR\_FUNCREMOVED**  
Diese Funktion wird nicht mehr unterstützt! (1216)

**#ERR\_COORDSRANGE**  
Die angegebenen Koordinaten sind außerhalb des zulässigen Bereichs! (1217)

**#ERR\_BADDIMENSIONS**  
Breiten- und Höhenangaben müssen größer als 0 sein! (1218)

**#ERR\_FINDSPRITE**  
Konnte Sprite %ld nicht finden! (1219)

**#ERR\_SPRITEONSCREEN**  
Sprite %ld ist nicht auf dem Bildschirm! (1220)



**#ERR\_PREPROCSYM**  
Unbekannter Präprozessorbefehl @%s! (1221) (Beachten Sie: Präprozessorbefehl ist das gleiche wie Präprozessor-Anweisung)

**#ERR\_UNKNOWNTAG**  
Unbekannte Option "%s"! (1222)

**#ERR\_MASKALPHA**  
Maske und Alphakanal können nicht zusammen benutzt werden! (1223)

**#ERR\_NOSPRITES**  
Bitte entfernen Sie zuerst alle Sprites! (1224)

**#ERR\_WRONGCLIPREG**  
Clipregion passt nicht zur Größe des Ausgabegeräts! (1225)

**#ERR\_NOCLIPREG**  
Bitte deaktivieren Sie die Clipregion bevor Sie die Ebenen anschalten! (1226)

**#ERR\_MODIFYSPRITE**  
Sprites, die sich gerade auf dem Bildschirm befinden, können nicht verändert werden! (1227)

**#ERR\_MODIFYSPRITE2**  
Verknüpfte Sprites können nicht verändert werden! (1228)

**#ERR\_ENDDOUBLEBUFFER**  
Bitte beenden Sie zuerst den Doublebuffermodus! (1229) (Beachten Sie: Doublebuffermodus ist der Doppelpuffermodus)

**#ERR\_DBTRANSWIN**  
Doublebuffering wird momentan nicht für transparente Bilder unterstützt! (1230) (Beachten Sie: Doublebuffering bedeutet Doppelpuffer)

**#ERR\_FINDMUSIC**  
Konnte Musik %ld nicht finden! (1231)

**#ERR\_MUSNOTPLYNG**  
Musik %ld spielt momentan nicht! (1232)

**#ERR\_SEEKRANGE**  
Angegebene Position ist außerhalb des gültigen Bereichs! (1233)

**#ERR\_MIXMUSMOD**  
Musik und Protrackermodule können nicht zur selben Zeit spielen! (1234)

**#ERR\_UNKNOWNMUSFMT**  
Unbekanntes Musikformat! (1235)

**#ERR\_MUSFMTSUPPORT**  
Aktuelles Musikformat unterstützt diese Funktion nicht! (1236) (Beachten Sie: Mit Funktion ist hier ein Befehl von Hollywood gemeint)

**#ERR\_TABLEORNIL**  
Es wurde ein Table oder Nil erwartet! (1237) (Beachten Sie: Table ist eine Tabelle)

**#ERR\_PROTMETATABLE**

Kann einen geschützten Metatable nicht verändern! (1238) (Beachten Sie: Metatable ist eine Metatabelle)

**#ERR\_ERRORCALLED**

Nicht definiert (1239)

**#ERR\_ADDTASK**

Fehler beim Installieren eines Tasks! (1240)

**#ERR\_TASKSETUP**

Fehler bei der Taskeinrichtung! (%s) (1241)

**#ERR\_READRANGE**

Kann nicht über das Dateiende hinaus lesen! (1242)

**#ERR\_BACKFILL**

Falsche Backfill-Konfiguration! (1243) (Beachten Sie: Backfill ist der Hintergrund)

**#ERR\_NODOUBLEBUFFER**

Doublebuffer ist nicht aktiv! (1244) (Beachten Sie: Doublebuffer ist der Doppelpuffer)

**#ERR\_STRTOOSHORT**

Angegebene Länge übertrifft die Stringlänge! (1245) (Beachten Sie: Mit String ist eine Zeichenkette/Zeichenfolge gemeint)

**#ERR\_CACHEERROR**

Fehler während der Bearbeitung des Gfx-Cache! (1246)

**#ERR\_STRTABLEARG**

Es wurde ein String im Tableargument "%s" erwartet! (1247) (Beachten Sie: Mit String ist eine Zeichenkette/Zeichenfolge gemeint)

**#ERR\_APPLET**

Es wurde keine Appletdatei übergeben! (1248)

**#ERR\_KEYFILE**

Fehler beim Laden des Keyfiles! (1249)

**#ERR\_NOTADIR**

%s ist kein Verzeichnis! (1250)

**#ERR\_UNKTEXTFMT**

Unbekanntes Textformattag nach eckiger Klammer! (1251)

**#ERR\_TEXTSYNTAX**

Syntaxfehler im Textformat! (1252)

**#ERR\_TEXTARG**

Nicht genügend Argumente für dieses Textformattag übergeben! (1253)

**#ERR\_DEFFONT**

Fehler beim Öffnen der Standardschrift! (1254)

**#ERR\_ANTIALIAS**  
Dieser Schrifttyp erlaubt keine Antialias-Ausgabe! (1255)

**#ERR\_CREATEPORT**  
Konnte Nachrichtenport nicht erzeugen! (1256)

**#ERR\_NOREXX**  
Der ARexx Server läuft nicht! (1257)

**#ERR\_REXXERR**  
Fehler vom REXX Interpreter! (%s) (1258)

**#ERR\_STRCALLBACK**  
Diese Callbackfunktion muss eine Zeichenkette zurückgeben!! (1259)

**#ERR\_PORTNOTAVAIL**  
Es gibt schon einen Port mit dem Namen %s! (1260)

**#ERR\_BAD8SVX**  
Fehlerhafte Daten im IFF 8SVX oder IFF 16SV Ton! (1261)

**#ERR\_CMPUNSUPPORTED**  
Diese Tondatei benutzt ein unbekanntes Kompressionsformat! (1262)

**#ERR\_BADWAVE**  
Fehlerhafte Daten im RIFF WAVE Ton! (1263)

**#ERR\_MUSNOTPAUSED**  
Diese Musik ist momentan nicht pausiert! (1264)

**#ERR\_CONFIG2**  
Nicht definiert (1265)

**#ERR\_EXETYPE**  
Unbekannter Programmtyp angegeben! (1266)

**#ERR\_OPENAUDIO**  
Kann Tonausgabegerät nicht öffnen! (1267)

**#ERR\_DATATYPESAVE**  
Der angegebene Datatype unterstützt kein Speichern! (1268)

**#ERR\_DATATYPESAVE2**  
Fehler beim Abspeichern über Datatype! (1269)

**#ERR\_LOADFRAME**  
Fehler beim Laden des Einzelbildes! (1270)

**#ERR\_LAYERSUPPORT2**  
Diese Funktion kann nicht benutzt werden, wenn Layers aktiviert sind! (1271)  
(Beachten Sie: Mit Layers sind Ebenen gemeint)

**#ERR\_SHORTIF**  
Das kurze IF-Statement muss sich auf einer einzigen Zeile befinden! (1272)

**#ERR\_SYSTOOOLD**  
Ihre Hollywood.sys Version ist zu alt! (1273)

- #ERR\_KEYNOTFOUND**  
Der Schlüssel "%s" wurde nicht in der Systembasis gefunden! (1274)
- #ERR\_FINDPORT**  
Der Port "%s" konnte nicht gefunden werden! (1275)
- #ERR\_TOOSMALL2**  
Der aktuelle Bildschirm ist nicht groß genug um die Auflösung %s darzustellen!  
(1276)
- #ERR\_SAVEPNG**  
Fehler beim Speichern des PNG-Bildes! (1277)
- #ERR\_NOTIGER**  
Hollywood benötigt mindestens Version 10.4 (Tiger) von macOS! (1278)
- #ERR\_STREAMASSAMPLE**  
Streamingformate können nicht als Sample geladen werden! (1279)
- #ERR\_AUDIOCONVERTER**  
Fehler beim Erstellen des Audioconverters für dieses Format! (1280)
- #ERR\_RENDERCALLBACK**  
Fehler beim Installieren der Rendercallback auf diesem Mixerbus! (1281)
- #ERR\_SETFILEATTR**  
Fehler beim Setzen der Dateiattribute! (1282)
- #ERR\_SETFILEDATE**  
Fehler beim Setzen des Dateidatums! (1283)
- #ERR\_SETFILECOMMENT**  
Fehler beim Setzen des Dateikommentars! (1284)
- #ERR\_INVALIDDATE**  
Ungültiges Datumsformat verwendet! (1285)
- #ERR\_LOCK2**  
Fehler beim Zugriff auf %s!! (1286)
- #ERR\_THREAD**  
Konnte Thread nicht einrichten! (1287)
- #ERR\_UNSUPPORTEDFEAT**  
Dieses Feature wird momentan nicht auf dieser Plattform unterstützt! (1288)
- #ERR\_NOCHANNEL**  
Konnte keinen Audiokanal für diesen Ton reservieren! (1289)
- #ERR\_CREATEEVENT**  
Fehler beim Erstellen eines unbenannten Ereignisobjekts! (1290)
- #ERR\_DSOUNDNOTIFY**  
Konnte Soundbenachrichtungsschnittstelle nicht erhalten! (1291)
- #ERR\_DSOUNDNOTIPOS**  
Fehler beim Setzen der Soundpufferpositionen! (1292)

**#ERR\_DSOUNDPLAY**  
Fehler beim Abspielen des Tons! (1293)

**#ERR\_AFILEPROP**  
Fehler beim Abfragen der Audiodateieigenschaften! (1294)

**#ERR\_DIRECTSHOW**  
Fehler beim Einrichten von DirectShow! (Code: #%ld) (1295)

**#ERR\_REGCLASS**  
Fehler beim Registrieren der Fensterklasse! (1296)

**#ERR\_TIMER**  
Fehler beim Einrichten der Timerfunktion! (1297)

**#ERR\_SEMAPHORE**  
Fehler beim Reservieren der Semaphore! (1298)

**#ERR\_8OR16BITONLY**  
Hollywood unterstützt momentan nur Töne der Bittiefe 8 und 16! (1299)

**#ERR\_DISPMINIMIZED**  
Diese Funktion kann nicht benutzt werden, während Hollywood minimiert ist!  
(1300) (Beachten Sie: Mit Funktion ist hier ein Befehl von Hollywood gemeint)

**#ERR\_COMMODITY**  
Fehler beim Erstellen des Commodities! (1301)

**#ERR\_MSGPORT**  
Fehler beim Einrichten des Nachrichtenports! (1302)

**#ERR\_TEXTCONVERT**  
Fehler beim Konvertieren des Textes nach Unicode! (1303)

**#ERR\_ATSUI**  
Fehler bei der Textoperation (ATSUI)! (1304)

**#ERR\_LFSYNTAX**  
Syntaxfehler in der Linkfile-Datenbank! (1305)

**#ERR\_ZLIBIO**  
Ein Zlib-IO-Fehler ist aufgetreten! (1306)

**#ERR\_ZLIBSTREAM**  
Ein Zlib-Stream-Fehler ist aufgetreten! (1307)

**#ERR\_ZLIBVERSION**  
Falsche zlib Version vorhanden! (1308)

**#ERR\_ZLIBDATA**  
Kaputte oder nicht vollständige zlib-Daten! (1309)

**#ERR\_PAKFORMAT**  
Unbekanntes Packformat! (1310)

**#ERR\_NOTXTLAYER**  
Angegebene Ebene ist keine Textebene! (1311)

**#ERR\_DDAUTOSCALE**

Autoscale kann nicht zusammen mit DisplayDesktop verwendet werden! (1312)  
(Beachten Sie: Autoscale ist Autoskalierung)

**#ERR\_NODISLAYERS**

Layers können nicht abgeschaltet werden, während die Layerscale-Engine aktiv ist! (1313) (Beachten Sie: Mit Layers sind Ebenen und mit Layerscale ist die Ebenenskalierung gemeint)

**#ERR\_LOCKEDOBJ**

Gesperrte Objekte können nicht verändert werden! (1314)

**#ERR\_WRITEJPEG**

Fehler beim Schreiben des JPEG-Bilds! (1315)

**#ERR\_DDRECVIDEO**

Skripte, die DisplayDesktop benutzen, können nicht aufgenommen werden! (1316)

**#ERR\_WRONGCMDRECVIDEO**

Dieser Befehl kann im Videorecorder-Modus nicht benutzt werden! (1317)

**#ERR\_FINDDIR**

Kann Verzeichnishandle %ld nicht finden! (1318)

**#ERR\_MUSNOTPLYNG2**

Diese Musik spielt momentan nicht! (1319)

**#ERR\_FINDPOINTER**

Konnte Mauszeiger %ld nicht finden! (1320)

**#ERR\_POINTERIMG**

Fehler beim Erstellen des Mauszeigers! (1321)

**#ERR\_READFUNC**

An dieser Position befindet sich keine Hollywood-Funktion! (1322)

**#ERR\_BADBASE64**

Ungültige Base64-Kodierung! (1323)

**#ERR\_NOHWFUNC**

Die angegebene Funktion ist keine Benutzerfunktion! (1324)

**#ERR\_SPRITEONSCREEN2**

Sprite befindet sich nicht auf dem Bildschirm! (1325)

**#ERR\_FINDASYNCDRAW**

Konnte die asynchrone Zeichenfunktion %ld nicht finden! (1326)

**#ERR\_FREECURPOINTER**

Aktuelle Mauszeiger dürfen nicht freigegeben werden! (1327)

**#ERR\_READTABLE**

Keine Hollywood-Tabelle an dieser Position gefunden! (1328)

**#ERR\_LAYERSWITCH**

Layermodus darf nicht verändert werden während asynchrone Zeichenfunktionen aktiv sind! (1329) (Beachten Sie: Mit Layer sind Ebenen gemeint)

**#ERR\_VIDEOSTRATEGY**  
Unbekannte Videostrategie angegeben! (1330)

**#ERR\_WRONGVSTRATEGY**  
Ungültige Videostrategie-Konfiguration! (1331)

**#ERR\_FINDFONT**  
Kann Schriftart %s nicht im System finden! (1332)

**#ERR\_LINKFONT**  
Die Schriftart %s kann nicht gebunden werden, da der Typ nicht unterstützt wird! (1333)

**#ERR\_FINDFONT2**  
Konnte Schriftart %ld nicht finden! (1334)

**#ERR\_FONTPATH**  
Schriftangabe darf keine Datei sein! (1335)

**#ERR\_FONTFORMAT**  
Schriftart ist in einem nicht unterstützten Format! (1336)

**#ERR\_NOCOORDCST**  
Sie dürfen hier keine Koordinatenkonstanten benutzen! (1337)

**#ERR\_ANIMDISK**  
Diese Funktion kann nicht mit disk-gepufferten Animationen verwendet werden! (1338) (Beachten Sie: Mit Funktion ist hier ein Befehl von Hollywood gemeint)

**#ERR\_SELECTANIM**  
Dieser Befehl kann nicht benutzt werden während SelectBrush() aktiviert ist! (1339)

**#ERR\_MODIFYAANIM**  
Sie dürfen die aktuell ausgewählte Animation nicht verändern! (1340)

**#ERR\_FINDANIMSTREAM**  
Konnte Animatenstrom %ld nicht finden! (1341)

**#ERR\_NEEDMORPHOS2**  
Diese Funktion benötigt mindestens MorphOS 2.0! (1342) (Beachten Sie: Mit Funktion ist hier ein Befehl von Hollywood gemeint)

**#ERR\_SMODEALPHA**  
Aktueller Bildschirmmodus unterstützt keine Fenster mit Alphatransparenz! (1343)

**#ERR\_FINDDISPLAY**  
Konnte Anzeige %ld nicht finden! (1344) (Beachten Sie: Mit Anzeige ist Display gemeint)

**#ERR\_MULTIBGPIC**  
Ein Hintergrundbild darf nicht mit mehreren Anzeigen verknüpft werden! (1345) (Beachten Sie: Mit Anzeigen sind Displays gemeint)

**#ERR\_FREEADISPLAY**

Aktuelle Anzeige darf nicht freigegeben werden! (1346) (Beachten Sie: Mit Anzeige ist Display gemeint)

**#ERR\_CLOSEDISPLAY**

Diese Funktion kann nicht benutzt werden, während die aktuelle Anzeige versteckt ist! (1347) (Beachten Sie: Mit Funktion ist hier ein Befehl von Hollywood und mit Anzeige das Display gemeint)

**#ERR\_ADDAPPICON**

Fehler beim Hinzufügen eines Piktogramms zur Workbench! (1348)

**#ERR\_SCREENSIZE**

Bildschirmmodus %s wird vom aktuellen Monitor nicht unterstützt! (1349)

**#ERR\_DIFFDEPTH**

Kann Bildmodus nicht wechseln wegen Unterschieden in der Farbauflösung! (1350)

**#ERR\_VIDRECMULTI**

Mehrere Anzeigen können nicht benutzt werden, wenn der Videorekorder aktiv ist! (1351) (Beachten Sie: Anzeigen sind Displays)

**#ERR\_VIDRECTTRANS**

Transparente Anzeigen können nicht benutzt werden, wenn der Videorekorder aktiv ist! (1352) (Beachten Sie: Anzeigen sind Displays)

**#ERR\_NEEDOS41**

Diese Funktion benötigt mindestens AmigaOS 4.1! (1353) (Beachten Sie: Mit Funktion ist hier ein Befehl von Hollywood gemeint)

**#ERR\_SYSIMAGE**

Fehler beim Holen eines Systembilds! (1354)

**#ERR\_SYSBUTTON**

Fehler beim Erstellen eines Systemknopfs! (1355)

**#ERR\_OPENANIM2**

Die Animation "%s" ist in einem unbekannten Format! (1356)

**#ERR\_OPENSOUND2**

Die Tondatei "%s" ist in einem unbekannten Format! (1357)

**#ERR\_LOADPICTURE2**

Das Bild "%s" ist in einem unbekannten Format! (1358)

**#ERR\_SIGNAL**

Fehler beim Reservieren eines Signals! (1359)

**#ERR\_ADDAPPWIN**

Fehler beim Hinzufügen eines Appfensters zur Workbench! (1360)

**#ERR\_CLIPFORMAT**

Unbekannte Daten in der Zwischenablage! (1361)

**#ERR\_SORTFUNC**

Ungültige Sortierfunktion übergeben! (1362)



**#ERR\_INISYNTAX**  
Syntaxfehler in der Konfigurationsdatei! (1363)

**#ERR\_CLIPOPEN**  
Kann die Zwischenablage nicht öffnen! (1364)

**#ERR\_CLIPREAD**  
Kann nicht aus der Zwischenablage lesen! (1365)

**#ERR\_SCALEBGPICT**  
Die Größe von einem mit einer Anzeige verknüpften Hintergrundbild darf nicht geändert werden! (1366) (Beachten Sie: Anzeige ist ein Display)

**#ERR\_SELECTBGPICT**  
Sie müssen zuerst die mit dem Hintergrundbild verknüpfte Anzeige aktivieren bevor Sie das Hintergrundbild verändern! (1367) (Beachten Sie: Anzeige ist ein Display)

**#ERR\_CLIPWRITE**  
Fehler beim Schreiben in die Zwischenablage! (1368)

**#ERR\_FINDLAYER**  
Die Ebene "%s" wurde im aktuellen Hintergrundbild nicht gefunden! (1369)

**#ERR\_INVINSERT**  
Ungültige Einfügeposition angegeben! (1370)

**#ERR\_ALREADYASYNC**  
Die angegebene Ebene hat schon eine verknüpfte asynchrone Zeichenfunktion! (1371)

**#ERR\_REMADLAYER**  
Ebene kann nicht entfernt werden, solange sie noch von einer asynchronen Zeichenfunktion benutzt wird! (1372)

**#ERR\_NAMETOOLONG**  
Der angegebene Name ist zu lang! (1373)

**#ERR\_GROUPNAMEUSED**  
Der angegebene Gruppenname wurde schon einer Ebene zugewiesen! (1374)

**#ERR\_REGISTRYREAD**  
Fehler beim Lesen des Registrierungsschlüssels %s! (1375)

**#ERR\_REGISTRYWRITE**  
Fehler beim Schreiben des Registrierungsschlüssels %s! (1376)

**#ERR\_SELECTBGPICT2**  
Die Grafikdaten eines mit einer Anzeige verknüpften Hintergrundbilds dürfen nicht verändert werden! (1377) (Beachten Sie: Anzeige ist ein Display)

**#ERR\_MODIFYABGPIC**  
Aktuelles Hintergrundbild darf nicht verändert werden! (1378)

**#ERR\_ADFWRONGDISP**  
Asynchrone Zeichenfunktion ist nicht mit der aktuellen Anzeige verknüpft! (1379) (Beachten Sie: Anzeige ist ein Display)

- #ERR\_ADFFREEDISP**  
Kann Anzeige nicht freigeben bevor nicht alle asynchronen Zeichenfunktionen  
freigegeben wurden! (1380) (Beachten Sie: Anzeige ist ein Display)
- #ERR\_SPRITELINK**  
Spriteverknüpfung kann nicht von einer Spriteverknüpfung erstellt werden!  
(1381)
- #ERR\_WRONGSPRITESIZE**  
Die angegebene Sprites müssen dieselbe Größe haben! (1382)
- #ERR\_TRANSBRUSH**  
Kann Pinsel nicht stutzen, da er vollständig transparent ist! (1383)
- #ERR\_DINPUT**  
Fehler beim Öffnen von DirectInput! (1384)
- #ERR\_JOYSTICK**  
Joystick kann nicht abgefragt werden! (1385)
- #ERR\_FT2** Fehler beim Initialisieren von freetype2! (1386)
- #ERR\_ICONDIMS**  
Angegebene Bilddatei ist nicht von der geforderten Größe (%s)! (1387)
- #ERR\_BRUSHTYPE**  
Diese Operation wird von dem angegebenen Pinseltyp nicht unterstützt! (1388)
- #ERR\_TFVBRUSH**  
Transformierte Pinsel können nicht als Ebene eingefügt werden! Transformieren  
Sie die Ebene statt den Pinsel! (1389)
- #ERR\_BGPICTYPE**  
Diese Operation wird vom angegebenen BGPic-Typ nicht unterstützt! (1390)
- #ERR\_TFVBRUSHBGPIC**  
Kann transformierten Vektorpinsel nicht in ein BGPic konvertieren! (1391)
- #ERR\_TFVBGPICBRUSH**  
Kann transformiertes Vektor-BGPic nicht in einen Pinsel konvertieren! (1392)
- #ERR\_FINDPATH**  
Pfad %ld konnte nicht gefunden werden! (1393)
- #ERR\_EMPTYPATH**  
Angegebener Pfad ist leer! (1394)
- #ERR\_VFONTTYPE**  
Sie müssen den eingebauten Schrifttreiber benutzen, um Vektortext zu erzeugen!  
(1395)
- #ERR\_VFONT**  
Fehler beim Einrichten der Vektorschrift! (1396)
- #ERR\_CREATESHORTCUT**  
Fehler beim Erstellen der Verknüpfung! (1397)

**#ERR\_NOACCESS**  
Zugriff verweigert! (1398)

**#ERR\_BADPLATFORM**  
Kompilieren für die Plattform "%s" wird von dieser Version nicht unterstützt!  
(1399)

**#ERR\_NEWHWPLUGIN**  
Dieses Plugin benötigt mindestens Hollywood %s! (1400)

**#ERR\_PLUGINVER**  
Benötige mindestens die Version %s! (1401)

**#ERR\_PLUGINARCH**  
Das Plugin ist inkompatibel mit der aktuellen Plattform (%s)! (1402)

**#ERR\_IMAGEERROR**  
Fehler in den Bilddaten in der Datei %s! (1403)

**#ERR\_RENDERADLAYER**  
Kann Ebene nicht zeichnen, weil sie mit einem asynchronen Zeichenobjekt verknüpft wurde! (1404)

**#ERR\_NOJOYATPORT**  
An diesem Port konnte kein Joystick gefunden werden! (1405)

**#ERR\_DEMO**  
Dieses Feature ist nicht in der Demoversion von Hollywood enthalten! (1406)

**#ERR\_DEMO2**  
In der Demoversion ist die Skriptgröße auf max. 800 Zeilen bzw. 32 Kilobyte beschränkt! (1407)

**#ERR\_DEMO3**  
Diese Demoversion ist abgelaufen! Wenn Sie das Programm weiterhin benutzen möchten, kaufen Sie bitte die Vollversion! (1408)

**#ERR\_FINDCLIENT**  
Konnte Verbindung %ld nicht finden! (1409)

**#ERR\_SOCKET**  
Der folgende Netzwerk-Fehler ist aufgetreten: %s (1410)

**#ERR\_OPEN\_SOCKET**  
Konnte Socket-Schnittstelle nicht initialisieren! (1411)

**#ERR\_FINDSERVER**  
Konnte Server %ld nicht finden! (1412)

**#ERR\_SOCKETOPT**  
Fehler beim Setzen der Socket-Optionen! (1413)

**#ERR\_PEERNAME**  
Fehler beim Abfragen des Peernamen! (1414)

**#ERR\_HOSTNAME**  
Fehler beim Abfragen des Hostnamen! (1415)

#ERR\_UNKPROTOCOL  
Unbekanntes Protokoll in URL! (1416)

#ERR\_BADURL  
Ungültige URL angegeben! (1417)

#ERR\_HTTPERROR  
Der HTTP-Fehler %ld ist aufgetreten! (1418)

#ERR\_HTTPTE  
HTTP-Transfermodus wird nicht unterstützt! (1419)

#ERR\_SENDDATA  
Ein Fehler ist während der Datenübertragung aufgetreten! (1420)

#ERR\_FTPERROR  
Der FTP-Fehler %ld ist aufgetreten! (1421)

#ERR\_RECVTIMEOUT  
Empfangstimeout überschritten! (1422)

#ERR\_RECVCLOSED  
Der entfernte Server hat die Verbindung getrennt! (1423)

#ERR\_RECVUNKNOWN  
Ein unbekannter Fehler ist aufgetreten während des Datenempfangs! (1424)

#ERR\_FILENOTFOUND  
Die Datei %s wurde auf diesem Server nicht gefunden! (1425)

#ERR\_FTPAUTH  
Zugriff verweigert mit dem angegebenen Benutzernamen/Passwort! (1426)

#ERR\_UPLOADFORBIDDEN  
Keine Erlaubnis die Datei auf %s hochzuladen! (1427)

#ERR\_SOCKNAME  
Fehler beim Abfragen des Socketnamen! (1428)

#ERR\_FINDUDPOBJECT  
Konnte UDP-Objekt %ld nicht finden! (1429)

#ERR\_BADIP  
Ungültige IP-Adresse angegeben! (1430)

#ERR\_XDISPLAY  
Fehler beim Verbinden mit dem X-Server! (1431)

#ERR\_CREATEGC  
Fehler beim Erstellen des Grafikkontext! (1432)

#ERR\_PIPE  
Fehler beim Erstellen der Pipe! (1433)

#ERR\_GTK  
Konnte GTK nicht öffnen! (1434)

#ERR\_NEEDCOMPOSITE  
Compositing muss eingeschaltet sein für Fenster mit Alphatransparenz! (1435)

**#ERR\_NOARGBVISUAL**  
Fehler beim Einrichten einer Visualinfo für ARGB-Grafiken! (1436)

**#ERR\_XFIXES**  
Die Xfixes-Erweiterung wird für dieses Feature benötigt! (1437)

**#ERR\_XCURSOR**  
Die Xcursor-Erweiterung wird für dieses Feature benötigt! (1438)

**#ERR\_ALSAPCM**  
Fehler beim Einrichten des ALSA-Systems! (#%ld) (1439)

**#ERR\_SETENV**  
Fehler beim Setzen der Umgebungsvariable! (1440)

**#ERR\_UNSETENV**  
Fehler beim Entfernen der Umgebungsvariable! (1441)

**#ERR\_XF86VIDMODEEXT**  
Wechseln des Bildschirmmodus benötigt die XFree86-VidModeExtension!  
(1442)

**#ERR\_NODISPMODES**  
Keine Anzeigemodi gefunden! (1443)

**#ERR\_UNKNOWNFILTER**  
Der Filter "%s" wurde nicht erkannt! (1444)

**#ERR\_NOFILTERNAME**  
Unbekannter Filtername im Table-Feld %s! (1445) (Beachten Sie: Table ist eine  
Tabelle)

**#ERR\_TABEXPECTED2**  
Untertable erwartet im Table "%s"! (1446) (Beachten Sie: Table ist eine Tabelle)

**#ERR\_SMPRANGE**  
Der angegebene Sample-Wert ist außerhalb des gültigen Bereichs! (1447)

**#ERR\_NOTENOUGHPIXELS**  
Table enthält nicht genügend Pixel für die angegebene Größe! (1448) (Beachten  
Sie: Table ist eine Tabelle)

**#ERR\_FINDVIDEO**  
Konnte Video %ld nicht finden! (1449)

**#ERR\_LOADVIDEO**  
Die Datei "%s" ist in einem unbekannten Videoformat! (1450)

**#ERR\_VIDNOTPLAYING**  
Kann Video nicht pausieren, da es nicht spielt! (1451)

**#ERR\_VIDNOTPAUSED**  
Kann Video nicht wiederaufnehmen, da es nicht pausiert ist! (1452)

**#ERR\_COLORSPACE**  
Fehler beim Abfragen des Farbraums! (1453)

**#ERR\_QUICKTIME**

Diese Funktion benötigt QuickTime! (1454) (Beachten Sie: Mit Funktion ist hier ein Befehl von Hollywood gemeint)

**#ERR\_VIDATTACHED**

Diese Funktion kann nicht benutzt werden, während Videos abgespielt werden! (1455) (Beachten Sie: Mit Funktion ist hier ein Befehl von Hollywood gemeint)

**#ERR\_FGRABVIDSTATE**

Kann Einzelbild nicht kopieren während das Video spielt oder pausiert ist! (1456)

**#ERR\_VIDEOFRAME**

Das angegebene Einzelbild ist außerhalb des gültigen Bereichs! (1457)

**#ERR\_VIDEOTRANS**

Videos können nicht auf transparenten BGPics abgespielt werden! (1458)

**#ERR\_LOADPLUGIN**

Fehler beim Laden des Plugins "%s"! (1459)

**#ERR\_VECGFXPLUGIN**

Diese Funktion benötigt ein Vectorgraphics-Plugin! (1460) (Beachten Sie: Mit Funktion ist hier ein Befehl von Hollywood gemeint)

**#ERR\_INVCAPIDX**

Ungültiger Capture-Index! (1461) (Beachten Sie: Mit Capture ist hier Übereinstimmung gemeint)

**#ERR\_INVPATCAP**

Ungültiges Pattern-Capture! (1462) (Beachten Sie: Mit Capture ist hier Übereinstimmung gemeint)

**#ERR\_MALFORMPAT1**

Ungültiges Pattern! (endet mit "%%") (1463)

**#ERR\_MALFORMPAT2**

Ungültiges Pattern! (vermisse "]"") (1464)

**#ERR\_UNBALANCEDPAT**

Unausgeglichenes Pattern! (1465)

**#ERR\_TOOMANYCAPTURES**

Zu viele Captures! (1466) (Beachten Sie: Mit Capture ist hier Übereinstimmung gemeint)

**#ERR\_MISSINGOPBRACK**

Vermisse "[" nach "%%f" in Pattern! (1467)

**#ERR\_UNFINISHEDCAPTURE**

Unfertiges Capture! (1468) (Beachten Sie: Mit Capture ist hier Übereinstimmung gemeint)

**#ERR\_TFIMAGE**

Fehler beim Transformieren des Bildes! (1469)

- #ERR\_DRAWPATH**  
Fehler beim Zeichnen des Pfads! (1470)
- #ERR\_MOBILE**  
Dieser Befehl ist nicht verfügbar in der mobilen Version von Hollywood! (1471)
- #ERR\_DDMOBILE**  
DisplayDesktop-Skripte werden nicht unterstützt für mobile Geräte! (1472)
- #ERR\_MULDISMOBILE**  
Mehrere Fenster werden nicht unterstützt auf mobilen Geräten! (1473)
- #ERR\_TRANSBGMOBILE**  
Transparente BGPics werden nicht unterstützt auf mobilen Geräten! (1474)
- #ERR\_MODIFYPSMP**  
Kann Sample nicht verändern, da es gerade spielt! (1475)
- #ERR\_TABCALLBACK**  
Callback-Funktion muss einen Table zurückgeben! (1476) (Beachten Sie: Table ist eine Tabelle)
- #ERR\_BADCALLBACKRET**  
Ungültiger Callback-Rückgabewert! (1477)
- #ERR\_NOCALLBACK**  
Dieser Befehl darf nicht von einer Callback-Funktion aufgerufen werden! (1478)
- #ERR\_LOWFREQ**  
Die angegebene Frequenz ist zu niedrig! (1479)
- #ERR\_FINDLAYERDATA**  
Element "%s" wurde in der angegebenen Ebene nicht gefunden! (1480)
- #ERR\_NODIRPATTERN**  
Filtermuster können nur mit Verzeichnissen verwendet werden! (1481)
- #ERR\_SEEKFORMAT**  
Dieses Dateiformat unterstützt nicht die Seek-Funktion! (1482) (Beachten Sie: Mit Seek ist eine bestimmte Sprungposition gemeint)
- #ERR\_PLUGIN\_TYPE**  
Unbekannter Plugin-Typ! (%s) (1483)
- #ERR\_NOMUSICCB**  
Dieser Befehl darf nur in einer Musik-Callback aufgerufen werden! (1484)
- #ERR\_NOFMBHANDLER**  
Sie müssen zuerst einen "FillMusicBuffer" Eventhandler installieren! (1485)  
(Beachten Sie: Eventhandler ist ein Ereignis-Handler)
- #ERR\_UNKNOWNIMGOUT**  
Unbekanntes Bildformat angegeben! (1486)
- #ERR\_SAVEIMAGE**  
Fehler beim Speichern des Bildes! (1487)

**#ERR\_UNKNOWNANMOUT**  
Unbekanntes Animformat angegeben! (1488)

**#ERR\_SAVEANIM**  
Fehler beim Speichern der Animation! (1489)

**#ERR\_UNKNOWNSMPOUT**  
Unbekanntes Sampleformat angegeben! (1490)

**#ERR\_SAVESAMPLE**  
Fehler beim Speichern des Samples! (1491)

**#ERR\_UDEXPECTED**  
Userdata wurde in Argument %ld erwartet! (1492)

**#ERR\_ASSERTFAILED**  
Assert-Bedingung nicht erfüllt! (1493)

**#ERR\_REQUIREPLUGIN**  
Dieses Programm benötigt %s! (1494)

**#ERR\_NOABSPATH**  
Absolute Pfadangaben sind hier nicht erlaubt! (1495)

**#ERR\_FINDOBJECTDATA**  
Element "%s" wurde in dem angegebenen Objekt nicht gefunden! (1496)

**#ERR\_HWBRUSH**  
Hardware-Pinsel können hier nicht benutzt werden! (1497)

**#ERR\_HWBRUSHFUNC**  
Diese Funktionalität wird für Hardware-Pinsel momentan nicht unterstützt!  
(1498)

**#ERR\_SAVERALPHA**  
Dieses Format unterstützt keinen Alphakanal! (1499)

**#ERR\_VIDPAUSED**  
Video ist pausiert. ResumeVideo() zum Fortsetzen benutzen! (1500)

**#ERR\_VIDPLAYING**  
Das Video spielt schon! (1501)

**#ERR\_PERCENTFORMATSTR**  
Ungültiges Prozentformat im Table-Argument "%s"! (1502) (Beachten Sie: Table ist eine Tabelle)

**#ERR\_SCRPIXFMT**  
Dieser Bildschirm benutzt ein inkompatibles Pixelformat! (1503)

**#ERR\_SATFREEDISP**  
Kann Display nicht freigeben bevor dessen zugeordnete Satelliten entfernt wurden! (1504)

**#ERR\_CREATEICON**  
Kann kein Piktogramm aus diesem Bild erstellen! (1505)



**#ERR\_GETSHORTCUT**  
Fehler beim Abfragen des vollständigen Pfades der Shortcut-Datei! (1506)

**#ERR\_UNKNOWNMIMETYPE**  
Unbekannter MIME-Typ bei der Dateieindung \*.%! (1507)

**#ERR\_NOMIMEVIEWER**  
Kann keinen Anzeiger für die Dateieindung \*.%s finden! (1508)

**#ERR\_JAVA**  
Kann Thread nicht an die Java VM anbinden! (1509)

**#ERR\_FINDACTIVITY**  
Kann Activity \"%s\" nicht finden! (1510)

**#ERR\_BEGINREFRESH**  
Dieser Befehl kann nicht im BeginRefresh()-Modus benutzt werden! (1511)

**#ERR\_DBVIDEOPLAYER**  
Videoobjekt wird schon als Layer in einem BGPic verwendet! (1512) (Beachten Sie: Mit Layer ist eine Ebenen gemeint)

**#ERR\_VIDEOPLAYER**  
Diese Funktion wird nicht von Videoplayern unterstützt! (1513) (Beachten Sie: Mit Layern sind Ebenen gemeint und mit Funktion ist hier ein Befehl von Hollywood gemeint)

**#ERR\_VIDEOPLAYERDRV**  
Videoplayer funktionieren nur mit Hollywoods plattform-unabhängigem Videoreiber! (1514) (Beachten Sie: Mit Layern sind Ebenen gemeint und mit Funktion ist hier ein Befehl von Hollywood gemeint)

**#ERR\_BADLAYERTYPE**  
Der angegebene Layertyp unterstützt diese Funktionalität nicht! (1515) (Beachten Sie: Mit Layer ist eine Ebenen gemeint)

**#ERR\_VIDSTOPPED**  
Das Video wurde schon angehalten! (1516)

**#ERR\_VIDLAYERFUNC**  
Benutzen Sie Funktionen der Layers-Bibliothek, um Attribute von Videoplayern zu ändern! (1517) (Beachten Sie: Mit Layern sind Ebenen gemeint und mit Funktion ist hier ein Befehl von Hollywood gemeint)

**#ERR\_SETADAPTER**  
Konnte den Adapter nicht installieren! (1518)

**#ERR\_DISPLAYADAPTERSUPPORT**  
Diese Funktionalität wird von diesem Display-Adapter nicht unterstützt! (1519)

**#ERR\_DLOPEN**  
Kann Plugin %s nicht laden! (1520)

**#ERR\_PLUGINSYMBOL**  
Fehler beim Auflösen des Plugin-Symbols: %s (1521)

- #ERR\_BITMAP**  
Fehler beim Reservieren der Bitmap! (1522)
- #ERR\_SATELLITE**  
Diese Funktionalität ist nicht verfügbar, wenn Display-Satelliten benutzt werden! (1523)
- #ERR\_READVIDEOPIXELS**  
Fehler beim Lesen der Pixel von der Hardware-Bitmap! (1524)
- #ERR\_HWDBFREEDISP**  
Kann Display nicht freigeben während Hardware-Doublebuffering aktiv ist! (1525) (Beachten Sie: Doublebuffering ist Doppelpuffer)
- #ERR\_HWBMCLOSEDISP**  
Kann keine Hardware-Bitmap reservieren solange das Display noch nicht initialisiert wurde! (1526)
- #ERR\_INCOMPATBRUSH**  
Dieser Hardware-Pinsel ist inkompatibel zu dem aktuellen Display! (1527)
- #ERR\_ADDSYSEVENT**  
Fehler beim Hinzufügen des Ereignisses! (1528)
- #ERR\_SEEKFILE**  
Konnte Dateiposition nicht ändern! (1529)
- #ERR\_CLOSEFILE**  
Fehler beim Schließen der Datei! (1530)
- #ERR\_FINDPLUGIN**  
Kann Plugin %s nicht finden! (1531)
- #ERR\_PLUGINDOUBLET**  
Das Plugin %s wurde schon geladen! (1532)
- #ERR\_APPLICATION**  
Fehler beim Anmelden der Anwendung! (1533)
- #ERR\_NEEDAPPLICATION**  
Diese Funktionalität kann nur von registrierten Anwendungen benutzt werden! (1534)
- #ERR\_FINDAPPLICATION**  
Konnte Anwendung %s nicht finden! (1535)
- #ERR\_SENDMESSAGE**  
Fehler beim Übertragen der Nachricht! (1536)
- #ERR\_FINDMENU**  
Konnte Menü %ld nicht finden! (1537)
- #ERR\_MENUCOMPLEXITY**  
Diese Menüdefinition ist zu komplex! (1538)
- #ERR\_CREATEMENU**  
Fehler beim Erstellen des Menüs! (1539)

**#ERR\_VISUALINFO**  
Konnte Visualinfo nicht abfragen! (1540)

**#ERR\_SETMENU**  
Fehler beim Setzen des Menüs! (1541)

**#ERR\_MENUATTACHED**  
Kann das Menü nicht freigeben während es noch an ein Display angebunden ist! (1542)

**#ERR\_FINDMENUITEM**  
Kann Menüitem %s nicht finden! (1543) (Beachten Sie: Mit Menüitem ist ein Menüpunkt gemeint)

**#ERR\_NOMENU**  
Das angegebene Display besitzt kein Menü! (1544)

**#ERR\_EMPTYMENUTREE**  
Leere Menübäume sind nicht erlaubt! (1545)

**#ERR\_TAGEXPECTED**  
Es wurde ein Tag erwartet! (1546)

**#ERR\_FULLSCREEN**  
Diese Funktion wird im Vollbildmodus nicht unterstützt! (1547) (Beachten Sie: Mit Funktion ist hier ein Befehl von Hollywood gemeint)

**#ERR\_CREATEDOCKY**  
Fehler beim Erstellen des Application-Docky! (1548)

**#ERR\_UPDATEICON**  
Fehler beim Aktualisieren des Docky-Piktogramms! (1549)

**#ERR\_DOUBLEMENU**  
Baum wurde für dieses Menü schon definiert! (1550)

**#ERR\_CONTEXTMENU**  
Kontextmenüs dürfen nur einen einzigen Baum benutzen! (1551)

**#ERR\_VECTORBRUSH**  
Diese Funktion ist nicht für Vektorpinsel verfügbar! (1552) (Beachten Sie: Mit Funktion ist hier ein Befehl von Hollywood gemeint)

**#ERR\_NOCONTEXTMENU**  
Diese Anwendung besitzt kein Kontextmenü! (1553)

**#ERR\_ACCELERATOR**  
Fehler beim Erstellen der Accelerator-Tabelle! (1554)

**#ERR\_FINDMONITOR**  
Konnte Monitor %ld nicht finden! (1555)

**#ERR\_MONITORFULLSCREEN**  
Der Monitor %ld befindet sich schon im Vollbildmodus! (1556)

**#ERR\_MONITORRANGE**  
Der angegebene Monitor ist außerhalb des zulässigen Bereichs! (1557)

- #ERR\_GETMONITORINFO**  
Fehler beim Holen der Monitoreigenschaften! (1558)
- #ERR\_SCREENMODE**  
Kann für dieses Display keinen passenden Bildschirmmodus finden! (1559)
- #ERR\_NOCOMPRESS**  
Der Hollywood Player unterstützt nur komprimierte Applets! (1560)
- #ERR\_GRABSCREEN**  
Fehler beim Lesen der Bildschirmpixel! (1561)
- #ERR\_ALLOCCHANNEL**  
Fehler beim Reservieren eines Audiokanals! (1562)
- #ERR\_REQUIRETAGFMT**  
Syntaxfehler im Tag-Format! (1563)
- #ERR\_ALLOCALPHA**  
Fehler beim Einrichten des Alphakanals! (1564)
- #ERR\_ALLOCMASK**  
Fehler beim Einrichten der Maske! (1565)
- #ERR\_OLDAPPLET**  
Diese Funktion ist nur verfügbar für Applets, die mit Hollywood %s oder besser kompiliert wurden! (1566) (Beachten Sie: Mit Funktion ist hier ein Befehl von Hollywood gemeint)
- #ERR\_MUSPAUSED**  
Die Musik ist pausiert. Benutzen Sie ResumeMusic() um die Wiedergabe fortzusetzen! (1567)
- #ERR\_MUSPLAYING**  
Die Musik spielt schon! (1568)
- #ERR\_CONSOLEARG**  
Ungültiger Parameter für dieses Konsole-Argument! (1569)
- #ERR\_FILESIZE**  
Fehler beim Ermitteln der Dateigröße! (1570)
- #ERR\_STAT**  
Fehler beim Untersuchen des Dateisystemobjekts! (1571)
- #ERR\_REQAUTH**  
Dieser Server benötigt Benutzer-Authentifizierung! (1572)
- #ERR\_MISSINGFIELD**  
Tablefeld "%s" muss angegeben werden! (1573) (Beachten Sie: Table ist eine Tabelle)
- #ERR\_NOTTRANSPARENCY**  
Das Bild "%s" besitzt keinen transparenten Stift! (1574)
- #ERR\_LEGACYPTMOD**  
Der Legacy-Soundtreiber unterstützt nicht das Abspielen mehrerer Protracker-Module! (1575)

**#ERR\_CHANNELRANGE**  
Der angegebene Audiokanal ist außerhalb des gültigen Bereichs! (1576)

**#ERR\_FILEFORMAT**  
Fehler im Dateiformat! (1577)

**#ERR\_LINKPLUGIN**  
Fehler beim Einbinden des Plugins %s! (1578)

**#ERR\_EXECUTE**  
Fehler beim Ausführen des Programms! (1579)

**#ERR\_AMIGAGUIDE**  
Fehler beim Öffnen der AmigaGuide-Datei %s! (1580)

**#ERR\_COMPLEXPATTERN**  
Pattern ist zu komplex! (1581)

**#ERR\_ESCREPLACE**  
Unzulässige Benutzung des Escape-Zeichens im Ersatzstring! (1582) (Beachten Sie: Mit String ist eine Zeichenkette/Zeichenfolge gemeint)

**#ERR\_INVREPLACE**  
Unzulässiger Ersatzwert! (1583)3)

**#ERR\_BADENCODING**  
Unbekannte Textkodierung! (1584)

**#ERR\_INVALIDUTF8**  
Ungültige UTF-8 Zeichenkette! (1585)(1585)

**#ERR\_DIFFENCODING**  
Kann Applet nicht einbinden, da es eine andere Textkodierung benutzt als das aktuelle Skript! (1586)

**#ERR\_DBLENCODING**  
Unzulässige Kombination von Textkodierungen! (1587)

**#ERR\_INVALIDUTF8ARG**  
Ungültiger UTF-8 String in Argument %d! (1588) (Beachten Sie: Mit String ist eine Zeichenkette/Zeichenfolge gemeint)

**#ERR\_CORETEXT**  
Fehler beim Zeichnen des Strings mit Core Text! (1589) (Beachten Sie: Mit String ist eine Zeichenkette/Zeichenfolge gemeint)

**#ERR\_COREFOUNDATION**  
Ein Core Foundation-Fehler ist aufgetreten! (1590)1590)

**#ERR\_FRAMEGRABBER**  
Fehler beim Extrahieren eines Einzelbilds aus dem Video! (1591)

**#ERR\_FINDSELECTOR**  
Kann Selektor %s nicht finden! (1592)

**#ERR\_FIRSTPREPROC**

Präprozessorbefehle zur bedingten Kompilierung müssen am Anfang der Zeile stehen! (1593) (Beachten Sie: Präprozessorbefehl ist das gleiche wie Präprozessor-Anweisung)

**#ERR\_ELSEIFATERELSE**

ELSEIF nach ELSE! (1594)

**#ERR\_ELSETWICE**

ELSE wurde zweimal benutzt! (1595)

**#ERR\_NOBLOCKBREAK**

Kein Block zum Unterbrechen! (1596)

**#ERR\_NOFALLTHROUGH**

Kein Block zum Durchfallen! (1597)

**#ERR\_TABEXPECTED3**

Es wurde ein Table erwarten! (1598) (Beachten Sie: Table ist eine Tabelle)

**#ERR\_EMPTYTABLE**

Table muss mindestens ein Element enthalten! (1599) (Beachten Sie: Table ist eine Tabelle)

**#ERR\_MOVEFILE**

Fehler beim Bewegen der Datei!(1600)

**#ERR\_RADIOGGLEMENU**

Menüeintrag kann nicht Radio- und Togglemodus benutzen! (1601)

**#ERR\_RANDOMIZE**

Fehler beim Erzeugen einer Zufallszahl! (1602)

**#ERR\_TRIALCOMPILE**

Das Kompilieren von Applets oder Programmen wird in der Demoversion nicht unterstützt! (1603)

**#ERR\_TRIALSAVEVID**

Videaufnahme wird in der Demoversion nicht unterstützt! (1604)

**#ERR\_TRIALLIMIT**

Die Demoversion unterstützt keine Skripte, die größer als 16 Kilobyte sind! (1605)

**#ERR\_TRIALINCLUDE**

Das Einbinden von Dateien wird in der Demoversion nicht unterstützt! (1606)

**#ERR\_VIDEOINIT**

Fehler beim Initialisieren des Anzeigegegeräts! (1607) (Videogerät)

**#ERR\_FINDICON**

Konnte Piktogramm %d nicht finden! (1608)

**#ERR\_ICONPARMS**

Die Parameter von Normal- und Aktivzustand sind unterschiedlich! (1609)

**#ERR\_ICONSIZE**  
Piktogrammgröße wurde zweimal benutzt! (1610)

**#ERR\_LOADICON**  
Piktogramm-Datei \"%s\" ist in einem unbekannten Format! (1611)

**#ERR\_ICONSTANDARD**  
Es darf nur eine Standard-Piktogrammgröße geben! (1612)

**#ERR\_ICONENTRY**  
Der angegebene Piktogrammindex ist außerhalb des gültigen Bereichs! (1613)

**#ERR\_ICONVECTOR**  
Vektorpinsel müssen das einzige Piktogrammbild sein! (1614)

**#ERR\_MULTIDISPLAYS**  
Hollywood unterstützt auf dieser Plattform nur ein einziges Display! (1615)

**#ERR\_TEXTURE**  
Fehler beim Erzeugen der Textur! (1616)

**#ERR\_SURFACE**  
Fehler beim Erzeugen der Oberfläche! (1617)

**#ERR\_RENDERER**  
Fehler beim Erzeugen des Renderers! (1618)

**#ERR\_FINDSERIAL**  
Konnte serielle Verbindung %d nicht finden! (1619)

**#ERR\_INITSERIAL**  
Fehler beim Initialisieren der seriellen Schnittstelle! (1620)

**#ERR\_OPENSERIAL**  
Fehler beim Öffnen der seriellen Schnittstelle %s! (1621)

**#ERR\_SERIALIO**  
Serieller Ein-/Ausgabefehler! (1622)

**#ERR\_SENDTIMEOUT**  
Zeitüberschreitung beim Senden! (1623)

**#ERR\_SENDUNKNOWN**  
Unbekannter Fehler während des Sendens! (1624)

**#ERR\_PLUGIN SUPPORT**  
Das Plugin unterstützt diese Funktion nicht! (1625)

**#ERR\_REWINDDIR**  
Fehler beim Zurücksetzen des Verzeichnisses! (1626)

**#ERR\_MONITORDIR**  
Fehler beim Beobachten des Verzeichnisses! (1627) (Beachten Sie: Mit Beobachten ist Überwachen gemeint)

**#ERR\_JAVAMETHOD**  
Die Java-Methode \"%s\" wurde nicht gefunden! (1628)

**#ERR\_NUMBEREXPECTED**  
Es wurde eine Zahl erwartet! (1629)

**#ERR\_CHANGEDIR**  
Konnte nicht das Verzeichnis nach %s wechseln! (1630)

**#ERR\_FUNCARG**  
Funktion im Tableargument \"%s\" erwartet! (1631) (Beachten Sie: Mit Tableargument ist Tabellenargument gemeint)

**#ERR\_BADYIELD**  
Yield-Versuch über Metamethoden/C-Aufrufgrenze! (1632)

**#ERR\_CYIELD**  
Yield ist nicht für C-Funktionen möglich! (1633)

**#ERR\_THREADEXPECTED**  
Koroutine wurde in Argument %d erwartet! (1634)

**#ERR\_YIELD**  
Dieser Fehler ist nur zum internen Gebrauch. (1635)

**#ERR\_DEADRESUME**  
Kann keine tote Koroutine fortsetzen! (1636)

**#ERR\_NONSUSPENDEDRESUME**  
Nicht angehaltene Koroutinen können nicht fortgesetzt werden! (1637)

**#ERR\_FORBIDMODAL**  
Dieser Befehl wurde durch ein Plugin deaktiviert! (1638)

**#ERR\_SERIALIZE**  
Fehler beim Serialisieren der Daten! (1639)

**#ERR\_SERIALIZETYPE**  
Dieser Datentyp kann nicht serialisiert werden! (1640)

**#ERR\_DESERIALIZE**  
Fehler beim Deserialisieren der Daten! (1641)

**#ERR\_FINDASYNCOBJ**  
Konnte asynchrone Operation %d nicht finden! (1642)

**#ERR\_NOPALETTE**  
Die Bilddatei \"%s\" besitzt keine Palette! (1643)

**#ERR\_NEEDPALETTEIMAGE**  
Die Bilddaten besitzen keine Palette! (1644)

**#ERR\_NOPALETTEIMAGE**  
Diese Funktion kann nicht mit Palettebildern verwendet werden! (1645) (Beachten Sie: Mit Funktion ist hier ein Befehl von Hollywood gemeint)

**#ERR\_PENRANGE**  
Paletttestift ist außerhalb des zulässigen Bereichs! (1646)

**#ERR\_DEPTHMISMATCH**  
Inkompatible Farbtiefe! (1647)



- #ERR\_ALLOCCHUNKY**  
Fehler beim Reservieren der Palettebitmap! (1648)
- #ERR\_FINDPALETTE**  
Konnte Palette %d nicht finden! (1649)
- #ERR\_DEPTHRANGE**  
Die angegebene Palettetiefe ist außerhalb des zulässigen Bereichs! (1650)
- #ERR\_BGPICPALETTE**  
Das aktuelle Hintergrundbild besitzt keine Palette! (1651)
- #ERR\_UNKNOWNPALETTE**  
Unbekannter Standardpalettetyp angegeben! (1652)
- #ERR\_PALETTEFILL**  
Der angegebene Füllstil kann nicht mit Palettebildern verwendet werden! (1653)
- #ERR\_DISPLAYDESKTOPPAL**  
Paletten können nicht mit einem Desktop-Display benutzt werden! (1654)
- #ERR\_DBPALETTE**  
Hardware-Doublebuffer können nicht im Palettemodus benutzt werden! (1655)  
(Beachten Sie: Doublebuffer ist der Doppelpuffer)
- #ERR\_UNKNOWNICNOUT**  
Unbekanntes Piktogrammformat angegeben! (1656)
- #ERR\_SAVEICON**  
Fehler beim Speichern des Piktogramms! (1657)
- #ERR\_PALETTEMODE**  
Diese Funktion kann nur im Palettemodus benutzt werden! (1658) (Beachten Sie: Mit Funktion ist hier ein Befehl von Hollywood gemeint)
- #ERR\_NOPALETTEMODE**  
Diese Funktion kann nicht im Palettemodus benutzt werden! (1659) (Beachten Sie: Mit Funktion ist hier ein Befehl von Hollywood gemeint)
- #ERR\_NORTG**  
Konnte CyberGraphX oder Picasso96 nicht finden. Um Hollywood ohne Cyber-GraphX oder Picasso96 zu benutzen, installieren Sie bitte das Plananarama-Plugin! (1660)
- #ERR\_GETMENUATTR**  
Fehler beim Abfragen der Menüattribute! (1661)
- #ERR\_SETMENUATTR**  
Fehler beim Setzen der Menüattribute! (1662)
- #ERR\_TRAYICON**  
Fehler beim Setzen des Traypiktogramms! (1663)
- #ERR\_FONTPATH2**  
Sie müssen den eingebauten Schrifttreiber benutzen, wenn Sie Schriftdateien direkt öffnen möchten! (1664)

**#ERR\_MEDIAFOUNDATION**

Ein Media-Foundation-Fehler ist aufgetreten! (1665)

**#ERR\_GETIFADDRS**

Fehler beim Abfragen der Schnittstellenadressen! (1666)

**#ERR\_TFVANIM**

Kann keine transformierte Vektoranimation als Ebene hinzufügen! Benutzen Sie Drawtags anstatt die Animation direkt zu transformieren! (1667)

**#ERR\_VECTORANIM**

Diese Funktionalität ist nicht für Vektoranimationen verfügbar! (1668)

**#ERR\_PLAYVIDEO**

Fehler beim Starten des Videos! (1669)

**#ERR\_TEXTCONVERT2**

Fehler bei der Textkonvertierung! (1670)

**#ERR\_TFVTEXTOBJ**

Kann kein transformiertes Vektortextobjekt als Ebene hinzufügen! Benutzen Sie Drawtags anstatt das Textobjekt direkt zu transformieren! (1671)

**#ERR\_GROUPNOTFOUND**

Die angegebene Ebenengruppe existiert nicht! (1672)

**#ERR\_MERGEDLAYER**

Diese Funktion wird für zusammengeführte Ebenen nicht unterstützt! (1673)

**#ERR\_REMMERGEDLAYER**

Kann Ebene nicht entfernen, da sie Teil einer zusammengeführten Ebene ist! (1674)

**#ERR\_ALLOCIMAGE**

Fehler bei der Einrichtung des Bildes! (1675)

**#ERR\_ADVANCEDCONSOLE**

Diese Funktion ist nur im erweiterten Konsolenmodus verfügbar! (1676)

**#ERR\_COLORTERMINAL**

Das Terminal unterstützt keinen Farbmodus! (1677)

**#ERR\_CONSOLE**

Ein Konsolenfehler ist aufgetreten! (1678)

**#ERR\_FINDCONWIN**

Kann das Konsolenfenster %d nicht finden! (1679)

**#ERR\_CONWIN**

Fehler beim Erstellen des Konsolenfensters! (1680)

**#ERR\_FREEPARENT**

Unterobjekt kann nicht vor dem Oberobjekt freigegeben werden! (1681)

**#ERR\_AMIGAINPUT**

Fehler beim Initialisieren von AmigaInput! (1682)

## 27.5 GetErrorMessage

### BEZEICHNUNG

GetErrorMessage – gibt einen Namen für einen Fehlercode zurück

### ÜBERSICHT

```
err$ = GetErrorMessage(code)
```

### BESCHREIBUNG

Dieser Befehl gibt einen Namen für den angegebenen Fehlercode zurück. Der Name beschreibt den Fehler des angegebenen Codes. Dieser Befehl sollte direkt nach `GetLastError()` aufgerufen werden, weil der Fehlerzustand Informationen enthalten könnte die verloren sind, wenn der nächste Befehl normal ausgeführt wird. Konditionale Befehle zwischen `GetLastError()` und `GetErrorMessage()` aufzurufen ist kein Problem. Bitte schauen Sie sich für weitere Informationen das Beispiel an.

Stellen Sie außerdem sicher, dass Sie die Dokumentation von `GetLastError()` gelesen haben, um detailliertere Informationen über manuelle Fehlerbehandlung zu haben.

Siehe [Abschnitt 27.4 \[Fehlercodes\]](#), Seite 524, für eine Liste aller in Hollywood definierten Fehlercodes.

### EINGABEN

`code` ein Fehlercode, wie er von `GetLastError()` zurückgegeben wird

### RÜCKGABEWERTE

`err$` eine Zeichenfolge mit der Beschreibung über den aufgetretenen Fehler

### BEISPIEL

Siehe [Abschnitt 27.6 \[GetLastError\]](#), Seite 563.

## 27.6 GetLastError

### BEZEICHNUNG

GetLastError – gibt den Fehlercode des letzten Befehls zurück

### ÜBERSICHT

```
code = GetLastError()
```

### BESCHREIBUNG

Dieser Befehl fragt Hollywoods internen Fehlerzustand ab und gibt das Ergebnis zurück. Der Zustand ist Null, wenn der letzte Befehl erfolgreich ausgeführt wurde. Falls der Befehl fehlschlug, wird ein Fehlercode zurückgegeben, der von Null verschieden ist. Dieser Fehlercode kann dann benutzt werden, um Hollywood nach einer Beschreibung für den aufgetretenen Fehler zu fragen. (Benutzen Sie hierfür `GetErrorMessage()`.)

Wichtiger Hinweis: Hollywoods interner Fehlerzustand wird auf Null zurückgesetzt, bevor ein Befehl ausgeführt wird. Deshalb wird der Fehlercode, den Sie mit `GetLastError()` erhalten, der des Befehls sein, der direkt vor `GetLastError()` aufgerufen wurde.

Wichtiger Hinweis Nr.2: Dieser Befehl ist nur sinnvoll, wenn die automatische Fehlerbehandlung abgestellt ist. Falls sie eingeschaltet ist, (was der Voreinstellung entspricht), wird die Fehlerbehandlung Ihr Skript sofort abbrechen, wenn ein Fehler auftritt. Dadurch

wird Ihr Skript niemals einen `GetLastError()`-Aufruf erreichen, wenn die automatische Fehlerbehandlung eingeschaltet ist. Deshalb müssen Sie vorher `ExitOnError()` mit `False` als Parameter aufrufen, um Hollywoods interne Fehlerbehandlung abzustellen.

Siehe [Abschnitt 27.4 \[Fehlercodes\]](#), [Seite 524](#), für eine Liste aller in Hollywood definierten Fehlercodes.

#### EINGABEN

keine

#### RÜCKGABEWERTE

code        Null falls erfolgreich; von Null verschieden bei aufgetretenem Fehler

#### BEISPIEL

```
ExitOnError(FALSE) ; Deaktiviert den automatischen Fehlerhandler
LoadBGPic(1,"blablabla") ; dieser Befehl wird fehlschlagen!
code=GetLastError()
If code<>0
    err$=GetErrorName(code)
    SystemRequest("Ein Fehler ist aufgetreten!",err$,"OK")
End
EndIf
```

Das oben genannte Stück Code zeigt, wie der Fehler, den `LoadBGPic()` erzeugt hat, zu behandeln ist. Es ist wichtig, dass kein weiterer Befehl zwischen `LoadBGPic()` und `GetLastError()` ist. Wäre dort ein weiterer Befehl, wäre das Fehlerergebnis von `LoadBGPic()` verloren.

## 27.7 RaiseOnError

#### BEZEICHNUNG

`RaiseOnError` – definiert eine benutzerdefinierte Fehlerbehandlung (V5.2)

#### ÜBERSICHT

`RaiseOnError(f)`

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um eine benutzerdefinierte Fehlerbehandlungsfunktion zu installieren. Immer wenn ein Fehler auftritt, wird dieser Befehl mit den folgenden vier Argumenten aufgerufen werden: Ein Fehlercode, eine Zeichenfolge, die den Fehler beschreibt, der Name des letzten Befehls und die aktuelle Zeilennummer.

Dies ist nützlich, wenn Sie nicht die automatische eingebaute Fehlerbehandlung von Hollywoods verwenden wollen. Bitte beachten Sie, dass in bestimmten Situationen der Name des letzten Befehls und die aktuelle Zeilennummer falsch sein können.

Beachten Sie auch, dass wenn ein Fehler in Ihrer benutzerdefinierten Fehlerbehandlungsfunktion auftritt, das Hollywood mit einem schwerwiegenden Fehler beendet wird. Daher sollten Sie die benutzerdefinierte Fehlerbehandlung so kurz und geradlinig wie möglich halten.

Um den benutzerdefinierten Fehlerhandler zu deinstallieren, übergeben Sie einfach `Nil` im Argument `f`.

Siehe [Abschnitt 27.4 \[Fehlercodes\]](#), [Seite 524](#), für eine Liste aller in Hollywood definierten Fehlercodes.

**EINGABEN**

`f`            Funktion, die aufgerufen wird, wenn ein Fehler auftritt.

**BEISPIEL**

```
Function p_ErrorFunc(code, msg$, cmd$, line)
    DebugPrint(code, msg$, cmd$, line)
EndFunction
```

```
RaiseOnError(p_ErrorFunc)
```

```
LoadBrush(1, "non_existing_brush.png")
```

Der obige Code installiert eine benutzerdefinierte Fehlerfunktion und versucht dann, einen nicht vorhandenen Pinsel zu laden. Dies führt dazu, dass die Fehlerfunktion aufgerufen wird und weitere Informationen auf das Debug-Gerät ausgegeben werden.



## 28 Grafikbibliothek

### 28.1 ARGB

#### BEZEICHNUNG

ARGB – kombiniert eine Farbe mit Alpha-Transparenz (V2.0)

#### ÜBERSICHT

```
color = ARGB(alpha, rgb)
```

#### BESCHREIBUNG

Dieser Befehl kombiniert einen Alpha-Transparenzwert mit einer Farbe im **RGB-Format**. Es wird eine 32-Bit **ARGB-Farbe** im Format \$aarrgbb zurückgegeben. Dies ist in Verbindung mit den Befehlen der Grafikbibliothek nützlich, weil sie auch Grafiken mit Alphakanaltransparenz zeichnen können.

Siehe auch **RGB()**, **GetRandomColor()** und **MixRGB()**.

#### EINGABEN

alpha      gewünschte Alphatransparenz (0-255)  
 rgb        Standard **RGB-Farbe**

#### RÜCKGABEWERTE

color      **ARGB-Farbe**

#### BEISPIEL

```
Box(#CENTER, #CENTER, 320, 240, ARGB(128, #WHITE))
```

Der obige Code zeichnet ein weißes Rechteck mit Alphatransparenz. Mischungsverhältnis beträgt 128 (50%).

### 28.2 ARGB-Farben

Eine ARGB-Farbe ist eine **RGB-Farbe**, welche mit Informationen der Alpha-Transparenz erweitert ist. Diese Schreibweise ist für die Befehle der Zeichnungsbibliothek nützlich, weil sie ermöglicht es Ihnen, die Alpha-Transparenz über die ARGB-Farbe anzugeben. Daher können Sie transparente Grafikgrundelemente mit der Grafikbibliothek von Hollywood zeichnen.

Eine Standard RGB-Farbe hat einen 24Bit-Wert, während eine ARGB-Farbe 32Bit verwendet. Die höchsten 8Bits werden für die Information der Alpha-Transparenz verwendet, die somit von 0 bis 255 reichen kann. Eine ARGB-Farbe sieht wie folgt aus:

**\$AARRGGBB**

Ein Alphawert von 0 bedeutet, dass überhaupt keine Transparenz vorhanden ist, ein Wert von 255 allerdings volle Transparenz. Dies ist eine Umkehrung des Formats, welches in **SetAlphaIntensity()** verwendet wird, wobei 255 0% Transparenz bedeutet und 0 100% Transparenz. Bitte beachten Sie das.

Sie können den **ARGB()** Befehl verwenden, um auf einfache Weise einen Alpha-Transparenzwert mit einer 24Bit RGB-Farbe zu kombinieren.

Siehe auch **RGB-Farben**.

## 28.3 BeginDoubleBuffer

### BEZEICHNUNG

BeginDoubleBuffer – startet Doppelpufferung für das aktuelle Display (V2.0)

### ÜBERSICHT

`BeginDoubleBuffer([hardware])`

### BESCHREIBUNG

Dieser Befehl setzt das aktuelle Display in den Doppelpuffermodus. Die Grafiken, die nach diesem Befehl gezeichnet wurden, werden nicht sichtbar sein, bis Sie den Befehl **Flip()** aufrufen. Doppelpufferung wird verwendet, um sichtbare Bildschirmaktualisierungen für den Benutzer zu vermeiden. Wie der Name schon sagt, haben Sie zwei Puffer im Doppelpuffermodus: Einen Vordergrundpuffer (für den Benutzer sichtbar) und einen Hintergrundpuffer (im Speicher). Ihre Bildschirmaktualisierungen werden immer im Hintergrundpuffer (Speicher) gezeichnet und wenn Sie damit fertig sind, rufen Sie den Befehl **Flip()** auf, um den Hintergrundpuffer nach vorne zu holen. Danach können Sie die nächsten Aktualisierungen vornehmen. Diese Technik sorgt dafür, dass kein Flackern sichtbar wird, weil **Flip()** immer das gesamte Display in einem Rutsch aktualisieren wird.

Doppelpuffer sind äußerst nützlich, wenn viele Grafiken in einer Bildschirmaktualisierung gezeichnet werden müssen. Wenn Sie nur ein kleines Bild bewegen müssen, sollten Sie besser Sprites verwenden, weil es damit schneller geht. Denken Sie daran, dass ein doppelt gepuffertes Display immer den ganzen Bildschirm aktualisiert. Wenn Sie also eine Anwendung in 640x480 bei 25 fps läuft, wird es etwas Arbeit für Hollywood bedeuten, weil ein Bildschirm von 640x480 25-mal in der Sekunde gezeichnet werden muss.

Doppelpuffer werden auf einer Pro-Display Basis installiert. Wenn Sie also **BeginDoubleBuffer()** aufrufen, wird nur das aktuell ausgewählte Display in den Doppelpuffermodus versetzt werden. Nicht alle Displays werden somit doppelt gepuffert! Wenn Sie alle Ihre Displays doppelt gepuffert haben wollen, dann müssen Sie **BeginDoubleBuffer()** für jedes einzelne Ihrer Display aufrufen.

Es gelten aber bestimmte Einschränkungen:

- Sie können Doppelpuffer nicht verwenden, wenn Ebenen eingeschaltet sind.
- Sie können keine Sprites mit doppelt gepufferten Displays verwenden.
- Das BGPic, welches beim Aufruf von **BeginDoubleBuffer()** aktiv ist, darf nicht transparent sein.

Ab Hollywood 5.0, gibt es das neue optionale Argument **hardware**, womit Sie Hardware-doppelpuffer aktivieren können. Auf unterstützten Systemen ist dies viel schneller als Softwaredoppelpufferung, da es vollständig im Videospeicher ausgeführt wird, der die GPU zum Zeichnen verwendet. Es gibt aber einige Einschränkungen: Wenn Sie einen Hardware-doppelpuffer verwenden, sollten Sie wann immer möglich Hardware-Pinsel benutzen. Alle anderen Zeichnungsbefehle sind viel langsamer! Nur bei Verwendung von Hardware-Pinsel können Sie eine vollständig beschleunigte Hardwarezeichnung erhalten. Mit normalen Zeichnungsbefehlen kann Hardware-doppelpuffer sogar langsamer sein, als es mit Softwaredoppelpuffer wäre. Dies ist insbesondere dann bei Grafiken der Fall, die einen Alphakanal verwenden, z.B. Antialiasing-Text oder Vektorformen. Daher sollten Sie versuchen, wann immer möglich Hardware-Pinsel zu verwenden, wenn Sie mit einem



Hardwaredoppelpuffer arbeiten. Siehe [Abschnitt 43.37 \[Informationen über Hardware-Pinsel\]](#), [Seite 940](#), für Details.

Bitte beachten Sie, dass derzeit Hardware-Doppelpufferung nur auf AmigaOS und Android unterstützt wird und standardmäßig voreingestellt ist. Allerdings können Plugins, die ein Display-Adaptermodul installieren, auch Hardware-Doppelpuffer für ihr Display-Adaptermodul unterstützen. In diesem Fall können Sie auch Hardware-Doppelpuffer auf anderen Systemen verwenden. Zum Beispiel können die GL Galore- und RebelSDL-Plugins Hardware-Doppelpuffer unter Windows, macOS und Linux verwenden. Siehe [Abschnitt 5.4 \[Vorhandene Plugins\]](#), [Seite 68](#), für Details.

Bitte beachten Sie auch, dass Hollywood auf einigen Systemen in den Singlepuffermodus fallen könnte. Daher ist es nicht sicher, dass der Befehl `Flip()` wirklich die Puffer wechselt. Es könnte auch einfach in den einzelnen Puffer zeichnen und dann alle Aktualisierungen wieder darauf anwenden.

Siehe auch `EndDoubleBuffer()` und `Flip()`.

## EINGABEN

**hardware** optional: Bei `True` wird Hardwaredoppelpuffer eingeschaltet (voreingestellt ist `False`, womit Softwaredoppelpuffer verwendet wird) (V5.0)

## BEISPIEL

```
BeginDoubleBuffer()
CreateBrush(1, 64, 64, #RED)
For k = -64 To 640
    Cls
    DisplayBrush(1, k, #CENTER)
    Flip
Next
EndDoubleBuffer()
```

Der obige Code bewegt ein rotes Rechteck von links außen nach rechts außen ohne sichtbares Flimmern. Dies ist kein gutes Beispiel, weil wir nur ein kleines Bild bewegen. Es ist eine Menge Aufwand, das ganze 640x480 Display nur für dieses kleine Bild zu aktualisieren, so sollten Sie in diesem Fall besser Sprites verwenden. Denken Sie daran, dass Doppelpufferung nur empfohlen wird, wenn eine Menge Grafiken zu aktualisieren/zeichnen gibt.

## 28.4 BeginRefresh

### BEZEICHNUNG

`BeginRefresh` – bereitet das Display für die optimierte Aktualisierung vor (V5.3)

### ÜBERSICHT

```
BeginRefresh([force])
```

### BESCHREIBUNG

Dieser Befehl bereitet das aktive Display für eine optimierte Aktualisierung vor. Auf Plattformen, die optimierte Aktualisierung unterstützen, bedeutet dies, dass alle Zeichnungsbefehle, die auf `BeginRefresh()` folgen, in die Warteschlange gestellt werden, bis

Sie den Befehl **EndRefresh()** aufrufen. Durch Zwischenspeichern aller Zeichnungsbefehle, die zwischen **BeginRefresh()** und **EndRefresh()** aufgerufen werden, kann letzteres Ihr gesamtes Display in einem einzigen Einzelbild aktualisieren, was auf vielen Plattformen neben der Verwendung eines doppelten Puffers die effizienteste Art des Zeichnens ist. Auf Plattformen, die die optimierte Aktualisierung nicht unterstützen, hat das Aufrufen von **BeginRefresh()** und **EndRefresh()** keinerlei Auswirkungen. Daher ist es sicher, es immer in Ihren Skripten zu verwenden und Hollywood wird es bei Bedarf automatisch aktivieren.

Die optimierte Aktualisierung wird immer unter Windows (außer wenn die Software-Wiedergabe aktiv ist), macOS (außer auf PowerPC), iOS und Android unterstützt. Auf allen anderen Plattformen wird die optimierte Aktualisierung nur unterstützt, wenn die Autoskalierung aktiv ist. Siehe [Abschnitt 24.24 \[Skalierungssysteme\]](#), [Seite 402](#), für Details.

Um das Konzept der optimierten Aktualisierung besser zu verstehen, müssen Sie wissen, dass auf einigen Systemen alle Zeichnungsoperationen immer zu einer vollständigen Neuzeichnung des gesamten Displays führen, selbst wenn nur ein einzelnes Pixel geändert wird. Stellen Sie sich nun folgende Situation vor: Ihr Spiel muss für jedes neue Einzelbild 48 Sprites auf den Bildschirm zeichnen. Auf Systemen, die immer das gesamte Display aktualisieren, auch wenn sich nur ein einzelnes Pixel geändert hat, bedeutet dies, dass Hollywood den gesamten Bildschirm 48 Mal neu zeichnen muss und dies 50 Mal pro Sekunde, wenn Ihr Spiel mit 50 fps läuft.

Pro Sekunde (!) wären also 2400 komplette Bildschirmaktualisierungen ( $48 * 50$ ) notwendig. Dies wird natürlich die Leistung Ihres Spiels erheblich beeinträchtigen. Wenn Sie stattdessen das Zeichnen der 48 Sprites in einen **BeginRefresh()**- und **EndRefresh()**-Abschnitt kapseln, muss Hollywood den Bildschirm nur einmal pro Einzelbild (50 Mal pro Sekunde) aktualisieren, was zu einem viel flüssigeren Erscheinungsbild führt. Daher ist die Verwendung von **BeginRefresh()** in diesem Fall unbedingt erforderlich, um eine gute Leistung Ihres Spiels zu gewährleisten.

Optimierte Aktualisierung kann auch die Leistung des Skripts erheblich verbessern, wenn automatische Skalierung aktiv ist. Wenn Sie normales Zeichnen mit Autoskalierung verwenden, wird Hollywood bei jeder gewünschten Abmessung seinen Auffrischungspuffer bei jedem einzelnen Zeichnungsbefehl skalieren. Wenn Sie hingegen hier einen **BeginRefresh()** Abschnitt verwenden, muss Hollywood beim Aufruf von **EndRefresh()** nur einmal seinen Auffrischungspuffer skalieren. Dies kann natürlich die Leistung auch erheblich verbessern.

Standardmäßig verwendet **BeginRefresh()** nur dann die optimierte Aktualisierung, wenn deren Verwendung entweder auf der Host-Plattform empfohlen wird oder wenn die automatische Skalierung aktiviert ist. Sie können dieses Verhalten ändern, indem Sie im optionalen Argument **force True** übergeben. Somit wird **BeginRefresh()** immer eine optimierte Aktualisierung verwenden. Dies wird aber nicht empfohlen, da eine optimierte Aktualisierung in einigen Fällen auch langsamer als die normal Aktualisierung sein kann. Aus diesem Grund sollten Sie die Entscheidung Hollywood überlassen, ob eine optimierte Aktualisierung verwendet wird oder nicht.

Bitte beachten Sie, dass Sie nur **BeginRefresh()** Abschnitte benutzen, wenn Ihr Skript seine Zeichnungen ohne Doppelpuffer darstellt. Wenn es aber einen Doppelpuffer ver-

wendet, wird das Zeichnen bereits mit einer Aktualisierung pro Einzelbild durchgeführt und somit ist in diesem Fall eine Verwendung von `BeginRefresh()` nicht erforderlich.

### EINGABEN

**force** optional: zwingt den Einsatz von optimierte Aktualisierung (**True**) oder die Entscheidung wird Hollywood überlassen (**False**). Siehe oben für weitere Informationen (voreingestellt ist **False**)

### BEISPIEL

```
BeginRefresh()
For Local k = 1 To 48
    DisplaySprite(k, sprites[k].x, sprites[k].y)
Next
EndRefresh()
```

Der obige Code aktualisiert die Positionen von 48 Sprites mit Hilfe eines `BeginRefresh()`-Abschnitts. Das bedeutet, dass auf Systemen, die immer das gesamte Display aktualisieren (zum Beispiel Android) Hollywood nur einmal den Bildschirm aktualisieren muss. Ohne `BeginRefresh()` würde Hollywood den Bildschirm 48 Mal aktualisieren müssen, was natürlich viel langsamer ist.

## 28.5 Blue

### BEZEICHNUNG

Blue – gibt den Blau-Anteil einer Farbe zurück (V1.9)

### ÜBERSICHT

```
b = Blue(color)
```

### BESCHREIBUNG

Dieser Befehl gibt den 8-bit Blauanteil einer Farbe im **RGB-Format** zurück.

Siehe auch **Green()** und **Red()**.

### EINGABEN

**color** eine Farbe im **RGB-Format**

### RÜCKGABEWERTE

**b** Blauanteil der Farbe (0-255)

### BEISPIEL

```
b = Blue($B2C024)
```

Gibt \$24 zurück, welches den Blauanteil der Farbe darstellt.

## 28.6 ClearScreen

### BEZEICHNUNG

ClearScreen – löscht den Bildschirm mit einem Übergangseffekt (1.0/5.0)

### ÜBERSICHT

```
[handle] = ClearScreen(effect, color[, table])
```

## BESCHREIBUNG

`ClearScreen()` ist ein Befehl, der mit der ersten Hollywoodversion im November 2002 eingeführt wurde; er wurde dann in der Version 1.5 entfernt, steht aber seit Version 5.0 wieder zur Verfügung.

Dieser Befehl löscht den Bildschirm mit einem von Hollywoods Übergangseffekten. Die Farbe ist ein RGB-Wert, der die Farbe des Übergangseffekt angibt.

Beachten Sie, dass dieser Befehl automatisch ein neues `BGPic` erstellt, welches mit der angegebenen Farbe gefüllt ist.

Ab Hollywood 5.0 verwendet dieser Befehl eine neue Syntax mit nur einer einzigen Tabelle als optionales Argument. Die alte Syntax wird weiterhin aus Kompatibilitätsgründen unterstützt. Die optionale Tabelle `table` kann den Übergangseffekt konfigurieren. Folgende Optionen sind möglich:

**Speed:** Legt die gewünschte Geschwindigkeit für den Übergang fest. Je höher der Wert, desto schneller wird der Effekt angezeigt werden. (Standardeinstellung ist `#NORMALSPEED`)

**Parameter:** Einige Übergangseffekte akzeptieren einen zusätzlichen Parameter, der hier angegeben werden kann. (Standardeinstellung ist `#RANDOMPARAMETER`)

**Async:** Sie können diese Option, um ein asynchrones Zeichnungsobjekt für diesen Übergang zu erstellen. Wenn Sie hier `True` angeben, wird `DisplayBGPicPartFX()` sofort verlassen, und es wird ein Handler für das asynchrone Zeichnungsobjekt kreiert, den Sie dann mit dem Befehl `AsyncDrawFrame()` verwenden können. Ein Beispielskript finden Sie unter dem Befehl `AsyncDrawFrame()`. Siehe [Abschnitt 19.1 \[AsyncDrawFrame\]](#), [Seite 231](#), für weitere Informationen über asynchrone Zeichnungsobjekte.

**X:** Setzt die neue X-Position für das Display. Wenn Sie das Display an seiner aktuellen X-Position behalten wollen, geben Sie die spezielle Konstante `#KEEPPPOSITION` an. Der Standardwert ist `#CENTER`.

**Y:** Legt die neue Y-Position für das Display fest. Wenn Sie das Display an seiner aktuellen Y-Position behalten wollen, geben Sie die spezielle Konstante `#KEEPPPOSITION` an. Der Standardwert ist `#CENTER`.

Werfen Sie einen Blick in die Dokumentation des Befehls `DisplayTransitionFX()`, um die Liste aller Übergangseffekte zu sehen, die verwendet werden können.

Siehe auch [Cls\(\)](#).

## EINGABEN

**effect** spezielle Effektkonstante (siehe `DisplayTransitionFX()`)

**color** Farbe, die für den Übergangseffekt verwendet wird

**table** optional: Tabelle, die den Übergangseffekt konfiguriert

## RÜCKGABEWERTE

**handle** optional: Handler auf ein asynchrones Zeichnungsobjekt; wird nur zurückgegeben, wenn `Async` auf `True` gesetzt wurde (siehe oben)

**BEISPIEL**

```
ClearScreen(#VBLINDS32, $000000, {Speed = 10})
```

Der obige Code löscht den Bildschirm mit der Farbe Schwarz und wendet den Übergangseffekt #VBLINDS32 mit der Geschwindigkeit 10 an.

**28.7 Collision****BEZEICHNUNG**

Collision – überprüft, ob zwei Objekte kollidieren (V2.0)

**ÜBERSICHT**

```
bool = Collision(type, ...)
bool = Collision(#BOX, x1, y1, width1, height1, x2, y2, width2, height2)
bool = Collision(#BRUSH, id1, x1, y1, id2, x2, y2)
bool = Collision(#LAYER, id1, id2)
bool = Collision(#SPRITE, id1, id2)
bool = Collision(#BRUSH_VS_BOX, id, x, y, x2, y2, width2, height2) (V4.5)
bool = Collision(#LAYER_VS_BOX, id, x, y, width, height) (V4.5)
bool = Collision(#SPRITE_VS_BOX, id, x, y, width, height) (V4.5)
bool = Collision(#SPRITE_VS_BRUSH, id1, id2, x, y) (V7.1)
```

**BESCHREIBUNG**

Dieser Befehl überprüft, ob zwei Objekte kollidieren. Abhängig vom Objekttyp gibt es mehrere Möglichkeiten, wie Sie diesen Befehl verwenden.

Die folgenden Kollisionstypen werden derzeit unterstützt:

- #BOX:** Sie haben die Position und Größe von zwei Rechtecken festgelegt und dieser Befehl wird entscheiden, ob sie kollidieren oder nicht. Dies ist eine schnelle Berechnung, ist aber wahrscheinlich für einige Zwecke nicht exakt genug.
- #BRUSH:** Überprüft, ob die beiden Pinsel, welche durch `id1` und `id2` angegeben wurden, zusammen kollidieren, wenn sie bei `x1` und `y1` sowie `x2` und `y2` angezeigt werden. Transparente Bereiche (Maske oder Alpha-Kanal) der Pinsel werden voll berücksichtigt, so dass Sie ein genaues Ergebnis erhalten, wenn Pixel kollidieren oder nicht.
- #LAYER:** Überprüft, ob die beiden von `id1` und `id2` angegebenen Ebenen kollidieren. Falls die Ebenen transparente Bereiche haben, werden diese berücksichtigt. Wenn Sie diesen Typ verwenden, müssen natürlich Ebenen aktiviert werden.
- #SPRITE:** Überprüft, ob die beiden in `id1` und `id2` angegeben Sprites miteinander kollidieren. Wenn die Sprites transparente Bereiche haben, werden diese berücksichtigt.
- #BRUSH\_VS\_BOX:** Überprüft, ob der angegebene Pinsel mit dem rechteckigen Bereich kollidiert. Wenn der Pinsel transparente Bereiche aufweist, werden diese berücksichtigt. (V4.5)

**#LAYER\_VS\_BOX:**

Überprüft, ob die angegebene Ebene mit dem rechteckigen Bereich kollidiert. Wenn die Ebene transparente Bereiche aufweist, werden diese berücksichtigt. (V4.5)

**#SPRITE\_VS\_BOX:**

Überprüft, ob der angegebene Sprite mit dem rechteckigen Bereich kollidiert. Wenn der Sprite transparente Bereiche aufweist, werden sie berücksichtigt. (V4.5)

**#SPRITE\_VS\_BRUSH:**

Prüft, ob das in `id1` angegebene Sprite mit dem in `id2` angegebenen Pinsel kollidiert, falls der Pinsel an der in `x` und `y` bezeichneten Position angezeigt wurde. (V7.1)

Siehe auch [Intersection\(\)](#).

**EINGABEN**

`type`        entweder `#BOX`, `#BRUSH`, `#SPRITE`, `#LAYER`, `#BRUSH_VS_BOX`, `#LAYER_VS_BOX`, `#SPRITE_VS_BOX` oder `#SPRITE_VS_BRUSH` (siehe oben)

`...`        optionale Argumente; hängen vom Typ ab (siehe oben), wobei `width` die Breite und `height` die Höhe des jeweiligen Rechtecks ist

**RÜCKGABEWERTE**

`bool`        `True` für Kollision, andernfalls `False`

**BEISPIEL**

```
Box(10, 10, 100, 100, #RED)
Box(70, 70, 100, 100, #BLUE)
b = Collision(#BOX, 10, 10, 100, 100, 70, 70, 100, 100)
Dies gibt True zurück, weil die Rechtecke miteinander kollidieren.
```

## 28.8 CreateClipRegion

**BEZEICHNUNG**

CreateClipRegion – erstellt eine Clip-Region (V2.0)

**ÜBERSICHT**

```
[id] = CreateClipRegion(id, type, ...)
[id] = CreateClipRegion(id, #BOX, x, y, width, height)
[id] = CreateClipRegion(id, #SHAPE, id, x, y)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um eine Clip-Region zu erstellen, die alle Grafikbefehle von Hollywood respektieren. `CreateClipRegion()` erzeugt eine neue Clip-Region und weist den Identifikator `id` zu. Wenn Sie `Nil` als ID angeben, wird `CreateClipRegion()` automatisch eine auswählen und zurückgeben. Beachten Sie, dass `CreateClipRegion()` nicht die Clip-Region aktiviert. Dies muss mit dem Befehl [SetClipRegion\(\)](#) erledigt werden.

Clip-Regionen sind nützlich, um den Bereich Ihres Display zu begrenzen, wo Grafiken angezeigt werden können. Z.B. wenn Sie ein Spielbildschirm mit zwei Teilen haben möchten: Den Levelbereich und die Statusleiste (Leben, Munition etc.). Es kann dann sinnvoll sein, eine Clip-Region zu installieren, die mit den Grenzen des Levelbereichs übereinstimmt, so dass Sprites nie außerhalb dieses Bereichs gezogen werden können. Alle Grafikbefehle von Hollywood werden diese Clip-Region respektieren. Somit werden keine Grafiken jemals außerhalb der Grenzen der Clip-Region gezeichnet.

Es gibt zwei verschiedene Clip-Regionstypen: Rechteckige (**#BOX**) und benutzerdefiniert geformte Clip-Regionen (**#SHAPE**). Das Zeichnen von rechteckigen Clip-Regionen ist in der Regel schneller. Wenn Sie eine rechteckige Clip-Region installieren möchten, müssen Sie seine **x**- und **y**-Position sowie Breite **width** und Höhe **height** angeben. Eine benutzerdefiniert geformte Clip-Region kann durch Angabe eines Pinsels installiert werden, dessen Maske der gewünschten Clip-Region entspricht. Außerdem müssen Sie die **x**- und **y**-Koordinaten, wo auf dem Display die Clip-Region positioniert wird. Wenn Sie z.B. eine Maske der Größe 320x240 verwenden, aber ein Display der Größe 640x480 haben, können Sie die Clip-Region auf dem Display zentrieren. Wenn Sie eine benutzerdefinierte Pinsel mit Maske als Clip-Region installieren, wird Hollywood in der Lage sein, auf alle sichtbaren Pixel der Maske zu zeichnen. Bei allen unsichtbaren Pixel der Maske wird die Grafik abgeschnitten.

Siehe [Abschnitt 28.31 \[SetClipRegion\]](#), Seite 592, für mehr Informationen über die Erstellung und Installation von Clip-Regionen.

Wenn Sie eine mit **CreateClipRegion()** erstellte Clip-Region aus dem Speicher löschen wollen, benutzen Sie den Befehl **FreeClipRegion()**.

Siehe auch **FreeClipRegion()** und **SetClipRegion()**.

#### EINGABEN

<b>id</b>	Identifikator für die Clip-Region oder <b>Nil</b> für die <b>automatische ID-Zuweisung</b>
<b>type</b>	Typ der zu installierenden Clip-Region: <b>#BOX</b> , <b>#SHAPE</b> oder <b>#NONE</b>
<b>...</b>	die folgenden Argumente hängen vom Typ ab (siehe oben), wobei <b>width</b> die Breite und <b>height</b> die Höhe des Rechtecks ist

#### RÜCKGABEWERTE

<b>id</b>	optional: Identifikator der Clip-Region; wird nur zurückgegeben, wenn Sie <b>Nil</b> in <b>id</b> angegeben haben (siehe oben)
-----------	--

#### BEISPIEL

Siehe [Abschnitt 28.31 \[SetClipRegion\]](#), Seite 592.

## 28.9 DisablePrecalculation

#### BEZEICHNUNG

**DisablePrecalculation** – schaltet Vorausberechnung ab (nur V1.5)

#### ÜBERSICHT

**DisablePrecalculation()**

**BESCHREIBUNG**

Achtung: Dieser Befehl wurde in Hollywood 1.9 entfernt. Holen Sie sich eine schnellere CPU.

Dieser Befehl schaltet Vorausberechnungen bei bestimmten Effekten ab.

**EINGABEN**

keine

## 28.10 DisableVWait

**BEZEICHNUNG**

DisableVWait – deaktiviert das vertikale Aktualisierungs-Timing (V9.0)

**ÜBERSICHT**

DisableVWait()

**BESCHREIBUNG**

Dieser Befehl deaktiviert das interne vertikale Aktualisierungs-Timing. `DisableVWait()` ist nur zum Debuggen nützlich. Sie sollten diesen Befehl nicht in Ihren Skripten verwenden.

Siehe auch `EnableVWait()`.

**EINGABEN**

keine

## 28.11 EnablePrecalculation

**BEZEICHNUNG**

EnablePrecalculation – schaltet Vorausberechnung an (nur V1.5)

**ÜBERSICHT**

EnablePrecalculation()

**BESCHREIBUNG**

Achtung: Dieser Befehl wurde in Hollywood 1.9 entfernt. Holen Sie sich eine schnellere CPU.

Dieser Befehl ermöglicht Vorausberechnung bei bestimmten aufwendigen Effekten von `DisplayTransitionFX()`. Effekte, die Vorausberechnung unterstützen, werden dann im voraus berechnet. Die Vorausberechnung von aufwendigen Effekten ist eigentlich nur auf 68k-Systemen sinnvoll; PPC-native Systeme sollten die Effekte auch in Echtzeit darstellen können.

**EINGABEN**

Keine



## 28.12 EnableVWait

### BEZEICHNUNG

EnableVWait – aktiviert das vertikale Aktualisierungs-Timing (V9.0)

### ÜBERSICHT

EnableVWait()

### BESCHREIBUNG

Dieser Befehl aktiviert das interne vertikale Aktualisierungs-Timing. `EnableVWait()` ist nur zum Debuggen nützlich. Sie sollten ihn nicht in Ihren Skripten verwenden.

Siehe auch `DisableVWait()`.

### EINGABEN

keine

## 28.13 EndDoubleBuffer

### BEZEICHNUNG

EndDoubleBuffer – stoppt Doppelpufferung für das aktuelle Display (V2.0)

### ÜBERSICHT

EndDoubleBuffer()

### BESCHREIBUNG

Dieser Befehl beendet den Doppelpuffermodus für das aktuelle Display. Siehe [Abschnitt 28.3 \[BeginDoubleBuffer\]](#), [Seite 568](#), für mehr Informationen über die Doppelpufferung.

Siehe auch `Flip()`.

### EINGABEN

Keine

### BEISPIEL

Siehe [Abschnitt 28.3 \[BeginDoubleBuffer\]](#), [Seite 568](#).

## 28.14 EndRefresh

### BEZEICHNUNG

EndRefresh – leert die Zeichnungswarteschlange (V5.3)

### ÜBERSICHT

EndRefresh()

### BESCHREIBUNG

Dieser Befehl zeichnet alle Grafiken, die seit dem letzten Aufruf von `BeginRefresh()` in die Warteschlange gestellt wurden. Siehe [Abschnitt 28.4 \[BeginRefresh\]](#), [Seite 569](#), für weitere Informationen über optimierte Aktualisierungsabschnitte sowie ein Beispiel.

### EINGABEN

Keine

**BEISPIEL**

Siehe [Abschnitt 28.4 \[BeginRefresh\]](#), Seite 569.

## 28.15 Flip

**BEZEICHNUNG**

Flip – wechselt den Vorder- mit dem Hintergrundpuffer aus (V2.0)

**ÜBERSICHT**

Flip([sync])

**BESCHREIBUNG**

Dieser Befehl bringt den Hintergrundpuffer nach vorne und nimmt den aktuellen Vordergrundpuffer nach hinten, so dass Sie die nächste Bildschirmaktualisierung durchführen können.

Ab Hollywood 5.0 existiert ein optionales Argument **sync**. Wenn dies auf **False** gesetzt wird, wartet Flip() nicht auf den vertikalen Neuaufbau, sondern wird den Hintergrundpuffer sofort zeichnen. Dies ist nützlich, wenn Sie den Doppelpuffer häufiger aktualisieren möchten als die Bildwiederholfrequenz des Monitors. Wenn nicht anders angegeben, ist **sync** standardmäßig auf **True** gesetzt, womit Flip() seine Doppelpuffer mit dem Neuaufbau des Monitors synchronisiert.

Bitte beachten Sie, dass Hollywood auf einigen Systemen in den Singlepuffermodus fallen könnte. Daher ist es nicht sicher, dass der Befehl **Flip()** wirklich die Puffer wechselt. Es könnte auch einfach in den einzelnen Puffer zeichnen und dann alle Aktualisierungen wieder darauf zeichnen.

Siehe [Abschnitt 28.3 \[BeginDoubleBuffer\]](#), Seite 568, für mehr Informationen über die Doppelpufferung.

Siehe auch [EndDoubleBuffer\(\)](#).

**EINGABEN**

<b>sync</b>	optional: <b>True</b> , wenn der Doppelpuffer mit dem Neuaufbau des Monitors synchronisiert, <b>False</b> wenn nicht auf den Neuaufbau des Monitors gewartet werden soll (voreingestellt ist <b>True</b> ) (V5.0)
-------------	---

**BEISPIEL**

Siehe [Abschnitt 28.3 \[BeginDoubleBuffer\]](#), Seite 568.

## 28.16 FreeClipRegion

**BEZEICHNUNG**

FreeClipRegion – löscht eine Clip-Region aus dem Speicher (V2.0)

**ÜBERSICHT**

FreeClipRegion(id)

**BESCHREIBUNG**

Dieser Befehl löscht eine mit CreateClipRegion() erstellte Clip-Region aus dem Speicher.

Sind Sprites an diese Clip-Region gebunden, werden sie herausgelassen und die Clip-Region wird sofort aus dem Speicher entfernt.

Mit Ebenen ist das Verhalten anders: Wenn es Ebenen gibt, die auf eine Clip-Region verweisen, dann wird Hollywood die Clip-Region behalten, bis diese Ebenen verschwunden sind. Sobald es keine weiteren Ebenen gibt, die auf diese Clip-Region verweisen, wird der Speicherbereiniger (Garbage-Collector) von Hollywood die Clip-Region automatisch aus dem Speicher löschen, vorausgesetzt Sie haben zuvor `FreeClipRegion()` für diese Clip-Region aufgerufen.

Bitte beachten Sie für weitere Informationen über Clip-Regionen auch die Dokumentation zum Befehl `CreateClipRegion()`.

Siehe auch `SetClipRegion()`.

#### EINGABEN

`id` ID der Clip-Region, die Sie aus dem Speicher löschen wollen

#### BEISPIEL

Siehe [Abschnitt 28.31 \[SetClipRegion\]](#), Seite 592.

## 28.17 GetFPSLimit

#### BEZEICHNUNG

`GetFPSLimit` – gibt die momentane maximale Anzahl Einzelbilder pro Sekunde zurück (V1.5)

#### ÜBERSICHT

```
fps = GetFPSLimit()
```

#### BESCHREIBUNG

Dieser Befehl gibt die aktuelle maximale Anzahl der Einzelbilder zurück, welche mit dem Befehl `SetFPSLimit()` gesetzt wurde. Wenn `SetFPSLimit()` nicht aktiv ist, wird 0 zurückgegeben werden

Siehe auch `VWait()`.

#### EINGABEN

keine

#### RÜCKGABEWERTE

`fps` aktuelle maximale Anzahl Einzelbilder pro Sekunde oder 0.

## 28.18 GetRandomColor

#### BEZEICHNUNG

`GetRandomColor` – wählt eine Zufallsfarbe (V4.7)

#### ÜBERSICHT

```
color = GetRandomColor()
```

**BESCHREIBUNG**

Dieser Befehl wählt einfach eine zufällige Farbe aus. Die Farbe wird als **RGB-Wert** zurückgegeben.

Siehe auch **ARGB()**, **RGB()**, **MixRGB()** und **GetRandomFX()**.

**EINGABEN**

Keine

**RÜCKGABEWERTE**

**color**      zufällig ausgewählte Farbe

## 28.19 GetRandomFX

**BEZEICHNUNG**

GetRandomFX – wählt einen Zufallsübergangseffekt aus (V4.0)

**ÜBERSICHT**

```
fx = GetRandomFX(objfx)
```

**BESCHREIBUNG**

Dieser Befehl wählt einfach einen zufälligen Übergangseffekt aus Hollywoods Palette von Übergangseffekten. Sie müssen im Argument **objfx** angeben, ob der Übergangseffekt für Objekte oder nur für die Hintergrundbilder verwendet werden soll. **GetRandomFX()** durchsucht dann die verfügbaren Effekte und wählt zufällig einen Effekt für Sie aus, den Sie dann den Spezialeffektbefehlen wie **DisplayBrushFX()** und **DisplayTransitionFX()** übergeben.

Normalerweise müssen Sie diesen Befehl nicht verwenden, weil Sie einfach bei allen Spezialeffektbefehlen **#RANDEFFECT** übergeben können. **GetRandomFX()** ist nur dann sinnvoll, wenn Sie Effekte filtern wollen. Somit ist es z.B. möglich, einen zufälligen Effekt auszuwählen, aber nicht **#WATER1**. In diesem Fall können Sie **GetRandomFX()** solange aufrufen, bis es einen Effekt gibt, der sich von denen unterscheidet, die Sie nicht wollen.

Siehe **Abschnitt 30.11 [DisplayTransitionFX]**, **Seite 630**, für die verschiedenen Typen von Hollywoods Übergangseffekten.

Siehe auch **GetRandomColor()**.

**EINGABEN**

**objfx**      **True**, wenn Sie einen Objekteffekt möchten, **False**, wenn Sie einen Hintergrundbildeffekt wünschen.

**RÜCKGABEWERTE**

**fx**      ein zufällig ausgewählter Übergangseffekt

**BEISPIEL**

```
Repeat
    fx = GetRandomFX(FALSE)
Until fx <> #WATER1
DisplayTransitionFX(2, fx)
```

Der obige Code wählt einen zufälligen Hintergrundbildeffekt aus, aber nie den `#WATER1` Effekt. Nach der Auswahl des Effekts wird Hintergrundbild 2 mit diesem angezeigt.

## 28.20 GetRealColor

### BEZEICHNUNG

`GetRealColor` – gibt den richtigen Wert der Farbe des aktuellen Bildschirms zurück (V1.5)

### ÜBERSICHT

```
color = GetRealColor(col)
```

### BESCHREIBUNG

Dieser Befehl gibt den richtigen Farbwert zurück, die die ausgewählte Farbe auf dem aktuellen Bildschirm darstellt. Auf 24-Bit und 32-Bit-Bildschirme ist der Farbwert immer der gleiche wie die Farbe, die Sie angegeben haben. Auf 15- und 16-Bit-Bildschirme ist der Farbwert von der Originalfarbe in den meisten Fällen etwas anders, da diese Bildschirme nur 65536 (16-Bit-Bildschirme) oder 32768 (15-Bit-Bildschirme) und nicht 16,7 Millionen Farben haben.

Dieser Befehl wird meist im Zusammenhang mit `ReadPixel()` verwendet.

### EINGABEN

`col`            Farbe, dessen richtigen Wert zu finden ist

### RÜCKGABEWERTE

`color`            Farbe, dessen Wert nun stimmt

### BEISPIEL

```
color = GetRealColor($FFFFFF)
```

Farbe, die folgenden Wert bekommt auf...

1) 24-bit und 32-bit Bildschirmen: \$FFFFFF 2) 16-bit Bildschirmen: \$F8FCF8 3) 15-bit Bildschirmen: \$F8F8F8

## 28.21 GrabDesktop

### BEZEICHNUNG

`GrabDesktop` – erstellt einen Pinsel aus dem Desktop-Bildschirm (V4.5)

### ÜBERSICHT

```
[id] = GrabDesktop(id[, table])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Inhalt des Desktop-Bildschirms in einen Pinsel zu kopieren. Sie müssen eine `id` oder `Nil` für den neuen Pinsel übergeben. Wenn Sie `Nil` angeben, wird `GrabDesktop()` automatisch eine freie ID für Sie auswählen.

Das optionale Argument `table` ist nützlich, wenn Sie nur einen Ausschnitt des Desktop-Bildschirms kopieren wollen. Folgende Tags stehen zur Verfügung:

**X, Y:**            Definiert die obere linke Ecke des Rechtecks auf dem Desktop, der kopiert werden soll. Voreingestellt ist 0:0.

**Width, Height:**

Definiert die Größe des Rechtecks, die von diesem Befehl kopiert werden soll. Standardmäßig ist die Größe des Desktops eingestellt.

**PubScreen:**

Dieser Tag wird nur in Versionen von Hollywood für den Amiga unterstützt. Es ermöglicht Ihnen, den Namen des Bildschirms anzugeben, die `GrabDesktop()` zu einem Pinsel kopieren sollte. In der Standardeinstellung wird `GrabDesktop()` immer den vordersten Bildschirm kopieren. (V5.3)

Siehe auch `SaveSnapshot()`.

**EINGABEN**

**id** ID des neuen Pinsel oder `Nil` für die automatische ID-Zuweisung  
**table** optional: Tabelle, um den zu kopierenden Ausschnitt zu definieren (siehe oben)

**RÜCKGABEWERTE**

**id** optional: Identifikator des Pinsels; Wird nur zurückgegeben werden, wenn Sie `Nil` als Argument 1 angegeben haben (siehe oben)

**BEISPIEL**

```
desktop_brush = GrabDesktop(Nil)
BrushToGray(desktop_brush)
DisplayBrush(desktop_brush, 0, 0)
```

Der obige Code kopiert den gesamten Desktop-Bildschirm in einen neuen Pinsel, wandelt den Pinsel in Graustufen um und zeigt ihn dann. Das Ergebnis wird ein Desktop-Bildschirm sein, der plötzlich seine Farbinformationen verloren hat.

## 28.22 Green

**BEZEICHNUNG**

Green – gibt den Grün-Anteil einer Farbe zurück (V1.9)

**ÜBERSICHT**

```
g = Green(color)
```

**BESCHREIBUNG**

Dieser Befehl gibt den 8-bit Grünanteil einer Farbe im **RGB-Format** zurück.

Siehe auch `Blue()` und `Red()`.

**EINGABEN**

**color** eine Farbe im **RGB-Format**

**RÜCKGABEWERTE**

**g** Grünanteil der Farbe (0-255)

**BEISPIEL**

```
g = Green($B2C024)
```

Gibt `$C0` zurück, welches den Grünanteil der Farbe darstellt.

## 28.23 Intersection

### BEZEICHNUNG

Intersection – berechnet die Fläche zweier überschneidenden Rechtecke (V6.1)

### ÜBERSICHT

```
ix, iy, iw, ih = Collision(x1, y1, w1, h1, x2, y2, w2, h2)
```

### BESCHREIBUNG

Dieser Befehl berechnet die Schnittfläche zwischen den beiden Rechtecken, deren Positionen und Abmessungen sie übergeben. Wenn die beiden Rechtecke sich nicht überschneiden, werden die zurückgegebenen Abmessungen 0 sein.

Siehe auch [Collision\(\)](#).

### EINGABEN

x1	x-Position des ersten Rechtecks
y1	y-Position des ersten Rechtecks
w1	Breite des ersten Rechtecks
h1	Höhe des ersten Rechtecks
x2	x-Position des zweiten Rechtecks
y2	y-Position des zweiten Rechtecks
w2	Breite des zweiten Rechtecks
h2	Höhe des zweiten Rechtecks

### RÜCKGABEWERTE

ix	x-Position der kreuzenden Fläche
iy	y-Position der kreuzenden Fläche
iw	Breite der kreuzenden Fläche
ih	Höhe der kreuzenden Fläche

### BEISPIEL

```
SetFillStyle(#FILLCOLOR)
Box(100, 100, 80, 100, #RED)
Box(160, 120, 100, 40, #YELLOW)
```

```
ix, iy, iw, ih = Intersection(100, 100, 80, 100, 160, 120, 100, 40)
```

```
Box(ix, iy, iw, ih, #GREEN)
```

Der obige Code berechnet den Schnittpunkt der roten und gelben Rechtecke und visualisiert ihn, indem er ein grünes Rechteck zeichnet.

## 28.24 IsPicture

### BEZEICHNUNG

IsPicture – stellt fest, ob ein Bild in einem unterstützten Format vorliegt

### ÜBERSICHT

```
ret, table = IsPicture(file$[, table])
```

### BESCHREIBUNG

Dieser Befehl überprüft, ob die in `file$` angegebene Datei in einem unterstützten Bildformat vorliegt. Wenn ja, wird der Befehl **True** zurückgeben, andernfalls **False**. Bei **True** können Sie das Bild mit **LoadBGPic()** oder **LoadBrush()** laden.

Neu in Hollywood 4.5: Dieser Befehl liefert nun als zweiter Rückgabewert eine Tabelle. Wenn die angegebene Datei ein Bild ist, wird die Tabelle einige Informationen über die Bilddatei enthalten. Die folgenden Felder der Rückgabetable werden initialisiert:

**Width:** Enthält die Breite des Bildes in Pixeln.

**Height:** Enthält die Höhe des Bildes in Pixeln.

**Depth:** Enthält die Bittiefe (Farbtiefe) des Bildes. (V9.0)

**Alpha:** **True**, wenn das Bild einen Alphakanal besitzt, andernfalls **False**.

**Vector:** **True**, wenn das Bild im skalierbaren Vektorformat ist, andernfalls **False**. (V5.0)

**Transparency:**

**True**, wenn das Bild einen monochromen Transparenzkanal besitzt, das heißt einen transparenten Stift in einem Palettenbasierten Bild, andernfalls **False** (V6.0)

**Format:** Dieses enthält das Bildformat, das Hollywood für diese Datei erkannt hat. Es kann eine der folgenden speziellen Konstanten sein:

**#IMGFMT\_BMP:**

BMP-Bild

**#IMGFMT\_PNG:**

PNG-Bild

**#IMGFMT\_JPEG:**

JPEG-Bild

**#IMGFMT\_ILBM:**

IFF ILBM-Bild

**#IMGFMT\_GIF:**

GIF-Bild

**#IMGFMT\_TIFF:**

TIFF-Bild

**#IMGFMT\_NATIVE:**

Das Bild hat ein Format, das vom Host-Betriebssystem geladen werden kann, auf dem Hollywood derzeit ausgeführt wird. Wenn



zum Beispiel ein Datentyp vorhanden ist, der das Bild unter AmigaOS laden kann, können Sie als Ergebnis `#IMGFMT_NATIVE` erhalten.

`#IMGFMT_PLUGIN:`

Das Bild hat ein Format, das von einem installierten Hollywood-Plugin unterstützt wird.

(V8.0)

Dieser Befehl ist viel schneller als `LoadBrush()` oder `LoadBGPic()`, weil es das Bild nicht lädt. Es wird nur der Formatbereich (Format Header) der Datei überprüft und teilt mit, ob Hollywood damit umgehen kann oder nicht. Beachten Sie jedoch, dass Bilder-Plugins, die Sie installiert haben, möglicherweise nicht so optimiert sind wie Hollywoods eingebaute Bilder-Lademodule. Daher können sie `IsPicture()` erheblich verlangsamen, da `IsPicture()` standardmäßig alle Plugins durchgeht, ob sie das Bild laden können und erst dann den in Hollywoods eingebaute Bild-Lademodule benutzt. Daher möchten Sie möglicherweise Plugins für `IsPicture()` deaktivieren, indem Sie den Tag `Loader` (siehe unten) auf `Inbuilt` setzen. Das bedeutet, dass `IsPicture()` niemals Plugins durchgeht, ob sie das Bild laden können. In diesem Fall wird nur das optimierte eingebaute Bild-Lademodul von Hollywood benutzt.

Ab Hollywood 6.0 akzeptiert dieser Befehl ein optionales Tabellenargument, womit weitere Konfigurationen möglich sind:

**Loader:** Mit diesem Tag können Sie ein oder mehrere Formatlademodule angeben, die dieses Bild laden sollen. Dies muss als eine Zeichenkette festgelegt werden, die den Namen eines oder mehrerer Lademodule enthält. Möglicherweise möchten Sie diese Einstellung auf `Inbuilt` setzen, um die bestmögliche Leistung zu erzielen. Es werden jedoch nur Formate erkannt, die von Hollywoods eingebautem Bild-Lademodul unterstützt werden (siehe oben für Einzelheiten). Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, die das angegebene Bild öffnen sollen. Dies muss als eine Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()` gesetzt wurde. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Siehe [Abschnitt 43.42 \[LoadBrush\]](#), [Seite 944](#), für eine Liste der unterstützten Bildformate.

## EINGABEN

`file$` die zu überprüfende Datei

**table** optional: Tabelle mit weiteren Argumenten (V6.0)

### RÜCKGABEWERTE

**ret** **True**, wenn das Bild in einem unterstützten Format vorliegt, andernfalls **False**

**table** Tabelle, welche mit den oben genannten Feldern initialisiert wird; nur gültig, wenn das Argument **ret True** ist

## 28.25 Matrix2D

### BEZEICHNUNG

Matrix2D – erstellt eine 2x2 Transformationsmatrix (V10.0)

### ÜBERSICHT

**m** = Matrix2D(**sx**, **sy**, **angle**)

### BESCHREIBUNG

Dieser Befehl kombiniert die durch **sx** und **sy** angegebenen Skalierungskoeffizienten sowie den durch **angle** angegebenen Rotationsfaktor zu einer 2x2-Transformationsmatrix und gibt diese zurück. Die 2x2-Transformationsmatrix wird als Tabelle zurückgegeben, in der die folgenden Felder initialisiert sind:

**sx**: Faktor der Skalierung für die x-Achse.

**rx**: Faktor der Drehung auf der x-Achse.

**ry**: Faktor der Drehung auf der y-Achse.

**sy**: Faktor der Skalierung für die y-Achse.

Sie können die Matrix dann an Befehle wie **TransformBox()** oder **TransformPoint()** übergeben.

### EINGABEN

**sx** x-Skalierungskoeffizienten

**sy** y-Skalierungskoeffizienten

**angle** Drehwinkel

### RÜCKGABEWERTE

**m** 2x2-Transformationsmatrix, die als Tabelle zurückgegeben wird

## 28.26 MixRGB

### BEZEICHNUNG

MixRGB – mischt zwei Farben (V2.0)

### ÜBERSICHT

**color** = MixRGB(**col1**, **col2**, **ratio**)

**BESCHREIBUNG**

Dieser Befehl mischt die beiden angegebenen Farben in einem bestimmten Verhältnis, welches im Bereich von 0 bis 255 liegt. Die zweite Farbe wird mit der ersten im angegebenen Verhältnis, d.h. wenn das Verhältnis gleich 0 ist, wird `col1` zurückgegeben und falls das Verhältnis 255 beträgt, wird `col2` zurückgegeben.

`ratio` kann auch eine prozentige Angabe enthalten, z.B. "50%".

Siehe auch `ARGB()`, `RGB()` und `GetRandomColor()`.

**EINGABEN**

`col1`      Farbe 1 im **RGB-Format**

`col2`      Farbe 2 im **RGB-Format**

`ratio`      Mischverhältnis (0 bis 255 oder prozentige Angabe)

**RÜCKGABEWERTE**

Farbe      die Mischfarbe

**BEISPIEL**

```
c = MixRGB(#RED, #BLUE, 128)
```

Dies wird eine Art rosa mischen.

## 28.27 Red

**BEZEICHNUNG**

Red – gibt den Rot-Anteil einer Farbe zurück (V1.9)

**ÜBERSICHT**

```
r = Red(color)
```

**BESCHREIBUNG**

Dieser Befehl gibt den 8-bit Rotanteil einer Farbe im **RGB-Format** zurück.

Siehe auch `Blue()` und `Green()`.

**EINGABEN**

`color`      eine Farbe im **RGB-Format**

**RÜCKGABEWERTE**

`r`            Rotanteil der Farbe (0-255)

**BEISPIEL**

```
r = Red($B2C024)
```

Gibt `$B2` zurück, welches den Rotanteil der Farbe darstellt.

## 28.28 RGB

### BEZEICHNUNG

RGB – mischt eine RGB-Farbe (V1.9)

### ÜBERSICHT

```
color = RGB(red, green, blue)
```

### BESCHREIBUNG

Dieser Befehl mischt eine **RGB-Farbe** aus ihren Rot-, Grün- und Blaukomponenten zusammen. Jede Komponente reicht von 0 bis 255.

Siehe auch **ARGB()**, **GetRandomColor()** und **MixRGB()**.

### EINGABEN

```
red      Rotanteil (0-255)
green    Grünanteil (0-255)
blue     Blauanteil (0-255)
```

### RÜCKGABEWERTE

```
color     eine Farbe im RGB-Format
```

### BEISPIEL

```
color = RGB(255, 255, 255) ; mischt Weiß
color = RGB(255, 0, 0)     ; mischt Rot
color = RGB(255, 255, 0)   ; mischt Gelb (Rot + Grün)
color = RGB(255, 0, 255)   ; mischt Magenta (Rot + Blau)
```

## 28.29 RGB-Farben

Hollywood akzeptiert Farben im 24Bit RGB-Format. Dies bedeutet, dass Sie 8-Bit pro Farbe zur Verfügung haben. RGB24 ist heute das gemeinsame Format für die Farbenangabe, z.B. wird es auch in HTML verwendet.

Sie werden in der Regel RGB-Farben in hexadezimaler Schreibweise beginnend mit einem '\$' Präfix angeben. Die allgemeine Syntax für eine RGB-Farbe in hexadezimaler Schreibweise ist

**\$RRGGBB**

RR: Farbintensität von Rot (maximum = 255 = \$FF = 100% Rot)

GG: Farbintensität von Grün (maximum = 255 = \$FF = 100% Grün)

BB: Farbintensität von Blau (maximum = 255 = \$FF = 100% Blau)

Farben werden erzeugt, indem die rote, grüne und blaue Komponenten miteinander vermischt werden. Jede Farbkomponente kann maximal einen Wert von 255 aufweisen.

Hier sind einige Farbbeispiele:

```
#BLACK  = $000000 - Schwarz ist 0% aller Komponenten
#WHITE  = $FFFFFF - Weiß ist 100% aller Komponenten
#RED    = $FF0000 - reines Rot ist 100% von Rot
#GREEN  = $00FF00 - 100% von Grün
```

**#BLUE** = \$0000FF - 100% von Blau  
**#YELLOW** = \$FFFF00 - Gelb wird gemixt aus 100% Rot und Grün  
**#FUCHSIA** = \$FF00FF - Magenta ist 100% von Rot und Blau  
**#GRAY** = \$939393 - Grau ist gleich viel von allem

Wenn Hollywood Sie nach einer RGB-Farbe fragt, können Sie Ihre individuelle Farbe in Hexadezimalschreibweise oder auch mit einer besonderen eingebauten Farbkonstanten angeben.

#BLACK	\$000000	
#MAROON	\$800000	
#GREEN	\$008000	
#OLIVE	\$808000	
#NAVY	\$000080	
#PURPLE	\$800080	
#TEAL	\$008080	
#GRAY	\$808080	
#SILVER	\$C0C0C0	
#RED	\$FF0000	
#LIME	\$00FF00	
#YELLOW	\$FFFF00	
#BLUE	\$0000FF	
#FUCHSIA	\$FF00FF	
#AQUA	\$00FFFF	
#WHITE	\$FFFFFF	

Siehe auch **ARGB-Farben**.

## 28.30 SaveSnapshot

### BEZEICHNUNG

SaveSnapshot – erstellt einen Schnappschuss des aktuellen Bildschirms (V2.0)

### ÜBERSICHT

SaveSnapshot(f\$[, mode, fmt, table])

### BESCHREIBUNG

Dieser Befehl macht eine Momentaufnahme des aktuellen Bildschirms und speichert sie in der in f\$ angegebenen Datei. Das Argument **mode** gibt an, welcher Bereich für die Momentaufnahme genommen werden soll. Dies kann eine der folgenden Konstanten sein:

#### #SNAPWINDOW:

Erfasst das komplette Hollywood-Display, d.h. mit Fensterrahmen. Dies ist die Standardeinstellung.

#### #SNAPDISPLAY:

Erfasst nur den Inhalt des Displays, d.h. ohne Fensterrahmen.

#### #SNAPDESKTOP:

Erfasst den gesamten Host-Bildschirm.

Beachten Sie, dass Sie `#SNAPDISPLAY` verwenden müssen, wenn das Display im Palettenmodus ist und Sie möchten, dass die Bilddatei ebenfalls palettenbasiert ist, da dies der einzige Fangmodus ist, der außer dem Inhalt des Displays keine anderen Grafiken enthält.

Das Argument `fmt` gibt das gewünschte Ausgabebildformat an. Dies kann entweder eine der folgenden Konstanten oder ein von einem Plugin bereitgestelltes Bildformat sein:

**#IMGFMT\_BMP:**

Windows-Bitmap. Hollywood unterstützt beim BMP-Format RGB- und Palettenbilder. `#IMGFMT_BMP` ist das von `SaveSnapshot()` verwendete Standardformat.

**#IMGFMT\_PNG:**

PNG-Format. Hollywood unterstützt beim PNG-Format RGB- und Palettenbilder. RGB-Bilder können auch einen Alphakanal und Palettenbilder können einen transparenten Stift haben. (V2.5)

**#IMGFMT\_JPEG:**

JPEG-Format. Beachten Sie, dass das JPEG-Format keine Alphakanäle oder palettenbasierte Grafiken unterstützt. Im Feld `Qualität` (siehe unten) können Sie die Qualitätsstufe für das JPEG-Bild festlegen (gültige Werte sind 0 bis 100, wobei 100 die beste Qualität ist). (V4.0)

**#IMGFMT\_GIF:**

GIF-Format. Da GIF-Bilder immer palettenbasiert sind, müssen RGB-Grafiken quantisiert werden, bevor sie als GIF exportiert werden können. Sie können die Tags `Colors` und `Dither` (siehe unten) verwenden, um entweder die Farbtiefe oder die Anzahl Farben in der Palette festzulegen, die dem Bild zugewiesen werden sollen und ob Dithering angewendet werden soll oder nicht. Wenn `#IMGFMT_GIF` mit einem Palettenmodus-Display verwendet wird, erfolgt keine Quantisierung. `#IMGFMT_GIF` unterstützt auch Palettenbilder mit einem transparenten Stift. (V4.5)

**#IMGFMT\_ILBM:**

IFF-ILBM-Format. Hollywood unterstützt beim IFF-ILBM-Format RGB- und Palettenbilder. Palettenbilder können auch einen transparenten Stift haben, allerdings werden Alphakanäle für dieses Ausgabeformat nicht unterstützt. (V4.5)

Mit der optionalen Tabelle `table` können Sie weitere Parameter konfigurieren:

**Dither:** Um Dithering zu aktivieren, setzen Sie dies auf `True`. Dieser Tag wird nur berücksichtigt, wenn das Zielformat palettenbasiert ist und die Quelldaten RGB sind. Der Standardwert ist `False`, was bedeutet, dass kein Dithering erfolgt.

**Depth:** Gibt die gewünschte Farbtiefe des Bildes an. Dies wird nur berücksichtigt, wenn das Format palettenbasiert ist und die Quelldaten RGB sind. Gültige Werte liegen zwischen 1 (= 2 Farben) und 8 (= 256 Farben). Die Voreinstellung ist 8. (V9.0)

- Colors:** Dies ist eine Alternative zum Tag **Depth**. Anstelle einer Bittiefe können Sie hier die Anzahl Farben für das Bild angeben. Auch dies wird nur für palettenbasierte Formate berücksichtigt, wenn die Quelldaten RGB sind. Gültige Werte liegen zwischen 1 und 256. Die Voreinstellung ist 256.
- Quality:** Hier können Sie einen Wert zwischen 0 und 100 für die Komprimierungsqualität bei verlustbehafteten Kompressionsformate angeben. Ein Wert von 100 bedeutet beste, 0 hingegen schlechteste Qualität. Dies wird nur bei Bildformaten berücksichtigt, die verlustbehaftete Kompression unterstützen. Der Standardwert ist 90, welcher eine ziemlich gute Qualität liefert.
- Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, die die angegebene Datei speichern sollen. Dies muss als eine Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen enthält. Standardmäßig wird der Adapter verwendet, der mit `SetDefaultAdapter()` gesetzt wurde. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V10.0)
- UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Hier ist eine Übersicht, die zeigt, welche Formate welche Tags unterstützen:

	<b>BMP</b>	<b>PNG</b>	<b>JPEG</b>	<b>GIF</b>	<b>ILBM</b>
Dither	Nein	Nein	Nein	Ja	Nein
Colors	Nein	Nein	Nein	Ja	Nein
Quality	Nein	Nein	Ja	Nein	Nein

Siehe auch [GrabDesktop\(\)](#).

## EINGABEN

- f\$** Datei für den Schnappschuss
- mode** optional: Bereich, der berücksichtigt wird (voreingestellt ist `#SNAPWINDOW`)
- fmt** optional: Bildformat: `#IMGFMT_BMP` (2.0), `#IMGFMT_PNG` (V2.5), `#IMGFMT_JPEG` (V4.0), `#IMGFMT_GIF` oder `#IMGFMT_ILBM` (voreingestellt ist `#IMGFMT_BMP`)
- table** optional: Tabelle für weitere Optionen (V4.5)

## BEISPIEL

```
SaveSnapshot("Snap.bmp")
```

Speichert einen Schnappschuss des Hollywoodfensters als Datei "Snap.bmp" ab.

## 28.31 SetClipRegion

### BEZEICHNUNG

SetClipRegion – aktiviert eine Clip-Region (V2.0)

### ÜBERSICHT

SetClipRegion(id)

### BESCHREIBUNG

Dieser Befehl installiert eine Clip-Region, die mit dem Befehl `CreateClipRegion()` zuvor erstellt wurde. Die Clip-Region wird aktiv, bis Sie `SetClipRegion()` mit dem speziellen Wert `#NONE` aufrufen, welcher dann die Clip-Region entfernt. Hollywood wird die Clip-Region automatisch löschen, wenn Sie ein neues Hintergrundbild anzuzeigen.

Wenn eine Clip-Region installiert ist, wird dies auch Auswirkungen auf die speziellen Koordinatenkonstanten von Hollywood haben, z.B. `#RIGHT` bedeutet dann die rechte Seite der Clip-Region. Randeinstellungen werden ebenfalls angepasst.

Sie können auch einen Clip-Region installieren, während `SelectBrush()` aktiv ist. Diese Clip-Region wird dann deaktiviert, wenn `EndSelect()` aufgerufen wird.

Wenn Sie eine Clip-Region auf dem Hauptbildschirm installieren und Sie rufen einen Hintergrundbildschirm-Wiedergabebefehl (Off-Screen-Rendering) auf (zum Beispiel `SelectBrush()`), wird die Clip-Region vorübergehend deaktiviert. Sie wird aber wiederhergestellt, sobald Sie den Befehl `EndSelect()` aufrufen.

Wenn Ebenen aktiv sind, kann jede Ebene ihre private Clip-Region haben. Falls die Ebene transformiert (skaliert oder rotiert) wird, wird auch ihre Clip-Region transformiert. Die Standard-Clip-Region einer Ebene ist die Clip-Region, welche aktiv war, als die Ebene erzeugt wurde. Sie können mit dem Stilelement `ClipRegion` des Befehls `SetLayerStyle()` die Clip-Region einer Ebene ändern.

Ausnahmen: Sie können `SetClipRegion()` nicht verwenden, wenn...

- das aktuelle Ausgabeziel ein Alphakanal ist, das heißt `SelectAlphaChannel()` ist aktiv
- das aktuelle Ausgabeziel eine Maske ist, das heißt `SelectMask()` ist aktiv

Siehe [Abschnitt 28.8 \[CreateClipRegion\]](#), Seite 574, für Details.

Siehe auch `FreeClipRegion()`.

### EINGABEN

id            Identifikator der installierten Clip-Region; benutzen Sie den Befehl `CreateClipRegion()`, um eine Clip-Region zu erstellen

### BEISPIEL

```
CreateClipRegion(1, #BOX, #CENTER, #CENTER, 320, 240)
SetClipRegion(1)
Circle(0, 0, 100, #RED)
Circle(439, 0, 100, #RED)
Circle(439, 279, 100, #RED)
Circle(0, 279, 100, #RED)
```

Installiert eine Clip-Region der Größe 320x240 in der Mitte eines 640x480 Display und zeichnet vier Kreise in alle Ecken. Jedoch werden aufgrund der Clip-Region nur Teile der Kreise sichtbar.



## 28.32 SetDrawTagsDefault

### BEZEICHNUNG

SetDrawTagsDefault – setzt die Standardwerte für die Standard-Tags zum Zeichnen (V5.0)

### ÜBERSICHT

SetDrawTagsDefault(table)

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die Standardwerte der Standard-Tags zum Zeichnen zu ändern. Die **Standard-Tags zum Zeichnen** sind allgemeine Optionen, die von den meisten Zeichnungsbefehlen von Hollywood unterstützt werden. Die Standard-Tags zum Zeichnen werden immer innerhalb einer optionalen Tabelle übergeben, die das letzte Argument eines Befehls darstellt. Wenn ein bestimmter Tag zum Zeichnen nicht angegeben ist, wird Hollywood auf eine interne Standardeinstellung zurückgreifen. Diese Standardeinstellung kann mit `SetDrawTagsDefault()` geändert werden.

Zum Beispiel, nehmen wir an, dass Sie immer einen Ankerpunkt von 0.5/0.5 anstelle von 0.0/0.0 verwenden möchten, welcher der Standardankerpunkt in Hollywood ist. Anstatt den ausdrücklich gewünschten Ankerpunkt bei allen Befehlen einzutragen, die Sie aufrufen, können Sie einfach diesen Ankerpunkt als neuen Standardankerpunkt definieren, den alle Zeichnungsbefehle verwenden sollen, wenn kein anderer Punkt gegeben ist. Siehe unten für ein Beispiel. Sie könnten auch `SetDrawTagsDefault()` verwenden, um die Standardeinfügeposition für Ebenen von vorderste auf hinterste Position zu ändern usw.

In der Tabelle, die Sie diesem Befehl übergeben müssen, können alle Tags enthalten sein, die in der **Dokumentation** der **Standard-Tags zum Zeichnen** aufgeführt sind. Für jeden Tag, den Sie benutzen, müssen Sie einen Standardwert angeben, der von Hollywood verwendet wird, wenn kein anderer Wert angegeben ist.

Siehe **Abschnitt 29.17 [Standard-Tags zum Zeichnen]**, Seite 613, für mehr Informationen über die Standard-Tags, die fast alle Zeichnungsbefehle von Hollywood unterstützen.

### EINGABEN

**table**      Tabelle mit einem oder mehr **Standard-Tags zum Zeichnen** und einem voreingestellten Wert für jeden Tag

### BEISPIEL

```
SetDrawTagsDefault({AnchorX = 0.5, AnchorY = 0.5})
DisplayBrush(1, 0, 0)
Box(100, 100, 200, 150, #RED)
```

Der obige Code setzt 0.5/0.5 als Standardankerpunkt. Die Aufrufe der Befehle `DisplayBrush()` und `Box()` werden dann diesen Ankerpunkt verwenden, da kein anderer Ankerpunkt angegeben wurde. In diesem Fall fallen die Zeichnungsbefehle wieder auf den Standardankerpunkt, der durch den Aufruf von `SetDrawTagsDefault()` verändert wurde.

## 28.33 SetFPSLimit

### BEZEICHNUNG

SetFPSLimit – limitiert die Einzelbilder pro Sekunde (V1.5)

### ÜBERSICHT

SetFPSLimit(fps)

### BESCHREIBUNG

Dieser Befehl schränkt die Anzahl der Einzelbilder pro Sekunde von bestimmten Hollywood-Befehlen ein, die `VWait()` aufrufen. Diese Befehle sind: `PlayAnim()`, `MoveBrush()`, `DisplayTransitionFX()` `Flip()` usw. Unter normalen Umständen ist es nicht erforderlich, diesen Befehl zu verwenden, da Hollywood die Geschwindigkeit intern automatisch einschränkt. Standardmäßig werden die oben aufgeführten Befehle nie mehr Einzelbilder pro Sekunde zeichnen, als der Monitor seine Anzeige auffrischt. Sie können dieses Verhalten mit diesem Befehl deaktivieren, aber es wird nicht empfohlen (es sei denn, Sie wissen wirklich, was Sie tun).

Das Beste ist, den Standard-Video-Synchronizer von Hollywood zu verwenden, um die beste Darstellung der Grafiken zu erhalten. Verwenden Sie diesen Befehl mit Vorsicht oder besser überhaupt nicht.

Wenn Sie 0 für `fps` angeben, wird der Standard-Video-Synchronizer wieder hergestellt.

Siehe auch `GetFPSLimit()` und `VWait()`.

### EINGABEN

<code>fps</code>	maximal zulässige Anzahl der Einzelbilder pro Sekunde oder 0, um wieder den Hollywood Standard-Video-Synchronizer zu benutzen.
------------------	--

## 28.34 TransformBox

### BEZEICHNUNG

TransformBox – wendet eine affine Transformation auf ein Rechteck an (V10.0)

### ÜBERSICHT

`tx, ty, tw, th = TransformBox(x, y, w, h, m[, origin, anchorx, anchory])`

### BESCHREIBUNG

Dieser Befehl wendet die durch `m` angegebene 2x2-Transformationsmatrix auf das durch `x`, `y`, `w` und `h` angegebene Rechteck an und gibt die Größe und die Koordinaten des transformierten Rechtecks in `tx`, `ty`, `tw` und `th` zurück. Der Parameter `origin` ist ein boolescher Wert, der angibt, ob das Rechteck zum Ursprung verschoben werden soll, bevor es transformiert wird. Die Werte `anchorx` und `anchory` geben den für die Transformation zu verwendenden Ankerpunkt an. Dies sollte ein Fließkommawert zwischen 0,0 und 1,0 sein, wobei 0,0 den linken/oberen Rand und 1,0 den rechten/unteren Rand angibt. `anchorx` und `anchory` sind beide standardmäßig 0, was bedeutet, dass die obere linke Ecke des Rechtecks der Standardankerpunkt ist.

Die Transformationsmatrix `m` muss als Tabelle übergeben werden, in der die folgenden Felder initialisiert sind:

<code>sx</code> :	Faktor der Skalierung für die x-Achse.
-------------------	--

**rx:** Faktor der Drehung auf der x-Achse.  
**ry:** Faktor der Drehung auf der y-Achse.  
**sy:** Faktor der Skalierung für die y-Achse.

Sie können den Befehl `Matrix2D()` verwenden, um eine solche Matrix zu erstellen. Siehe [Abschnitt 28.25 \[Matrix2D\]](#), [Seite 586](#), für Details.

#### EINGABEN

**x** linker Rand des Rechtecks  
**y** oberer Rand des Rechtecks  
**w** Breite des Rechtecks  
**h** Höhe des Rechtecks  
**m** Tabelle mit einer 2x2-Transformationsmatrix  
**origin** optional: ob das Rechteck zum Ursprung verschoben werden soll, bevor es transformiert wird (voreingestellt ist `True`)  
**anchorx** optional: horizontaler Ankerpunkt (voreingestellt ist 0, was linker Rand bedeutet)  
**anchory** optional: vertikaler Ankerpunkt (voreingestellt ist 0, was der obere Rand bedeutet)

#### RÜCKGABEWERTE

**tx** transformierter linker Rand  
**ty** transformierter oberer Rand  
**tw** transformierte Breite  
**th** transformierte Höhe

## 28.35 TransformPoint

#### BEZEICHNUNG

`TransformPoint` – wendet eine affine Transformation auf einen Punkt an (V10.0)

#### ÜBERSICHT

`tx, ty = TransformPoint(px, py, m)`

#### BESCHREIBUNG

Dieser Befehl wendet die durch `m` angegebene 2x2-Transformationsmatrix auf den durch `px` und `py` angegebenen Punkt an und gibt die Koordinaten des transformierten Punktes in `tx` und `ty` zurück. Die Transformationsmatrix `m` muss als Tabelle übergeben werden, die folgende Felder initialisiert hat:

**sx:** Faktor der Skalierung für die x-Achse.  
**rx:** Faktor der Drehung auf der x-Achse.  
**ry:** Faktor der Drehung auf der y-Achse.

**sy:** Faktor der Skalierung für die y-Achse.

Sie können den Befehl `Matrix2D()` verwenden, um eine solche Matrix zu erstellen. Siehe [Abschnitt 28.25 \[Matrix2D\]](#), [Seite 586](#), für Details.

#### EINGABEN

**px** horizontale Punktkoordinate  
**py** vertikale Punktkoordinate  
**m** Tabelle mit einer 2x2-Transformationsmatrix

#### RÜCKGABEWERTE

**tx** transformierte horizontale Position  
**ty** transformierte vertikale Position

## 28.36 VWait

#### BEZEICHNUNG

VWait – wartet bis zum nächsten Bildschirmaufbau

#### ÜBERSICHT

`VWait()`

#### BESCHREIBUNG

Dieser Befehl hält das Programm an, bis der nächste Bildschirm aufgebaut wird.

Siehe auch [GetFPSLimit\(\)](#) und [SetFPSLimit\(\)](#).

#### EINGABEN

keine

## 29 Grafikgrundelemente

### 29.1 Arc

#### BEZEICHNUNG

Arc – zeichnet eine Teilellipse (V2.0)

#### ÜBERSICHT

Arc(x, y, xradius, yradius, start, end[, color], table)

#### BESCHREIBUNG

Dieser Befehl zeichnet eine Teilellipse an der angegebenen Position x und y mit den angegebenen Radien xradius und yradius sowie der Farbe color (RGB-Wert) in dem Stil, welcher in den Befehlen `SetFormStyle()` und `SetFillStyle()` konfiguriert wurde. Den Argumenten start und end übergeben Sie den Start- und Endwinkel der Ellipse in Grad. Wenn Sie eine geschlossene Ellipse zeichnen möchten, muss das Startargument 0 und der Endwinkel 360 sein. In diesem Fall wäre das Verwenden des Befehls `Ellipse()` natürlich einfacher.

Die Breite der Teilellipse ist  $xradius * 2 + 1$  (Mittelpunkt) und die Höhe wird mit  $yradius * 2 + 1$  (Mittelpunkt) berechnet.

Wenn Ebenen aktiviert sind, wird dieser Befehl eine neue Ebene des Typs #ARC dem Ebenenstapel hinzuzufügen.

Neu in Hollywood 2.0: color kann auch ein ARGB-Wert zum Zeichnen von Alpha-Transparenz sein.

Ab Hollywood 4.5 wird eine neue Syntax für diesen Befehl verwendet. Es wird nur noch mit einer einzigen Tabelle als optionales Argument gearbeitet. Die alte Syntax wird weiterhin aus Kompatibilitätsgründen unterstützt. Die optionale Tabelle table kann verwendet werden, um den Stil des Bogens zu konfigurieren. Folgende Optionen sind möglich:

##### Clockwise:

Sie können mit diesem Tag angeben, ob der elliptischen Bogen im Uhrzeigersinn (True) gezeichnet wird. Wenn Sie ihn auf False setzen, wird Arc() die Winkel gegen den Uhrzeigersinn verbinden. Dieser Tag standardmäßig auf True gesetzt. (V2.5)

Darüber hinaus kann die optionale Tabelle table auch eine oder mehrere der Standard-Tags für alle Zeichnungsbefehle enthalten. Siehe Abschnitt 29.17 [Standard-Tags zum Zeichnen], Seite 613, für weitere Informationen über die Standard-Tags, die fast alle Zeichnungsbefehle von Hollywood unterstützen.

Bitte beachten Sie, dass aus historischen Gründen die Position, die für diesen Befehl in den beiden Argumenten x und y zu übergeben sind, wirklich die linke obere Ecke des Begrenzungsrechtecks des elliptischen Bogens sind. Dies könnte verwirrend sein, da traditionell elliptische Bögen im Verhältnis zu ihrem Mittelpunkt gezogen werden. Durch einen Designfehler in Hollywood 1.0 weicht aber Hollywood leider von diesem Standard ab.

Beachten Sie, dass beim Zeichnen auf ein palettenbasiertes Ziel und der Palettenmodus auf `#PALETTEMODE_PEN` eingestellt ist, dieser Befehl mit dem Stift zeichnet, der mit `SetDrawPen()` gesetzt wurde, anstatt der Farbe, die dem Befehl übergeben wurde.

Siehe auch `Box()`, `Circle()`, `Ellipse()`, `Line()`, `Plot()`, `Polygon()` und `Cls()`.

## EINGABEN

<code>x</code>	x-Position der Ellipse (linke obere Ecke des Begrenzungsrechtecks)
<code>y</code>	y-Position der Ellipse (linke obere Ecke des Begrenzungsrechtecks)
<code>xradius</code>	x-Radius der Ellipse
<code>yradius</code>	y-Radius der Ellipse
<code>start</code>	Startwinkel in Grad (muss positiv sein)
<code>end</code>	Endwinkel in Grad (muss auch positiv sein)
<code>color</code>	optional: RGB- oder ARGB-Farbe (voreingestellt ist <code>#BLACK</code> ) <code>color</code> ist optional, da es nicht erforderlich ist, wenn Sie eine Maske oder einen Alpha-Kanal benutzen
<code>table</code>	optional: Tabelle mit weiteren Argumenten; kann das oben aufgeführte und/oder eines der Standard-Tags zum Zeichnen sein. (V4.5)

## BEISPIEL

```
Arc(0, 0, 183, 183, 45, 315, #WHITE)
```

```
Circle(164, 33, 16, #BLACK)
```

Zeichnet eine Pac-Manform.

## 29.2 Box

### BEZEICHNUNG

`Box` – zeichnet ein Rechteck

### ÜBERSICHT

```
Box(x, y, width, height[, color], table)
```

### BESCHREIBUNG

Dieser Befehl zeichnet ein Rechteck von der Position `x/y` mit der angegebenen Breite `width` und Höhe `height` sowie der Farbe `color` (RGB-Wert). Das Rechteck wird in dem Stil gezeichnet, wie es mit dem Befehl `SetFormStyle()` angegeben wurde und die Füllung des Rechtecks kann durch den Befehl `SetFillStyle()` konfiguriert werden.

Wenn Ebenen aktiviert sind, wird dieser Befehl eine neue Ebene des Typs `#BOX` dem Ebenenstapel hinzuzufügen.

Neu in Hollywood 2.0: Die Farbe kann auch ein ARGB-Wert für Alphakanal-Zeichnung sein.

Ab Hollywood 4.5 verwendet dieser Befehl eine neue Syntax mit nur einer einzigen Tabelle als optionales Argument. Die alte Syntax wird aus Kompatibilitätsgründen weiterhin

unterstützt. Das optionale `Tabelle`-Argument kann verwendet werden, um den Stil des Rechtecks zu konfigurieren. Folgende Optionen sind möglich:

**RoundLevel:**

Mit dieser Option können Sie ein Rechteck mit runden Ecken angeben. `RoundLevel` ist eine Prozentzahl, die angibt, wie rund die Ecken sein sollen (0 = gar nicht rund, 100 = absolut rund). Voreingestellt ist 0 (keine runde Ecken) (V1.9).

**CornerA, CornerB, CornerC, CornerD:**

Mit diesen vier Optionen können Sie die Feinabstimmung der Eckenabrundung des Rechtecks vornehmen. Sie können eine Rundungsstufe in Prozent (0 = gar nicht rund, 100 = absolut rund) für jede Ecke des Rechtecks angeben, so dass Sie ein Rechteck erstellen können, bei dem nicht alle Ecken abgerundet sind oder wo die verschiedenen Ecken unterschiedliche Rundungen haben. Diese Option setzt alle Einstellungen von `RoundLevel` außer Kraft. (V5.0)

Darüber hinaus kann die optionale Tabelle `table` auch eine oder mehrere der **Standard-Tags** für alle **Zeichnungsbefehle** enthalten. Siehe **Abschnitt 29.17 [Standard-Tags zum Zeichnen]**, **Seite 613**, für weitere Informationen über die Standard-Tags, die fast alle Hollywood Zeichnungsbefehle unterstützen.

Beachten Sie, dass beim Zeichnen auf ein palettenbasiertes Ziel und der Palettenmodus auf `#PALETTEMODE_PEN` eingestellt ist, dieser Befehl mit dem Stift zeichnet, der mit `SetDrawPen()` gesetzt wurde, anstatt der Farbe, die dem Befehl übergeben wurde.

Siehe auch `Arc()`, `Circle()`, `Ellipse()`, `Line()`, `Plot()`, `Polygon()` und `Cls()`.

## EINGABEN

<code>x</code>	Anfangs-X-Wert
<code>y</code>	Anfangs-Y-Wert
<code>width</code>	gewünschte Breite
<code>height</code>	gewünschte Höhe
<code>color</code>	optional: <b>RGB-</b> oder <b>ARGB-Farbe</b> (voreingestellt ist <code>#BLACK</code> ) <code>color</code> ist optional, da es nicht erforderlich ist, wenn Sie eine Maske erstellen oder einen Alpha-Kanal verwenden
<code>table</code>	optional: Tabelle mit weiteren Argumenten; kann das oben aufgeführte und/oder eines der <b>Standard-Tags zum Zeichnen</b> sein. (V4.5)

## BEISPIEL

```
Box(0, 0, 640, 480, #YELLOW)
```

Dieser Code zeichnet ein gelbes Rechteck als Rahmen von der Größe 640x480.

```
Box(0, 0, 300, 200, #RED, {RoundLevel = 25})
```

Dieses Beispiel zeichnet ein rotes Rechteck mit 25-prozentig abgerundeten Ecken.

## 29.3 Circle

### BEZEICHNUNG

Circle – zeichnet einen Kreis

### ÜBERSICHT

```
Circle(x, y, radius[, color], table)
```

### BESCHREIBUNG

Dieser Befehl zeichnet einen Kreis von der Position **x**/**y** mit dem angegebenen Radius **radius** und der Farbe **color** (RGB-Wert). Der Kreis wird in dem Stil gezeichnet, wie es mit dem Befehl **SetFormStyle()** angegeben wurde und die Füllung des Kreises kann durch den Befehl **SetFillStyle()** konfiguriert werden.

Die Breite und Höhe des Kreises ist **radius** \* 2 + 1 (Kreismitte)

Wenn Ebenen aktiviert sind, wird dieser Befehl eine neue Ebene des Typs **#CIRCLE** dem Ebenenstapel hinzuzufügen.

Neu in Hollywood 2.0: **color** kann auch ein **ARGB-Wert** zum Zeichnen von Alpha-Transparenz sein.

Ab Hollywood 4.5 kann mit der optionale Tabelle **table** auch eine oder mehrere der **Standard-Tags für alle Zeichnungsbefehle** enthalten. Siehe **Abschnitt 29.17 [Standard-Tags zum Zeichnen]**, **Seite 613**, für weitere Informationen über die Standard-Tags, die fast alle Hollywood Zeichnungsbefehle unterstützen.

Bitte beachten Sie, dass aus historischen Gründen die Position, die für diesen Befehl in den beiden Argumenten **x** und **y** zu übergeben sind, wirklich die linke obere Ecke des Begrenzungsrechtecks des Kreises sind. Dies könnte verwirrend sein, da traditionell Kreise im Verhältnis zu ihrem Mittelpunkt gezogen werden. Durch einen Designfehler in Hollywood 1.0 weicht aber Hollywood leider von diesem Standard ab.

Beachten Sie, dass beim Zeichnen auf ein palettenbasiertes Ziel und der Palettenmodus auf **#PALETTEMODE\_PEN** eingestellt ist, dieser Befehl mit dem Stift zeichnet, der mit **SetDrawPen()** gesetzt wurde, anstatt der Farbe, die dem Befehl übergeben wurde.

Siehe auch **Arc()**, **Box()**, **Ellipse()**, **Line()**, **Plot()**, **Polygon()** und **Cls()**.

### EINGABEN

<b>x</b>	x-Position (linke obere Ecke des Begrenzungsrechtecks)
<b>y</b>	y-Position (linke obere Ecke des Begrenzungsrechtecks)
<b>radius</b>	Radius des Kreises
<b>color</b>	optional: <b>RGB-</b> oder <b>ARGB-Farbe</b> (voreingestellt ist <b>#BLACK</b> ) <b>color</b> ist optional, da es nicht erforderlich ist, wenn Sie eine Maske oder einem Alpha-Kanal verwenden
<b>table</b>	optional: Tabelle mit weiteren Argumenten; kann eines der <b>Standard-Tags zum Zeichnen</b> sein. (V4.5)



## 29.4 Cls

### BEZEICHNUNG

Cls – löscht den Bildschirm (V2.0)

### ÜBERSICHT

Cls([color])

### BESCHREIBUNG

Dieser Befehl löscht schnell das aktuelle Ausgabeziel. Sein Verhalten hängt davon ab, was zur Zeit als Ausgabeziel ausgewählt ist.

Wenn das Ausgabeziel ein Display ist, wird die aktuelle Hintergrundfüllung wieder hergestellt und alle Ebenen sowie Sprites werden entfernt. Das Farbbargument `color` wird in diesem Fall nicht verwendet.

Wenn das Ausgabeziel ein Pinsel ist, wird er mit der angegebenen Farbe `color` gelöscht.

Wenn das Ausgabeziel ein Hintergrundbild ist, das heißt wenn bei `SelectBGPic()` `mode` auf `#SELMODE_LAYERS` eingestellt ist, werden alle Ebenen dieses Bildes entfernt. Das Farbbargument wird in diesem Fall nicht verwendet.

Wenn das Ausgabeziel eine Maske ist, wird `Cls()` die Pixel unter Verwendung des Modus `mode` deaktivieren, der durch `SetMaskMode()` installiert wurde. Das Farbbargument wird in diesem Fall nicht verwendet.

Wenn das Ausgabeziel ein Alphakanal ist, wird `Cls()` die Pixel mit der Intensität füllen, welche mit `SetAlphaIntensity()` angegeben wurde. Das Farbbargument wird in diesem Fall nicht verwendet.

Beachten Sie, dass, wenn das aktuelle Zeichnungsziel palettenbasiert ist und der Palettenmodus auf `#PALETTEMODE_PEN` eingestellt ist, dieser Befehl den Bildschirm unter Verwendung des Stifts löscht, der mit `SetDrawPen()` gesetzt wurde, anstatt der Farbe, die dem Befehl übergeben wurde.

Siehe auch `ClearScreen()`.

### EINGABEN

`color` optional: RGB-Farbe (voreingestellt ist `#BLACK`); nur erforderlich, wenn `SelectBrush()` aktiv ist

## 29.5 Ellipse

### BEZEICHNUNG

Ellipse – zeichnet eine Ellipse

### ÜBERSICHT

Ellipse(x, y, xradius, yradius[, color], table))

### BESCHREIBUNG

Dieser Befehl zeichnet eine Ellipse an der angegebenen Position `x` und `y` mit den angegebenen Radien `xradius` und `yradius` sowie der Farbe `color` (RGB-Wert). Die Ellipse wird in dem Stil gezeichnet, wie es mit dem Befehl `SetFormStyle()` angegeben wurde und die Füllung der Ellipse kann durch den Befehl `SetFillStyle()` konfiguriert werden.

Die Breite der Ellipse ist `xradius * 2 + 1` (Mittelpunkt) und die Höhe wird mit `yradius * 2 + 1` (Mittelpunkt) berechnet.

Wenn Ebenen aktiviert sind, wird dieser Befehl eine neue Ebene des Typs `#ELLIPSE` dem Ebenenstapel hinzuzufügen.

Neu in Hollywood 2.0: `color` kann auch ein **ARGB-Wert** zum Zeichnen von Alpha-Transparenz sein.

Ab Hollywood 4.5 kann mit der optionale Tabelle `table` auch eine oder mehrere der **Standard-Tags für alle Zeichnungsbefehle** enthalten. Siehe **Abschnitt 29.17 [Standard-Tags zum Zeichnen]**, **Seite 613**, für weitere Informationen über die Standard-Tags, die fast alle Hollywood Zeichnungsbefehle unterstützen.

Bitte beachten Sie, dass aus historischen Gründen die Position, die für diesen Befehl in den beiden Argumenten `x` und `y` zu übergeben sind, wirklich die linke obere Ecke des Begrenzungsrechtecks des elliptischen Bogens sind. Dies könnte verwirrend sein, da traditionell elliptische Bögen im Verhältnis zu ihrem Mittelpunkt gezogen werden. Durch einen Designfehler in Hollywood 1.0 weicht aber Hollywood leider von diesem Standard ab.

Beachten Sie, dass beim Zeichnen auf ein palettenbasiertes Ziel und der Palettenmodus auf `#PALETTEMODE_PEN` eingestellt ist, dieser Befehl mit dem Stift zeichnet, der mit `SetDrawPen()` gesetzt wurde, anstatt der Farbe, die dem Befehl übergeben wurde.

Siehe auch `Arc()`, `Box()`, `Circle()`, `Line()`, `Plot()`, `Polygon()` und `Cls()`.

## EINGABEN

<code>x</code>	x-Position der Ellipse (linke obere Ecke des Begrenzungsrechtecks)
<code>y</code>	y-Position der Ellipse (linke obere Ecke des Begrenzungsrechtecks)
<code>xradius</code>	x-Radius der Ellipse
<code>yradius</code>	y-Radius der Ellipse
<code>color</code>	optional: <b>RGB-</b> oder <b>ARGB-Farbe</b> (voreingestellt ist <code>#BLACK</code> ) <code>color</code> ist optional, da es nicht erforderlich ist, wenn Sie eine Maske oder einem Alpha-Kanal verwenden
<code>table</code>	optional: Tabelle mit weiteren Argumenten; kann eines der <b>Standard-Tags zum Zeichnen</b> sein. (V4.5)

## 29.6 GetFillStyle

### BEZEICHNUNG

`GetFillStyle` – gibt den aktuellen Füllstil zurück (V7.1)

### ÜBERSICHT

```
style[, t] = GetFillStyle()
```

### BESCHREIBUNG

Dieser Befehl gibt den aktuellen Füllstil zurück, der mit `SetFillStyle()` gesetzt wurde. Der Rückgabewert `style` wird entweder auf `#FILLNONE`, `#FILLCOLOR`, `#FILLGRADIENT` oder `#FILLTEXTURE` gesetzt. Siehe **Abschnitt 29.14 [SetFillStyle]**, **Seite 610**, für Details.

Wenn `style` `#FILLNONE` ist, gibt `GetFillStyle()` auch eine Tabelle `t` mit den folgenden Feldern zurück:

**Thickness:**

Die Umrisssdicke in Pixeln.

Wenn `style` `#FILLGRADIENT` ist, wird die Rückgabetable die folgenden Felder enthalten:

**Type:** Farbverlaufstyp. Dies wird entweder `#LINEAR`, `#CONICAL` oder `#RADIAL` sein.

**StartColor:**

Die Startfarbe des Farbverlaufs.

**EndColor:**

Die Endfarbe des Farbverlaufs.

**Angle:** Der Winkel des Farbverlaufs. Wird nur von den Typen `#LINEAR` und `#CONICAL` unterstützt.

**CenterX, CenterY:**

Der Mittelpunkt des Farbverlaufs. Wird nur von den Typen `#RADIAL` und `#CONICAL` unterstützt.

**Balance:** Der Balancepunkt des Farbverlaufs. Wird nur von `#CONICAL` unterstützt.

**Border:** Die Rahmengröße des Farbverlaufs. Wird nur von `#RADIAL` unterstützt.

**Colors:** Wenn der Farbverlauf mehr als zwei Farben verwendet, enthält dieses Feld eine Tabelle mit allen Farben und ihren Stopppunkten.

Siehe [Abschnitt 30.6 \[CreateGradientBGPic\]](#), [Seite 624](#), für mehr Informationen über Farbverläufe.

Wenn `style` `#FILLTEXTURE` ist, enthält die Rückgabetable die folgenden Felder:

**Brush:** Der Identifikator des Pinsels für die Textur.

**X, Y:** Pixelversatz innerhalb des Pinsels, der als Startversatz für die Texturierung verwendet wurde.

Siehe [Abschnitt 29.14 \[SetFillStyle\]](#), [Seite 610](#), für mehr Informationen über die Füllstile.

Siehe auch [GetFormStyle\(\)](#), [GetLineWidth\(\)](#), [SetFormStyle\(\)](#) und [SetLineWidth\(\)](#).

## EINGABEN

keine

## RÜCKGABEWERTE

**style** der momentan verwendete Füllstil

**t** optional: Tabelle mit zusätzlichen Stilinformationen (siehe oben)

## 29.7 GetFormStyle

### BEZEICHNUNG

GetFormStyle – gibt den aktuellen Zeichnungsstil zurück (V7.1)

### ÜBERSICHT

```
style[, t] = GetFormStyle()
```

### BESCHREIBUNG

Dieser Befehl gibt den aktuellen Zeichnungsstil zurück, der mit `SetFormStyle()` gesetzt wurde. Der Rückgabewert `style` wird auf eine Kombination der Flags `#ANTIALIAS`, `#SHADOW` und `#BORDER` gesetzt. Siehe [Abschnitt 29.15 \[SetFormStyle\]](#), [Seite 611](#), für Details. Vor Hollywood 9.0 wurde dieser Stil `#EDGE` genannt.

Wenn `#SHADOW` gesetzt ist, gibt `GetFormStyle()` auch eine Tabelle als zweiten Rückgabewert zurück, die die folgenden Felder enthält:

**ShadowColor:**

Die Schattenfarbe.

**ShadowSize:**

Die Grösse bzw. der Abstand des Schattens von der Form in Pixeln.

**ShadowDir:**

Die Richtung des Schattens. Dies ist eine der [Richtungskonstanten](#).

Wenn `#BORDER` gesetzt ist, wird auch eine Tabelle mit den folgenden Feldern zurückgegeben:

**BorderColor:**

Die Farbe des Rahmens.

**BorderSize:**

Die Dicke des Rahmens in Pixel.

Siehe [Abschnitt 29.15 \[SetFormStyle\]](#), [Seite 611](#), für mehr Informationen über die Zeichnungsstile.

Siehe auch [GetFillStyle\(\)](#), [GetLineWidth\(\)](#), [SetFillStyle\(\)](#) und [SetLineWidth\(\)](#).

### EINGABEN

keine

### RÜCKGABEWERTE

`style` eine Kombination der Zeichnungsstile-Flags

`t` optional: Tabelle mit zusätzlichen Informationen über den Zeichnungsstil (siehe oben)

## 29.8 GetLineWidth

### BEZEICHNUNG

GetLineWidth – gibt die Breite der Umrisslinie zurück (V7.1)

**ÜBERSICHT**

`t = GetLineWidth()`

**BESCHREIBUNG**

Dieser Befehl gibt die aktuelle Umrissliniendicke zurück, die mit `SetLineWidth()` oder `SetFillStyle()` eingestellt wurde. Siehe [Abschnitt 29.16 \[SetLineWidth\]](#), Seite 613, für Details.

Siehe auch `GetFillStyle()`, `GetFormStyle()`, `SetFillStyle()` und `SetFormStyle()`.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`t`                    aktuelle Umrissliniendicke

## 29.9 Line

**BEZEICHNUNG**

Line – zeichnet eine Linie

**ÜBERSICHT**

`Line(x1, y1, x2, y2[, color], table)`

**BESCHREIBUNG**

Dieser Befehl zeichnet eine Linie vom Punkt `x1` und `y1` nach dem Punkt `x2` und `y2` in der in `color` angegebenen Farbe (RGB-Wert).

Wenn Sie Linien mit Antialiasing zeichnen wollen, verwenden Sie den Befehl `SetFormStyle()` und setzen bei `style` `#ANTIALIAS` ein.

Wenn Ebenen aktiviert sind, wird dieser Befehl eine neue Ebene des Typs `#LINE` dem Ebenenstapel hinzuzufügen.

Neu in Hollywood 2.0: `color` kann auch ein **ARGB-Wert** zum Zeichnen mit Alpha-Transparenz sein.

Ab Hollywood 4.5 verwendet dieser Befehl eine neue Syntax mit nur einer einzigen Tabelle als optionales Argument. Die alte Syntax wird aus Kompatibilitätsgründen weiterhin unterstützt. Das optionale Tabellenargument kann verwendet werden, um den Stil der Linie zu konfigurieren. Folgende Optionen sind möglich:

**Thickness:**

Dieser Tag kann die Stärke der Linie festlegen, die Sie zeichnen möchten. Der Mindestwert hier ist 1, die eine normale Linie ziehen wird. Dies ist auch der Standard, wenn Sie diesen Tag nicht angeben. (V2.5)

**Arrowhead:**

Mit diesem Tag können Sie die Linie in einen Pfeil umwandeln. Er kann auf eines der folgenden Tags gesetzt werden:

**#ARROWHEAD\_NONE:**

Keine Pfeilspitze. Dies ist die Voreinstellung.

**#ARROWHEAD\_SINGLE:**

Fügt eine Pfeilspitze ans Ende der Linie hinzu.

**#ARROWHEAD\_DOUBLE:**

Fügt eine Pfeilspitze am Anfang und am Ende der Linie hinzu.

Voreingestellt ist **#ARROWHEAD\_NONE**. (V9.1)

Darüber hinaus kann die optionale Tabelle **table** auch eine oder mehrere der **Standard-Tags für alle Zeichnungsbefehle** enthalten. Siehe **Abschnitt 29.17 [Standard-Tags zum Zeichnen]**, Seite 613, für weitere Informationen über die Standard-Tags, die fast alle Hollywood Zeichnungsbefehle unterstützen.

Beachten Sie, dass beim Zeichnen auf ein palettenbasiertes Ziel und der Palettenmodus auf **#PALETTEMODE\_PEN** eingestellt ist, dieser Befehl mit dem Stift zeichnet, der mit **SetDrawPen()** gesetzt wurde, anstatt der Farbe, die dem Befehl übergeben wurde.

Siehe auch **Arc()**, **Box()**, **Circle()**, **Ellipse()**, **Plot()**, **Polygon()** und **Cls()**.

**EINGABEN**

<b>x1</b>	Anfangs-X-Wert
<b>y1</b>	Anfangs-Y-Wert
<b>x2</b>	Ziel-X-Wert
<b>y2</b>	Ziel-Y-Wert
<b>color</b>	optional: <b>RGB-</b> oder <b>ARGB-Farbe</b> (voreingestellt ist <b>#BLACK</b> ) <b>color</b> ist optional, da es nicht erforderlich ist, wenn Sie eine Maske oder einem Alpha-Kanal benutzen
<b>table</b>	optional: Tabelle mit weiteren Argumenten; kann das oben aufgeführte und/oder eines der <b>Standard-Tags zum Zeichnen</b> sein. (V4.5)

**BEISPIEL**

```
Line(0, 0, 639, 479, #WHITE)
```

```
Line(639, 0, 0, 479, #WHITE)
```

Der obige Code wird ein weißes Kreuz zeichnen.

## 29.10 Plot

**BEZEICHNUNG**

**Plot** – zeichnet einen Punkt

**ÜBERSICHT**

```
Plot(x, y[, color])
```

**BESCHREIBUNG**

Dieser Befehl zeichnet einen einzelnen Pixel in der angegebenen Farbe auf das Display.

**Plot()** funktioniert nur mit deaktivierten Ebenen. Wenn Sie einen Punkt in der Größe 1x1 bei aktivierten Ebenen haben möchten, können Sie **Box()** verwenden.

Neu in Hollywood 2.0: **color** kann auch ein **ARGB-Wert** zum Zeichnen mit Alpha-Transparenz sein.

Beachten Sie, dass beim Zeichnen auf ein palettenbasiertes Ziel und der Palettenmodus auf `#PALETTEMODE_PEN` eingestellt ist, dieser Befehl mit dem Stift zeichnet, der mit `SetDrawPen()` gesetzt wurde, anstatt der Farbe, die dem Befehl übergeben wurde.

Siehe auch `Arc()`, `Box()`, `Circle()`, `Ellipse()`, `Line()`, `Polygon()` und `Cls()`.

#### EINGABEN

<code>x</code>	Position <code>x</code>
<code>y</code>	Position <code>y</code>
<code>color</code>	optional: <b>RGB-</b> oder <b>ARGB-Farbe</b> (voreingestellt ist <code>#BLACK</code> ) <code>color</code> ist optional, da es nicht erforderlich ist, wenn Sie eine Maske oder einem Alpha-Kanal verwenden

#### BEISPIEL

```
Plot(#CENTER, #CENTER, #RED)
```

Zeichnet in der Mitte des Displays einen roten Punkt.

## 29.11 Polygon

#### BEZEICHNUNG

Polygon – zeichnet ein Vieleck (V1.9)

#### ÜBERSICHT

```
Polygon(x, y, vertices, count[[, color], table])
```

#### BESCHREIBUNG

Dieser Befehl zeichnet ein Vieleck aus einer Tabelle von Eckpunkten. Das Argument `vertices` ist eine Tabelle mit `x/y`-Punkte, die verwendet werden, um das Vieleck zu zeichnen. Der Parameter `count` gibt an, wie viele Ecken Ihr Vieleck hat. Darüber hinaus müssen Sie die Farbe für das Vieleck angeben (im RGB-Format). Das Vieleck wird in dem Stil gezeichnet, wie es mit dem Befehl `SetFormStyle()` angegeben wurde und die Füllung kann durch den Befehl `SetFillStyle()` konfiguriert werden. Denken Sie auch daran, dass der letzte Eckpunkt das Vieleck schließen sollte, also gleich dem ersten Eckpunkt sein muss.

Die Ecken können auch als negative Zahl und/oder als Fließkommazahl angegeben werden, um die beste Genauigkeit zu erreichen. Hollywood wird sie nicht abrunden, bevor es die endgültige Rasterung vornimmt.

Wenn Ebenen aktiviert sind, wird dieser Befehl eine neue Ebene des Typs `#POLYGON` dem Ebenenstapel hinzuzufügen.

Neu in Hollywood 2.0: `color` kann auch ein **ARGB-Wert** zum Zeichnen von Alpha-Transparenz sein.

Ab Hollywood 4.5 kann mit der optionale Tabelle `table` auch eine oder mehrere der **Standard-Tags für alle Zeichnungsbefehle** enthalten. Siehe **Abschnitt 29.17 [Standard-Tags zum Zeichnen]**, **Seite 613**, für weitere Informationen über die Standard-Tags, die fast alle Hollywood Zeichnungsbefehle unterstützen.

Beachten Sie, dass beim Zeichnen auf ein palettenbasiertes Ziel und der Palettenmodus auf `#PALETTEMODE_PEN` eingestellt ist, dieser Befehl mit dem Stift zeichnet, der mit `SetDrawPen()` gesetzt wurde, anstatt der Farbe, die dem Befehl übergeben wurde.

Siehe auch `Arc()`, `Box()`, `Circle()`, `Ellipse()`, `Line()`, `Plot()` und `Cls()`.

#### EINGABEN

<code>x</code>	Position x des Vielecks auf dem Display
<code>y</code>	Position y des Vielecks auf dem Display
<code>vertices</code>	Tabelle mit den Eckpunkten
<code>count</code>	Anzahl der Ecken in der Tabelle
<code>color</code>	optional: RGB- oder ARGB-Farbe (voreingestellt ist <code>#BLACK</code> ) <code>color</code> ist optional, da es nicht erforderlich ist, wenn Sie eine Maske oder einem Alpha-Kanal verwenden
<code>table</code>	optional: Tabelle mit weiteren Argumenten; kann eines der Standard-Tags zum Zeichnen sein. (V4.5)

#### BEISPIEL

```
v = {}
v[0] = 0      ;X1
v[1] = 100    ;Y1
v[2] = 50     ;X2
v[3] = 0      ;Y2
v[4] = 100    ;X3
v[5] = 100    ;Y3
v[6] = 0      ;X4
v[7] = 100    ;Y4
Polygon(#CENTER, #CENTER, v, 4, #RED)
```

Der obige Code wird ein rotes Dreieck in die Mitte des Displays zeichnen.

```
Polygon(#CENTER, #CENTER, v, 4, #RED, {Rotate = 45})
```

Dieser Code zeichnet ein rotes Dreieck, welches um 45 Grad gedreht ist.

## 29.12 ReadPixel

#### BEZEICHNUNG

`ReadPixel` – liest einen Pixel aus dem Ausgabeziel (V1.5)

#### ÜBERSICHT

```
col = ReadPixel(x, y)
```

#### BESCHREIBUNG

Dieser Befehl liest den Pixel, der durch die Koordinaten `x` und `y` bestimmt wird, vom aktuellen Ausgabeziel und gibt seine Farbe in `col` zurück.

Bitte beachten: Die Farbe, welche dieser Befehl liefert, kann etwas von der ursprünglichen Farbe an dieser Stelle abweichen. Zum Beispiel, wenn Sie einen Pinsel haben, der komplett Weiß ist (das heißt alle Pixel haben die Farbe `$FFFFFF`). Wenn Sie `ReadPixel()`



verwenden, um nun die Farbe eines beliebigen Pixels von diesem Pinsel zu lesen, werden Sie die Farbe \$FFFFFF nur erhalten, wenn Hollywood auf einem Bildschirm mit 24-Bit oder 32-Bit läuft. Wenn Hollywood auf einem Bildschirm mit 16-Bit ausgeführt wird, werden Sie die Farbe \$FCF8FC, auf einem 15-Bit würden Sie \$F8F8F8 erhalten. Dies liegt daran, dass diese Bildschirme nicht 16,7 Millionen Farben haben, sondern nur 65536 (16-Bit) oder 32768 (15-Bit). Sie können den Befehl `GetRealColor()` verwenden, um herauszufinden, welche Farbe die angegebenen Farbe auf dem aktuellen Bildschirm darstellt.

Wenn das aktuelle Ausgabeziel die Maske eines Pinsels ist, dann gibt `ReadPixel()` entweder 0 für einen unsichtbaren oder 1 für einen sichtbaren Pixel zurück.

Wenn das aktuelle Ausgabeziel ein Alphakanal eines Pinsels ist, dann gibt `ReadPixel()` den Transparenzgrad des angeforderten Pixels zurück, der von 0 (volle Transparenz) bis 255 (keine Transparenz) gehen kann.

### EINGABEN

x            Position x des Pixels  
y            Position y des Pixels

### RÜCKGABEWERTE

col            Farbe oder Transparenzgrad des Pixels

### BEISPIEL

```
CreateBrush(1, 320, 256)
SelectBrush(1)
SetFillStyle(#FILLCOLOR)
Box(0, 0, 320, 256, #GREEN)
a = ReadPixel(100, 100)
EndSelect()
```

Der obige Code wird ein grünes Rechteck als Pinsel 1 erstellen und liest die Farbe des Pixel bei 100:100 ein. Die Variable a wird den Wert von #GREEN erhalten, weil der ganze Pinsel mit Grün gefüllt ist.

## 29.13 Richtungskonstanten

Die Richtungskonstanten werden von den Befehlen `SetFontStyle()` und `SetFormStyle()` verwendet, um die Richtung des Schattens des Text- oder Grafikobjekt zu konfigurieren. Es stehen acht Richtungskonstanten zur Verfügung und sie entsprechen den Punkten des Kompasses:

```
#SHDWNORTH        ;Norden
#SHDWNORTHEAST    ;Nord-Ost
#SHDWEAST          ;Osten
#SHDWSOUTHEAST    ;Süd-Ost
#SHDWSOUTH        ;Süden
#SHDWSOUTHWEST    ;Süd-West
#SHDWWEST          ;Westen
#SHDWNORTHWEST    ;Nord-West
```

## 29.14 SetFillStyle

### BEZEICHNUNG

SetFillStyle – stellt die Füllung der Grundelemente ein (V1.5)

### ÜBERSICHT

```
SetFillStyle(style)
SetFillStyle(#FILLGRADIENT, type, startcol, endcol[, angle, t]) (V2.0)
SetFillStyle(#FILLTEXTURE, brush_id[, x, y]) (V2.0)
SetFillStyle(#FILLNONE[, thickness]) (V2.0)
```

### BESCHREIBUNG

Mit diesem Befehl können Sie die Füllung für die Grundelemente `Arc()`, `Box()`, `Circle()`, `Ellipse()` und `Polygon()` einstellen. Standardmäßig wird der Füllstil `#FILLNONE` gesetzt, was bedeutet, dass nur die Umrisse gezeichnet werden.

Zur Zeit werden folgende Arten unterstützt:

#### #FILLCOLOR:

Füllt das Objekt mit Farbe.

#### #FILLNONE:

Das Objekt wird nicht gefüllt, sondern es werden nur die Umrisse gezeichnet. Ab Hollywood 2.0 können Sie auch eine Linienstärke angeben. Der Standardwert für `thickness` ist 1, was bedeutet, der Umriss wird in der Dicke eines einzelnen Pixels gemalt. Beachten Sie, dass `#FILLNONE` nur die Formstile `#SHADOW` und `#BORDER` unterstützt, wenn Ebenen aktiviert sind. Sonst wird kein Schatten oder Rahmen gezogen werden.

#### #FILLGRADIENT:

Füllt das Objekte mit einem Farbverlauf. Sie müssen drei weitere Argumente angeben, die den Typ des Farbverlaufs sowie seine Farben einstellen; Zusätzlich können Sie das optionale Argument `angle` für den Winkel angeben, um die Ausrichtung festzulegen. Für weitere Einstellungen können Sie das letzte optionale Tabellenargument benutzen. Siehe [Abschnitt 30.6 \[CreateGradientBGPic\]](#), [Seite 624](#), für Details. (V2.0)

#### #FILLTEXTURE:

Füllt das Objekte mit einer Textur. Sie müssen ein zusätzliches Argument angeben, welches die ID des Pinsels angibt, die für die Texturierung verwendet werden soll. Bitte beachten Sie, dass etwaige Transparenzkanäle (Maske oder Alpha-Kanal), die der Pinsel aufweisen kann, derzeit nicht durch Texturierung unterstützt werden. Die optionalen Parameter `x` und `y` sind neu ab Hollywood 4.6 vorhanden. Sie ermöglichen es, einen Versatz im Texturpinsel anzugeben. Texturierung wird dann von diesem im Pinsel versetzt beginnen. Der Standardwert für dieses Argument ist 0/0 und bedeutet, dass es in der oberen linken Ecke im Texturpinsel beginnt. (V2.0)

Siehe auch `GetFillStyle()`, `GetFormStyle()`, `GetLineWidth()`, `SetFormStyle()` und `SetLineWidth()`.

### EINGABEN

`style`      einer der oben genannten Stile

... zusätzliche Argumente, die vom gewählten Stil abhängen

### BEISPIEL

```
SetFillStyle(#FILLCOLOR)
Box(0, 0, 320, 256, #RED)
```

An der Position 0:0 wird ein rotgefülltes Rechteck mit der Größe von 320x256 gezeichnet.

## 29.15 SetFormStyle

### BEZEICHNUNG

SetFormStyle – stellt den Zeichnungsstil der Grundelemente ein (V2.5)

### ÜBERSICHT

```
SetFormStyle(style[, t])
```

### FRÜHERE SYNTAX

```
SetFormStyle(#SHADOW, color, distance, direction)
SetFormStyle(#BORDER, color, size)
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Zeichnungsstil der Grundelemente zu konfigurieren. Er beeinflusst das Aussehen der Befehle `Arc()`, `Box()`, `Circle()`, `Ellipse()`, `Line()` und `Polygon()`.

Der Stil muss mit einer der folgenden vordefinierten Konstanten eingestellt werden:

**#NORMAL** Dadurch werden sämtliche Formstile zurück gesetzt.

#### #ANTIALIAS

Dies setzt den Zeichnungsstil auf Antialiasing; somit werden die Grafikgrundelemente mit Antialiasing gezeichnet. Braucht zwar mehr Zeit, sieht aber besser aus.

**#SHADOW** Die Grundelemente werden nun mit einem Schatten gezeichnet. Das zweite Argument `color` gibt die Schattenfarbe an. Diese Farbe kann entweder ein **RGB-** oder **ARGB-Wert** sein. Schatten-Transparenz wird voll unterstützt. Das dritte Argument `distance` gibt den Abstand des Schattens von der Hauptform in Pixeln und das vierte Argument `direction` gibt die Richtung des Schattens an. Dies muss einer der **Richtungskonstanten** sein. Bitte beachten Sie, dass **#SHADOW** nicht die Füllung **#FILLNONE** unterstützt, wenn Ebenen deaktiviert sind.

**#BORDER** Ein Rahmen mit der angegebenen Größe wird um das Grundelement gezeichnet werden. Das zweite Argument `color` gibt die Farbe für den Rahmen an. Diese Farbe kann entweder ein **RGB-** oder **ARGB-Wert** sein. Rahmentransparenz wird voll unterstützt. Das dritte Argument `size` gibt die gewünschte Dicke des Rahmens in Pixeln an. Bitte beachten Sie, dass **#BORDER** nicht die Füllung **#FILLNONE** unterstützt, wenn Ebenen deaktiviert sind. Vor Hollywood 9.0 wurde dieser Stil **#EDGE** genannt.

Um mehrere Formstile in einem einzigen Aufruf zu kombinieren, benutzen Sie einfach den senkrechten Balken (`|`) (bit-oder). Z.B. ein Aufruf von

`SetFormStyle(#SHADOW|#BORDER)` aktiviert die Schatten- und Rahmenzeichnungsstile. Natürlich schließt sich der Stil `#NORMAL` gegenseitig gegenüber den anderen Zeichnungsstilen aus und ist mit keinem anderen Stil kombinierbar.

Ab Hollywood 9.0 verwendet `SetFormStyle()` eine neue Syntax, die ein optionales Tabellenargument akzeptiert, das die folgenden Tags unterstützt:

**ShadowDir:**

Gibt die Richtung des Schattens an. Dies muss auf eine von Hollywoods **Richtungskonstanten** gesetzt werden. Dieser Tag wird nur berücksichtigt, wenn der Stil `#SHADOW` gesetzt wurde (siehe oben). (V9.0)

**ShadowColor:**

Gibt die Farbe des Schattens an. Dies muss ein **ARGB-Wert** sein, der eine Transparenzeinstellung enthalten kann. Dieser Tag wird nur berücksichtigt, wenn der Stil `#SHADOW` gesetzt wurde (siehe oben). (V9.0)

**ShadowSize:**

Gibt die Länge des Schattens an. Dieser Tag wird nur berücksichtigt, wenn der Stil `#SHADOW` gesetzt wurde (siehe oben). (V9.0)

**BorderColor:**

Gibt die Farbe des Rahmens an. Dies muss ein **ARGB-Wert** sein, der eine Transparenzeinstellung enthalten kann. Dieser Tag wird nur berücksichtigt, wenn der Stil `#BORDER` gesetzt wurde (siehe oben). (V9.0)

**BorderSize:**

Gibt die Dicke des Rahmens an. Dieser Tag wird nur berücksichtigt, wenn der Stil `#BORDER` gesetzt wurde (siehe oben). (V9.0)

Bitte beachten Sie, dass der Befehl `Line()` weder `#SHADOW` noch `#BORDER` unterstützt. Er akzeptiert nur den Stil `#ANTIALIAS`.

Siehe auch `GetFillStyle()`, `GetFormStyle()`, `GetLineWidth()`, `SetFillStyle()` und `SetLineWidth()`.

## EINGABEN

- `style` spezielle Stilkonstante (siehe Liste oben)
- `t` optional: Tabellenargument mit weiteren Optionen (siehe oben) (V9.0)

## BEISPIEL

```
SetFormStyle(#ANTIALIAS)
```

Der Aufruf oben ermöglicht Formen mit Antialiasing.

```
SetFormStyle(#SHADOW, {ShadowColor = ARGB(128, $939393),
    ShadowSize = 16, ShadowDir = #SHDWSOUTHEAST})
```

Der obige Code ermöglicht einen halbtransparenten grauen Schatten, der 16 Pixel südöstlich der Hauptform positioniert wird.

## 29.16 SetLineWidth

### BEZEICHNUNG

SetLineWidth – stellt die Breite der Umrisslinie ein (V5.0)

### ÜBERSICHT

SetLineWidth(width)

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die Umrisslinienstärke zu ändern, wenn die Füllung #FILLNONE eingestellt ist. Sie können auch die Linienstärke mit dem Befehl `SetFillStyle()` ändern, aber mit `SetLineWidth()` wird der Code besser lesbar.

Siehe auch `GetFillStyle()`, `GetFormStyle()`, `GetLineWidth()` und `SetFormStyle()`.

### EINGABEN

width      gewünschte Linienstärke für die Umrisszeichnung

## 29.17 Standard-Tags zum Zeichnen

Ab Hollywood 4.0 akzeptieren die meisten der Zeichnungsbefehle eine optionale Tabelle als Argument. Zum Beispiel können Sie die Tags angeben, die die Grafik skaliert oder automatisch dreht, bevor sie angezeigt wird.

Viele der Standard-Tags zum Zeichnen funktionieren nur, wenn **Ebenen** aktiviert sind. Einige können jedoch auch ohne Ebenen verwendet werden. Beachten Sie, dass die Grafikdaten des Quellobjekts nie verändert werden. Wenn Ebenen aktiviert sind, wird Hollywood die neue Ebene einfügen und sofort jede Transformation anwenden, bevor die Ebene sichtbar wird. Wenn das Ebenensystem ausgeschaltet ist, wird von der ursprünglichen Grafik eine Kopie gemacht. Die Originalgrafikdaten werden niemals geändert (im Gegensatz zu Befehlen wie `ScaleBrush()`, `RotateBrush()`, `TransformBrush()` usw., die die Daten von Pinseln ändern). So können Sie sicher sein, dass Sie immer die beste Qualität erhalten, wenn die Transformationen mit den Standard-Tags zum Zeichnen anwenden, weil Hollywood immer die Originalgrafikdaten verwendet wird.

Für jeden Standard-Zeichnungs-Tag gibt es eine Voreinstellung, falls der Tag nicht angegeben wurde. Sie können diese Standardeinstellung mit dem Befehl `SetDrawTagsDefault()` ändern. Dies ist sehr nützlich, wenn Sie dauerhaft einen anderen Standardwert für einen bestimmten Tag verwenden möchten. Zum Beispiel wenn Sie dauerhaft einen Ankerpunkt von 0.5 / 0.5 anstelle von 0.0 / 0.0 haben wollen oder Sie ändern den Einsatz der Standard-Ebenenposition von vorderste zu hinterste usw.

Die folgenden Standard-Tags sind derzeit definiert:

#### Width, Height:

Wenn Sie diese Tags angeben, wird das Objekt auf die angegebene Größe geändert und anschließend angezeigt. (V4.0)

**Rotate:** Dieser Tag wird die Grafik in Gradeinheiten drehen und dann anzeigen. (V4.0)

#### SmoothScale:

Wenn dieser Tag auf **True** gesetzt ist, wird beim Skalieren und/oder Drehen das Objekt mit Antialias interpoliert. Das sieht besser aus, ist aber auch langsamer. (V4.0)

**ScaleX, ScaleY:**

Das ist eine Alternative, um eine Objektgröße zu ändern. Sie müssen hier einen Fließkommawert übergeben, der den Skalierungsfaktor angibt. Zum Beispiel bedeutet 0.5 halb so groß, 2.0 hingegen doppelt so groß. Dies ist besonders praktisch, wenn Sie die Proportionen des Objekts beibehalten möchten. Wenn Sie den gleichen Faktor für **ScaleX** und **ScaleY** verwenden, wird das Seitenverhältnis der Grafik erhalten bleiben. Bitte beachten Sie, dass **ScaleX/ScaleY** und **Width/Height** nicht gleichzeitig gebraucht werden können. Entweder verwenden Sie **ScaleX/ScaleY** oder **Width/Height**. (V4.5)

**Transform:**

Mit diesem Tag können Sie eine 2x2 Transformationsmatrix angeben. Transformationsmatrizen sind nützlich, wenn Sie gleichzeitig Skalierung und Rotation anwenden oder ein Objekt spiegeln wollen. Sie übergeben eine Tabelle, welche die vier Bestandteile einer 2x2 Transformationsmatrix in der folgenden Reihenfolge enthält: **sx**, **rx**, **ry**, **sy**. Siehe [Abschnitt 43.78 \[TransformBrush\]](#), [Seite 980](#), für weitere Informationen über Transformationsmatrizen. Bitte beachten Sie, dass der Tag **Transform** und folgende sich gegenseitig ausschließen: **Width/Height/ScaleX/ScaleY/Rotate**. (V4.5)

**AnchorX, AnchorY:**

Sie können diese beiden Tags verwenden, um den Ankerpunkt des Grafikobjekts anzugeben (manchmal wird der Ankerpunkt auch "Hot Spot" genannt). Der Ankerpunkt kann jeder Punkt zwischen 0.0/0.0 (oben links des Grafikobjekts) und 1.0/1.0 (untere rechte Ecke des Grafikobjekts) sein. Das Zentrum des Grafikobjekts würde durch Ankerpunkt 0.5/0.5 definiert werden. Alle Transformationen (Skalierung, Drehung usw.) werden relativ zum Ankerpunkt angewendet. Auch ist die Position eines Objekts immer relativ zu dem Ankerpunkt. Werfen wir einen Blick auf den folgenden Code:

```
DisplayBrush(1, 0, 0, {AnchorX = 0.5, AnchorY = 0.5})
```

Mit dieser Befehlszeile wird der Pinsel im Zentrum von 0:0 erscheinen, weil der Ankerpunkt bei 0.5/0.5 eingestellt ist. Wenn die linke obere Ecke des Pinsels bei 0:0 angezeigt werden soll, müssen Sie einen Ankerpunkt von 0.0/0.0 verwenden (was auch der Standardankerpunkt ist). Ankerpunkte werden meistens nur dann verwendet, wenn **Ebenen** aktiviert sind. Sie können jedoch auch **AnchorX/Y** bei ausgeschalteten Ebenen verwenden. Siehe [Abschnitt 25.40 \[SetLayerAnchor\]](#), [Seite 438](#), für weitere Informationen über das Ankerpunktekonzept. (V4.5)

**Hidden:**

Hier können Sie eine Ebene erstellen, die zunächst verborgen bleibt. Wenn Sie diesen Tag auf **True** setzen, werden die Grafiken als Ebene eingefügt werden. Allerdings werden sie nicht angezeigt, da die Ebene ausgeblendet ist. Diese Funktionalität läuft nur mit **aktivierten Ebenen**. Der Standardwert ist **False**. (V4.5)

**InsertPos:**

Hier können Sie die Einfügeposition für die neue Ebene angeben. Die Ebene wird an dieser Stelle eingefügt und schiebt alle anderen Ebenen in der Hierarchie nach unten. Die erste Ebene ist an Position 1. Geben Sie hier eine Position von 0 an, wird die Ebene als letzte eingefügt. Dies ist auch die Standardeinstellung.

Sie können auch den Namen einer Ebene angeben, an dessen Position die neue Ebene eingefügt werden soll. Natürlich funktioniert diese Funktionalität nur mit **aktivierten Ebenen**. (V4.5)

**Name:** Dieser Tag kann verwendet werden, um für die Ebene zum Zeitpunkt der Erstellung einen Namen zuzuweisen. Das ist so ziemlich das gleiche, wie wenn direkt nach dem Erstellen einer Ebene der Befehl **SetLayerName()** angewandt wird. Mit diesem Tag ersparen Sie etwas Tipparbeit und macht den Code besser lesbar. Dieser Tag wird nur berücksichtigt, wenn **Ebenen** aktiviert sind. Standardmäßig werden keine Namen den Ebenen, sondern nur IDs zugewiesen. (V4.5)

**Group:** Wenn Ebenen aktiviert sind, können Sie diesen Tag verwenden, um die Ebene direkt zum Zeitpunkt der Erstellung an eine Gruppe anzuhängen. Dies ist so ziemlich dasselbe wie der Aufruf des Befehls **GroupLayer()** für die Ebene direkt nach dem Erstellen. Das Angeben des Tags **Group** hier erspart Ihnen nur etwas Tipparbeit und macht den Code besser lesbar. Dieser Tag wird nur verarbeitet, wenn **Ebenen** aktiviert sind. Siehe **Abschnitt 25.16 [GroupLayer]**, **Seite 418**, für Details. (V10.0)

**Transparency:**

Sie können mit diesem Tag eine globale Transparenz für dieses Grafikobjekt einstellen. Dies kann einen Wert im Bereich von 0 (keine Transparenz) bis 255 (volle Transparenz) sein oder alternativ eine Zeichenfolge, die einen Prozentsatz (zum Beispiel "50%" für die Hälfte der Transparenz) enthält. Der Standardwert ist 0. (V4.5)

**GlobalTransparency:**

Dies wird nur unterstützt, wenn Ebenen aktiviert sind. Wenn er auf **True** gesetzt ist, wird die Transparenzeinstellung der Ebene, die mit dem Tag **Transparency** oder mit Befehlen wie **SetLayerTransparency()** festgelegt wurde, auch auf den Schatten der Ebene angewendet. Aus irgendeinem Grund hat Hollywood dies nie standardmäßig getan und um die Kompatibilität mit Skripten aufrechtzuerhalten, die das alte Verhalten erwarten, wurde die Funktionalität mit einem neuen Tag hinzugefügt. Wenn Sie das neue Verhalten global für alle Ihre Ebenen aktivieren möchten, verwenden Sie einfach den Befehl **SetDrawTagsDefault()**, wobei **GlobalTransparency** auf **True** gesetzt ist. (V9.1)

**Tint:** Mit diesem Tag können Sie eine globale Farbtönung für dieses Grafikobjekt angeben. Dies kann einen Wert von 0 (keine Tönung) bis hin zu 255 (Vollfarbtönung) oder alternativ eine Zeichenfolge sein, die einen Prozentsatz (zum Beispiel "50%" für mittlere Farbtönung) enthält. Wenn Sie etwas anderes als 0 gesetzt haben, müssen Sie auch eine Farbe im Tag **TintColor** einsetzen (siehe unten). Der Standardwert ist 0. Ab Hollywood 5.0 ist dieser Tag mit dem Filter **Tint** direkt verbunden. Somit gibt der Tag **Tint** das gleiche wie ein **Tint**-Filter in der unten angegebenen **Filters**-Tabelle an. (V4.5)

**TintColor:**

Nur erforderlich, wenn Sie auch den Tag **Tint** angegeben haben. In diesem Fall müssen Sie hier eine RGB-Farbe angeben, die für die Tönung verwendet wird. (V4.5)



**Shadow:** Wenn Sie diesen Tag auf **True** setzen, wird das Grafikobjekt mit einem Schatten gezeichnet werden. Sie können das Aussehen des Schattens mit den Tags **ShadowDir**, **ShadowSize**, **Shadow** und **ShadowRadius** konfigurieren. Siehe unten für weitere Informationen. Dieser Tag wird nur unterstützt, wenn **Ebenen** eingeschaltet sind. Siehe [Abschnitt 25.47 \[SetLayerShadow\]](#), [Seite 448](#), für Details. (V5.0)

**ShadowDir:** Gibt die Richtung des Schattens an. Dies muss mit einer **Richtungskonstanten** von Hollywood eingestellt werden. Dieser Tag wird nur berücksichtigt, wenn **Shadow** auf **True** gesetzt wurde (siehe oben). (V5.0)

**ShadowColor:** Stellt die Farbe des Schattens ein. Dies muss ein **ARGB-Wert** sein und kann eine transparente Einstellung haben. Auch dieser Tag wird nur gehandhabt, wenn **Shadow** auf **True** gesetzt wurde (siehe oben). (V5.0)

**ShadowPen:** Wenn der Palettenmodus auf **#PALETTEMODE\_PEN** eingestellt ist und das Zeichnungsziel eine Palette verwendet, wird der Schatten mit dem hier angegebenen Stift gezeichnet, anstatt mit der Farbe, die im Tag **ShadowColor** von oben angegeben wurde. (V9.0)

**ShadowSize:** Mit diesem Tag wird die Größe bzw. der Abstand des Schattens in Pixel angegeben. Wenn der Tag **Shadow** auf **True** gesetzt wurde, wird dieser Tag berücksichtigt, ansonsten wird er ignoriert (siehe oben). (V5.0)

**ShadowRadius:** Definiert den Radius des Schattens. Wird nur berücksichtigt, wenn für **Shadow** **True** gesetzt wurde (siehe oben). (V5.0)

**Border:** Dieser Tag kann verwendet werden, um die Grafik mit einem Rahmen zu zeichnen. Sie können das Aussehen des Rahmens mit den Tags **BorderSize** und **BorderColor** konfigurieren. Bitte siehe unten für weitere Informationen. Dieser Tag wird nur unterstützt, wenn **Ebenen** eingeschaltet sind. Siehe [Abschnitt 25.41 \[SetLayerBorder\]](#), [Seite 439](#), für Details. (V5.0)

**BorderColor:** Dieser Tag gibt die Farbe des Rahmens an. Dies muss ein **ARGB-Wert** sein, der eine Transparenzeinstellung enthält. Dieser Tag funktioniert nur dann, wenn der Tag **Border** auf **True** gesetzt wurde (siehe oben). (V5.0)

**BorderPen:** Wenn der Palettenmodus auf **#PALETTEMODE\_PEN** eingestellt ist und das Zeichnungsziel eine Palette verwendet, wird der Rahmen mit dem hier angegebenen Stift gezeichnet, anstatt mit der Farbe, die oben im Tag **BorderColor** angegeben wurde. (V9.0)

**BorderSize:** Damit lässt sich die Rahmendicke einstellen. Läuft nur, wenn **Border** auf **True** gesetzt wurde (siehe oben). (V5.0)



**Filters:** Dieser Tag kann verwendet werden, um einen Filter auf diesem Grafikobjekt anzuwenden. Sie müssen diesem Tag eine Tabelle übergeben, die die gewünschte Konfigurationen der einzelnen Filter beschreibt. Siehe [Abschnitt 25.43 \[SetLayerFilter\]](#), [Seite 441](#), für weitere Informationen wie diese Tabelle aussehen muss.. Beachten Sie, dass obwohl die Dokumentation auf die Ebenen-Bibliothek verweist, der Tag **Filters** tatsächlich auch dann funktioniert, wenn die Ebenen deaktiviert sind. (V5.0)



## 30 Hintergrundbildbefehle

### 30.1 Überblick

Hintergrundbilder (BGPics) sind in Hollywood sehr wichtig, denn jedes Display muss ein ihm zugewiesenes Hintergrundbild haben. Wenn Ihr Display sichtbar wird, zeigt es am Anfang dem Benutzer das Hintergrundbild. Dieses Hintergrundbild wird dann als "Arbeitsbereich" Ihres Skripts fungieren, d.h. der Bereich, den Sie für das Zeichnen von Grafiken verwenden können. Dieser Bereich steht voll zu Ihrer Verfügung und Sie können ihn gestalten, wie Sie wollen.

Das Hintergrundbild muss immer gleich groß wie das Display sein. Wenn Sie also die Displaygröße z.B. mit `ChangeDisplaySize()` ändern, wird Ihr Hintergrundbild automatisch skaliert und an die neuen Dimensionen angepasst. Wenn Ihr Fenster veränderbar ist, kann der Benutzer auch die Displaygröße anpassen. In diesem Fall wird Hollywood intern `ChangeDisplaySize()` aufrufen und die neuen Dimensionen für das Hintergrundbild übernehmen.

Wenn Sie ein neues Hintergrundbild z.B. mit dem Befehl `DisplayBGPic()` anzeigen möchten und die Abmessungen des neuen Hintergrundbildes von den Abmessungen Ihres aktuellen abweichen, wird das Display so angepasst, dass es den neuen Dimensionen des Hintergrundbildes entspricht.

Beim starten des Programmes zeigt Hollywood das Hintergrundbild an, dem die ID 1 zugewiesen wurde. Wenn Sie mit der Präprozessor-Anweisung `@BGPIC` kein Hintergrundbild mit dem Identifikator 1 deklariert haben, wird Hollywood automatisch dieses Hintergrundbild für Sie erstellen und Ihrem Display zuweisen. Das Hintergrundbild verwendet den selben Füllstil und die gleiche Abmessungen, welche Sie mit der Präprozessor-Anweisung `@DISPLAY` für das Display 1 in Ihrem Skript angegeben haben.

Hier ist ein minimales Skript, welches ein Display mit der Bilddatei `test.jpg` zeigt:

```
@BGPIC 1, "test.jpg"
WaitLeftMouse
End
```

### 30.2 BGPIC

#### BEZEICHNUNG

BGPIC – lädt ein Hintergrundbild für den späteren Gebrauch vor (V2.0)

#### ÜBERSICHT

```
@BGPIC id, filename$[, table]
```

#### BESCHREIBUNG

Diese Präprozessor-Anweisung lädt das in `filename$` angegeben Hintergrundbild unter der Kennung `id`. Wenn Sie 1 als `id` benutzen, dann wird dieses Bild als Anfangshintergrundbild verwendet werden, wenn Hollywood Ihr Display zeigt.

Bildformate, die auf allen Plattformen unterstützt werden, sind PNG, JPEG, BMP, IFF ILBM, GIF und Plugins für Bilder. Je nach Plattform, worauf Hollywood ausgeführt wird, können mehr Bildformate unterstützt werden. Zum Beispiel auf den Amigakompatiblen Systemen wird Hollywood in der Lage sein, alle Formate über `Bilddatatypes`

zu öffnen. Unter Windows kann `@BGPIC` auch Bildformate laden, die von der Windows-Imaging-Komponente unterstützt werden.

Ab Hollywood 5.0 kann dieser Befehl auch Vektorformate wie SVG laden, wenn Sie eine entsprechendes Plugin installieren. Die Verwendung eines Vektorbildes als BGPic hat den Vorteil, dass wenn sich die Größe des Displays ändert (z.B. weil der Benutzer das Fenster in der Größe ändert) kann das BGPic ohne Einbußen in der Qualität an die neue Größe angepasst werden, da Vektor-BGPics ohne Qualitätsverluste stufenlos skaliert werden können. Siehe [Abschnitt 30.16 \[Vektor-BGPics\]](#), [Seite 642](#), für weitere Informationen zu Vektor-BGPics.

Das dritte Argument ist optional. Es ist eine Tabelle, die für weitere Möglichkeiten des Ladevorgangs verwendet werden kann. Die folgenden Tags stehen dafür zur Verfügung:

**Transparency:**

Dieser Tag kann verwendet werden, um eine Farbe in **RGB**-Notation, anzugeben, die im BGPic transparent erscheinen soll.

**LoadAlpha:**

Setzen Sie diesen Tag auf **True**, wenn der Alpha-Kanal des Bildes auch geladen werden soll. Bitte beachten Sie, dass nicht alle Bilder einen Alpha-Kanal haben und dass nicht alle Bildformate in der Lage sind, Alphakanaldaten zu speichern. Es wird vorgeschlagen, dass Sie das PNG-Format verwenden, wenn Sie Alphakanaldaten benötigen. Dieser Tag ist standardmäßig **False**. (V4.5)

**Link:** Setzen Sie diesen Tag auf **False**, wenn Sie dieses BGPic nicht in die ausführbare Datei/das Applet einbinden wollen, wenn Sie Ihr Skript kompilieren. Dieser Tag ist standardmäßig auf **True** gesetzt, was bedeutet, dass das BGPic mit der ausführbaren Datei/dem Applet beim Kompilieren verknüpft wird.

**FillStyle:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), [Seite 634](#), für Details. (V5.0)

**FillColor:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), [Seite 634](#), für Details. (V5.0)

**TextureBrush:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), [Seite 634](#), für Details. (V5.0)

**TextureX, TextureY:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), [Seite 634](#), für Details. (V5.0)

**GradientStyle:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), [Seite 634](#), für Details. (V5.0)

**GradientAngle:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), [Seite 634](#), für Details. (V5.0)

**GradientStartColor, GradientEndColor:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), [Seite 634](#), für Details. (V5.0)

**GradientCenterX, GradientCenterY:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), [Seite 634](#), für Details. (V5.0)

**GradientBalance:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), Seite 634, für Details. (V5.0)

**GradientBorder:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), Seite 634, für Details. (V5.0)

**GradientColors:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), Seite 634, für Details. (V5.0)

**ScaleWidth, ScaleHeight:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), Seite 634, für Details. (V5.3)

**SmoothScale:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), Seite 634, für Details. (V5.3)

**Loader:** Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), Seite 634, für Details. (V6.0)

**Adapter:** Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), Seite 634, für Details. (V6.0)

**LoadTransparency:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), Seite 634, für Details. (V6.0)

**LoadPalette:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), Seite 634, für Details. (V9.0)

**FillPen:** Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), Seite 634, für Details. (V9.0)

**TransparentPen:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), Seite 634, für Details. (V9.0)

**UserTags:**

Siehe [Abschnitt 30.13 \[LoadBGPic\]](#), Seite 634, für Details. (V10.0)

Wenn Sie **Transparency**, **LoadTransparency** oder **LoadAlpha** verwenden, wird Ihr Display automatisch die Transparenzeinstellungen des BGPic annehmen, wenn es angezeigt wird. Mit anderen Worten, wenn Sie ein Display mit Transparenz haben möchten, laden Sie einfach ein transparentes BGPic und zeigen es an.

Wenn Sie den Tag **LoadPalette** auf **True** setzen, wird Ihr Display zu einem Palettenmodus-Display, sobald das BGPic angezeigt wird. Palettenmodus-Displays verhalten sich anders als normale True-Color-Displays und bei ihrer Verwendung sind einige Dinge zu beachten. Siehe [Abschnitt 24.16 \[Palettenmodus-Displays\]](#), Seite 393, für Details.

Bitte beachten Sie, dass die Tags **Transparency**, **LoadTransparency** und **LoadAlpha** sich gegenseitig ausschließen. Ein BGPic kann nur eine Transparenzeinstellung haben!

Wenn Sie manuell Hintergrundbilder laden möchten, verwenden Sie den Befehl **LoadBGPic()**.

## EINGABEN

<b>id</b>	ein Wert, der dieses Hintergrundbild später im Code identifiziert; wenn <b>id</b> 1 ist, dann wird das Bild als Anfangshintergrundbild verwendet werden
<b>filename\$</b>	Bilddatei, die Sie laden möchten
<b>table</b>	optional: eine Tabelle mit weiteren Einstellungen; siehe Erklärungen oben

**BEISPIEL**

```
@BGPIC 1, "MyBG.png"
```

Deklariert "MyBG.png" als Anfangshintergrundbild (wird angezeigt, wenn Hollywood startet).

```
@BGPIC 1, "MyBG.png", {Transparency = $FF0000}
```

Ist das gleiche wie oben, aber das Bild ist jetzt transparent (Transparenzfarbe ist Rot = \$FF0000).

```
@BGPIC 1, "4MB_uncompressed_picture.bmp", {Link = False}
```

Dieser Code lädt das angegebene Bild und sagt Hollywood, dass es beim Kompilieren nie dieses Bild verknüpfen soll, weil es so groß ist.

### 30.3 BrushToBGPic

**BEZEICHNUNG**

BrushToBGPic – kopiert einen Pinsel in ein Hintergrundbild

**ÜBERSICHT**

```
BrushToBGPic(bbrushid, bgpicid)
```

**BESCHREIBUNG**

Dieser Befehl erstellt eine Kopie des in **bbrushid** angegebenen Pinsels und konvertiert ihn in ein Hintergrundbild, auf das dann mit der ID **bgpicid** zugegriffen werden kann. Alles wird geklont, deshalb ist das Hintergrundbild unabhängig vom originalen Pinsel (Sie könnten ihn zum Beispiel nach dieser Operation aus dem Speicher löschen und das Hintergrundbild bleibt trotzdem benutzbar!).

**EINGABEN**

**bbrushid**    Pinsel, der geklont werden soll

**bgpicid**    Kennung für das neue Hintergrundbild

**BEISPIEL**

```
BrushToBGPic(1,2)
```

```
DisplayBGPic(2)
```

Dieses Beispiel kopiert Pinsel 1 als Hintergrundbild 2 und zeigt dieses dann an.

### 30.4 CopyBGPic

**BEZEICHNUNG**

CopyBGPic – kopiert ein Hintergrundbild (V4.0)

**ÜBERSICHT**

```
[id] = CopyBGPic(source, dest)
```

**BESCHREIBUNG**

Dieser Befehl kloniert das Hintergrundbild **source** und erstellt eine Kopie davon als Hintergrundbild mit der ID **dest**. Wenn Sie **Nil** im Argument **dest** angeben, wird eine ID für dieses Hintergrundbild automatisch ausgewählt und in **id** abgelegt. Das neue Hintergrundbild ist unabhängig vom alten, so dass Sie das Quellhintergrundbild löschen könnten, nachdem es geklonert wurde.

Bitte beachten Sie, dass nur die einfachen Grafikdaten des Hintergrundbildes geklonert werden. `CopyBGPic()` kloniert nicht alle Ebenen, Sprites oder Schaltflächen, die dem Hintergrundbild zugewiesen wurden.

**EINGABEN**

**source** ID des Quellhintergrundbildes  
**dest** ID des neuen BGPic oder **Nil** für die automatische ID-Zuweisung

**RÜCKGABEWERTE**

**id** optional: Identifikator des BGPic in **dest**; Wird nur zurückgegeben werden, wenn Sie **Nil** als Argument 1 angegeben haben (siehe oben)

**BEISPIEL**

```
CopyBGPic(1, 10)
FreeBGPic(1)
```

Der obige Code erstellt ein neues Hintergrundbild 10, das die gleichen Grafikdaten wie Hintergrundbild 1 enthält. Dann wird Hintergrundbild 1 aus dem Speicher gelöscht, da es nicht mehr benötigt wird.

## 30.5 CreateBGPic

**BEZEICHNUNG**

CreateBGPic – erzeugt ein leeres Hintergrundbild (V1.5)

**ÜBERSICHT**

```
[id] = CreateBGPic(id, width, height[[, color], table])
```

**BESCHREIBUNG**

Dieser Befehl erzeugt ein neues Hintergrundbild mit der angegebenen Höhe **height** und Breite **width** und füllt es mit der in **color** angegebenen Farbe. Wenn keine Farbe übergeben wurde, wird das Hintergrundbild schwarz gefüllt. Wenn Sie beim Argument **id** **Nil** angeben, wird `CreateBGPic()` für dieses Hintergrundbild automatisch eine Zahl wählen und Ihnen übergeben.

Ab Hollywood 9.0 gibt es ein optionales Tabellenargument, mit dem Sie weitere Optionen angeben können. Die folgenden Tabellen-Tags werden derzeit unterstützt:

**Palette:** Wenn dieser Tag auf den Identifikator einer Palette gesetzt ist, erstellt Hollywood ein Palettenhintergrundbild für Sie. Paletten können mit Befehlen wie `CreatePalette()` oder `LoadPalette()` erstellt werden. Alternativ können Sie diesen Tag auch auf eine von Hollywoods integrierten Paletten setzen, z.B. `#PALETTE_AGA`. Siehe [Abschnitt 41.36 \[SetStandardPalette\]](#), [Seite 872](#), für eine Liste der integrierten Paletten.

**FillPen:** Wenn der Tag **Palette** gesetzt ist (siehe oben), können Sie mit diesem Tag den Stift einstellen, mit dem der Hintergrund des Hintergrundbildes ausgefüllt werden soll. Beachten Sie, dass der Parameter **color**, der an **CreateBGPic()** übergeben wird, ignoriert wird, wenn **Palette True** ist. Aus diesem Grund können Sie mit diesem Tag einen Stift angeben, der beim Initialisieren der Pixel des Hintergrundbildes verwendet wird. Voreingestellt ist 0.

**TransparentPen:**

Wenn **Palette** auf **True** gesetzt ist, kann mit diesem Tag ein Stift angegeben werden, der im neuen Hintergrundbild transparent gemacht werden soll. Der Standardwert ist **#NOPEN**, was bedeutet, dass kein transparenter Stift vorhanden sein soll.

## EINGABEN

<b>id</b>	Identifikator des neuen Hintergrundbildes
<b>width</b>	Breite des Hintergrundbildes
<b>height</b>	Höhe des Hintergrundbildes
<b>color</b>	optional: <b>RGB-Farbe</b> für die Füllung (voreingestellt ist <b>#BLACK</b> )
<b>table</b>	optional: Tabelle mit weiteren Optionen (siehe oben) (V9.0)

## RÜCKGABEWERTE

<b>id</b>	optional: Identifikator des BGPic; wird nur zurück gegeben, wenn Sie als erstes Argument <b>Nil</b> angegeben haben.
-----------	--

## BEISPIEL

```
CreateBGPic(2, 640, 480, #RED)
```

Diese Zeile erzeugt ein neues rotes Hintergrundbild mit der ID 2 und den Abmessungen 640x480.

# 30.6 CreateGradientBGPic

## BEZEICHNUNG

**CreateGradientBGPic** – erstellt ein neues Hintergrundbild mit einem Farbverlauf (V2.0)

## FRÜHERER NAME

**CreateRainbowBGPic** (V1.0 - V1.9)

## ÜBERSICHT

```
[id] = CreateGradientBGPic(id, type, startcolor, endcolor[, width,
                           height, angle, table])
```

## BESCHREIBUNG

Dieser Befehl kann verwendet werden, um ein neues Hintergrundbild mit einem Farbverlauf zu erstellen. Wenn Sie **Nil** als **id** angeben, wird automatisch eine ID für dieses Hintergrundbild ausgewählt und in **id** zurückgegeben. **type** gibt die Art des Farbverlaufs an, den Sie verwenden möchten. Die folgenden Verlaufstypen sind derzeit verfügbar:



**#LINEAR**, **#RADIAL** und **#CONICAL**. Wenn **width** (Breite) und **height** (Höhe) nicht angegeben werden, ist die Abmessungen dieselbe wie die Dimensionen des aktuellen Displays. **angle** (Winkel) ermöglicht es Ihnen, einen Drehwinkel in Grad für den Verlauf anzugeben. Dieses Winkelargument wird nur durch die Verlaufstypen **#LINEAR** und **#CONICAL** unterstützt. Radialfarbverläufe können nicht gedreht werden.

Das optionale Tabellenargument **table** kann für weitere Optionen verwendet werden. Die folgenden Tags sind derzeit verfügbar:

**CenterX, CenterY:**

Diese beiden Tags können verwendet werden, um den Mittelpunkt des Farbverlaufs anzugeben. Lineare Verläufe haben keinen Mittelpunkt, darum werden diese beiden Variablen nur dann berücksichtigt, wenn Sie Farbverläufe vom Typ **#RADIAL** oder **#CONICAL** verwenden. Der Mittelpunkt muss als Fließkommawert angegeben werden, der zwischen 0.0 (linke/obere Ecke) und 1.0 (rechts/unten Ecke) ist. Wenn nichts angegeben wurde, wird als Standard 0.5 genommen, so dass der Mittelpunkt des Verlaufes in der Mitte des Bildes ist. (V5.0)

**Border:** Dieser Tag kann verwendet werden, um die Rahmengröße für den Verlauf des Typs **#RADIAL** einzustellen. Für die anderen Verlaufstypen wird dieser Tag ignoriert. Die Rahmengröße des radialen Verlaufes muss ein Fließkommawert zwischen 0.0 und 1.0 sein. Voreingestellt ist 0.0, das keinen Rahmen bedeutet. (V5.0)

**Balance:** Dieser Tag kann verwendet werden, um den Gleichgewichtspunkt für den Verlauf des Typs **#CONICAL** einzustellen. Für die anderen Verlaufstypen wird dieser Tag ignoriert. Der Gleichgewichtspunkt des konischen Verlaufes muss ein Fließkommawert zwischen 0.0 und 1.0 sein. Der Standardwert ist 0.5. Beachten Sie, dass dies nur beim Erstellen eines zweifarbigen Farbverlaufs verwendet wird. Wenn Sie einen mehrfarbigen Farbverlauf mit der Tabelle **Colors** erstellen, wird **Balance** ignoriert, da Sie mit der Tabelle **Colors** die Farben im Farbverlauf mithilfe von Farbstopps individuell ausgleichen können. (V5.0)

**Colors:** Mit diesem Tag können Sie Verläufe erstellen, die mehrere Farben enthalten. Dieser Tag ist eine Tabelle, die eine Folge von abwechselnden Farben und Stoppwerte enthält. Die Farben müssen im **RGB-Format** angegeben werden. Der Stoppwert ist ein Fließkommawert zwischen 0.0 und 1.0 und definiert die Position, wo die entsprechende Farben in dem Verlauf verschmolzen werden. Eine Position von 0.0 bedeutet die Startposition des Verlaufes und eine Position von 1.0 ist die Endposition. Bitte beachten Sie, dass die Stoppositionen in aufsteigender Reihenfolge sortiert werden müssen, das heißt von 0.0 bis 1.0. Wenn Sie diesen Tag angeben, werden die Farben in den Argumenten **startcolor** und **endcolor** ignoriert und Hollywood wird nur die in diesem Tag angegebenen Farben verwenden. (V5.0)

## EINGABEN

<b>id</b>	ID des neuen BGPic oder <b>Nil</b> für die <b>automatische ID-Zuweisung</b>
<b>type</b>	Typ des Farbverlaufs; siehe oben für die verschiedenen Typen

**startcolor**      **RGB-Wert** für die Startfarbe

**endcolor**      **RGB-Wert** als Endfarbe

**width**          optional: gewünschte Breite für das Hintergrundbild (Standard: aktuelle Displaybreite)

**height**        optional: gewünschte Höhe für das Hintergrundbild (Standard: aktuelle Displayhöhe)

**angle**          optional: Drehwinkel für den Farbverlauf (Voreingestellt ist 0)

**table**          optional: Tabelle, welche weitere Optionen ermöglicht; siehe oben für eine Beschreibung der verfügbaren Optionen

### RÜCKGABEWERTE

**id**              optional: Identifikator des BGPic; Wird nur zurückgegeben werden, wenn Sie **Nil** als Argument 1 angegeben haben (siehe oben)

### BEISPIEL

```
CreateGradientBGPic(2, #LINEAR, #BLACK, #BLUE)
DisplayBGPic(2)
```

Erstellt einen von oben nach unten gehender Verlauf, welcher als Hintergrundbild 2 mit der Farbe Schwarz verblassend ins blaue zeigt.

```
CreateGradientBGPic(2, #LINEAR, 0, 0, 640, 480, 0, {Colors = {#RED, 0,
    #BLUE, 0.25, #GREEN, 0.5, #YELLOW, 0.75, #BLACK, 1}})
DisplayBGPic(2)
```

Der Code erstellt einen Verlauf mit mehreren Farben. Dieser Verlauf versucht, das Aussehen der berühmten Amiga Kupferstäbe zu replizieren.

## 30.7 CreateTexturedBGPic

### BEZEICHNUNG

CreateTexturedBGPic – erzeugt ein neues Hintergrundbild, das mit einem Pinsel texturiert ist

### ÜBERSICHT

```
[id] = CreateTexturedBGPic(id, brushid[, width, height, x, y])
```

### BESCHREIBUNG

Dieser Befehl erzeugt Ihnen ein neues Hintergrundbild und überzieht es mit einer Textur, die aus dem durch **brushid** angegebenen Pinsel gestaltet wird. Wenn Sie **Nil** als **id** angeben, wird automatisch eine ID für dieses Hintergrundbild ausgewählt und in **id** zurückgegeben. Werden **width** und **height** weggelassen, werden die Abmessungen des aktuellen Displays verwendet.

Die optionalen Parameter **x** und **y** sind neu in Hollywood 4.6. Sie ermöglichen es, einen Versatz des Texturpinsels anzugeben. Texturierung wird dann im Pinsel versetzt beginnen. Der Standardwert für dieses Argument ist 0/0 und bedeutet, dass es in der oberen linken Ecke im Texturpinsel beginnt.

**EINGABEN**

<b>id</b>	ID des neuen BGPic oder <b>Nil</b> für die automatische ID-Zuweisung
<b>brushid</b>	ID des Pinsels, welcher für die Textur benutzt wird
<b>width</b>	optional: gewünschte Breite für das Hintergrundbild (Standard: aktuelle Displaybreite)
<b>height</b>	optional: gewünschte Höhe für das Hintergrundbild (Standard: aktuelle Displayhöhe)
<b>x</b>	optional: Startpunkt x im Texturpinsel (V4.6)
<b>y</b>	optional: Startpunkt y im Texturpinsel (V4.6)

**RÜCKGABEWERTE**

<b>id</b>	optional: Identifikator des BGPic; wird nur zurückgegeben werden, wenn Sie <b>Nil</b> als Argument 1 angegeben haben (siehe oben)
-----------	---

**BEISPIEL**

```
CreateTexturedBGPic(2,1)
DisplayBGPic(2)
```

Erzeugt ein Hintergrundbild, das mit Pinsel 1 texturiert wurde und zeigt es an.

## 30.8 DisplayBGPic

**BEZEICHNUNG**

DisplayBGPic – zeigt ein neues Hintergrundbild an

**ÜBERSICHT**

```
DisplayBGPic(id[, args])
```

**BESCHREIBUNG**

Dieser Befehl ändert das Hintergrundbild auf das durch **id** angegebene. Weichen die Abmessungen dieses Bildes vom aktuellen ab, wird die Displaygröße entsprechend angepasst.

Neu in Hollywood 4.0: Sie können eine Tabelle im optionalen zweiten Argument **args** übergeben, um weitere Möglichkeiten anzugeben. Derzeit kann die Tabelle folgende Tags enthalten:

<b>X:</b>	Gibt die neue Position X für das Display an. Wenn das Display seine aktuelle X-Position beibehalten soll, geben Sie die spezielle Konstante <b>#KEEPPPOSITION</b> an. Der Standardwert ist <b>#CENTER</b> .
<b>Y:</b>	Gibt die neue Position Y für das Display an. Wenn das Display seine aktuelle Y-Position beibehalten soll, geben Sie die spezielle Konstante <b>#KEEPPPOSITION</b> an. Der Standardwert ist <b>#CENTER</b> .

**EINGABEN**

<b>id</b>	ID des neuen Hintergrundbildes
<b>args</b>	optional: weitere Konfigurationsoptionen (V4.0)

**BEISPIEL**

```
DisplayBGPic(2)
```

Stellt Hintergrundbild 2 dar und passt die Fenstergröße falls nötig an.

```
DisplayBGPic(2, {X = #RIGHT, Y = #BOTTOM})
```

Zeigt das Hintergrundbild 2 in der unteren rechten Ecke des aktuellen Desktop.

**30.9 DisplayBGPicPart****BEZEICHNUNG**

DisplayBGPicPart – stellt einen Teil eines Hintergrundbildes dar

**ÜBERSICHT**

```
DisplayBGPicPart(id, x, y, width, height[, dx, dy, table])
```

**BESCHREIBUNG**

Dieser Befehl stellt einen Ausschnitt des durch `id` angegebenen Hintergrundbildes auf dem Bildschirm dar. Der Ausschnitt wird durch `x`, `y` und seine Breite `width` sowie Höhe `height` definiert.

Wenn Ebenen aktiviert sind, wird dieser Befehl eine neue Ebene des Typs `#BGPICPART` dem Ebenenstapel hinzuzufügen.

Ab Hollywood 4.5 verwendet dieser Befehl eine neue Syntax. Die alte Syntax wird aus Kompatibilitätsgründen weiterhin unterstützt. Bei neuen Skripten sollten Sie die neue Syntax verwenden. Sie akzeptiert eine Tabelle als letztes Argument, wo Sie einige weitere Optionen angeben können:

**Layers:** Wenn Sie dies auf `True` setzen, werden die Ebenen (wenn sie aktiviert sind) oder die Vordergrundgrafiken (wenn Ebenen deaktiviert sind) des Hintergrundbildes auch gezeichnet. Dies ist z.B. nützlich, wenn Sie eine exakte Kopie eines Hintergrundbildes in einem Pinsel erstellen. Bitte beachten Sie, dass wenn Ebenen deaktiviert sind, können Sie dieses Argument nur verwenden, wenn `id` die ID des aktuellen Hintergrundbildes ist, weil Hollywood nicht den gesamten Inhalt des Vordergrund aller Hintergrundbilder halten kann, wenn Ebenen deaktiviert sind.

Darüber hinaus kann die optionale Tabelle `table` auch eine oder mehrere der **Standard-Tags für alle Zeichnungsbefehle** enthalten. Siehe **Abschnitt 29.17 [Standard-Tags zum Zeichnen]**, **Seite 613**, für weitere Informationen über die Standard-Tags, die fast alle Zeichnungsbefehle von Hollywood unterstützen.

**EINGABEN**

<code>id</code>	ID des Quellhintergrundbildes
<code>x</code>	linke Ecke
<code>y</code>	rechte Ecke
<code>width</code>	Breite des Ausschnittes
<code>height</code>	Höhe des Ausschnittes

<code>dx</code>	optional: Zielposition des Ausschnitts in X-Richtung (voreingestellt ist x) (V1.5)
<code>dy</code>	optional: Zielposition des Ausschnitts in Y-Richtung (voreingestellt ist y) (V1.5)
<code>table</code>	optional: weitere Konfigurationsmöglichkeiten

**BEISPIEL**

```
DisplayBGPicPart(2,0,0,100,100)
```

Stellt die ersten 100 Zeilen und Spalten des Hintergrundbildes 2 auf dem Bildschirm an Position 0:0 dar.

```
width = GetAttribute(#DISPLAY, 0, #ATTRWIDTH)
height = GetAttribute(#DISPLAY, 0, #ATTRHEIGHT)
id = GetAttribute(#DISPLAY, 0, #ATTRBGPIC)
CreateBrush(1, width, height)
SelectBrush(1)
DisplayBGPicPart(id, 0, 0, width, height, 0, 0, {Layers = TRUE})
EndSelect
```

Dieser Code erstellt eine Kopie des aktuellen Displayinhalts als Pinsel 1.

## 30.10 DisplayBGPicPartFX

**BEZEICHNUNG**

DisplayBGPicPartFX – stellt einen Teil eines Hintergrundbildes mit einem Effekt dar

**ÜBERSICHT**

```
[handle] = DisplayBGPicPartFX(id, x, y, width, height[, table])
```

**BESCHREIBUNG**

Dies ist eine erweiterte Version des Befehls `DisplayBGPicPart()`. Er tut das gleiche, allerdings mit einem zusätzlichen Übergangseffekt.

Wenn Ebenen aktiviert sind, wird dieser Befehl eine neue Ebene des Typs `#BGPICPART` dem Ebenenstapel hinzuzufügen.

Ab Hollywood 4.5 verwendet dieser Befehl eine neue Syntax. Die alte Syntax wird aus Kompatibilitätsgründen weiterhin unterstützt. Das optionale Tabellenargument kann den Übergangseffekt konfigurieren. Folgende Optionen sind möglich:

**Type:** Hier geben Sie den gewünschten Übergangseffekt an. Siehe [Abschnitt 30.11 \[DisplayTransitionFX\]](#), [Seite 630](#), für eine Liste aller unterstützten Übergangseffekte. (Voreingestellt ist `#RANOMEFFECT`)

**Speed:** Legt die gewünschte Geschwindigkeit für den Übergang fest. Je höher der Wert, den Sie hier angeben, desto schneller wird der Effekt angezeigt werden. (Standardeinstellung ist `#NORMALSPEED`)

**Parameter:** Einige Übergangseffekte akzeptieren einen zusätzlichen Parameter, der hier angegeben werden kann. (Standardeinstellung ist `#RANDOMPARAMETER`)

- Async:** Sie können diesen Tag verwenden, um ein asynchrones Zeichnungsobjekt für diesen Übergang zu erstellen. Wenn Sie hier **True** angeben, wird `DisplayBGPicPartFX()` sofort verlassen und es wird ein Handler für das asynchrone Zeichnungsobjekt kreiert, den Sie dann mit dem Befehl `AsyncDrawFrame()` verwenden können. Ein Beispieldokument finden Sie unter dem Befehl `AsyncDrawFrame()`. Siehe [Abschnitt 19.1 \[AsyncDrawFrame\]](#), [Seite 231](#), für weitere Informationen über asynchrone Zeichnungsobjekte.
- DX:** Zielposition des Ausschnitts in x-Richtung. (voreingestellt ist x als Parameter 2)
- DY:** Zielposition des Ausschnitts in y-Richtung. (voreingestellt ist y als Parameter 3)
- Layers:** Bei **True** werden die Ebenen des Hintergrundbildes auch angezeigt (erfordert aktivierte Ebenen). (Standardeinstellung **False**)

#### EINGABEN

- id** ID des Quellhintergrundbildes
- x** linke Ecke
- y** obere Ecke
- width** Breite des Ausschnittes
- height** Höhe des Ausschnittes
- table** optional: Konfiguration der Übergangseffekte

#### RÜCKGABEWERTE

- handle** optional: Handler auf ein asynchrones Zeichnungsobjekt; wird nur zurückgegeben, wenn **Async** auf **True** gesetzt wurde (siehe oben)

#### BEISPIEL

```
DisplayBGPicPartFX(2, 0, 0, 100, 100, #HSTRIPES32, 10) ; alte Syntax
```

ODER

```
DisplayBGPicPartFX(2, 0, 0, 100, 100, {Type = #HSTRIPES32,
    Speed = 10}) ; neue Syntax
```

Zeigt die ersten 100 Pixel und Zeilen aus dem Hintergrundbild 2 auf dem Bildschirm mit dem Übergangseffekt `#HSTRIPES32` bei der Geschwindigkeit 10.

## 30.11 DisplayTransitionFX

#### BEZEICHNUNG

`DisplayTransitionFX` – ändert das Hintergrundbild mit einem Übergangseffekt

#### ÜBERSICHT

```
[handle] = DisplayTransitionFX(id[, table])
```

## BESCHREIBUNG

Dieser Befehl stellt ein neues Hintergrundbild mit einem Übergangseffekt dar. Eine Liste aller verfügbaren Effekte ist weiter unten. Sie müssen auch die Geschwindigkeit des Übergangs angeben, die entweder eine der speziellen Geschwindigkeitskonstanten sein kann (`#SLOWSPEED`, `#NORMALSPEED`, `#FASTSPEED`) oder eine benutzerdefinierte Feinabstimmung mit numerischem Wert. Als Faustregel für den Geschwindigkeitsparameter: Je höher der Wert, desto schneller läuft der Übergang.

Für die beste Wirkung sollte das neue Hintergrundbild die gleichen Abmessungen haben wie das alte. Ist dies nicht der Fall ist, wird das alte Hintergrundbild auf die Größe des neuen skaliert werden.

Beachten Sie, dass transparente BGPics nicht mit einen Übergangseffekt angezeigt werden können. Es ist auch nicht möglich, ein BGPic ohne Transparenz mit einem Übergangseffekt anzuzeigen, wenn das aktuelle BGPic eine Transparenz besitzt. Damit dieser Befehl funktioniert, dürfen das aktuelle und das neue BGPic nicht transparent sein.

Ab Hollywood 4.5 verwendet dieser Befehl eine neue Syntax. Die alte Syntax wird aus Kompatibilitätsgründen weiterhin unterstützt. Das optionale Tabellenargument kann den Übergangseffekt konfigurieren. Folgende Optionen sind möglich:

- Type:** Hier geben Sie den gewünschten Übergangseffekt an. Beachten Sie die Liste unten mit den möglichen Effekten. (Voreingestellt ist `#RANDEFFECT`)
- Speed:** Legt die gewünschte Geschwindigkeit für den Übergang fest. Je höher der Wert, den Sie hier angeben, desto schneller wird der Effekt angezeigt werden. (Standardeinstellung ist `#NORMALSPEED`)
- Parameter:** Einige Übergangseffekte akzeptieren einen zusätzlichen Parameter, der hier angegeben werden kann. (Standardeinstellung ist `#RANDOMPARAMETER`)
- Async:** Sie können diesen Tag verwenden, um für diesen Übergang ein asynchrones Zeichnungsobjekt zu erstellen. Wenn Sie hier `True` angeben, wird `DisplayBGPicPartFX()` sofort verlassen, und es wird ein Handler für das asynchrone Zeichnungsobjekt kreiert, den Sie dann mit dem Befehl `AsyncDrawFrame()` verwenden können. Ein Beispielskript finden Sie unter dem Befehl `AsyncDrawFrame()`. Siehe [Abschnitt 19.1 \[AsyncDrawFrame\]](#), [Seite 231](#), für weitere Informationen über asynchrone Zeichnungsobjekte.
- X:** Zielposition des neuen Hintergrundbildes in x-Richtung. Wenn das Display seine aktuelle x-Position beibehalten soll, geben Sie die spezielle Konstante `#KEEPPPOSITION` an. Der Standardwert ist `#CENTER`.
- Y:** Zielposition des neuen Hintergrundbildes in y-Richtung. Wenn das Display seine aktuelle y-Position beibehalten soll, geben Sie die spezielle Konstante `#KEEPPPOSITION` an. Der Standardwert ist `#CENTER`.

Folgende Effekte sind derzeit verfügbar:

- Horizontale Streifen: `#HSTRIPES2`, `#HSTRIPES4`, `#HSTRIPES16`, `#HSTRIPES32`
- Vertikale Streifen: `#VSTRIPES2`, `#VSTRIPES8`, `#VSTRIPES16`, `#VSTRIPES32`

- Schnelle horizontale Jalousien: #HBLINDS8, #HBLINDS16, #HBLINDS32, #HBLINDS64, #HBLINDS128
- Schnelle vertikale Jalousien: #VBBLINDS8, #VBLINDS16, #VBLINDS32, #VBLINDS64, #VBLINDS128
- Horizontal Vorhang: #HOPENCURTAIN, #HCLOSECURTAIN
- Vertical Vorhang: #VOPENCURTAIN, #VCLOSECURTAIN
- Horizontale Linien: #HLINES, #HLINES2
- Vertikale Linien: #VLINES, #VLINES2
- Enthüllen: #REVEALLEFT, #REVEALRIGHT, #REVEALTOP, #REVEALBOTTOM
- Balken: #BARS
- Viertel: #QUARTERS
- Überblenden: #CROSSFADE
- Farbübergang: #FADE; optionales Argument gibt die zu verwendende Farbe an
- Einblenden: #BLEND; optionales Argument gibt die Farbe für das Mischen an

Ab Hollywood 1.5 kamen neue Effekte dazu:

- Rechteck von innen: #RECTCENTER, #RECTNORTH, #RECTNORTHEAST, #RECTEAST, #RECTSOUTHEAST, #RECTSOUTH, #RECTSOUTHWEST, #RECTWEST, #RECTNORTHWEST
- Rectangle von außen: #RECTBACKCENTER, #RECTBACKNORTH, #RECTBACKNORTHEAST, #RECTBACKEAST, #RECTBACKSOUTHEAST, #RECTBACKSOUTH, #RECTBACKSOUTHWEST, #RECTBACKWEST, #RECTBACKNORTHWEST
- Hereinbewegen: #SCROLLLEFT, #SCROLLRIGHT, #SCROLLTOP, #SCROLLBOTTOM (das optionale Argument erlaubt die Angabe eines Effektes, der für die Bewegung genutzt werden soll; alle Effekte von `MoveBrush()` können hier angegeben werden)
- Streckungen: #STRETCHLEFT, #STRETCHRIGHT, #STRETCHTOP, #STRETCHBOTTOM, #HSTRETCHCENTER, #VSTRETCHCENTER
- Hineinzoomen: #ZOOMCENTER, #ZOOMNORTH, #ZOOMNORTHEAST, #ZOOMWEST, #ZOOMSOUTHEAST, #ZOOMSOUTH, #ZOOMSOUTHWEST, #ZOOMNORTHWEST
- Fließen: #HFLOWTOP, #HFLOWBOTTOM, #VFLOWLEFT, #VFLOWRIGHT
- Tore: #HOPENGATE, #HCLOSEGATE, #VOPENGATE, #VCLOSEGATE (B)
- Verschieben: #PUSHLEFT, #PUSHRIGHT, #PUSHTOP, #PUSHBOTTOM (B)
- Puzzle: #PUZZLE
- Diagonal: #DIAGONAL
- Hereinrollen: #ROLLTOP
- Tapete: #WALLPAPERTOP
- Allgemeine vertikale Streifen: #VSTRIPES; optionales Argument gibt die Anzahl der darzustellenden Streifen an
- Allgemeine horizontale Streifen: #HSTRIPES; optionales Argument gibt die Anzahl der darzustellenden Streifen an

Hollywood 1.9 bringt folgende neue Effekte mit:

- Hereinbewegen: #SCROLLNORTHEAST, #SCROLLSOUTHEAST, #SCROLLNORTHWEST, #SCROLLSOUTHWEST (das optionale Argument erlaubt die Angabe eines Effektes,



- der für die Bewegung genutzt werden soll; alle Effekte von `MoveBrush()` können hier angegeben werden)
- Enthüllen im Uhrzeigersinn: `#CLOCKWIPE`
  - Über Stern darstellen: `#STAR`
  - Bild geteilt bewegen: `#HSTRANGEPUSH`, `#VSTRANGEPUSH` (B)
  - Diaprojektor: `#SLIDELEFT`, `#SLIDERIGHT`, `#SLIDETOP`, `#SLIDEBOTTOM` (B)
  - Spirale: `#SPIRAL`
  - Schweizer Kreuz Effekt: `#SWISS`
  - Vier Rechtecke: `#QUADRECT`
  - Teilungseffekt: `#HSPLIT`, `#VSPLIT`
  - Hoch und runter: `#UPNDOWN`
  - Karteikarteneffekt: `#CARDTOP`, `#CARDBOTTOM` (B)
  - Sonnenaufgang: `#SUN`
  - Wasserwellen: `#WATER1`, `#WATER2`, `#WATER3`, `#WATER4` (!)
  - Strudel-Effekt: `#STRUDEL` (!)
  - Bild auflösen: `#DISSOLVE`
  - Pixel heranzoomen: `#PIXELZOOM1`
  - Pixel heranzoomen 2: `#PIXELZOOM2` (B)
  - Hoher Zoomeffekt: `#ZOOMIN`, `#ZOOMOUT` (B)
  - Bild pressen: `#CRUSHLEFT`, `#CRUSHRIGHT`, `#CRUSHTOP`, `#CRUSHBOTTOM` (B)
  - Münzeneffekt: `#VFLIPCOIN`, `#VLOWFLIPCOIN`, `#HFLIPCOIN`, `#HLOWFLIPCOIN` (B)
  - Bild herunterlassen: `#TURNDOWNTOP`, `#TURNDOWNBOTTOM`, `#TURNDOWNLEFT`, `#TURNDOWNRIGHT` (B)
  - Schreibmaschine: `#TYPEWRITER` (T) [wird seit V3.1 nicht mehr unterstützt und ist gelöscht]
  - Tapete: `#WALLPAPERLEFT` (!)
  - Hereinrollen: `#ROLLEFT`
  - Ultimativer Effekt: `#RANDEFFECT`

Wenn Sie `#RANDEFFECT` wählen, sucht sich Hollywood zufällig irgendeinen Effekt aus allen möglichen aus. Sehr nützlich, wenn Sie eine Diashow darstellen. Wenn Sie `#RANDEFFECT` in der 68k-Version aufrufen, wird Hollywood keinen der High-End Effekte auswählen, um Ihr Skript nicht lahmzulegen.

Legende:

- (B): Effekt kann nur mit Hintergrundbildern verwendet werden.
- (O): Effekt kann nur mit Objekten verwendet werden (Pinsel, Ebenen usw., aber nicht mit Hintergrundbilder).
- (T): Effekt kann nur mit Textobjekten benutzt werden.
- (!): High-End-Effekt und das bedeutet, dass er sehr viel Rechenleistung benötigt, um reibungslos abzulaufen. Sie können ihn auf 68k laufen lassen, aber es ist kein Spaß, weil er etwa 4 Minuten oder so für den Übergang braucht. Sie sollten "!" nur mit schnellen Rechnern verwenden.

**EINGABEN**

`id`            Identifikator des darzustellenden Hintergrundbildes  
`table`        optional: Konfiguration des Übergangseffektes

**RÜCKGABEWERTE**

`handle`       optional: Handler auf ein asynchrones Zeichnungsobjekt; wird nur zurückgegeben, wenn `Async` auf `True` gesetzt wurde (siehe oben)

**BEISPIEL**

`DisplayTransitionFX(2, #HSTRIPES32, 10)`    ; alte Syntax

ODER

`DisplayTransitionFX(2, {Type = #HSTRIPES32, Speed = 10})`    ; neue Syntax  
 Stellt Hintergrundbild 2 mit dem `#HSTRIPES32` Effekt bei einer Geschwindigkeit von 10 dar.

## 30.12 FreeBGPic

**BEZEICHNUNG**

`FreeBGPic` – gibt von einem Hintergrundbild belegten Speicher frei

**ÜBERSICHT**

`FreeBGPic(id)`

**BESCHREIBUNG**

Dieser Befehl gibt den Speicher frei, der von dem durch `id` angegebenen Hintergrundbild belegt wird. Normalerweise ist das unnötig, da Hollywood allen Speicher freigibt wenn es beendet wird. Falls Ihnen jedoch der Speicher ausgeht und Sie das Bild selbst entfernen möchten, benutzen Sie diesen Befehl.

**EINGABEN**

`id`            Identifikator des Hintergrundbildes

## 30.13 LoadBGPic

**BEZEICHNUNG**

`LoadBGPic` – lädt ein Hintergrundbild

**ÜBERSICHT**

`[id] = LoadBGPic(id, filename$, table)`

**BESCHREIBUNG**

Dieser Befehl lädt das Bild `filename$` in den Speicher und weist `id` die Nummer zu. Wenn Sie `Nil` als `id` angeben, wird automatisch eine ID für dieses Hintergrundbild ausgewählt und in `id` zurückgegeben.

Bildformate, die auf allen Plattformen unterstützt werden, sind PNG, JPEG, BMP, IFF ILBM, GIF und Plugins für Bilder. Je nach Plattform, worauf Hollywood ausgeführt

wird, können mehr Bildformate unterstützt werden. Zum Beispiel auf den Amigakompatiblen Systemen wird Hollywood in der Lage sein, alle Formate über `Bilddatatypes` zu öffnen. Unter Windows kann `LoadBGPic()` auch Bildformate laden, die von der Windows-Imaging-Komponente unterstützt werden.

Ab Hollywood 5.0 kann dieser Befehl auch Vektorformate wie SVG laden, wenn Sie eine entsprechende Plugin installieren. Die Verwendung eines Vektorbildes als BGPic hat den Vorteil, dass wenn sich die Größe des Displays ändert (z.B. weil der Benutzer das Fenster in der Größe ändert) kann das BGPic ohne Einbußen in der Qualität an die neue Größe angepasst werden, da Vektor-BGPics ohne Qualitätsverluste stufenlos skaliert werden können. Siehe [Abschnitt 30.16 \[Vektor-BGPics\]](#), [Seite 642](#), für weitere Informationen zu Vektor-BGPics.

Das dritte Argument `table` ist optional. Es ist eine Tabelle, die weitere Möglichkeiten für den Ladevorgang verwendet werden können. Die folgenden Tags stehen dafür zur Verfügung:

**Transparency:**

Dieser Tag kann verwendet werden, um eine Farbe in **RGB**-Notation, anzugeben, die im BGPic transparent erscheinen soll.

**LoadAlpha:**

Setzen Sie diesen Tag auf **True**, wenn der Alpha-Kanal des Bildes auch geladen werden soll. Bitte beachten Sie, dass nicht alle Bilder einen Alpha-Kanal haben und dass nicht alle Bildformate in der Lage sind, Alphakanaldaten zu speichern. Es wird vorgeschlagen, dass Sie das PNG-Format verwenden, wenn Sie Alphakanaldaten benötigen. Dieser Tag ist standardmäßig **False**. (V4.5)

**FillStyle:**

Dieser Tag ermöglicht es Ihnen, einen gefüllten Hintergrund mit einem Füllstil für dieses BGPic zu definieren. Dieser Füllstil wird nur verwendet, wenn das BGPic transparente Bereiche hat, das heißt, wenn Sie entweder die Tags **Transparency** oder **LoadAlpha** verwenden oder wenn Sie ein Bild in einem Format laden, die einen Alphakanal enthält. **FillStyle** ist nützlich, vor allem im letzteren Fall, weil einige Formate immer einen Alpha-Kanal zurückgeben, aber die meiste Zeit werden Sie diesen Alphakanal nicht verwenden, wenn ein solches Bild als ein Hintergrundbild geladen wird. Siehe [Abschnitt 29.14 \[SetFillStyle\]](#), [Seite 610](#), für Informationen über alle verfügbaren Füllstile. (V5.0)

**FillColor:**

Wenn beim Tag **FillStyle** **#FILLCOLOR** gesetzt wurde, können Sie mit diesem Tag die RGB-Farbe definieren, die zur Füllung verwendet werden soll. (V5.0)

**TextureBrush:**

Wenn beim Tag **FillStyle** **#FILLTEXTURE** gesetzt wurde, können Sie mit diesem Tag die ID des Pinsels angeben, die für die Texturierung verwendet werden soll. (V5.0)

**TextureX, TextureY:**

Diese Tags steuern den Startpunkt innerhalb des Texturpinsels und werden nur unterstützt, wenn `FillStyle` auf `#FILLTEXTURE` gesetzt wurde. Siehe [Abschnitt 29.14 \[SetFillStyle\]](#), [Seite 610](#), für Details. (V5.0)

**GradientStyle:**

Wenn `FillStyle` auf `#FILLGRADIENT` gesetzt wurde, können Sie diesen Tag nutzen, um den Farbverlauftyp anzugeben. Dies kann `#LINEAR`, `#RADIAL` oder `#CONICAL` sein. (V5.0)

**GradientAngle:**

Gibt die Ausrichtung des Farbverlaufs an, wenn bei `FillStyle` `#FILLGRADIENT` eingestellt ist. Der Winkel wird in Grad ausgedrückt. Nur für die beiden `GradientStyle` `#LINEAR` und `#CONICAL`. (V5.0)

**GradientStartColor, GradientEndColor:**

Verwenden Sie diese beiden Farben, um den Farbverlauf zu konfigurieren, wenn `FillStyle` auf `#FILLGRADIENT` gesetzt wurde. (V5.0)

**GradientCenterX, GradientCenterY:**

Legt den Mittelpunkt für die `GradientStyle` des Typs `#RADIAL` oder `#CONICAL` fest. Muss ein Fließkommawert zwischen 0.0 und 1.0 sein. Siehe [Abschnitt 30.6 \[CreateGradientBGPic\]](#), [Seite 624](#), für Details. (V5.0)

**GradientBalance:**

Dieser Tag steuert den Balancepunkt für Farbverläufe des Typs `#CONICAL`. Muss ein Fließkommawert zwischen 0.0 und 1.0 sein. Siehe [Abschnitt 30.6 \[CreateGradientBGPic\]](#), [Seite 624](#), für Details. (V5.0)

**GradientBorder:**

Dieser Tag steuert die Rahmengröße der Farbverläufe des Typs `#RADIAL`. Muss ein Fließkommawert zwischen 0.0 und 1.0 sein. Siehe [Abschnitt 30.6 \[CreateGradientBGPic\]](#), [Seite 624](#), für Details. (V5.0)

**GradientColors:**

Dieser Tag kann verwendet werden, um einen Farbverlauf zwischen mehr als zwei Farben zu erzeugen. Dies muss mit einer Tabelle gesetzt werden, die Sequenzen von abwechselnden Farben und Stoppwerte enthält. Wenn dieser Tag verwendet wird, werden die Tags `GradientStartColor` und `GradientEndColor` ignoriert. Siehe [Abschnitt 30.6 \[CreateGradientBGPic\]](#), [Seite 624](#), für Details. (V5.0)

**ScaleWidth, ScaleHeight:**

Diese Tags können verwendet werden, um eine skalierte Version des Bildes zu laden. Wenn der Bildtreiber skaliertes Laden unterstützt, kann das die Geschwindigkeit erheblich steigern, wenn Sie zum Beispiel nur eine Version in Thumbnailgröße eines großen Bildes laden möchten. Wenn der Bildtreiber skaliertes Laden nicht unterstützt, wird zuerst das volle Bild geladen, bevor es skaliert wird. Dies ist nicht wesentlich schneller als das Bild nach dem Laden von Hand zu skalieren. Sie können die Größe entweder als direkter Wert übergeben oder Sie benutzen einen Prozentsatz als Zeichenfolge (z.B. `Scalewidth = "200%"`). (V5.3)

**SmoothScale:**

Wenn **ScaleWidth** oder **ScaleHeight** eingestellt ist, können Sie mit diesem Tag festlegen, ob Hollywood bei der Skalierung Antialiasing verwenden soll oder nicht. Der Standardwert ist **False** und bedeutet kein Antialiasing. Beachten Sie, dass Skalierung mit Antialiasing viel langsamer ist als die normale Skalierung. (V5.3)

**Loader:** Mit diesem Tag können Sie ein oder mehrere Formatlademodule angeben, die dieses BGPic laden sollen. Dies muss als eine Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Lademodulen enthält. Standardmäßig wird der mit **SetDefaultLoader()** eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen sollen. Dies muss als eine Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Adaptermodulen enthält. Standardmäßig wird der mit **SetDefaultAdapter()** eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**LoadTransparency:**

Ist dieser Tag auf **True** gesetzt, wird die monochrome Transparenz des Bildes geladen. Bitte beachten Sie, dass dieser Tag speziell für monochrome Transparenzkanäle ist, das heißt der transparente Stift ist für ein Palettenbasiertes Bild ausgelegt. Wenn Sie den Alpha-Kanal eines Bildes laden möchten, stellen Sie den Tag **LoadAlpha** auf **True**. Dieser Tag ist auf **False** voreingestellt. (V6.0)

**LoadPalette:**

Wenn dieser Tag auf **True** gesetzt ist, lädt Hollywood das BGPic als Paletten-BGPic. Dies bedeutet, dass Sie die Palette des BGPic abrufen und verändern können, was für bestimmte Effekte wie Farbwechsel nützlich ist. Sie können Stifte auch mit dem Tag **TransparentPen** (siehe unten) oder dem Tag **LoadTransparency** (siehe oben) transparent machen. Paletten-BGPics haben auch den Vorteil, dass sie weniger Speicher benötigen, da ein Pixel nur 1 Byte Speicher anstelle von 4 Byte für 32-Bit-Bilder benötigt. Beachten Sie, dass wenn Sie den Tag **LoadPalette** auf **True** setzen, Ihr Display zu einem Palettenmodus-Display wird, sobald das BGPic angezeigt wird. Palettenmodus-Displays verhalten sich anders als normale True-Color-Displays und bei ihrer Verwendung sind einige Dinge zu beachten. Siehe [Abschnitt 24.16 \[Palettenmodus-Displays\]](#), [Seite 393](#), für Details. Dieser Tag ist standardmäßig **False**. (V9.0)

**FillPen:** Wenn der Tag **LoadPalette** auf **True** gesetzt wurde (siehe oben) und im Bild ein transparenter Stift vorhanden ist, können Sie mit dem Tag **FillPen** die Füllfarbe für alle transparenten Bereiche im Bild festlegen. Dies ist die Palette, die dem Tag **FillColor** entspricht, das nur für Bilder ohne Paletten verwendet wird. (V9.0)

**TransparentPen:**

Wenn der Tag `LoadPalette` auf `True` gesetzt wurde (siehe oben), kann mit dem Tag `TransparentPen` ein Stift definiert werden, der transparent gemacht werden soll. Stifte werden ab 0 gezählt. Alternativ können Sie auch den Tag `LoadTransparency` auf `True` setzen, um Hollywood zu zwingen, den transparenten Stift zu verwenden, der in der Bilddatei gespeichert ist (sofern das Bildformat die Speicherung von transparenten Stiften unterstützt). Dieser Tag ist standardmäßig `#NOPEN`. (V9.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Wenn Sie `Transparency`, `LoadTransparency` oder `LoadAlpha` verwenden, wird Ihr Display automatisch die Transparenzeinstellungen des BGPic annehmen, wenn es angezeigt wird. Mit anderen Worten, wenn Sie ein Display mit Transparenz haben möchten, laden Sie einfach ein transparentes BGPic und zeigen es an.

Bitte beachten Sie, dass sich die Tags `Transparency`, `LoadTransparency` und `LoadAlpha` gegenseitig ausschließen. Ein BGPic kann nur eine Transparenzeinstellung haben!

Dieser Befehl gibt es auch als Präprozessor: Verwenden Sie `@BGPIC`, um Hintergrundbilder vorzuladen!

Siehe auch `IsPicture()`.

**EINGABEN**

`id` ID des neuen BGPic oder `Nil` für die [automatische ID-Zuweisung](#)

`filename$`  
Datei zum Laden

`table` optional: weitere Konfigurationsmöglichkeiten für den Ladevorgang

**RÜCKGABEWERTE**

`id` optional: Identifikator des BGPic; Wird nur zurückgegeben werden, wenn Sie `Nil` als Argument 1 angegeben haben (siehe oben)

**BEISPIEL**

```
LoadBGPic(2, "MyBG.iff", {Transparency = $00FF00})
```

Das lädt "MyBG.iff" als Hintergrundbild 2 mit Farbe Grün als Transparenzmaske.

## 30.14 ScaleBGPic

**BEZEICHNUNG**

`ScaleBGPic` – skaliert ein Hintergrundbild (V1.5)

**ÜBERSICHT**

```
ScaleBGPic(id, width, height[, smooth])
```

**BESCHREIBUNG**

Dieser Befehl skaliert das Hintergrundbild, welches in `id` angegeben wurde, auf die neue Größe, die durch `width` (Breite) und `height` (Höhe) angegeben wurde. Sie können diesen Befehl nicht mit dem Hintergrundbild benutzen, welches gerade angezeigt wird. Wenn Sie die Dimensionen dieses Hintergrundbildes verändern möchten, müssen Sie den Befehl `ChangeDisplaySize()` benutzen.

Hollywood behält immer das Originalbild jedes Hintergrundbildes im Speicher, so dass Sie das nicht machen müssen. Jede Skalierung, die `ScaleBGPic()` durchführt, benutzt das Originalbild als Ausgangsbild und nicht eine evtl. schon skalierte Version. `ScaleBrush()` dagegen hat keinen Puffer für den Originalpinsel. Das Originalbild wird im Puffer nur gelöscht, wenn Sie mit `SelectBGPic()` ein anderes Hintergrundbild wählen.

Neu in V2.0: Sie können `#KEEPASPRAT` entweder für die Breite oder Höhe einsetzen. Hollywood wird dann die Größe automatisch anpassen, indem es das Seitenverhältnis des Hintergrundbildes berechnet. Außerdem können Sie für die Höhe und Breite auch eine Prozentzahl in Form einer Zeichenkette angeben. z.B. "50%".

Ab Version 2.5 dürfen Sie im Argument `smooth True` angeben, damit bei der Skalierung der Grafik das Glättungsverfahren Antialias verwendet wird.

**EINGABEN**

<code>id</code>	Kennung des zu skalierenden Hintergrundbildes
<code>width</code>	gewünschte Breite für das Hintergrundbild
<code>height</code>	gewünschte Höhe für das Hintergrundbild
<code>smooth</code>	optional: Antialias bei der Skalierung verwenden oder nicht (V2.5)

**30.15 SelectBGPic****BEZEICHNUNG**

SelectBGPic – wählt ein Hintergrundbild als Ausgabeziel (V1.5)

**ÜBERSICHT**

```
SelectBGPic(id[, mode, combomode])
```

**BESCHREIBUNG**

Dieser Befehl selektiert das durch `id` angegebene Hintergrundbild als aktuelles Ausgabeziel. Dieser Befehl kann in verschiedenen Modi verwendet werden. Der übliche Modus ist der Ebenenmodus (`#SELMODE_LAYERS`), der auch der Standardmodus ist. Ebenenmodus bedeutet, alle Grafikdatenausgaben von Hollywood werden als Ebenen diesem Hintergrundbild hinzugefügt werden. Daher müssen Sie Ebenen aktivieren, bevor Sie diesen Befehl im Ebenenmodus benutzen. Ihr Hintergrundbild wird nie im Ebenenmodus geändert werden, es erhält nur mehr und mehr Ebenen.

Alternativ können Sie auch die Modi `#SELMODE_NORMAL` oder `#SELMODE_COMBO` verwenden. Diese Modi werden Ihre Daten des BGPic ändern. Sie können nur auf BGPics verwendet werden, die derzeit nicht einem Display zugewiesen sind. `#SELMODE_NORMAL` bedeutet, dass nur die Farbkanäle des BGPic geändert werden, wenn Sie reinzeichnen. Der Transparenzkanal des BGPic (kann entweder eine Maske oder ein Alpha-Kanal sein)



wird nie verändert werden. Sie können dies ändern, indem Sie `#SELMODE_COMBO` im optionalen Argument `mode` angeben. Wenn Sie diesen Modus verwenden, werden alle Befehle für Hollywoodgrafiken, die nach `SelectBGPic()` aufgerufen werden, die Farbe und den Transparenzkanal des BGPic ändern. Wenn das BGPic keinen Transparenzkanal hat, verhält sich `#SELMODE_COMBO` gleich wie `#SELMODE_NORMAL`.

Ab Hollywood 5.0 können Sie das optionale Argument `combomode` verwenden, um festzulegen, wie sich `#SELMODE_COMBO` verhalten soll. Ist `combomode` auf 0 gesetzt, werden die Farb- und Transparenzinformation aller Pixel in dem Quellenbild in jedem Fall zu dem Zielbild kopiert, selbst wenn die Pixel unsichtbar sind. Dies ist das Standardverhalten. Wenn `combomode` auf 1 gesetzt ist, werden nur die sichtbaren Pixel in das Zielbild kopiert. Dies bedeutet, wenn der Alphawert eines Pixels in dem Quellenbild 0 ist (unsichtbar), wird es nicht in das Zielbild kopiert werden.

Hollywood 6.0 stellt den neuen `combomode` 2 zur Verfügung. In diesem Fall wird Hollywood die Farbkanäle und den Alpha-Kanal des Quellbildes in die Farbe und Alpha der Zielbildkanäle mischen. Wenn Sie später das Zielbild zeichnen, wird es so aussehen, als ob die beiden Bilder hintereinander auf jeweils des anderen gezeichnet wurde. Bitte beachten Sie, dass das Argument `combomode` nur zusammen mit `#SELMODE_COMBO` unterstützt wird. Es hat keine Wirkung, wenn sie es mit den anderen Modi verwenden.

Beachten Sie, dass wenn Sie `#SELMODE_NORMAL` oder `#SELMODE_COMBO` verwenden, werden die ursprünglichen Grafiken des BGPic modifiziert. Sie werden immer auf die Originalgrafiken des BGPic zeichnen.

Stellen Sie sich vor, Sie haben 640x480 BGPic, welches Sie derzeit mit dem Befehl `ChangeDisplaySize()` auf 800x600 skaliert haben. Wenn Sie jetzt `SelectBGPic()` mit `#SELMODE_NORMAL` oder `#SELMODE_COMBO` auf dieses BGPic anwenden, werden Sie tatsächlich in das 640x480 Bild zeichnen. Das 800x600 Bild wird aktualisiert, wenn `EndSelect()` aufgerufen wird. Hollywood wird dann die Original-Grafik auf die aktuellen Ausgabegröße des BGPic skalieren, aber Ihre Zeichnungsbefehle werden immer auf dem originalen BGPic ausgeführt.

Ein alternativer Weg, um in die Transparenzkanäle eines BGPic zu zeichnen, ist, die Befehle `SelectMask()` oder `SelectAlphaChannel()` zu verwenden. Diese beiden Befehle werden jedoch die Schreibdaten nur auf den Transparenzkanal beschränken. Sie werden nicht den Farbkanal beeinflussen. Wenn Sie beide Kanäle (Farbe und Transparenz) ändern wollen, müssen Sie das Argument `mode` beim Befehl `SelectBGPic()` auf `#SELMODE_COMBO` setzen.

Wenn Sie mit dem Zeichnen in Ihr BGPic fertig sind und möchten, dass Ihr Display wieder das Ausgabeziel wird, rufen Sie einfach den Befehl `EndSelect()` auf.

Nur Befehle mit direkter Grafikausgabe können mit `SelectBGPic()` verwendet werden. Sie können nicht animierte Befehle wie `MoveBrush()` oder `DisplayBrushFX()` aufrufen, während `SelectBGPic()` aktiv ist.

Wenn `mode` auf `#SELMODE_LAYERS` gesetzt ist, kann `SelectBGPic()` auch nützlich sein. So können Sie mehrere Änderungen an den Ebenen des aktuellen BGPic erledigen, ohne dass nach jeder Änderung das BGPic aktualisiert wird. Zum Beispiel können Sie auf einmal 100 neue Ebenen einfügen. Das wäre ziemlich langsam, wenn man es in der herkömmlichen Art und Weise tun würde, weil Hollywood das Display hundertmal aktualisiert. Um dies zu vermeiden, können Sie einfach `SelectBGPic()` aufrufen, die 100 Ebe-



nen einfügen und Hollywood wird das Display nicht aktualisieren, bevor Sie `EndSelect()` aufgerufen haben. Innerhalb eines `SelectBGPic()`-`EndSelect()` Block können Sie so viele Änderungen vornehmen, wie Sie möchten. Sie werden nicht vor dem Befehl `EndSelect()` gezeichnet werden. Siehe unten für ein Beispiel.

Siehe auch `EndSelect()`, `SelectAlphaChannel()`, `SelectBrush()`, `SelectMask()`, `SelectLayer()` und `SelectAnim()`.

## EINGABEN

<code>id</code>	Hintergrundbild, welches als Ziel benutzt werden soll
<code>mode</code>	optional: Modus mit <code>#SELMODE_LAYERS</code> , <code>#SELMODE_NORMAL</code> oder <code>#SELMODE_COMBO</code> als Parameter (siehe oben); Voreingestellt ist <code>#SELMODE_LAYERS</code> (V4.5)
<code>combomode</code>	optional: Modus, wenn <code>#SELMODE_COMBO</code> aktiv ist (siehe oben); voreingestellt ist 0 (V5.0)

## BEISPIEL

```
EnableLayers()
SelectBGPic(2)
TextOut(#CENTER, #CENTER, "Hello World")
Box(0, 0, 100, 100, #RED)
Box(#RIGHT, #BOTTOM, 100, 100, #BLUE)
EndSelect()
DisplayBGPic(2)
```

Der obige Code wählt das Hintergrundbild 2 als aktuelles Ziel und fügt drei Ebenen (eine Text und zwei Rechteck) ein. Danach wird das Display als Ausgabeziel ausgewählt und Hintergrundbild 2 mit drei Ebenen dargestellt.

```
SetFillStyle(#FILLCOLOR)
EnableLayers
SelectBGPic(1) ; wir gehen davon aus, 1 ist unser aktuelles BGPic
; erstellt 100 Ebenen
For Local k = 1 To 100
    Box(Rnd(540), Rnd(380), 100, 100, RGB(Rnd(255), Rnd(255), Rnd(255)))
Next
EndSelect ; nun werden die 100 Ebenen aufs Mal dargestellt!
```

Dieser Code stellt den oben diskutierten Fall dar. Sie müssen viele Änderungen vornehmen und Sie möchten aus Performancegründen das Zeichnen auf später verschieben. In unserem Fall wollen wir 100 Ebenen dem aktuellen BGPic hinzuzufügen. So kapseln wir diesen Code durch einen `SelectBGPic()`-`EndSelect` Block. Hollywood wird die 100 Ebenen hinzufügen und wird sie in einem Rutsch zeichnen, wenn `EndSelect()` aufgerufen wird. Das ist viel schneller, als ohne `SelectBGPic()`, weil in diesem Fall jeder Aufruf von `Box()` eine Aktualisierung verursachen würde.

## 30.16 Vektor-BGPic

Wenn Sie ein Vektorbild mit den Befehlen `LoadBGPic()` oder `@BGPIC` laden, werden Sie eine spezielle Art von Hintergrundbild erhalten: Ein Vektor-BGPic. Wenn Sie normale Bilder wie PNG, JPEG etc. laden, werden Sie immer ein Raster-BGPic bekommen. Sie können mit dem Befehl `GetAttribute()` und dem Attribut `#ATTRTYPE` den Typ eines BGPic herausfinden.

Der Vorteil eines Vektor-BGPic ist, Sie können die Größe ohne Qualitätsverlust ändern und/oder transformieren. Zum Beispiel, wenn der Benutzer ein Display in der Größe ändert, kann sein BGPic ohne Qualitätsopfer an die neue Größe angepasst werden. So ist es möglich, Skripte zu erstellen, die frei skalierbar sind. Alles was Sie tun müssen, ist Vektor-BGPics, Vektorpinsel und Vektortext (z.B. TrueType-Schriftarten) zu verwenden.

Neben von Vektor-Bildformate erzeugten Vektor-BGPics, gibt es in Hollywood auch einige andere Arten von Vektor-BGPics. Zum Beispiel erzeugen die Befehle `CreateGradientBGPic()` und `CreateTexturedBGPic()` auch Vektor-BGPics, die unendlich skaliert werden können.

## 31 IPC-Bibliothek

### 31.1 CreatePort

#### BEZEICHNUNG

CreatePort – erstellt einen Nachrichtenport für Ihr Skript (V5.0)

#### ÜBERSICHT

CreatePort(name\$)

#### BESCHREIBUNG

Mit diesem Befehl wird ein Nachrichtenport für Ihr Skript erstellt und ihm der angegebene Name zugewiesen. Um Nachrichten von `SendMessage()` zu empfangen, muss Ihr Skript einen Nachrichtenport haben. Andere Hollywood-Anwendungen können dann mit Ihrem Skript kommunizieren, indem sie Nachrichten an diesen Port senden. Alle Nachrichten, die zu Ihrem Nachrichtenport gelangen, werden an die Callback-Funktion weitergeleitet, welche Sie mit dem Befehl `InstallEventHandler()` dem Ereignis-Handler `OnUserMessage` zugewiesen haben. Wenn Sie den Ereignis-Handler nicht installieren, erhalten Sie keine Benachrichtigungen über eingehende Nachrichten.

Bitte beachten Sie, dass Nachrichtenportnamen immer in Groß-/Kleinschreibung notiert werden. Somit bedeuten "MYPORT" und "myport" zwei unterschiedliche Nachrichtenports. Aus Stilgründen wird empfohlen, dass Sie nur Großbuchstaben für Ihren Anschlussnamen verwenden. Außerdem muss jeder Nachrichtenport im System eindeutig sein. Wenn Sie einen Portnamen angeben, der bereits verwendet wird, schlägt dieser Befehl fehl. Stellen Sie daher sicher, dass Sie einen eindeutigen Namen verwenden.

Bitte beachten Sie, dass jedes Hollywood-Skript nur einen Nachrichtenport haben kann. Wenn Sie bereits einen Nachrichtenport angelegt haben und dieser Befehl erneut aufgerufen wird, wird der alte Nachrichtenport gelöscht.

Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), [Seite 483](#), Für weitere Informationen, wie die Callback-Funktion des Benutzers aufgerufen wird.

#### EINGABEN

name\$      gewünschter Name für Ihren Nachrichten-Port

#### BEISPIEL

```
Function p_EventFunc(msg)
  Switch msg.action
  Case "OnUserMessage"
    Switch msg.command
    Case "EXIT"
      DebugPrint("Exit received! Quitting now.")
    End
  Default
    Local t = SplitStr(msg.args, "\0")
    DebugPrint(msg.command, "called with", msg argc, "argument(s)")
    For Local k = 1 To msg argc
      DebugPrint("Argument", k .. ":", t[k - 1])
    End For
  End Switch
End Function
```

```

        Next
    EndSwitch
EndSwitch
EndFunction
CreatePort("MY_COOL_PORT_123")
InstallEventHandler({OnUserMessage = p_EventFunc})
Repeat
    WaitEvent
Forever

```

Speichern Sie den obigen Code als ein Hollywood-Skript und führen Sie ihn mit Hollywood aus. Dann speichern Sie den folgenden Code als neues Hollywood-Skript und führen ihn aus:

```

SendMessage("MY_COOL_PORT_123", "INIT", "Value1", "Value2", "Value3")
SendMessage("MY_COOL_PORT_123", "DO_SOMETHING", "Argument1")
SendMessage("MY_COOL_PORT_123", "EXIT")

```

Der obige Code kommuniziert dann mit dem ersten Skript. Sie sehen, dass die Nachrichten aus der Debug-Ausgabe von Skript Nummer eins ankommen.

## 31.2 SendMessage

### BEZEICHNUNG

SendMessage – sendet eine Nachricht an den Nachrichtenport (V5.0)

### ÜBERSICHT

```
SendMessage(port$, cmd$[, ...])
```

### BESCHREIBUNG

Dieser Befehl sendet den in `cmd$` angegebene Nachricht an den in `port$` angegebenen Nachrichtenport. Der in `cmd$` angegebene Befehl darf keine Leerzeichen enthalten. Darüber hinaus können Sie eine unbegrenzte Anzahl von Argumenten an den Nachrichtenport senden. Geben Sie sie einfach als optionales Argumente nach dem Befehlsnamen ein. Die optionalen Argumente müssen als Zeichenfolge übergeben werden.

Der Port der in `port$` angegeben wurde, muss zuvor durch den Aufruf von `CreatePort()` erstellt worden sein. Bitte beachten Sie die Groß- und Kleinschreibung im Portnamen, d.h. "MYPORT" und "myport" bezeichnen zwei verschiedene Nachrichtenports. Aus Stillgründen sind Portnamen meist nur in Großbuchstaben.

Die Nachricht wird an den angegebenen Nachrichtenport in Form eines `OnUserMessage`-Ereignisses gesendet, das an die Callback-Funktion weitergeleitet wird, die Sie bei der Installation des Ereignisses in `InstallEventHandler()` angegeben haben.

### EINGABEN

<code>port\$</code>	Name des zu adressierenden Anschlusses
<code>cmd\$</code>	Die Befehle die Sie an diesen Port senden möchten
<code>...</code>	optional: Zusätzliche Zeichenfolgenargumente zum Senden an den Port

**BEISPIEL**

Siehe [Abschnitt 31.1 \[CreatePort\]](#), Seite 643.



## 32 Joystick-Bibliothek

### 32.1 ConfigureJoystick

#### BEZEICHNUNG

ConfigureJoystick – stellt die Joystick-Optionen ein (V10.0)

#### ÜBERSICHT

ConfigureJoystick(t)

#### BESCHREIBUNG

Mit diesem Befehl können verschiedene Joystick-Optionen konfiguriert werden. Als einzigen Funktionsparameter `t` müssen Sie eine Tabelle übergeben. Diese Tabelle kann die folgenden Tags enthalten:

UseAmigaInput:

Dieser Tag wird nur von AmigaOS 4 unterstützt. Wenn Sie ihn auf `True` setzen, wird das AmigaInput-System von OS4 verwendet, um Joystick-Status abzufragen. Der Vorteil der Verwendung von AmigaInput anstelle von `lowlevel.library`, was die Voreinstellung bei Amiga ist, besteht darin, dass Sie mehr als 7 Schaltflächen abfragen können und Benutzer nicht die AmigaInput-Voreinstellungen verwenden müssen, um ihre Controller den Lowlevel.library-Ports zuzuordnen.

#### EINGABEN

`t`            Tabelle mit einer oder mehreren Optionen (siehe oben)

### 32.2 CountJoysticks

#### BEZEICHNUNG

CountJoysticks – ermittelt die Anzahl der derzeit eingesteckten Joysticks (V4.6)

#### ÜBERSICHT

`r = CountJoysticks()`

#### BESCHREIBUNG

Dieser Befehl gibt die Anzahl der derzeit eingesteckt Joysticks zurück. Sie können dann die einzelnen Joysticks mit den Befehlen `JoyDir()` und `JoyButton()` abfragen. Dieser Befehl ist nützlich, um zu überprüfen, ob überhaupt ein Joystick zur Verfügung steht. Wenn 0 zurück gegeben wird, dann sind derzeit keine Joysticks eingesteckt, die von Hollywood erkannt werden.

Siehe auch `JoyAxisX()`, `JoyAxisY()`, `JoyAxisZ()` und `JoyHat()`.

#### EINGABEN

Keine

#### RÜCKGABEWERTE

`r`            Anzahl eingesteckter Joysticks oder 0 für keinen

## 32.3 JoyAxisX

### BEZEICHNUNG

JoyAxisX – gibt die Stellung der x-Achse des Joysticks zurück (V10.0)

### ÜBERSICHT

```
state = JoyAxisX(port[, idx])
```

### BESCHREIBUNG

Dieser Befehl gibt die aktuelle Stellung der x-Achse des Joysticks an dem durch **port** angegebenen Anschluss zurück. Die Stellung der x-Achse wird im Bereich von -1000 bis 1000 zurückgegeben. Das optionale Argument **idx** kann verwendet werden, um den Index des zu verwendenden Joysticks anzugeben, falls es mehrere Joysticks an einem Controller gibt. Joystick-Indizes beginnen bei 0.

**port** kann von 0 bis zur Anzahl der aktuell angeschlossenen Joysticks minus 1 liegen. Sie können die Anzahl der aktuell verfügbaren Joysticks mit dem Befehl **CountJoysticks()** ermitteln. Bitte beachten Sie, dass Port 0 unter AmigaOS den Standard-Joystick-Port anspricht, obwohl dies Port 1 auf klassischer Amiga-Hardware ist. Hollywood wechselt diese Ports für plattformübergreifende Konsistenz, wobei Port 0 immer auf den Standard-Joystick verweist.

Siehe auch **JoyAxisY()**, **JoyAxisZ()**, **JoyButton()**, **JoyDir()** und **JoyHat()**.

### EINGABEN

<b>port</b>	Port-Nummer (normalerweise 0 für den Standard-Joystick-Anschluss)
<b>idx</b>	optional: abzufragender Joystick-Index (Standardwert: 0)

### RÜCKGABEWERTE

<b>state</b>	Stellung der x-Achse des Joysticks im Bereich von -1000 bis 1000
--------------	--

## 32.4 JoyAxisY

### BEZEICHNUNG

JoyAxisY – gibt die Stellung der y-Achse des Joysticks zurück (V10.0)

### ÜBERSICHT

```
state = JoyAxisY(port[, idx])
```

### BESCHREIBUNG

Dieser Befehl gibt die aktuelle Stellung der y-Achse des Joysticks an dem durch **port** angegebenen Anschluss zurück. Die Stellung der y-Achse wird im Bereich von -1000 bis 1000 zurückgegeben. Das optionale Argument **idx** kann verwendet werden, um den Index des zu verwendenden Joysticks anzugeben, falls es mehrere Joysticks an einem Controller gibt. Joystick-Indizes beginnen bei 0.

**port** kann von 0 bis zur Anzahl der aktuell angeschlossenen Joysticks minus 1 liegen. Sie können die Anzahl der aktuell verfügbaren Joysticks mit dem Befehl **CountJoysticks()** ermitteln. Bitte beachten Sie, dass Port 0 unter AmigaOS den Standard-Joystick-Port anspricht, obwohl dies Port 1 auf klassischer Amiga-Hardware ist. Hollywood wechselt



diese Ports für plattformübergreifende Konsistenz, wobei Port 0 immer auf den Standard-Joystick verweist.

Siehe auch `JoyAxisX()`, `JoyAxisZ()`, `JoyButton()`, `JoyDir()` und `JoyHat()`.

#### EINGABEN

**port** Port-Nummer (normalerweise 0 für den Standard-Joystick-Anschluss)  
**idx** optional: abzufragender Joystick-Index (Standardwert: 0)

#### RÜCKGABEWERTE

**state** Stellung der y-Achse des Joysticks im Bereich von -1000 bis 1000

## 32.5 JoyAxisZ

#### BEZEICHNUNG

JoyAxisZ – gibt die Stellung der z-Achse des Joysticks zurück (V10.0)

#### ÜBERSICHT

```
state = JoyAxisZ(port[, idx])
```

#### BESCHREIBUNG

Dieser Befehl gibt die aktuelle Stellung der z-Achse des Joysticks an dem durch **port** angegebenen Anschluss zurück. Die Stellung der z-Achse wird im Bereich von -1000 bis 1000 zurückgegeben. Das optionale Argument **idx** kann verwendet werden, um den Index des zu verwendenden Joysticks anzugeben, falls es mehrere Joysticks an einem Controller gibt. Joystick-Indizes beginnen bei 0.

**port** kann von 0 bis zur Anzahl der aktuell angeschlossenen Joysticks minus 1 liegen. Sie können die Anzahl der aktuell verfügbaren Joysticks mit dem Befehl `CountJoysticks()` ermitteln. Bitte beachten Sie, dass Port 0 unter AmigaOS den Standard-Joystick-Port anspricht, obwohl dies Port 1 auf klassischer Amiga-Hardware ist. Hollywood wechselt diese Ports für plattformübergreifende Konsistenz, wobei Port 0 immer auf den Standard-Joystick verweist.

Siehe auch `JoyAxisX()`, `JoyAxisY()`, `JoyButton()`, `JoyDir()` und `JoyHat()`.

#### EINGABEN

**port** Port-Nummer (normalerweise 0 für den Standard-Joystick-Anschluss)  
**idx** optional: abzufragender Joystick-Index (Standardwert: 0)

#### RÜCKGABEWERTE

**state** Stellung der z-Achse des Joysticks im Bereich von -1000 bis 1000

## 32.6 JoyButton

#### BEZEICHNUNG

JoyButton – prüft, ob ein Joystickknopf gedrückt ist (V1.5)

#### FRÜHERER NAME

JoyFire (V1.5 - V9.1)

**ÜBERSICHT**

```
pressed = JoyButton(port[, button])
```

**BESCHREIBUNG**

Dieser Befehl gibt **True** zurück, wenn ein Knopf des Joysticks, der in den durch **port** angegebenen Anschluss eingesteckt ist, gedrückt wurde. Andernfalls wird **False** zurückgegeben. Das optionale Argument **button** gibt an, nach welchem Knopf gesucht werden soll. Wenn Sie nach einem bestimmten Knopf suchen, geben Sie die Nummer von diesem Knopf an (muss zwischen 1 und 32 liegen). Wenn Sie nach mehreren Knöpfen suchen, geben Sie 0 an und dieser Befehl gibt eine 32-Bit-Maske zurück, in der jedes der 32 Bit die Stellung des Knopfes anzeigt (gedrückt oder nicht gedrückt).

**port** kann von 0 bis zur Anzahl der aktuell angeschlossenen Joysticks minus 1 liegen. Sie können die Anzahl der aktuell verfügbaren Joysticks mit dem Befehl **CountJoysticks()** ermitteln. Bitte beachten Sie, dass Port 0 unter AmigaOS den Standard-Joystick-Port anspricht, obwohl dies Port 1 auf klassischer Amiga-Hardware ist. Hollywood wechselt diese Ports für plattformübergreifende Konsistenz, wobei Port 0 immer auf den Standard-Joystick verweist.

Siehe auch **JoyAxisX()**, **JoyAxisY()**, **JoyAxisZ()**, **JoyDir()** und **JoyHat()**.

**EINGABEN**

<b>port</b>	Port-Nummer (normalerweise 0 für den Standard-Joystick-Anschluss)
<b>button</b>	optional: Knopf, nach dem gesucht werden soll, oder 0 für alle Knöpfe (Standardwert ist 1, was bedeutet, dass nach dem ersten, d.h. dem Auslöseknopf, gesucht wird) (V4.6)

**RÜCKGABEWERTE**

<b>pressed</b>	<b>True</b> , wenn der Knopf gedrückt ist, sonst <b>FALSE</b> ; wenn Sie 0 für <b>button</b> übergeben haben, ist dies eine 32-Bit-Maske, die den Zustand aller 32 Knöpfe angibt
----------------	--

**BEISPIEL**

```
While fire = FALSE
    fire = JoyButton(0)
    VWait
Wend
```

Der obige Code wartet, bis der Benutzer auf "Feuer" drückt.

## 32.7 JoyDir

**BEZEICHNUNG**

JoyDir – gibt die Richtung des Joysticks zurück (V1.5)

**ÜBERSICHT**

```
dir = JoyDir(port[, idx])
```

**BESCHREIBUNG**

Dieser Befehl gibt die Richtung des Joysticks zurück, der in den durch **port** angegebenen Port eingesteckt ist. Das optionale Argument **idx** kann verwendet werden, um den Index

des zu verwendenden Joysticks anzugeben, falls es mehrere Joysticks an einem Controller gibt. Joystick-Indizes beginnen bei 0.

Eine der folgenden Stellungen wird zurückgegeben:

```
#JOYUP      Der Joystick wird nach oben gedrückt
#JOYUPRIGHT
              Der Joystick wird nach oben rechts gedrückt
#JOYRIGHT
              Der Joystick wird nach rechts gedrückt
#JOYDOWNRIGHT
              Der Joystick wird nach unten rechts gedrückt
#JOYDOWN    Der Joystick wird nach unten gedrückt
#JOYDOWNLEFT
              Der Joystick wird nach unten links gedrückt
#JOYLEFT    Der Joystick wird nach links gedrückt
#JOYUPLEFT
              Der Joystick wird nach oben links gedrückt
#JOYNODIR
              Nirgends (keine Richtung selektiert)
```

`port` kann von 0 bis zur Anzahl der aktuell angeschlossenen Joysticks minus 1 liegen. Sie können die Anzahl der aktuell verfügbaren Joysticks mit dem Befehl `CountJoysticks()` ermitteln. Bitte beachten Sie, dass Port 0 unter AmigaOS den Standard-Joystick-Port anspricht, obwohl dies Port 1 auf klassischer Amiga-Hardware ist. Hollywood wechselt diese Ports für plattformübergreifende Konsistenz, wobei Port 0 immer auf den Standard-Joystick verweist.

Siehe auch `JoyAxisX()`, `JoyAxisY()`, `JoyAxisZ()`, `JoyButton()` und `JoyHat()`.

## EINGABEN

```
port      Port-Nummer (normalerweise 0 für den Standard-Joystick-Anschluss)
idx       optional: abzufragender Joystick-Index (Standardwert: 0)
```

## RÜCKGABEWERTE

```
dir       aktuelle Joystick-Stellung (eine der Konstanten von oben)
```

## BEISPIEL

```
While state <> #JOYRIGHT
    state = JoyDir(0)
    VWait
Wend
```

Der obige Code wartet, bis der Benutzer den Joystick am Anschluss 0 nach rechts bewegt.

## 32.8 JoyHat

### BEZEICHNUNG

JoyHat – gibt die Stellung des Rundblickschalters zurück (V10.0)

### ÜBERSICHT

```
state = JoyHat(port[, idx])
```

### BESCHREIBUNG

Dieser Befehl gibt die Stellung des Rundblickschalters (auch POV-Schalter (Point Of View), D-Pads, Coolie Hat, Hat Switch oder Steuerkreuz genannt) des Joysticks an dem durch `port` angegebenen Port zurück. Die zurückgegebene Stellung ist -1, wenn sich der Rundblickschalter in der Mitte befindet, ansonsten wird ein Wert zwischen 0 und 27000 zurückgegeben. Das optionale Argument `idx` kann verwendet werden, um den Index des zu verwendenden Rundblickschalters anzugeben, falls es mehrere an einem Controller gibt. Indizes beginnen bei 0.

`port` kann von 0 bis zur Anzahl der aktuell angeschlossenen Joysticks minus 1 liegen. Sie können die Anzahl der aktuell verfügbaren Joysticks mit dem Befehl `CountJoysticks()` ermitteln. Bitte beachten Sie, dass Port 0 unter AmigaOS den Standard-Joystick-Port anspricht, obwohl dies Port 1 auf klassischer Amiga-Hardware ist. Hollywood wechselt diese Ports für plattformübergreifende Konsistenz, wobei Port 0 immer auf den Standard-Joystick verweist.

Siehe auch `JoyAxisX()`, `JoyAxisY()`, `JoyAxisZ()`, `JoyButton()` und `JoyDir()`.

### EINGABEN

<code>port</code>	Port-Nummer (normalerweise 0 für den Standard-Joystick-Anschluss)
<code>idx</code>	optional: abzufragender Joystick-Index (Standardwert: 0)

### RÜCKGABEWERTE

<code>state</code>	Stellung des Rundblickschalters, von -1 (Mitte) bis 27000
--------------------	---

## 33 Konsolenbibliothek

### 33.1 AllocConsoleColor

#### BEZEICHNUNG

AllocConsoleColor – weist die Konsolenfarbe zu (V10.0)

#### ÜBERSICHT

```
col = AllocConsoleColor(color)
```

#### PLATTFORMEN

Linux, macOS, Windows

#### BESCHREIBUNG

Dieser Befehl weist der aktuellen Konsole die durch `color` angegebene Farbe zu und gibt sie zurück. Sie können sie dann zur aktiven Farbe machen, indem Sie sie an Befehle wie `SetConsoleColor()` übergeben. Die Farbzuordnung ist notwendig, da standardmäßig nur wenige vordefinierte ANSI-Farben wie `#BLACK`, `#WHITE`, `#RED` etc. zur Verfügung stehen. Wenn Sie also benutzerdefinierte Farben verwenden möchten, müssen Sie diese zuerst zuweisen.

Wenn Sie mit einer von diesem Befehl zugewiesenen Farbe fertig sind, rufen Sie `FreeConsoleColor()` auf, um die Farbe zu löschen. Dies ist wichtig, um sicherzustellen, dass Ihnen die Farben nicht ausgehen.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

#### EINGABEN

`color`      zuzuweisende Farbe; diese muss als **RGB-Farbe** übergeben werden

#### RÜCKGABEWERTE

`col`      zugewiesene Farbe

#### BEISPIEL

```
EnableAdvancedConsole()
SetConsoleColor(1, AllocConsoleColor($FFA500))
ConsolePrint("Hello World")
RefreshConsole()
```

Der obige Code gibt die Zeichenkette "Hello World" in Orange aus. Beachten Sie, dass Sie unter normalen Umständen die Konsolenfarbe wieder freigeben sollten, wenn Sie damit fertig sind. Dieser Teil wurde aus Gründen der Lesbarkeit weggelassen.

### 33.2 BeepConsole

#### BEZEICHNUNG

BeepConsole – spielt den Console-Piep ab (V10.0)

#### ÜBERSICHT

```
BeepConsole()
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl lässt die akustische Glocke (oder Piep-Ton) auf dem Terminal ertönen. Wenn das nicht möglich ist, ruft er `FlashConsole()` auf.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

keine

### 33.3 ClearConsole

**BEZEICHNUNG**

ClearConsole – löscht die Konsole (Clear wird True) (V10.0)

**ÜBERSICHT**

`ClearConsole()`

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl löscht die Konsole, indem sie den aktuellen Hintergrundzeichensatz mit `SetConsoleBackground()` in jede Zelle des Fensters kopiert. `ClearConsole()` setzt auch das Flag `Clear` des Befehls `SetConsoleOptions()` für das aktuelle Fenster auf `True`, um sicherzustellen, dass das Fenster bei der nächsten Aktualisierung gelöscht wird. Wenn Sie das nicht möchten, verwenden Sie stattdessen den Befehl `EraseConsole()`. Siehe [Abschnitt 33.20 \[EraseConsole\]](#), [Seite 665](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

keine

### 33.4 ClearConsoleStyle

**BEZEICHNUNG**

ClearConsoleStyle – löscht den Konsolenstil (V10.0)

**ÜBERSICHT**

`ClearConsoleStyle(style)`

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl löscht den/die angegebenen Stil(e) im aktuellen Konsolenfenster. Der Parameter `style` kann eines oder mehrere der Konsolenstil-Flags sein, wie es in der Dokumentation `SetConsoleStyle()` beschrieben ist. Siehe [Abschnitt 33.55 \[SetConsoleStyle\]](#), [Seite 690](#), für Details. Da alle Stil-Flags Bitmasken sind, können Sie mehrere Stile mit dem bitweisen OR-Operator (`|`) kombinieren.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

`style`        eine oder mehrere zu löschende Stil-Flags

## 33.5 CloseConsole

**BEZEICHNUNG**

`CloseConsole` – schließt das Konsolenfenster (V10.0)

**ÜBERSICHT**

`CloseConsole()`

**PLATTFORMEN**

Windows

**BESCHREIBUNG**

Damit wird ein Konsolenfenster geschlossen, das zuvor mit `OpenConsole()` geöffnet wurde. Dies wird nur unter Windows unterstützt, da nur Windows zwischen Nicht-Konsolen- und Konsolenprogrammen unterscheidet. Siehe [Abschnitt 33.44 \[OpenConsole\]](#), [Seite 681](#), für Details.

**EINGABEN**

keine

## 33.6 ConsolePrint

**BEZEICHNUNG**

`ConsolePrint` – gibt die Argumente in der Konsole aus (V8.0)

**ÜBERSICHT**

`ConsolePrint(...)`

**BESCHREIBUNG**

Dieser Befehl gibt alle von Ihnen angegebenen Argumente in der Konsole aus. Sie können unbeschränkt viele Argumente angeben, die beliebig sein können. Wenn Sie mehrere Argumente an diesen Befehl übergeben, werden diese mit einem Leerzeichen ausgegeben, um sie voneinander zu trennen.

`ConsolePrint()` fügt automatisch ein Zeilenvorschubzeichen an das Ende der Ausgabe an. Wenn Sie das nicht möchten, verwenden Sie stattdessen `ConsolePrintNR()`. Siehe [Abschnitt 33.7 \[ConsolePrintNR\]](#), [Seite 656](#), für Details.

**EINGABEN**

... mindestens einen Wert, der in der Konsole ausgegeben werden soll

**BEISPIEL**

```
ConsolePrint("The user entered", name$, "as his name and", age,  
            "as his age!")
```

## 33.7 ConsolePrintNR

**BEZEICHNUNG**

ConsolePrintNR – gibt die Argumente in der Konsole ohne Zeilenumbruch aus (V8.0)

**ÜBERSICHT**

```
ConsolePrintNR(...)
```

**BESCHREIBUNG**

Dies funktioniert genauso wie der Befehl `ConsolePrint()`, fügt jedoch kein Zeilenumbruch an die Zeichenkette an.

Siehe [Abschnitt 33.6 \[ConsolePrint\]](#), [Seite 655](#), für Details.

**EINGABEN**

... mindestens einen Wert, der in der Konsole ausgegeben werden soll

**BEISPIEL**

```
ConsolePrintNR("Hello ")  
ConsolePrintNR("World!")  
ConsolePrintNR("\n")
```

Dies entspricht dem Befehl `ConsolePrint("Hello World!")`.

## 33.8 ConsolePrompt

**BEZEICHNUNG**

ConsolePrompt – liest Benutzereingaben von der Konsole (V8.0)

**ÜBERSICHT**

```
s$ = ConsolePrompt(p$)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um den Benutzer zur Eingabe einer Zeichenkette in der Konsole aufzufordern. `ConsolePrompt()` zeigt den in `p$` angegebene Zeichenkette als Eingabeaufforderungstext an und stoppt die Ausführung des Skripts, bis der Benutzer eine Zeichenkette eingegeben und seine Eingabe mit der RETURN-Taste bestätigt hat. Die Zeichenkette wird dann von diesem Befehl zurückgegeben.

**EINGABEN**

`p$` Text, der dem Benutzer gezeigt wird

**RÜCKGABEWERTE**

`s$` Zeichenkette, welche der Benutzer eingegeben hat



**BEISPIEL**

```
name$ = ConsolePrompt("What is your name? ")
age$ = ConsolePrompt("And your age? ")
home$ = ConsolePrompt("Where do you live? ")
ConsolePrint("Your name is", name$, "and you are", age$,
             "years old and live in", home$, "!")
```

Der obige Code veranschaulicht die Verwendung des Befehls `ConsolePrompt()`.

## 33.9 CopyConsoleWindow

**BEZEICHNUNG**

`CopyConsoleWindow` – kopiert den Text aus einem anderen Konsolenfenster (V10.0)

**ÜBERSICHT**

```
CopyConsoleWindow(id[, overlay, srcx, srcy, dstx1, dsty1, dstx2, dsty2])
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl kopiert den Text aus dem durch `id` angegebenen Konsolenfenster in das derzeit aktive Konsolenfenster. Wenn der Parameter `overlay` `True` ist, werden nur Nicht-Leerzeichen kopiert, d.h. alle leere Zeichen werden ignoriert. Wenn die optionalen Argumente nach `overlay` weggelassen werden, werden nur die Zeichen des Quellfensters kopiert, die sich mit dem Zielfenster überschneiden, so dass die Zeichen an derselben physikalischen Position auf dem Bildschirm erscheinen.

Wenn Sie die optionalen Argumente nach `overlay` angeben, müssen sich die beiden Fenster nicht überlappen. Die Argumente `srcx` und `srcy` geben die obere linke Ecke des zu kopierenden Bereichs an. Die Argumente `dstx1`, `dsty1`, `dstx2` und `dsty2` geben den Bereich innerhalb des Zielfensters an, in den kopiert werden soll. Alle Positionen müssen in Zeichen angegeben werden.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

<code>id</code>	ID des Konsolenfensters, aus dem kopiert werden soll
<code>overlay</code>	optional: ob nur Nicht-Leerzeichen übersprungen werden sollen oder nicht (Standardwert <code>False</code> )
<code>srcx1</code>	optional: x-Versatz im Fenster der Quellkonsole
<code>srcy1</code>	optional: y-Versatz im Fenster der Quellkonsole
<code>dstx1</code>	optional: Anfang des x-Versatz im Zielfenster
<code>dsty1</code>	optional: Anfang des y-Versatz im Zielfenster
<code>dstx2</code>	optional: Ende des x-Versatz im Zielfenster
<code>dsty2</code>	optional: Ende des y-Versatz im Zielfenster

## 33.10 CreateConsoleWindow

### BEZEICHNUNG

CreateConsoleWindow – erstellt ein neues Konsolenfenster (V10.0)

### ÜBERSICHT

```
[id] = CreateConsoleWindow(id, cols, rows, x, y[, parent, rel])
```

### PLATTFORMEN

Linux, macOS, Windows

### BESCHREIBUNG

Dieser Befehl erstellt ein neues Fenster mit der angegebenen Anzahl von Spalten und Zeilen. Die obere linke Ecke des Fensters befindet sich an den durch `x` und `y` angegebenen Koordinaten. Um ein neues Vollbildfenster zu erstellen, setzen Sie einfach `cols`, `rows`, `x` und `y` alle auf 0. Das neue Fenster erhält den in `id` angegebenen Identifikator. Wenn Sie `Nil` in `id` übergeben, wählt `CreateConsoleWindow()` automatisch eine ID aus und gibt sie zurück.

Wenn Sie das Argument `parent` auf den Identifikator eines vorhandenen Konsolenfensters setzen, wird das angegebene Fenster als übergeordnetes Fenster des neuen Konsolenfensters festgelegt, so dass das neue Konsolenfenster zu einem Unterfenster wird. Sie können `-1` in `parent` übergeben, um das standardmäßige Konsolenfenster als übergeordnetes Fenster festzulegen. Wenn Sie `rel` auf `True` setzen, werden die `x`- und `y`-Koordinaten als relativ zum Ursprung des übergeordneten Fensters statt relativ zum Ursprung des Bildschirms interpretiert.

Nachdem Sie das Konsolenfenster erstellt haben, können Sie es mit `SelectConsoleWindow()` zum aktiven Konsolenfenster machen. Siehe [Abschnitt 33.49 \[SelectConsoleWindow\]](#), Seite 685, für Details.

Beachten Sie, dass ein Konsolenfenster nicht dasselbe ist wie ein Fenster auf Ihrem Desktop. Ein Konsolenfenster ist lediglich ein bestimmter Bereich in der Konsole, der als eigenes Fenster betrachtet wird. Dies erleichtert das Erstellen von textuellen Benutzeroberflächen, da Sie Ihren Konsolenbildschirm in mehrere Fenster aufteilen können und dann alle Zeichnungen automatisch an den Fensterrändern abgeschnitten werden.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), Seite 664, für Details.

### EINGABEN

<code>id</code>	ID für das Konsolenfenster oder <code>Nil</code> für die automatische ID-Zuweisung
<code>cols</code>	Anzahl der Spalten für das Fenster
<code>rows</code>	Anzahl der Zeilen für das Fenster
<code>x</code>	Versatz des linken Fensters
<code>y</code>	Versatz des oberen Fensters
<code>parent</code>	optional: Identifikator eines Konsolenfensters, das als übergeordnetes Fenster verwendet werden soll (Voreingestellt ist <code>-1</code> , was bedeutet, dass der Bildschirm das übergeordnete Fenster ist)

`rel` optional: ob x und y relativer Versatz angegeben wird oder nicht (Standardwert: `False`)

## RÜCKGABEWERTE

`id` optional: Identifikator des neuen Konsolenfensters; wird nur zurückgegeben, wenn Sie `Nil` als Argument 1 übergeben (siehe oben)

## BEISPIEL

```
EnableAdvancedConsole()
w, h = GetConsoleSize()
CreateConsoleWindow(1, 20, 20, (w - 20) / 2, (h - 20) / 2)
SelectConsoleWindow(1)
DrawConsoleBorder()
RefreshConsole()
```

Der obige Code erstellt ein neues 20x20-Fenster, zieht einen Rahmen darum und zentriert es auf dem Bildschirm.

## 33.11 DecomposeConsoleChr

### BEZEICHNUNG

`DecomposeConsoleChr` – extrahiert ein formatiertes Zeichen (V10.0)

### ÜBERSICHT

```
ch, style, pen = DecomposeConsoleChr(c)
```

### PLATTFORMEN

Linux, macOS, Windows

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die einzelnen Komponenten eines formatierten Zeichens zu extrahieren, das von `MakeConsoleChr()` zusammengesetzt wurde. Sie müssen das formatierte Zeichen im Parameter `c` übergeben und `DecomposeConsoleChr()` gibt den Unicode-Codepunkt des Zeichens, die Format-Flags und den Stift zurück, der zum Zeichnen des Zeichens verwendet werden soll. Siehe [Abschnitt 33.42 \[MakeConsoleChr\]](#), [Seite 678](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

### EINGABEN

`c` formatiertes Zeichen, wie von `MakeConsoleChr()` zusammengestellt

## RÜCKGABEWERTE

`ch` Unicode-Codepunkt des Zeichens

`style` Konsolen-Format-Flags

`pen` Stift, der verwendet werden soll, um zu zeichnen

## 33.12 DeleteConsoleChr

### BEZEICHNUNG

DeleteConsoleChr – löscht ein Konsolenzeichen (V10.0)

### ÜBERSICHT

DeleteConsoleChr()

### PLATTFORMEN

Linux, macOS, Windows

### BESCHREIBUNG

Dieser Befehl löscht das Zeichen unter dem Cursor im aktuellen Fenster. Alle Zeichen rechts in derselben Zeile werden um eine Position nach links verschoben und das letzte Zeichen in der Zeile wird mit einem Leerzeichen aufgefüllt. Die Cursorposition wird nicht verändert.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

### EINGABEN

keine

## 33.13 DeleteConsoleLine

### BEZEICHNUNG

DeleteConsoleLine – löscht eine Konsolenzeile (V10.0)

### ÜBERSICHT

DeleteConsoleLine()

### PLATTFORMEN

Linux, macOS, Windows

### BESCHREIBUNG

Dieser Befehl löscht die Zeile unter dem Cursor im aktuellen Fenster. Alle Zeilen unterhalb der aktuellen Zeile werden um eine Zeile nach oben verschoben. Die unterste Zeile des Fensters wird geleert. Die Cursorposition ändert sich nicht.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

### EINGABEN

keine

## 33.14 DisableAdvancedConsole

### BEZEICHNUNG

DisableAdvancedConsole – beendet den erweiterten Konsolenmodus (V10.0)

**ÜBERSICHT**`DisableAdvancedConsole()`**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl beendet den erweiterten Konsolenmodus und versetzt die Konsole wieder in den normalen Modus. Das Beenden des erweiterten Konsolenmodus löscht auch die Konsole und stellt den ursprünglichen Inhalt wieder her, bevor der erweiterte Modus auf der Konsole gestartet wurde.

Um den erweiterten Konsolenmodus zu starten, rufen Sie den Befehl `EnableAdvancedConsole()` auf.

**EINGABEN**

keine

## 33.15 DrawConsoleBorder

**BEZEICHNUNG**`DrawConsoleBorder` – zeichnet einen Rahmen um die Konsole (V10.0)**ÜBERSICHT**`DrawConsoleBorder([ls, rs, ts, bs, tl, tr, bl, br])`**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl zeichnet einen Rahmen um den Rand des aktuellen Konsolenfensters. Alle Argumente müssen Zeichencodes sein, die das Zeichen angeben, das an der jeweiligen Position stehen soll. Der Parameter `ls` gibt das Zeichen an, das auf der linken Seite des Rahmens gezeichnet werden soll, `rs` gibt die rechte Seite des Rahmens an, `ts` die obere Seite und `bs` die untere Seite. Der Parameter `tl` gibt das Zeichen an, das in der oberen linken Ecke des Rahmens gezeichnet werden soll, `tr` gibt die obere rechte Ecke des Rahmens an, `bl` die untere linke Ecke und `br` die untere rechte Ecke. Wenn ein Parameter 0 ist, verwendet `DrawConsoleBorder()` sein Standardzeichen für die angegebene Rahmenposition.

Zeichen müssen als numerische Werte übergeben werden, nicht als Zeichenketten. Bei normalen Zeichen ist dieser Wert einfach der Unicode-Codepunkt des jeweiligen Zeichens, z.B. 65 für 'A'. Sie können jedoch auch einen Sonderzeichencode übergeben, der von dem Befehl `MakeConsoleChr()` zusammengesetzt wird. Dieser Befehl ermöglicht es Ihnen, Textformatierungsstile in den Zeichencode einzufügen und unterstützt auch spezielle Zeichencodes wie Pfeile oder Rahmenteile. Siehe [Abschnitt 33.42 \[MakeConsoleChr\]](#), [Seite 678](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

<b>ls</b>	optional: Zeichen für die linke Seite des Rahmens (Standardwert: 0)
<b>rs</b>	optional: Zeichen für die rechte Seite des Rahmens (Standardwert: 0)
<b>ts</b>	optional: Zeichen für die obere Seite des Rahmens (Standardwert: 0)
<b>bs</b>	optional: Zeichen für die untere Seite des Rahmens (Standardwert: 0)
<b>tl</b>	optional: Zeichen für die obere linke Ecke des Rahmens (Standardwert: 0)
<b>tr</b>	optional: Zeichen für die obere rechte Ecke des Rahmens (Standardwert: 0)
<b>bl</b>	optional: Zeichen für die untere linke Ecke des Rahmens (Standardwert: 0)
<b>br</b>	optional: Zeichen für die untere rechte Ecke des Rahmens (Standardwert: 0)

**BEISPIEL**

```
EnableAdvancedConsole()
DrawConsoleBorder()
RefreshConsole()
```

Der obige Code zeichnet einen Rahmen mit den Standardzeichen um das aktuelle Konsolenfenster.

## 33.16 DrawConsoleBox

**BEZEICHNUNG**

`DrawConsoleBox` – zeichnet ein Rechteck um die Konsole (V10.0)

**ÜBERSICHT**

```
DrawConsoleBox([xc, yc])
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl zeichnet ein Rechteck um den Rand des aktuellen Konsolenfensters. Die Argumente `xc` und `yc` müssen Zeichencodes sein, die das Zeichen angeben, das für die horizontalen und vertikalen Linien verwendet werden soll. Wenn ein Parameter 0 ist, verwendet `DrawConsoleBox()` sein Standardzeichen für die angegebene Rahmenposition.

Zeichen müssen als numerische Werte übergeben werden, nicht als Zeichenketten. Bei normalen Zeichen ist dieser Wert einfach der Unicode-Codepunkt des jeweiligen Zeichens, z.B. 65 für 'A'. Sie können jedoch auch einen Sonderzeichencode übergeben, der von dem Befehl `MakeConsoleChr()` zusammengesetzt wird. Dieser Befehl ermöglicht es Ihnen, Textformatierungsstile in den Zeichencode einzufügen und unterstützt auch spezielle Zeichencodes wie Pfeile oder Rahmenteile. Siehe [Abschnitt 33.42 \[MakeConsoleChr\]](#), [Seite 678](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

- `xc` optional: Zeichen für horizontalen Rahmen (Standardwert: 0)  
`yc` optional: Zeichen für vertikalen Rahmen (Standardwert 0)

**BEISPIEL**

```
EnableAdvancedConsole()  
DrawConsoleBox()  
RefreshConsole()
```

Der obige Code zeichnet einen Rahmen mit den Standardzeichen um das aktuelle Konsolenfenster.

## 33.17 DrawConsoleHLine

**BEZEICHNUNG**

`DrawConsoleHLine` – zeichnet eine horizontale Linie in die Konsole (V10.0)

**ÜBERSICHT**

```
DrawConsoleHLine([ch, n])
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl zeichnet eine horizontale Linie mit dem in `ch` angegebenen Zeichencode. Standardmäßig wird die Zeile bis zum Ende des Fensters verlängert. Alternativ können Sie den Parameter `n` verwenden, um `DrawConsoleHLine()` mitzuteilen, wie viele Zeichen gezeichnet werden sollen. Durch das Übergeben von 0 in `ch` verwendet `DrawConsoleHLine()` ein Standardzeichen. Die Cursorposition wird durch diesen Befehl nicht vorgerückt.

Zeichen müssen als numerische Werte übergeben werden, nicht als Zeichenketten. Bei normalen Zeichen ist dieser Wert einfach der Unicode-Codepunkt des jeweiligen Zeichens, z.B. 65 für 'A'. Sie können jedoch auch einen Sonderzeichencode übergeben, der von dem Befehl `MakeConsoleChr()` zusammengesetzt wird. Dieser Befehl ermöglicht es Ihnen, Textformatierungsstile in den Zeichencode einzufügen und unterstützt auch spezielle Zeichencodes wie Pfeile oder Rahmenteile. Siehe [Abschnitt 33.42 \[MakeConsoleChr\]](#), [Seite 678](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

- `ch` optional: Zeichen, mit dem gezeichnet werden soll (Standardwert: 0)  
`n` optional: Anzahl der zu zeichnenden Zeichen (Standardwert ist die Anzahl der Spalten)

## 33.18 DrawConsoleVLine

### BEZEICHNUNG

DrawConsoleVLine – zeichnet eine vertikale Linie in die Konsole (V10.0)

### ÜBERSICHT

`DrawConsoleVLine([ch, n])`

### PLATTFORMEN

Linux, macOS, Windows

### BESCHREIBUNG

Dieser Befehl zeichnet eine vertikale Linie unter Verwendung des in `ch` angegebenen Zeichencodes. Standardmäßig wird die Linie bis zum Ende des Fensters verlängert. Alternativ können Sie den Parameter `n` verwenden, um `DrawConsoleVLine()` mitzuteilen, wie viele Zeichen gezeichnet werden sollen. Durch das Übergeben von 0 in `ch` verwendet `DrawConsoleVLine()` ein Standardzeichen. Die Cursorposition wird durch diesen Befehl nicht verändert.

Zeichen müssen als numerische Werte übergeben werden, nicht als Zeichenketten. Bei normalen Zeichen ist dieser Wert einfach der Unicode-Codepunkt des jeweiligen Zeichens, z.B. 65 für 'A'. Sie können jedoch auch einen Sonderzeichencode übergeben, der von dem Befehl `MakeConsoleChr()` zusammengesetzt wird. Dieser Befehl ermöglicht es Ihnen, Textformatierungsstile in den Zeichencode einzufügen und unterstützt auch spezielle Zeichencodes wie Pfeile oder Rahmenteile. Siehe [Abschnitt 33.42 \[MakeConsoleChr\]](#), [Seite 678](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

### EINGABEN

<code>ch</code>	optional: Zeichen, mit dem gezeichnet werden soll (Standardwert: 0)
<code>n</code>	optional: Anzahl der zu zeichnenden Zeichen (Standardwert ist die Anzahl der Zeilen)

## 33.19 EnableAdvancedConsole

### BEZEICHNUNG

EnableAdvancedConsole – versetzt die Konsole in den erweiterten Modus (V10.0)

### ÜBERSICHT

`EnableAdvancedConsole()`

### PLATTFORMEN

Linux, macOS, Windows

### BESCHREIBUNG

Dieser Befehl löscht den Konsolenbildschirm und versetzt die Konsole in den erweiterten Modus. Die meisten Befehle der Konsolenbibliothek erfordern, dass sich die Konsole im erweiterten Modus befindet, da Möglichkeiten wie freie Cursorpositionierung, Farben,



Eingabebehandlung usw. im Standardkonsolenmodus nicht verfügbar sind. Daher ist `EnableAdvancedConsole()` normalerweise der erste Befehl, der aufgerufen wird, wenn Sie erweiterte Kontrolle über die Konsole haben möchten.

Um zum normalen Konsolenmodus zurückzukehren, rufen Sie den Befehl `DisableAdvancedConsole()` auf.

Bitte lesen Sie auch das Kapitel über [Hollywood und die Konsole](#), um mehr über die Verwendung von Hollywood im Konsolenmodus zu erfahren. Siehe [Abschnitt 3.1 \[Konsolenmodus\]](#), [Seite 31](#), für Details.

## EINGABEN

keine

## BEISPIEL

```
EnableAdvancedConsole()
w, h = GetConsoleSize()
s$ = "Hello World!"
SetConsoleCursor((w - StrLen(s$)) / 2, h / 2)
ConsolePrintNR(s$)
RefreshConsole()
```

Der obige Code gibt die Zeichenkette "Hello World" zentriert in der Konsole aus.

## 33.20 EraseConsole

### BEZEICHNUNG

`EraseConsole` – löscht die Konsole (`Clear` wird nicht `True`) (V10.0)

### ÜBERSICHT

`EraseConsole()`

### PLATTFORMEN

Linux, macOS, Windows

### BESCHREIBUNG

Dieser Befehl löscht die Konsole, indem sie den aktuellen Hintergrundzeichensatz mit `SetConsoleBackground()` in jede Zelle des Fensters kopiert. Im Gegensatz zu `ClearConsole()` setzt `EraseConsole()` das Flag `Clear` des Befehls `SetConsoleOptions()` für das aktuelle Fenster nicht auf `True`. Wenn Sie das möchten, verwenden Sie stattdessen `ClearConsole()`. Siehe [Abschnitt 33.3 \[ClearConsole\]](#), [Seite 654](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

## EINGABEN

keine

## 33.21 FlashConsole

### BEZEICHNUNG

FlashConsole – lässt die Konsole aufblitzen (V10.0)

### ÜBERSICHT

FlashConsole()

### PLATTFORMEN

Linux, macOS, Windows

### BESCHREIBUNG

Mit diesem Befehl wird der Bildschirm aufgeblitzt, indem der Vorder- und Hintergrund jeder Zelle invertiert wird, eine Pause eingelegt wird und dann die ursprünglichen Attribute wiederhergestellt werden.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

### EINGABEN

keine

## 33.22 FormatConsoleLine

### BEZEICHNUNG

FormatConsoleLine – legt den Stil und die Farbe für mehrere Zeichen fest (V10.0)

### ÜBERSICHT

FormatConsoleLine(*n*, *style*[, *pen*, *fgcolor*, *bgcolor*])

### PLATTFORMEN

Linux, macOS, Windows

### BESCHREIBUNG

Dieser Befehl legt den angegebenen Stil und die Farbe für die nächsten *n* Zeichen in der aktuellen Zeile fest, ohne den vorhandenen Text zu ändern oder die vorhandenen Stil- oder Farbeinstellungen zu ändern. Das Übergeben von -1 für *n* bedeutet, dass das Format auf alle Zeichen bis zum Ende der aktuellen Zeile angewendet wird.

Alle vom Befehl `SetConsoleStyle()` unterstützten Konsolenstile können im Parameter *style* übergeben werden. Siehe [Abschnitt 33.55 \[SetConsoleStyle\]](#), [Seite 690](#), für Details. Wenn Sie keine Stileinstellungen ändern möchten, können Sie den speziellen Stil `#CONSOLESTYLE_NONE` für *style* übergeben. In diesem Fall wendet `FormatConsoleLine()` keine Stile an.

Mit den optionalen Parametern *pen*, *fgcolor* und *bgcolor* kann die Farbe der Zeichen geändert werden. Sie können genauso verwendet werden wie bei `SetConsoleColor()`. Siehe [Abschnitt 33.52 \[SetConsoleColor\]](#), [Seite 687](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

<b>n</b>	Anzahl der zu formatierenden Zeichen oder -1 für alle Zeichen in der aktuellen Zeile
<b>style</b>	Formatierungsstil, der angewendet werden soll, oder <code>#CONSOLESTYLE_NONE</code> , um das Argument zu ignorieren
<b>pen</b>	optional: zu verwendender Farbstift oder 0, um die Textfarbe auf den Standardwert zurückzusetzen (Standardwert ist 0)
<b>fgcolor</b>	optional: gewünschte Vordergrundfarbe für den Stift; dies muss entweder eine <b>RGB-Farbe</b> aus der obigen Liste oder eine von <code>AllocConsoleColor()</code> zugewiesene Farbe sein (voreingestellt ist <code>#NOCOLOR</code> )
<b>bgcolor</b>	optional: gewünschte Hintergrundfarbe für den Stift; dies muss entweder eine <b>RGB-Farbe</b> aus der obigen Liste oder eine von <code>AllocConsoleColor()</code> zugewiesene Farbe sein (standardmäßig <code>#NOCOLOR</code> )

**33.23 FreeConsoleColor****BEZEICHNUNG**

`FreeConsoleColor` – löscht die Farbe der Konsole (V10.0)

**ÜBERSICHT**

`FreeConsoleColor(color)`

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl löscht die von `AllocConsoleColor()` zugewiesene Konsolenfarbe. Es ist wichtig, diesen Befehl aufzurufen, wenn Sie eine Farbe nicht mehr benötigen, um sicherzustellen, dass Ihnen die Farben nicht ausgehen.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

<b>color</b>	die zu löschende Farbe; diese muss mit <code>AllocConsoleColor()</code> zugewiesen worden sein
--------------	--

**33.24 FreeConsoleWindow****BEZEICHNUNG**

`FreeConsoleWindow` – löscht ein Konsolenfenster (V10.0)

**ÜBERSICHT**

`FreeConsoleWindow(id)`

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl löscht das durch `id` angegebene Konsolenfenster. Dieses muss zuvor von `CreateConsoleWindow()` zugewiesen worden sein. Siehe [Abschnitt 33.10 \[CreateConsoleWindow\]](#), [Seite 658](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

`id` ID des zu löschenden Konsolenfensters

## 33.25 GetAllocConsoleColor

**BEZEICHNUNG**

`GetAllocConsoleColor` – gibt die zugewiesene Farbe zurück (V10.0)

**ÜBERSICHT**

```
color = GetAllocConsoleColor(c)
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl gibt die **RGB-Farbe** einer mit `AllocConsoleColor()` zugewiesenen Konsolenfarbe zurück. Sie müssen die von `AllocConsoleColor()` zugewiesene Farbe im Parameter `c` übergeben.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

`c` zugewiesene Farbe, die ermittelt werden soll

**RÜCKGABEWERTE**

`color` zugewiesene Farbe als **RGB-Farbe**

## 33.26 GetConsoleBackground

**BEZEICHNUNG**

`GetConsoleBackground` – gibt das Zeichen vom Konsolenhintergrund zurück (V10.0)

**ÜBERSICHT**

```
ch = GetConsoleBackground()
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl gibt das aktuell für die Hintergrundfüllung verwendete Zeichen zurück. Dieses Zeichen kann mit `SetConsoleBackground()` gesetzt werden. Beachten Sie, dass das Zeichen auch ein Sonderzeichen sein kann, das vom Befehl `MakeConsoleChr()` erstellt wurde. Um solche Zeichen zu extrahieren, können Sie den Befehl `DecomposeConsoleChr()` verwenden. Siehe [Abschnitt 33.11 \[DecomposeConsoleChr\]](#), Seite 659, für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), Seite 664, für Details.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`ch`            das zum Ausfüllen verwendete Zeichen

## 33.27 GetConsoleChr

**BEZEICHNUNG**

`GetConsoleChr` – gibt das Zeichen an der aktuellen Cursorposition zurück (V10.0)

**ÜBERSICHT**

```
ch = GetConsoleChr()
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl ruft das Zeichen an der aktuellen Cursorposition ab und gibt es zurück. Beachten Sie, dass das Zeichen auch ein Sonderzeichen sein kann, das mit dem Befehl `MakeConsoleChr()` erstellt wurde. Um solche Zeichen zu extrahieren, können Sie den Befehl `DecomposeConsoleChr()` verwenden. Siehe [Abschnitt 33.11 \[DecomposeConsoleChr\]](#), Seite 659, für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), Seite 664, für Details.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`ch`            Zeichen an der aktuellen Cursorposition

## 33.28 GetConsoleColor

**BEZEICHNUNG**

`GetConsoleColor` – gibt die Farbe der Konsole zurück (V10.0)

**ÜBERSICHT**

```
pen, fgcolor, bgcolor = GetConsoleColor([cursor])
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl gibt die aktive Farbe des aktuellen Konsolenfensters zurück. Wenn das optionale Argument `cursor` auf `True` gesetzt ist, wird die Farbe an der aktuellen Cursorposition zurückgegeben. Andernfalls wird die Farbe des aktuellen Fensters zurückgegeben.

`GetConsoleColor()` gibt drei Werte zurück: Den derzeit aktiven Stift, die aktuellen Vorder- und Hintergrundfarben.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

`cursor` optional: `True`, um die Farbe an der Cursorposition zu erhalten, `False`, um die Farbe des aktuellen Fensters zu ermitteln (Voreingestellt ist `False`)

**RÜCKGABEWERTE**

`pen` aktueller Farbstift

`fgcolor` aktuelle Vordergrundfarbe als [RGB-Farbe](#)

`bgcolor` aktuelle Hintergrundfarbe als [RGB-Farbe](#)

## 33.29 GetConsoleControlChr

**BEZEICHNUNG**

`GetConsoleControlChr` – gibt ein Standardsteuerzeichen zurück (V10.0)

**ÜBERSICHT**

```
ch = GetConsoleControlChr(ctrl)
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl gibt den Zeichencode eines Standard-Konsolensteuerzeichens zurück. Folgende Steuerzeichen werden derzeit unterstützt und können im Parameter `ctrl` übergeben werden:

`#CONSOLECHR_KILL:`

Gibt das KILL-Zeichen zurück.

`#CONSOLECHR_ERASE:`

Gibt das ERASE-Zeichen zurück.

`#CONSOLECHR_WORD:`

Gibt das DELETEWORD-Zeichen zurück.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

`ctrl` zu erhaltendes Steuerzeichen; muss eines aus der obigen Liste sein

**RÜCKGABEWERTE**

`ch` das angeforderte Steuerzeichen im Format des aktuellen Terminals

### 33.30 GetConsoleCursor

**BEZEICHNUNG**

`GetConsoleCursor` – gibt die Position des Konsolencursors zurück (V10.0)

**ÜBERSICHT**

```
x, y = GetConsoleCursor()
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl gibt die Position des Cursors im aktuellen Fenster zurück. Die zurückgegebene Position ist relativ zur oberen linken Ecke des Fensters, die (0,0) ist.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`x` aktuelle x-Position

`y` aktuelle y-Position

### 33.31 GetConsoleOrigin

**BEZEICHNUNG**

`GetConsoleOrigin` – gibt die Ursprungskoordinaten der Konsole zurück (V10.0)

**ÜBERSICHT**

```
x, y = GetConsoleOrigin()
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl gibt die x- und y-Ursprungskoordinaten des aktuellen Konsolenfensters zurück.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

keine

**RÜCKGABEWERTE**

x            x-Ursprungsposition

y            y-Ursprungsposition

### 33.32 GetConsoleSize

**BEZEICHNUNG**

GetConsoleSize – gibt die Konsolenabmessungen zurück (V10.0)

**ÜBERSICHT**

```
cols, rows = GetConsoleSize()
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl gibt die Größe des aktuellen Fensters zurück. Die Größe wird als Anzahl der im aktuellen Fenster verfügbaren Spalten und Zeilen zurückgegeben. Eine typische Terminalgröße ist 80x24, also 24 Zeilen mit 80 Zeichen pro Zeile.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

keine

**RÜCKGABEWERTE**

cols        Anzahl der Spalten im Fenster

rows        Anzahl der Zeilen im Fenster

### 33.33 GetConsoleStr

**BEZEICHNUNG**

GetConsoleStr – liest eine Zeichenkette von der Konsole (V10.0)

**ÜBERSICHT**

```
s$ = GetConsoleStr()
```

**PLATTFORMEN**

Linux, macOS, Windows



**BESCHREIBUNG**

Dieser Befehl liest alle Zeichen ab der aktuellen Cursorposition bis zum Zeilenende und gibt sie als Zeichenkette zurück.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`s$` die von der Konsole gelesene Zeichenkette

### 33.34 GetConsoleStyle

**BEZEICHNUNG**

`GetConsoleStyle` – gibt den Konsolenstil zurück (V10.0)

**ÜBERSICHT**

```
style = GetConsoleStyle([cursor])
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl gibt den Stil des aktuellen Konsolenfensters zurück. Wenn das optionale Argument `cursor` auf `True` gesetzt ist, wird der Stil an der aktuellen Cursorposition zurückgegeben. Andernfalls wird der Stil für das aktuelle Fenster ermittelt.

Der Rückgabewert `style` ist eine Bitmaske von Style-Flags. Siehe [Abschnitt 33.55 \[SetConsoleStyle\]](#), [Seite 690](#), für alle verfügbaren Konsolenstile.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

`cursor` optional: `True`, um den Stil an der Cursorposition zu ermitteln, `False`, um den Stil für das aktuelle Fenster zu erhalten (Standardwert ist `False`)

**RÜCKGABEWERTE**

`style` Stil-Flags

### 33.35 GetConsoleWindow

**BEZEICHNUNG**

`GetConsoleWindow` – gibt das aktive Konsolenfenster zurück (V10.0)

**ÜBERSICHT**

```
id = GetConsoleWindow()
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl gibt das derzeit aktive Konsolenfenster zurück. Wenn kein Konsolenfenster mit `SelectConsoleWindow()` aktiviert wurde, wird -1 zurückgegeben.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`id` ID des aktiven Konsolenfensters oder -1, wenn der Standardbildschirm aktiv ist

### 33.36 HaveConsole

**BEZEICHNUNG**

HaveConsole – prüft, ob eine Konsole vorhanden ist (V10.0)

**ÜBERSICHT**

`ok = HaveConsole()`

**BESCHREIBUNG**

Dieser Befehl gibt `True` zurück, wenn das Programm eine angehängte Konsole hat. Dies ist nur von Nutzen, wenn Sie die Nicht-Konsolenversion von Hollywood unter Windows verwenden. In diesem Fall gibt es zunächst keine Konsole, sondern sie muss manuell mit `OpenConsole()` geöffnet werden. Auf allen anderen Plattformen und in der Konsolenversion von Hollywood unter Windows gibt es immer eine Konsole, sodass dieser Befehl immer `True` zurückgibt. Siehe [Abschnitt 33.44 \[OpenConsole\]](#), [Seite 681](#), für Details.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`ok` `True`, wenn eine Konsole verfügbar ist, sonst `False`

### 33.37 HideConsoleCursor

**BEZEICHNUNG**

HideConsoleCursor – blendet den Konsolencursor aus (V10.0)

**ÜBERSICHT**

`HideConsoleCursor()`

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl blendet den Konsolencursor aus.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

keine

## 33.38 InitConsoleColor

**BEZEICHNUNG**

`InitConsoleColor` – legt die Hinter- und Vordergrundfarbe fest (V10.0)

**ÜBERSICHT**

```
InitConsoleColor(pen[, fgcolor, bgcolor])
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Mit diesem Befehl kann die Vordergrundfarbe des durch `pen` festgelegten Konsolenstifts auf `fgcolor` und die Hintergrundfarbe auf `bgcolor` gesetzt werden. Der Stift kann dann mit `SetConsoleColor()` als Textstift eingestellt werden. Siehe [Abschnitt 33.52 \[SetConsoleColor\]](#), [Seite 687](#), für Details.

Der angegebene Stift muss größer als 0 sein. Die Anzahl der verfügbaren Stifte hängt von der Konsole ab. Normalerweise sind 256 Stifte verfügbar, was bedeutet, dass die höchste Stiftnummer, die Sie verwenden können, 255 ist, aber für die beste Kompatibilität sollten Sie niedrigere Stiftnummern verwenden, da möglicherweise nicht alle Konsolen 256 Stifte haben.

Die Argumente `fgcolor` und `bgcolor` können entweder **RGB-Werte** oder Farben sein, die mit `AllocConsoleColor()` zugewiesen werden. Beachten Sie, dass standardmäßig nur wenige Farben verfügbar sind und ohne Zuordnung in `fgcolor` oder `bgcolor` übergeben werden können. Diese sind: `#BLACK`, `#WHITE`, `#RED`, `#GREEN`, `#BLUE`, `#YELLOW`, `#AQUA` (Cyan) und `#FUCHSIA` (Magenta). Wenn Sie andere Farben verwenden möchten, müssen Sie diese zuerst mit `AllocConsoleColor()` zuweisen. Siehe [Abschnitt 33.1 \[AllocConsoleColor\]](#), [Seite 653](#), für Details.

Beachten Sie, dass Terminals möglicherweise eine abgedunkelte Version der hier angegebenen Farben verwenden. Seien Sie also nicht überrascht, wenn Ihre Farbe grau erscheint, obwohl Sie Weiß angegeben haben. Viele Terminals sind so konfiguriert, dass sie Grau als Weiß behandeln. Wenn Sie dies vermeiden möchten, weisen Sie benutzerdefinierte Farben mit `AllocConsoleColor()` zu.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

`pen`            der zu verwendende Farbstift

- fgcolor** optional: gewünschte Vordergrundfarbe; dies muss entweder eine **RGB-Farbe** aus der obigen Liste oder eine durch **AllocConsoleColor()** zugewiesene Farbe sein (Standardwert ist **#NOCOLOR**)
- bgcolor** optional: gewünschte Hintergrundfarbe; dies muss entweder eine **RGB-Farbe** aus der obigen Liste oder eine durch **AllocConsoleColor()** zugewiesene Farbe sein (Standardwert ist **#NOCOLOR**)

**BEISPIEL**

```
EnableAdvancedConsole()
SetConsoleStyle(#CONSOLESTYLE_BOLD)
InitConsoleColor(1, #BLACK, #WHITE)
SetConsoleColor(1)
ConsolePrint("Hello World")
RefreshConsole()
```

Der obige Code gibt die Zeichenkette "Hello World" in Schwarz auf weißem Hintergrund aus.

**33.39 InsertConsoleChr****BEZEICHNUNG**

InsertConsoleChr – fügt ein Konsolenzeichen ein (V10.0)

**ÜBERSICHT**

```
InsertConsoleChr(ch)
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl fügt das durch **ch** angegebene Zeichen an der aktuellen Cursorposition in das aktuelle Konsolenfenster ein. Alle Zeichen rechts vom Cursor werden nach rechts verschoben, wobei die Möglichkeit besteht, dass die Zeichen ganz rechts in der Zeile verloren gehen. Der Cursor wird nicht vorgerückt.

Das Zeichen muss als numerischer Wert übergeben werden, nicht als Zeichenkette. Für normale Zeichen gibt **ch** einfach den Unicode-Codepunkt des jeweiligen Zeichens an, z.B. 65 für 'A'. Das Argument **ch** kann jedoch auch ein Sonderzeichencode sein, der vom Befehl **MakeConsoleChr()** zusammengesetzt wird. Mit diesem Befehl können Sie Textformatierungsstile in den Zeichencode einfügen und sie unterstützt auch spezielle Zeichencodes wie Pfeile oder Rahmenbits. Siehe [Abschnitt 33.42 \[MakeConsoleChr\]](#), [Seite 678](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit **EnableAdvancedConsole()** aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

**ch** das einzufügende Zeichen

**BEISPIEL**

```
EnableAdvancedConsole()  
InsertConsoleChr('A')  
RefreshConsole()
```

Der obige Code fügt das Zeichen 'A' in die Konsole ein.

## 33.40 InsertConsoleLine

**BEZEICHNUNG**

InsertConsoleLine – fügt eine Konsolenzeile ein (V10.0)

**ÜBERSICHT**

```
InsertConsoleLine()
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl fügt eine Leerzeile oberhalb der aktuellen Zeile ein. Die untere Zeile geht dabei verloren.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

keine

## 33.41 InsertConsoleStr

**BEZEICHNUNG**

InsertConsoleStr – fügt eine Konsolenzeichenkette ein (V10.0)

**ÜBERSICHT**

```
InsertConsoleStr(s$)
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl fügt die durch `s$` angegebenen Zeichenkette an der aktuellen Cursorposition in das aktuelle Konsolenfenster ein. Alle Zeichen rechts vom Cursor werden nach rechts verschoben, wobei die Möglichkeit besteht, dass die Zeichen ganz rechts in der Zeile verloren gehen. Der Cursor wird nicht vorgerückt.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

`s$`            die einzufügende Zeichenkette

**BEISPIEL**

```
EnableAdvancedConsole()
InsertConsoleStr("Hello World!")
RefreshConsole()
```

Der obige Code gibt die Zeichenkette "Hello World!" auf der Konsole aus.

**33.42 MakeConsoleChr****BEZEICHNUNG**

MakeConsoleChr – fügt den Stil und die Farbe in den Zeichencode ein (V10.0)

**ÜBERSICHT**

```
ch = MakeConsoleChr(c[, style, pen])
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um einen formatierten Zeichencode zu erstellen, der an alle Befehle übergeben werden kann, die einen numerischen Zeichencode akzeptieren, z.B. `InsertConsoleChr()`. `MakeConsoleChr()` nimmt das in `c` übergebene Zeichen und wendet die in `style` übergebenen Stileinstellungen und die in `pen` übergebenen Farbeinstellungen des Stifts darauf an.

Das Zeichen muss von seinem numerischen Unicode-Codepunkt übergeben werden, nicht als Zeichenkette. Für das Zeichen "A" müssten Sie also in `c` den Wert 65 übergeben. Alternativ können Sie in `c` auch eines der folgenden Sonderzeichen übergeben:

```
#CONSOLECHR_BLOCK:
    Voller Block

#CONSOLECHR_BOARD:
    Schachbrettmuster

#CONSOLECHR_BTEE:
    Senkrecht nach oben ("T" umgedreht)

#CONSOLECHR_BULLET:
    Aufzählungszeichen

#CONSOLECHR_CKBOARD:
    Schachbrett

#CONSOLECHR_DARROW:
    Pfeil unten

#CONSOLECHR_DEGREE:
    Grad

#CONSOLECHR_DIAMOND:
    Raute

#CONSOLECHR_GEQUAL:
    Größer gleich
```

**#CONSOLECHR\_HLINE:**  
Horizontale Linie

**#CONSOLECHR\_LANTERN:**  
Laterne

**#CONSOLECHR\_LARROW:**  
Pfeil links

**#CONSOLECHR\_LEQUAL:**  
Kleiner gleich

**#CONSOLECHR\_LLCORNER:**  
(Ecke) unten links

**#CONSOLECHR\_LRCORNER:**  
(Ecke) unten rechts

**#CONSOLECHR\_LTEE:**  
Ergibt nicht ("T" links)

**#CONSOLECHR\_NEQUAL:**  
Ungleich

**#CONSOLECHR\_PI:**  
Pi

**#CONSOLECHR\_PLMINUS:**  
Plusminuszeichen

**#CONSOLECHR\_PLUS:**  
Pluszeichen

**#CONSOLECHR\_RARROW:**  
Pfeil rechts

**#CONSOLECHR\_RTEE:**  
Ergibt ("T" rechts)

**#CONSOLECHR\_S1:**  
Bildzeile 1

**#CONSOLECHR\_S3:**  
Bildzeile 3

**#CONSOLECHR\_S7:**  
Bildzeile 7

**#CONSOLECHR\_S9:**  
Bildzeile 9

**#CONSOLECHR\_STERLING:**  
Pfund Sterling Symbol

**#CONSOLECHR\_TTEE:**  
Senkrecht nach unten ("T")

**#CONSOLECHR\_UARROW:**  
Pfeil oben

**#CONSOLECHR\_ULCORNER:**  
(Ecke) oben links

**#CONSOLECHR\_URCORNER:**  
(Ecke) oben rechts

**#CONSOLECHR\_VLINE:**  
Vertikale Linie

Der Parameter **style** unterstützt alle Konsolenstile, die vom Befehl **SetConsoleStyle()** angeboten werden. Siehe [Abschnitt 33.55 \[SetConsoleStyle\]](#), [Seite 690](#), für Details. Wenn Sie keine Stil-Flags setzen wollen, können Sie für **style** den speziellen Stil **#CONSOLESTYLE\_NONE** übergeben. In diesem Fall wendet **MakeConsoleChr()** keine Stile an.

Mit dem optionalen Parameter **pen** kann festgelegt werden, mit welchem Stift das Zeichen gezeichnet werden soll. Vorder- und Hintergrundfarben dieses Stifts können mit dem Befehl **InitConsoleColor()** initialisiert werden. Siehe [Abschnitt 33.38 \[InitConsoleColor\]](#), [Seite 675](#), für Details.

Verwenden Sie den Befehl **DecomposeConsoleChr()**, um ein Zeichen, das Stil- oder Farbformatierungen enthält, um die einzelnen Komponenten eines formatierten Zeichens zu extrahieren. Siehe [Abschnitt 33.11 \[DecomposeConsoleChr\]](#), [Seite 659](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit **EnableAdvancedConsole()** aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

## EINGABEN

<b>c</b>	Zeichen als Unicode-Codepunkt
<b>style</b>	Formatierungsstil, der angewendet werden soll, oder <b>#CONSOLESTYLE_NONE</b> , um das Argument zu ignorieren
<b>pen</b>	optional: zu verwendender Stift oder -1, um die Festlegung eines Stifts zu überspringen (Standardwert: -1)

## RÜCKGABEWERTE

<b>ch</b>	formatierter Zeichencode
-----------	--------------------------

## 33.43 MoveConsoleWindow

### BEZEICHNUNG

**MoveConsoleWindow** – verschiebt das Konsolenfenster (V10.0)

### ÜBERSICHT

**MoveConsoleWindow(id, x, y)**

### PLATTFORMEN

Linux, macOS, Windows



**BESCHREIBUNG**

Dieser Befehl verschiebt das durch `id` angegebene Konsolenfenster an die durch `x` und `y` angegebene Position. Diese Position muss in Zeichen angegeben werden und darf nicht außerhalb des Bildschirms liegen.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

<code>id</code>	ID des zu verschiebenden Konsolenfensters
<code>x</code>	gewünschte x-Position für das Fenster
<code>y</code>	gewünschte y-Position für das Fenster

## 33.44 OpenConsole

**BEZEICHNUNG**

OpenConsole – öffnet ein Konsolenfenster (V10.0)

**ÜBERSICHT**

`OpenConsole()`

**PLATTFORMEN**

Windows

**BESCHREIBUNG**

Windows unterscheidet zwischen Nicht-Konsolen- und Konsolenprogrammen. Aus diesem Grund kann Hollywood zwei verschiedene Arten von Programmen unter Windows kompilieren: Nicht-Konsolenprogramme und Konsolenprogramme. Der Unterschied besteht darin, dass Konsolenprogramme automatisch eine Konsole öffnen, wenn sie gestartet werden, während Nicht-Konsolenprogramme dies nicht tun. Es ist jedoch möglich, eine Konsole in Nicht-Konsolenprogrammen manuell zu öffnen, indem Sie diesen Befehl aufrufen.

Auf allen anderen Plattformen gibt es keine solche Unterscheidung, weshalb dieser Befehl nur unter Windows und nur dann unterstützt wird, wenn Sie die Nicht-Konsolenversion von Hollywood verwenden.

Um die von diesem Befehl geöffnete Konsole zu schließen, rufen Sie einfach den Befehl `CloseConsole()` auf.

Bitte lesen Sie auch das Kapitel über den [Konsolenmodus](#), um mehr über die Verwendung von Hollywood im Konsolenmodus zu erfahren. Siehe [Abschnitt 3.1 \[Konsolenmodus\]](#), [Seite 31](#), für Details.

**EINGABEN**

keine

## 33.45 ReadConsoleKey

### BEZEICHNUNG

ReadConsoleKey – liest eine Konsolentaste (V10.0)

### ÜBERSICHT

```
key = ReadConsoleKey()
```

### PLATTFORMEN

Linux, macOS, Windows

### BESCHREIBUNG

Dieser Befehl liest ein Zeichen aus dem aktuellen Konsolenfenster. Standardmäßig wartet er, bis der Benutzer eine Taste drückt. Wenn Sie möchten, dass der Befehl nicht auf eine Taste wartet, müssen Sie die Option `Delay` in `SetConsoleOptions()` auf `False` setzen. In diesem Fall gibt `ReadConsoleKey()` `#CONSOLEKEY_NONE` zurück, falls keine Taste gedrückt wurde.

Wenn `ReadConsoleKey()` so eingestellt ist, dass er wartet, bis eine Taste gedrückt wird, wird sein Verhalten auch von der Einstellung `CBreak` von `SetConsoleOptions()` beeinflusst. Wenn `CBreak` auf `True` gesetzt ist, was auch die Standardeinstellung ist, reicht ein Tastendruck aus, damit `ReadConsoleKey()` die Blockierung beendet. Wenn `CBreak` auf `False` gesetzt ist, wird der Befehl blockiert, bis ein Zeilenumbruch erfolgt.

Standardmäßig wird die Taste, die der Benutzer drückt, in der Konsole als Echo ausgegeben. Wenn Sie das nicht möchten, setzen Sie den Tag `Echo` in `SetConsoleOptions()` auf `False`.

Schließlich kann `ReadConsoleKey()` auch Funktionstasten wie F1, F2, Cursortasten, ESC usw. lesen. Wenn Sie möchten, dass `ReadConsoleKey()` Funktionstasten unterstützt, müssen Sie die Konsole in den Tastaturmodus versetzen, indem Sie den Tag `Keypad` in `SetConsoleOptions()` auf `True` setzen. Wenn Sie das getan haben, kann `ReadConsoleKey()` auch die folgenden Funktionstasten zurückgeben:

```
#CONSOLEKEY_ENTER  
#CONSOLEKEY_UP  
#CONSOLEKEY_DOWN  
#CONSOLEKEY_RIGHT  
#CONSOLEKEY_LEFT  
#CONSOLEKEY_BACKSPACE  
#CONSOLEKEY_DEL  
#CONSOLEKEY_F1  
#CONSOLEKEY_F2  
#CONSOLEKEY_F3  
#CONSOLEKEY_F4  
#CONSOLEKEY_F5  
#CONSOLEKEY_F6  
#CONSOLEKEY_F7  
#CONSOLEKEY_F8  
#CONSOLEKEY_F9  
#CONSOLEKEY_F10  
#CONSOLEKEY_F11
```

```
#CONSOLEKEY_F12
#CONSOLEKEY_F13
#CONSOLEKEY_F14
#CONSOLEKEY_F15
#CONSOLEKEY_F16
#CONSOLEKEY_INSERT
#CONSOLEKEY_HOME
#CONSOLEKEY_END
#CONSOLEKEY_PRINT
#CONSOLEKEY_PAGEUP
#CONSOLEKEY_PAGEDOWN
#CONSOLEKEY_IC
#CONSOLEKEY_EIC
```

Wie Sie sehen, beeinflussen viele Optionen vom Befehl `SetConsoleOptions()` das Verhalten von `ReadConsoleKey()`. Siehe [Abschnitt 33.54 \[SetConsoleOptions\]](#), [Seite 689](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

#### EINGABEN

keine

#### RÜCKGABEWERTE

`key` Taste, die gedrückt wurde, oder `#CONSOLEKEY_NONE`, wenn keine Taste gedrückt wurde und sich die Konsole im `NoDelay`-Modus befindet

## 33.46 ReadConsoleStr

#### BEZEICHNUNG

`ReadConsoleStr` – liest eine Konsolenzeichenkette (V10.0)

#### ÜBERSICHT

```
s$ = ReadConsoleStr()
```

#### PLATTFORMEN

Linux, macOS, Windows

#### BESCHREIBUNG

Dieser Befehl liest eine Zeichenkette aus dem aktuellen Konsolenfenster und gibt ihn zurück. Im Gegensatz zu `ReadConsoleKey()` ist `ReadConsoleStr()` nicht mit der Option `Delay` von `SetConsoleOptions()` kompatibel. Er funktioniert nur korrekt, wenn `Delay` auf `True` gesetzt ist, was auch die Voreinstellung ist. Dies bedeutet, dass `ReadConsoleStr()` immer blockieren soll, bis eine Eingabe auf der Konsole erfolgt.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

keine

**RÜCKGABEWERTE**

s\$ von der Konsole gelesene Zeichenkette

## 33.47 RefreshConsole

**BEZEICHNUNG**

RefreshConsole – aktualisiert die Konsole (V10.0)

**ÜBERSICHT**

RefreshConsole()

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl zeichnet die Änderungen im aktuellen Konsolenfenster auf den Bildschirm. Um das Zeichnen zu optimieren, wird die Konsole normalerweise nicht automatisch aktualisiert, wenn Änderungen an ihrem Inhalt vorgenommen werden. Stattdessen muss `RefreshConsole()` normalerweise manuell aufgerufen werden, damit die Konsole neu gezeichnet wird. Wenn Sie möchten, dass Hollywood die Konsole automatisch aktualisiert, können Sie das Flag `Immediate` in `SetConsoleOptions()` auf `True` setzen, aber dies kann zu flackerndem Zeichnen führen.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

keine

## 33.48 ScrollConsole

**BEZEICHNUNG**

ScrollConsole – scrollt durch die Konsolenzeilen (V10.0)

**ÜBERSICHT**

ScrollConsole(n)

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl scrollt das Konsolenfenster um die im Argument `n` angegebenen Zeilen nach oben oder unten. Ein positiver Wert in `n` scrollt die Konsole `n` Zeilen nach oben, während ein negativer `n` die Konsole `n` Zeilen nach unten scrollt.

Bevor Sie diesen Befehl verwenden können, müssen Sie den Tag `Scroll` in `SetConsoleOptions` auf `True` setzen. Siehe [Abschnitt 33.54 \[SetConsoleOptions\]](#), [Seite 689](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

#### EINGABEN

`n`            Anzahl der Zeilen, die nach oben oder unten gescrollt werden; positive Werte werden nach oben, negative Werte nach unten gescrollt

### 33.49 SelectConsoleWindow

#### BEZEICHNUNG

SelectConsoleWindow – aktiviert das Konsolenfenster (V10.0)

#### ÜBERSICHT

SelectConsoleWindow(id)

#### PLATTFORMEN

Linux, macOS, Windows

#### BESCHREIBUNG

Dieser Befehl macht das durch `id` angegebene Konsolenfenster zum aktiven. Alle Befehle der Konsolenbibliothek wirken dann auf dieses Konsolenfenster. Der Parameter `id` muss der Identifikator eines Konsolenfensters sein, das von `CreateConsoleWindow()` zugewiesen wurde. Siehe [Abschnitt 33.10 \[CreateConsoleWindow\]](#), [Seite 658](#), für Details. Der spezielle Wert `-1` kann in `id` übergeben werden, um den Standard-Konsolenbildschirm zum aktiven Fenster zu machen.

Achten Sie darauf, dass Sie `SelectConsoleWindow()` nicht mit den gleichnamigen Befehlen `SelectBrush()`, `SelectBGPic()`, `SelectAnim()`, `SelectMask()` und `SelectAlphaChannel()` verwechseln. Alle diese Befehle erfordern, dass Sie `EndSelect()` aufrufen, wenn Sie mit ihnen fertig sind, aber bei `SelectConsoleWindow()` müssen Sie dies nicht anwenden. Tatsächlich funktioniert er auf eine völlig andere Weise, so dass Sie niemals `EndSelect()` für `SelectConsoleWindow()` aufrufen dürfen. Wenn Sie zum zuvor aktiven Konsolenfenster zurückkehren möchten, müssen Sie `SelectConsoleWindow()` erneut aufrufen. Der Aufruf von `EndSelect()` zum Wiederherstellen des zuvor aktiven Konsolenfensters wird definitiv nicht funktionieren.

Um das derzeit aktive Konsolenfenster zu ermitteln, rufen Sie den Befehl `GetConsoleWindow()` auf. Siehe [Abschnitt 33.35 \[GetConsoleWindow\]](#), [Seite 673](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

#### EINGABEN

`id`            id des Konsolenfensters, das aktiviert werden soll

#### BEISPIEL

```
EnableAdvancedConsole()
w, h = GetConsoleSize()
```

```
CreateConsoleWindow(1, 20, 20, (w - 20) / 2, (h - 20) / 2)
SelectConsoleWindow(1)
DrawConsoleBorder()
RefreshConsole()
```

Der obige Code erstellt ein neues 20x20-Fenster, zieht einen Rahmen darum und zentriert es auf dem Bildschirm.

## 33.50 SetAllocConsoleColor

### BEZEICHNUNG

SetAllocConsoleColor – ändert die zugewiesene Farbe (V10.0)

### ÜBERSICHT

```
SetAllocConsoleColor(c, color)
```

### PLATTFORMEN

Linux, macOS, Windows

### BESCHREIBUNG

Dieser Befehl ändert eine mit `AllocConsoleColor()` zugewiesene Farbe. Sie müssen die von `AllocConsoleColor()` zugewiesene Farbe im Parameter `c` und die neue Farbe im Parameter `color` (als **RGB-Farbe**) übergeben.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

### EINGABEN

<code>c</code>	die zu ändernde zugewiesene Farbe
<code>color</code>	neue Farbe als <b>RGB-Farbe</b>

## 33.51 SetConsoleBackground

### BEZEICHNUNG

SetConsoleBackground – legt den Konsolenhintergrund fest (V10.0)

### ÜBERSICHT

```
SetConsoleBackground(ch)
```

### PLATTFORMEN

Linux, macOS, Windows

### BESCHREIBUNG

Dieser Befehl füllt den Hintergrund des aktuellen Konsolenfensters mit dem durch `ch` angegebenen Zeichen. Das Zeichen muss als numerischer Wert übergeben werden, nicht als Zeichenkette. Für normale Zeichen gibt `ch` einfach den Unicode-Codepunkt des jeweiligen Zeichens an, z.B. 65 für 'A'. Das Argument `ch` kann jedoch auch ein Sonderzeichencode sein, der vom Befehl `MakeConsoleChr()` zusammengesetzt wird. Mit diesem Befehl

können Sie Textformatierungsstile in den Zeichencode einfügen und er unterstützt auch spezielle Zeichencodes wie Pfeile oder Rahmenbits. Siehe [Abschnitt 33.42 \[MakeConsoleChr\]](#), [Seite 678](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

## EINGABEN

`ch`            das zu verwendende Zeichen zum Ausfüllen des Hintergrunds

## BEISPIEL

```
EnableAdvancedConsole()
SetConsoleBackground('=')
RefreshConsole()
```

Der obige Code füllt den Hintergrund der Konsole mit dem Zeichen '='.

## 33.52 SetConsoleColor

### BEZEICHNUNG

`SetConsoleColor` – legt die Konsolenfarbe fest (V10.0)

### ÜBERSICHT

```
SetConsoleColor(pen[, fgcolor, bgcolor])
```

### PLATTFORMEN

Linux, macOS, Windows

### BESCHREIBUNG

Mit diesem Befehl kann der angegebene Stift als Textstift eingestellt werden. Wenn auch die Argumente `fgcolor` und `bgcolor` angegeben werden, wird der Stift mit den in `fgcolor` und `bgcolor` übergebenen Farben initialisiert, bevor er zum Textstift wird. Dies ist nur möglich, wenn die Stiftnummer größer als 0 ist. Stift 0 ist für die Standardtextfarbe reserviert. Wenn Sie also 0 in `pen` übergeben, werden `fgcolor` und `bgcolor` ignoriert und die Textfarbe wird auf die Standardtextfarbe der Konsole zurückgesetzt.

Die Anzahl der verfügbaren Stifte hängt von der Konsole ab. Normalerweise sind 256 Stifte verfügbar, was bedeutet, dass die höchste Stiftnummer, die Sie verwenden können, 255 ist. Für die beste Kompatibilität sollten Sie jedoch niedrigere Stiftnummern verwenden, da möglicherweise nicht alle Konsolen 256 Stifte haben.

Die Argumente `fgcolor` und `bgcolor` können entweder **RGB-Werte** oder Farben sein, die mit `AllocConsoleColor()` zugewiesen werden. Beachten Sie, dass standardmäßig nur wenige Farben verfügbar sind und ohne Zuordnung in `fgcolor` oder `bgcolor` übergeben werden können. Diese sind: `#BLACK`, `#WHITE`, `#RED`, `#GREEN`, `#BLUE`, `#YELLOW`, `#AQUA` (Cyan) und `#FUCHSIA` (Magenta). Wenn Sie andere Farben verwenden möchten, müssen Sie diese zuerst mit `AllocConsoleColor()` zuweisen. Siehe [Abschnitt 33.1 \[AllocConsoleColor\]](#), [Seite 653](#), für Details.

Beachten Sie, dass Terminals möglicherweise eine abgedunkelte Version der hier angegebenen Farben verwenden. Seien Sie also nicht überrascht, wenn Ihre Farbe grau erscheint,

obwohl Sie Weiß angegeben haben. Viele Terminals sind so konfiguriert, dass sie Grau als Weiß behandeln. Wenn Sie dies vermeiden möchten, weisen Sie benutzerdefinierte Farben mit `AllocConsoleColor()` zu.

Um die Vorder- und Hintergrundfarbe eines Stifts zu initialisieren, ohne ihn zum aktiven Textstift zu machen, können Sie den Befehl `InitConsoleColor()` verwenden. Siehe [Abschnitt 33.38 \[InitConsoleColor\]](#), [Seite 675](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

## EINGABEN

<code>pen</code>	zu verwendender Farbstift oder 0, um die Textfarbe auf die Standardwerte zurückzusetzen
<code>fgcolor</code>	optional: gewünschte Vordergrundfarbe für den Stift; dies muss entweder eine <b>RGB-Farbe</b> aus der obigen Liste oder eine durch <code>AllocConsoleColor()</code> zugewiesene Farbe sein (Standardwert ist <code>#NOCOLOR</code> )
<code>bgcolor</code>	optional: gewünschte Vordergrundfarbe für den Stift; dies muss entweder eine <b>RGB-Farbe</b> aus der obigen Liste oder eine durch <code>AllocConsoleColor()</code> zugewiesene Farbe sein (Standardwert ist <code>#NOCOLOR</code> )

## BEISPIEL

```
EnableAdvancedConsole()
SetConsoleStyle(#CONSOLESTYLE_BOLD)
SetConsoleColor(1, #BLACK, #WHITE)
ConsolePrint("Hello World")
RefreshConsole()
```

Der obige Code gibt die Zeichenkette "Hello World" in Schwarz auf weißem Hintergrund aus.

## 33.53 SetConsoleCursor

### BEZEICHNUNG

`SetConsoleCursor` – legt die Position des Konsolencursors fest (V10.0)

### ÜBERSICHT

```
SetConsoleCursor(x, y)
```

### PLATTFORMEN

Linux, macOS, Windows

### BESCHREIBUNG

Dieser Befehl bewegt den Cursor des aktuellen Fensters an die durch `x` und `y` angegebene Stelle. Dadurch wird der physische Cursor des Terminals nicht bewegt, sondern erst wenn `RefreshConsole()` aufgerufen wird. Die angegebene Position bezieht sich auf die linke obere Ecke des Fensters, die (0,0) ist.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.



**EINGABEN**

x            gewünschte x-Position  
 y            gewünschte y-Position

**BEISPIEL**

```
EnableAdvancedConsole()
w, h = GetConsoleSize()
s$ = "Hello World!"
SetConsoleCursor((w - StrLen(s$)) / 2, h / 2)
ConsolePrintNR(s$)
RefreshConsole()
```

Der obige Code gibt die Zeichenkette "Hello World" zentriert in der Konsole aus.

**33.54 SetConsoleOptions****BEZEICHNUNG**

SetConsoleOptions – konfiguriert die Konsoleneinstellungen (V10.0)

**ÜBERSICHT**

```
SetConsoleOptions(table)
```

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Mit diesem Befehl können mehrere Einstellungen konfiguriert werden, die bestimmen, wie sich die Konsole im erweiterten Konsolenmodus verhalten soll. Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

`SetConsoleOptions()` akzeptiert ein einzelnes Tabellenargument, das die folgenden Tags enthalten kann:

**Delay:**        Wenn dieser Tag auf `False` gesetzt ist, wartet `ReadConsoleKey()` nicht, bis eine Taste gedrückt wird. Standardmäßig `True`.

**HalfDelay:**

Wenn `Delay` auf `True` gesetzt ist, kann `HalfDelay` verwendet werden, um ein Zeitlimit in Zehntelsekunden anzugeben. Dies veranlasst `ReadConsoleKey()`, für diesen Zeitraum zu blockieren, bevor `#CONSOLEKEY_NONE` zurückgegeben wird, wenn kein Schlüssel empfangen wurde. Falls gesetzt, muss dieser Wert zwischen 1 und 255 liegen.

**Echo:**        Wenn dieser Tag auf `False` gesetzt ist, werden getippte Zeichen nicht in der Konsole ausgegeben. Der Standardwert ist `True`.

**Keypad:**     Wenn dieser Tag auf `True` gesetzt ist, kann `ReadConsoleKey()` auch Funktionstasten wie F1, F2, Cursortasten, ESC usw. lesen. Der Standardwert ist `False`.

- Scroll:** Wenn dieser Tag auf **True** gesetzt ist, wird die Konsole automatisch gescrollt, wenn das Schreiben über das Ende des Konsolenfensters hinausgeht. Der Standardwert ist **False**.
- Clear:** Wenn dieser Tag auf **True** gesetzt ist, löscht **RefreshConsole()** den Bildschirm vollständig und zeichnet den gesamten Bildschirm neu. Der Standardwert ist **False**.
- Leave:** Wenn dieser Tag auf **True** gesetzt ist, wird der Cursor dort belassen, wo er bei einer Bildschirmaktualisierung zufällig stehen bleibt. Dies kann für Anwendungen nützlich sein, bei denen der Cursor nicht verwendet wird, da es die Notwendigkeit von Cursorbewegungen reduziert. Wenn möglich wird der Cursor unsichtbar gemacht, wenn diese Option aktiviert ist. Die Voreinstellung ist **False**.
- Immediate:** Wenn dieser Tag auf **True** gesetzt ist, wird das Konsolenfenster bei jeder Änderung aktualisiert. Der Standardwert ist **False**.
- CBreak:** Wenn **Delay** auf **True** gesetzt ist, steuert **CBreak**, welche Zeichen dazu führen können, dass **ReadConsoleKey()** die Blockierung beendet. Wenn **CBreak** auf **True** gesetzt ist, reicht ein Tastendruck aus, damit **ReadConsoleKey()** die Blockierung beendet. Wenn **CBreak** auf **False** gesetzt ist, wird **ReadConsoleKey()** jedoch blockieren, bis ein Zeilenumbruch erfolgt.
- Newline:** Wenn dies auf **True** gesetzt ist, werden Zeilenumbrüche bei der Eingabe in Wagenrückläufe umgewandelt. Wenn Sie das nicht möchten, setzen Sie **Newline** auf **False**. Der Standardwert ist **True**.

## EINGABEN

- table** Tabelle mit einer oder mehreren zu ändernden Einstellungen (siehe oben)

## 33.55 SetConsoleStyle

### BEZEICHNUNG

SetConsoleStyle – stellt den Konsolenstil ein (V10.0)

### ÜBERSICHT

SetConsoleStyle(style)

### PLATTFORMEN

Linux, macOS, Windows

### BESCHREIBUNG

Dieser Befehl setzt den Stil des aktuellen Konsolenfensters auf denjenigen, der im Parameter **style** übergeben wurde. Dies kann eines oder mehrere der folgenden Flags sein:

#### #CONSOLESTYLE\_NORMAL:

Setzt den Textstil auf den Standard zurück. Dies kann nicht mit anderen Stilen kombiniert werden.

**#CONSOLESTYLE\_BOLD:**  
Text wird fett.

**#CONSOLESTYLE\_ITALIC:**  
Text wird kursiv.

**#CONSOLESTYLE\_UNDERLINED:**  
Unterstreicht den Text.

**#CONSOLESTYLE\_STANDOUT:**  
Hebt den Text hervor.

**#CONSOLESTYLE\_BLINK:**  
Lässt den Text blinken.

**#CONSOLESTYLE\_REVERSE:**  
Kehrt die Farbe des Textes um.

**#CONSOLESTYLE\_DIM:**  
Halbheller Effekt für Text. Dies wird nicht überall unterstützt.

**#CONSOLESTYLE\_PROTECT:**  
Geschützter Modus für Text. Dies wird nicht überall unterstützt.

**#CONSOLESTYLE\_INVISIBLE:**  
Unsichtbarer Text. Dies wird nicht überall unterstützt.

**#CONSOLESTYLE\_ALTCHARSET:**  
Verwendet den alternativen Zeichensatz.

Beachten Sie, dass alle Stil-Flags Bitmasken sind, so dass Sie mehrere Stile mit dem bitweisen OR-Operator (`|`) kombinieren können.

Um einen oder mehrere Konsolenstile zu löschen, verwenden Sie den Befehl `ClearConsoleStyle()`. Siehe [Abschnitt 33.4 \[ClearConsoleStyle\]](#), [Seite 654](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

## EINGABEN

`style` eine oder mehrere zu setzende Stil-Flags (siehe oben)

## BEISPIEL

```
EnableAdvancedConsole()
SetConsoleStyle(#CONSOLESTYLE_BOLD|#CONSOLESTYLE_UNDERLINED)
ConsolePrint("Hello World!")
RefreshConsole()
```

Der obige Code gibt den Text "Hello World!" in fett und unterstrichen aus.

### 33.56 SetConsoleTitle

#### BEZEICHNUNG

SetConsoleTitle – legt den Konsolentitel fest (V10.0)

#### ÜBERSICHT

SetConsoleTitle(*t\$*)

#### PLATTFORMEN

Windows

#### BESCHREIBUNG

Dieser Befehl ändert den Titel des Konsolenfensters in den in *t\$* angegebenen Text. Dies wird derzeit nur unter Windows unterstützt.

#### EINGABEN

*t\$*            neuer Titel für das Konsolenfenster

### 33.57 ShowConsoleCursor

#### BEZEICHNUNG

ShowConsoleCursor – zeigt den Konsolencursor an (V10.0)

#### ÜBERSICHT

ShowConsoleCursor([*normal*])

#### PLATTFORMEN

Linux, macOS, Windows

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Konsolencursor anzuzeigen. Mit dem optionalen Argument *normal* kann eingestellt werden, ob Sie den normalen oder den Block-Cursor haben möchten. Wenn *normal* auf *True* gesetzt ist, wird der normale Cursor angezeigt. Wenn er auf *False* gesetzt ist, wird der Blockcursor verwendet. Standardmäßig zeigt ShowConsoleCursor() den normalen Cursor (*True*).

Sie müssen den erweiterten Konsolenmodus mit [EnableAdvancedConsole\(\)](#) aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

#### EINGABEN

*normal*       optional: ob der normale (*True*) oder der Blockcursor (*False*) angezeigt werden soll (Voreingestellt ist *True*)

### 33.58 StartConsoleColorMode

#### BEZEICHNUNG

StartConsoleColorMode – versetzt die Konsole in den Farbmodus (V10.0)

#### ÜBERSICHT

StartConsoleColorMode([*defcolor*])

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl versetzt die Konsole in den Farbmodus. Normalerweise ist es nicht notwendig, diesen Befehl aufzurufen, da alle Befehle und Funktionen der Konsolenbibliothek, die im Farbmodus arbeiten, den Farbmodus automatisch aktivieren. Es gibt jedoch eine Ausnahme: Wenn Sie die Standardfarben nicht verwenden möchten, müssen Sie den Farbmodus manuell starten und das Argument `defcolor` auf `False` setzen. In diesem Fall wird der Farbmodus aktiviert, ohne jedoch die Standardfarben zu verwenden. Beachten Sie, dass es in diesem Fall wichtig ist, `StartConsoleColorMode()` vor allen anderen Befehlen und Funktionen der Konsolenbibliothek aufzurufen, die den Farbmodus aktivieren könnten. Wenn Sie diesen Befehl also verwenden, ist es am besten, ihn direkt nach `EnableAdvancedConsole()` aufzurufen.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

`defcolor` optional: ob die Standardfarben verwendet werden sollen (`True`) oder nicht (`False`) (Voreingestellt ist `True`)

## 33.59 TouchConsoleWindow

**BEZEICHNUNG**

`TouchConsoleWindow` – erzwingt eine Aktualisierung des gesamten Fensters (V10.0)

**ÜBERSICHT**

`TouchConsoleWindow(id)`

**PLATTFORMEN**

Linux, macOS, Windows

**BESCHREIBUNG**

Dieser Befehl markiert das gesamte durch `id` angegebene Konsolenfenster als aktualisierungsbedürftig. Standardmäßig werden nur die Bereiche im Fenster, die sich geändert haben, zur Aktualisierung markiert. Wenn Sie eine Aktualisierung des gesamten Fensters erzwingen möchten, rufen Sie `TouchConsoleWindow()` darauf auf.

Der Parameter `id` muss der Identifikator eines Konsolenfensters sein, das vom Befehl `CreateConsoleWindow()` zugewiesen wurde. Siehe [Abschnitt 33.10 \[CreateConsoleWindow\]](#), [Seite 658](#), für Details.

Sie müssen den erweiterten Konsolenmodus mit `EnableAdvancedConsole()` aktivieren, bevor Sie diesen Befehl verwenden können. Siehe [Abschnitt 33.19 \[EnableAdvancedConsole\]](#), [Seite 664](#), für Details.

**EINGABEN**

`id` ID des zu verwendenden Konsolenfensters



## 34 Lokalisierungsbibliothek

### 34.1 Übersicht

Hollywood ermöglicht es Ihnen, Ihre Programme zu lokalisieren, indem Sie Text-Zeichenketten aus so genannten Katalogdateien importieren, die Sie mit dem Befehl `OpenCatalog()` oder der Präprozessor-Anweisung `@CATALOG` öffnen. Aus Kompatibilitätsgründen sind diese Kataloge im Format `IFF CTLG`, welches von Commodore in den frühen 1990er Jahren eingeführt wurde. Aber dies wird nicht empfohlen, weil `IFF CTLG` keine ordnungsgemäße Unicode-Unterstützung hat und nur dann korrekt funktioniert, wenn die Systemlokalisierung dem des Katalogs entspricht.

So empfiehlt es sich, stattdessen Hollywoods eigenes Katalogformat zu verwenden. Dieses alternative Katalogformat ist einfach eine Textdatei, die in der UTF-8-Zeichencodierung eine Liste von Katalogzeichenketten aufführt, welche eine Zeichenfolge pro Zeile enthält. Wenn Zeilenumbrüche in einem Katalog mit Zeichenfolgen auftreten, müssen Sie sie als `"\n"` schreiben. Schrägstriche müssen als `"\"` geschrieben werden. Wenn Sie eine Zeichenfolge über mehrere Zeilen hinweg teilen müssen, fügen Sie einen einzelnen umgekehrten Schrägstrich an das Ende der jeweiligen Zeile an. Ein solcher einzelner Schrägstrich am Ende der Zeile signalisiert Hollywood, dass die Zeichenfolge in der nächsten Zeile fortgesetzt wird. Wenn am Ende der Zeile kein Schrägstrich vorhanden ist, weiß Hollywood, dass dies das Ende der Zeichenfolge ist. Leere Zeilen werden vollständig ignoriert und können aus Gründen der Lesbarkeit eingefügt werden.

Die Textdatei muss in der UTF-8-Zeichencodierung mit oder ohne BOM sein. Eine Bemerkung kann eingefügt werden, indem ein Semikolon (;) als erstes Zeichen in der Zeile verwendet wird. Dies lässt Hollywood den Rest der Zeile ignorieren. Ein Semikolon (;) an anderen Positionen in einer Zeile haben keine Bedeutung für den Parser. Wenn das erste Zeichen Ihrer Zeichenfolge ein Semikolon (;) sein muss, müssen Sie ihm einen umgekehrten Schrägstrich (\) voranstellen, um dem Parser mitzuteilen, dass dies kein Kommentar ist.

Hier ist ein einfaches Beispiel für eine Katalogdefinition im eigenen Hollywood-Katalogformat:

```
; Das ist ein Kommentar
Das ist die erste Zeichenfolge!
;
Das ist die zweite Zeichenkette!
;
Das ist die dritte Zeichenfolge \nund geht über zwei Zeilen!
;
Das ist die vierte Zeichenkette \
und hat nur eine Zeile \
die aber über vier Zeilen \
in der Katalog-Datei verteilt ist!
;
Das ist die fünfte und letzte Zeichenfolge!
```

Um eine Zeichenkette aus der Katalogdatei zu erhalten, rufen Sie einfach `GetCatalogString()` auf und übergeben Sie den Index der Zeichenkette, die Sie abrufen möchten. Um beispiels-

weise die fünfte Zeichenkette aus der oben angezeigten Katalogdatei abzurufen, müssen Sie Folgendes tun:

```
Print(GetCatalogString(4, "Vorgegebene Zeichenkette"))
```

Die Zeichenkette, die im zweiten Argument an `GetCatalogString()` übergeben wird, ist die Standardzeichenkette und wird nur zurückgegeben, wenn der angeforderte Zeichenkettenindex in der Katalogdatei nicht gefunden wurde.

## 34.2 CATALOG

### BEZEICHNUNG

CATALOG – lädt einen Katalog für die spätere Verwendung vor (V9.0)

### ÜBERSICHT

```
@CATALOG name$[, table]
```

### BESCHREIBUNG

Mit dieser Präprozessor-Anweisung kann der angegebene Katalog in der Sprache des Benutzers vorgeladen werden. Wenn kein Katalog in der Sprache des Benutzers vorhanden ist, zeigt diese Präprozessor-Anweisung keinen Fehler an. Dies ist normal, da Sie für jeden Eintrag, den Sie mit dem Befehl `GetCatalogString()` aus einem Katalog abrufen möchten, immer Standardzeichenketten angeben müssen. Daher ist es kein Problem, wenn `@CATALOG` einen Katalog angibt, der nicht existiert. In diesem Fall wird `GetCatalogString()` einfach auf die Standardzeichenketten zurückgreifen, ohne einen Fehler auszulösen.

Beachten Sie, dass es derzeit nur einen einzigen Katalog pro Anwendung geben kann. Daher sollte `@CATALOG` nur einmal pro Skript verwendet werden. Beachten Sie auch, dass `name$` kein Dateiname sein darf, sondern der Name eines Katalogs, der in einer Hollywood-Katalogverzeichnisstruktur gespeichert ist. Siehe [Abschnitt 34.11 \[OpenCatalog\]](#), [Seite 712](#), für Details zur Organisation einer solchen Hollywood-Katalogstruktur.

Der Vorteil der Verwendung von `@CATALOG` anstelle von `OpenCatalog()` besteht darin, dass bei Verwendung von `@CATALOG` die Kataloge für alle verfügbaren Sprachen beim Kompilieren Ihres Skripts mit Ihrem Programm oder Applet verknüpft werden. Dies erleichtert die Verteilung Ihres Projekts, da Sie die Katalogverzeichnisstruktur nicht in Ihr Programm einbinden müssen. Auch beim Kompilieren von Hollywood-Applets für mobile Systeme wie Android oder iOS ist es viel besser, alle externen Dateien mit dem Applet zu verknüpfen.

Zusätzlich zum Parameter `name$` akzeptiert `@CATALOG` auch ein optionales Tabellenargument. Die folgenden Felder im Tabellenargument sind derzeit verfügbar:

**Link:** Setzen Sie dieses Feld auf `False`, wenn Sie nicht möchten, dass dieser Katalog beim Kompilieren Ihres Skripts mit Ihrem Programm/Applet verknüpft wird. Dieses Feld ist standardmäßig auf `True` gesetzt. Dies bedeutet, dass der Katalog mit Ihrem Programm/Applet verknüpft wird, wenn sich Hollywood im Kompilierungsmodus befindet.

Um einen Katalog während der Laufzeit zu laden, verwenden Sie den Befehl `OpenCatalog()`. Siehe [Abschnitt 34.11 \[OpenCatalog\]](#), [Seite 712](#), für Details.



**EINGABEN**

name\$      Name des zu öffnenden Katalogs  
 table      optional: Tabelle mit weiteren Optionen

**BEISPIEL**

```
@CATALOG "Hollywood.catalog"

; dies ist unser standardmäßiger englischer Katalog
def$ = {}
def$[0] = "Welcome to Hollywood!"
def$[1] = "Written by Andreas Falkenhahn"
def$[2] = "What do you wanna do?"
```

```
; Wenn Hollywood.catalog nicht in der Sprache
; des Benutzers verfügbar ist, werden die
; englischen Zeichenketten verwendet
For k = 0 To 2
  c$[k] = GetCatalogString(k, def$[k])
Next
```

Der obige Code öffnet "Hollywood.catalog" und gibt die ersten drei Einträge aus diesem Katalog aus.

### 34.3 CloseCatalog

**BEZEICHNUNG**

CloseCatalog – schließt einen offenen Katalog

**ÜBERSICHT**

```
CloseCatalog()
```

**BESCHREIBUNG**

Dieser Befehl wird den aktuell geöffneten Katalog schließen. Sie müssen diesen Befehl nicht wirklich aufrufen, weil Hollywood den Katalog von selbst schließt, wenn es beendet wird.

**EINGABEN**

keine

**BEISPIEL**

Siehe [Abschnitt 34.11 \[OpenCatalog\]](#), Seite 712.

### 34.4 FormatDate

**BEZEICHNUNG**

FormatDate – formatiert das Datum und die Uhrzeit gemäß der Formatvorlage (V10.0)

**ÜBERSICHT**

```
d$ = FormatDate(fmt$, date$[, isdst])
```

## BESCHREIBUNG

Dieser Befehl formatiert das in `date$` übergebene Datum gemäß der in `fmt$` übergebenen Formatvorlage und gibt das Ergebnis zurück. Dadurch können Sie Formatvorlagen für Datum und Uhrzeit, die von `GetLocaleInfo()` zurückgegeben werden, in menschenlesbare Datumsangaben umwandeln. Die in `date$` übergebene Datumszeichenkette muss in der von Hollywood verwendeten Standardnotation für Datum und Uhrzeit vorliegen:

`dd-mmm-yyyy hh:mm:ss`

Der `dd`-Teil ist eine zweistellige Datumsangabe (mit führenden Nullen) und der `mmm`-Teil ist eine Zeichenkette mit drei Zeichen, die den Monat angibt. Dies kann `Jan`, `Feb`, `Mar`, `Apr`, `May`, `Jun`, `Jul`, `Aug`, `Sep`, `Oct`, `Nov` oder `Dec` sein. `yyyy` ist eine vierstellige Jahresangabe, während `hh` die Stunden, `mm` die Minuten und `ss` die Sekunden angeben.

Die in `date$` übergebene Datumsvorlage kann Zeichenkettenliterals sowie die folgenden Token enthalten:

<code>%a</code>	Abgekürzter Name des Wochentags
<code>%A</code>	Name des Wochentags
<code>%b</code>	Abgekürzter Name des Monats
<code>%B</code>	Name des Monats
<code>%d</code>	Zahl des Tages mit führenden Nullen
<code>%-d</code>	Zahl des Tages ohne führenden Nullen
<code>%H</code>	Stunde im 24-Stunden-Stil mit führenden Nullen
<code>%-H</code>	Stunde im 24-Stunden-Stil ohne führenden Nullen
<code>%I</code>	Stunde im 12-Stunden-Stil mit führenden Nullen
<code>%-I</code>	Stunde im 12-Stunden-Stil ohne führenden Nullen
<code>%m</code>	Monatszahl mit führenden Nullen
<code>%-m</code>	Monatszahl ohne führenden Nullen
<code>%M</code>	Anzahl der Minuten mit führenden Nullen
<code>%-M</code>	Anzahl der Minuten ohne führenden Nullen
<code>%S</code>	Anzahl der Sekunden mit führenden Nullen
<code>%-S</code>	Anzahl der Sekunden ohne führenden Nullen
<code>%y</code>	Jahr mit zwei Ziffern und führenden Nullen
<code>%-y</code>	Jahr mit zwei Ziffern ohne führenden Nullen
<code>%Y</code>	Jahr mit vier Ziffern und führenden Nullen

Beachten Sie, dass abhängig von der Plattform, auf der Hollywood ausgeführt wird, möglicherweise einige weitere Token unterstützt werden, aber nur die oben aufgeführten funktionieren garantiert auf allen Plattformen.

Nur die oben aufgeführten Token werden in `fmt$` ersetzt. Alle anderen Token werden nicht ersetzt und verbleiben als Zeichenkettenliterals in der zurückgegebenen Zeichenkette. Wenn Sie ein Prozentzeichen als Zeichenkettenliteral in der Datumsvorlage haben möchten, müssen Sie zwei Prozentzeichen (`%%`) angeben.

Das optionale Argument `isdst` gibt an, ob an dem angegebenen Datum die Sommerzeit aktiv ist oder nicht. Normalerweise müssen Sie dieses Argument nicht angeben, da Hollywood diese Information automatisch aus der Zeitzonendatenbank abfragt. Die Weitergabe dieser Information ist nur dann notwendig, wenn die Zeitangabe mehrdeutig ist, d.h. wenn bei der Umstellung von Sommerzeit auf Winterzeit eine bestimmte Zeitspanne (normalerweise eine Stunde) in der Nacht wiederholt wird. In Deutschland werden beispielsweise die Uhren bei der Umstellung von Sommerzeit auf Winterzeit von 3 Uhr morgens auf 2 Uhr morgens zurückgestellt. Das bedeutet, dass die Stunde zwischen 2 Uhr und 3 Uhr morgens zweimal vorkommt: Einmal in der Sommerzeit und einmal in der Winterzeit (Normalzeit). Mit dem Argument `isdst` können Sie angeben, auf welche Stunde Sie sich beziehen.

#### EINGABEN

<code>fmt\$</code>	Datumvorlage-Zeichenkette
<code>date\$</code>	Datumszeichenkette in der Hollywood-Datumsnotation
<code>isdst</code>	optional: ob die Sommerzeit zum angegebenen Datum aktiv ist oder nicht (Standardwert ist -1, was bedeutet, dass diese Information aus der lokalen Zeitzonendatenbank abgerufen werden sollte)

#### RÜCKGABEWERTE

<code>d\$</code>	formatiertes Datum
------------------	--------------------

#### BEISPIEL

```
d$ = FormatDate(GetLocaleInfo().DateTimeFormat, getDate(#DATELOCAL))
DebugPrint(d$)
```

Der obige Code gibt das Datum und die Uhrzeit entsprechend den Regeln des aktuellen Gebietsschemas formatiert aus.

## 34.5 GetCatalogString

#### BEZEICHNUNG

`GetCatalogString` – holt eine Zeichenkette aus einem Katalog

#### ÜBERSICHT

```
s$ = GetCatalogString(id, default$)
```

#### BESCHREIBUNG

Dieser Befehl extrahiert den Text mit der Nummer `id` aus der momentan geöffneten Katalogdatei. Falls kein Katalog geöffnet ist, wird die in `default$` definierte Zeichenkette zurückgegeben.

#### EINGABEN

<code>id</code>	gibt die Nummer des verlangten Texts an (beginnend bei 0)
<code>default\$</code>	Text, der zurückgegeben wird wenn kein Katalog geöffnet ist oder kein Text mit der angegebenen <code>id</code> vorhanden ist

#### BEISPIEL

Siehe [Abschnitt 34.11 \[OpenCatalog\]](#), Seite 712.

## 34.6 GetCountryInfo

### BEZEICHNUNG

GetCountryInfo – gibt Informationen über das Land zurück (V7.1)

### ÜBERSICHT

```
t = GetCountryInfo(ctry)
```

### BESCHREIBUNG

Mit diesem Befehl können Sie zusätzliche Informationen zu einem Land abrufen. Sie müssen im Argument `ctry` eine von Hollywoods Länderkonstanten übergeben. Siehe [Abschnitt 34.9 \[GetSystemCountry\]](#), Seite 702, für eine Länderliste.

`GetCountryInfo()` gibt dann eine Tabelle mit den folgenden Feldern zurück:

Alpha2: Alpha-2 Ländercode, der in ISO 3166-1 definiert ist.

Alpha3: Alpha-3 Ländercode, der in ISO 3166-1 definiert ist.

### EINGABEN

`ctry` eine der in Hollywood definierten Länderkonstanten (siehe oben)

### RÜCKGABEWERTE

`t` Tabelle mit Länderinformationen

### BEISPIEL

```
t = GetCountryInfo(GetSystemCountry())
Print(t.Alpha2, t.Alpha3)
```

Auf einem deutschen System wird dies "DE" und "DEU" ausgegeben.

## 34.7 GetLanguageInfo

### BEZEICHNUNG

GetLanguageInfo – gibt Informationen über die Sprache zurück (V7.1)

### ÜBERSICHT

```
t = GetLanguageInfo(lang)
```

### BESCHREIBUNG

Mit diesem Befehl können Sie zusätzliche Informationen zu einer Sprache abrufen. Sie müssen im Argument `lang` eine der Sprachkonstanten von Hollywood übergeben. Siehe [Abschnitt 34.10 \[GetSystemLanguage\]](#), Seite 707, für eine Liste der Sprachen.

`GetLanguageInfo()` gibt dann eine Tabelle mit den folgenden Feldern zurück:

Code: Sprachcode mit zwei Buchstaben gemäß ISO 639.

Name: Der ISO-Name der Sprache.

### EINGABEN

`lang` eine der in Hollywood definierten Sprachkonstanten (siehe oben)

### RÜCKGABEWERTE

`t` Tabelle mit Informationen über die Sprache

**BEISPIEL**

```
t = GetLanguageInfo(GetSystemLanguage())
Print(t.Code, t.Name)
```

Auf einem deutschen System wird dies "DE" und "german" ausgegeben.

## 34.8 GetLocaleInfo

**BEZEICHNUNG**

GetLocaleInfo – gibt Informationen über das Landesschema zurück (V10.0)

**ÜBERSICHT**

```
t = GetLocaleInfo()
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um Informationen über das derzeit aktive Landesschema abzurufen. `GetLocaleInfo()` gibt dann eine Tabelle mit den folgenden initialisierten Feldern zurück:

**DecimalPoint:**

Enthält den Dezimalpunkt des Landesschemas, z.B. "." auf einem englischen und "," auf einem deutschen System.

**ThousandSeparator:**

Enthält das Tausendertrennzeichen des Landesschemas, z.B. "," auf einem englischen und "." auf einem deutschen System.

**Currency:**

Enthält das Standardwährungssymbol des Landesschemas, z.B. "€" auf einem deutschen System.

**DateFormat:**

Enthält die Vorlage für das lange Datumsformat für das aktuelle Landesschema. Siehe [Abschnitt 34.4 \[FormatDate\]](#), [Seite 697](#), für eine Beschreibung der einzelnen Formatvorlagen.

**ShortDateFormat:**

Enthält die Vorlage für das kurze Datumsformat für das aktuelle Landesschema. Siehe [Abschnitt 34.4 \[FormatDate\]](#), [Seite 697](#), für eine Beschreibung der einzelnen Formatvorlagen.

**TimeFormat:**

Enthält die Zeitformatvorlage für das aktuelle Landesschema. Siehe [Abschnitt 34.4 \[FormatDate\]](#), [Seite 697](#), für eine Beschreibung der einzelnen Formatvorlagen.

**DateTimeFormat:**

Enthält die kombinierte Datums- und Zeitformatvorlage für das aktuelle Landesschema. Siehe [Abschnitt 34.4 \[FormatDate\]](#), [Seite 697](#), für eine Beschreibung der einzelnen Formatvorlagen.

**Days:**

Enthält ein Array mit den Namen der Wochentage im aktuellen Landesschema, z.B. "Monday", "Tuesday", usw.

- AbDays:** Enthält ein Array mit den abgekürzten Namen der Wochentage im aktuellen Landesschema, z.B. "Mon", "Tue", usw.
- Months:** Enthält ein Array mit den Monatsnamen im aktuellen Landesschema, z.B. "January", "February" usw.
- AbMonths:** Enthält ein Array mit den abgekürzten Monatsnamen im aktuellen Landesschema, z.B. "Jan", "Feb" usw.
- Language:** Der Name der Sprache in der Sprache des Landesschema, z.B. "deutsch" auf einem deutschen System. Beachten Sie, dass die genaue Zeichenfolge, die Sie hier erhalten, vom Host-Betriebssystem abhängt, so dass dies nicht wirklich portabel ist.

**EINGABEN**

keine

**RÜCKGABEWERTE**

t            Tabelle mit Landesschema-Informationen

**BEISPIEL**

```
d$ = FormatDate(GetLocaleInfo().DateTimeFormat, GetDate(#DATELOCAL))
DebugPrint(d$)
```

Der obige Code gibt das Datum und die Uhrzeit entsprechend den Regeln des aktuellen Landesschemas formatiert aus.

## 34.9 GetSystemCountry

**BEZEICHNUNG**

GetSystemCountry – ruft das aktuelle Land des Benutzers ab (V5.0)

**ÜBERSICHT**

```
cntry = GetSystemCountry()
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die Ländereinstellung des aktuellen Systems abzurufen. Folgende Länder werden derzeit unterstützt:

```
#COUNTRY_UK (0)
#COUNTRY_USA (1)
#COUNTRY_AUSTRALIA (2)
#COUNTRY_BELGIUM (3)
#COUNTRY_BULGARIA (4)
#COUNTRY_BRAZIL (5)
#COUNTRY_CANADA (6)
#COUNTRY_CZECHREPUBLIC (7)
#COUNTRY_DENMARK (8)
#COUNTRY_GERMANY (9)
#COUNTRY_SPAIN (10)
```

#COUNTRY\_FRANCE (11)  
#COUNTRY\_GREECE (12)  
#COUNTRY\_ITALY (13)  
#COUNTRY\_LIECHTENSTEIN (14)  
#COUNTRY\_LITHUANIA (15)  
#COUNTRY\_LUXEMBOURG (16)  
#COUNTRY\_HUNGARY (17)  
#COUNTRY\_MALTA (18)  
#COUNTRY\_MONACO (19)  
#COUNTRY\_NETHERLANDS (20)  
#COUNTRY\_NORWAY (21)  
#COUNTRY\_POLAND (22)  
#COUNTRY\_PORTUGAL (23)  
#COUNTRY\_ROMANIA (24)  
#COUNTRY\_RUSSIA (25)  
#COUNTRY\_SANMARINO (26)  
#COUNTRY\_SLOVAKIA (27)  
#COUNTRY\_SLOVENIA (28)  
#COUNTRY\_SWITZERLAND (29)  
#COUNTRY\_FINLAND (30)  
#COUNTRY\_SWEDEN (31)  
#COUNTRY\_TURKEY (32)  
#COUNTRY\_IRELAND (33)  
#COUNTRY\_AUSTRIA (34)  
#COUNTRY\_ICELAND (35)  
#COUNTRY\_ANDORRA (36)  
#COUNTRY\_UKRAINE (37)  
#COUNTRY\_UNKNOWN (38)  
#COUNTRY\_AFGHANISTAN (39)  
#COUNTRY\_ALANDISLANDS (40)  
#COUNTRY\_ALBANIA (41)  
#COUNTRY\_ALGERIA (42)  
#COUNTRY\_AMERICANSAMOA (43)  
#COUNTRY\_ANGOLA (44)  
#COUNTRY\_ANGUILLA (45)  
#COUNTRY\_ANTARCTICA (46)  
#COUNTRY\_ANTIGUAANDBARBUDA (47)  
#COUNTRY\_ARGENTINA (48)  
#COUNTRY\_ARMENIA (49)  
#COUNTRY\_ARUBA (50)  
#COUNTRY\_AZERBAIJAN (51)  
#COUNTRY\_BAHAMAS (52)  
#COUNTRY\_BAHRAIN (53)  
#COUNTRY\_BANGLADESH (54)  
#COUNTRY\_BARBADOS (55)  
#COUNTRY\_BELARUS (56)  
#COUNTRY\_BELIZE (57)

#COUNTRY\_BENIN (58)  
#COUNTRY\_BERMUDA (59)  
#COUNTRY\_BHUTAN (60)  
#COUNTRY\_BOLIVIA (61)  
#COUNTRY\_BESISLANDS (62)  
#COUNTRY\_BOSNIAANDHERZEGOVINA (63)  
#COUNTRY\_BOTSWANA (64)  
#COUNTRY\_BOUVETISLAND (65)  
#COUNTRY\_BRUNEI (66)  
#COUNTRY\_BURKINAFASO (67)  
#COUNTRY\_BURUNDI (68)  
#COUNTRY\_CAMBODIA (69)  
#COUNTRY\_CAMEROON (70)  
#COUNTRY\_CAPEVERDE (71)  
#COUNTRY\_CAYMANISLANDS (72)  
#COUNTRY\_CENTRALAFRICANREPUBLIC (73)  
#COUNTRY\_CHAD (74)  
#COUNTRY\_CHILE (75)  
#COUNTRY\_CHINA (76)  
#COUNTRY\_CHRISTMASISLAND (77)  
#COUNTRY\_COCOSISLANDS (78)  
#COUNTRY\_COLOMBIA (79)  
#COUNTRY\_COMOROS (80)  
#COUNTRY\_CONGO (81)  
#COUNTRY\_COOKISLANDS (82)  
#COUNTRY\_COSTARICA (83)  
#COUNTRY\_IVORYCOAST (84)  
#COUNTRY\_CROATIA (85)  
#COUNTRY\_CUBA (86)  
#COUNTRY\_CURACAO (87)  
#COUNTRY\_CYPRUS (88)  
#COUNTRY\_DJIBOUTI (89)  
#COUNTRY\_DOMINICA (90)  
#COUNTRY\_DOMINICANREPUBLIC (91)  
#COUNTRY\_DRCONGO (92)  
#COUNTRY\_ECUADOR (93)  
#COUNTRY\_EGYPT (94)  
#COUNTRY\_ELSALVADOR (95)  
#COUNTRY\_EQUATORIALGUINEA (96)  
#COUNTRY\_ERITREA (97)  
#COUNTRY\_ESTONIA (98)  
#COUNTRY\_ETHIOPIA (99)  
#COUNTRY\_FALKLANDISLANDS (100)  
#COUNTRY\_FAROEISLANDS (101)  
#COUNTRY\_FIJI (102)  
#COUNTRY\_FRENCHGUIANA (103)  
#COUNTRY\_FRENCHPOLYNESIA (104)



#COUNTRY\_GABON (105)  
#COUNTRY\_GAMBIA (106)  
#COUNTRY\_GEORGIA (107)  
#COUNTRY\_GHANA (108)  
#COUNTRY\_GIBRALTAR (109)  
#COUNTRY\_GREENLAND (110)  
#COUNTRY\_GRENADA (111)  
#COUNTRY\_GUADELOUPE (112)  
#COUNTRY\_GUAM (113)  
#COUNTRY\_GUATEMALA (114)  
#COUNTRY\_GUERNSEY (115)  
#COUNTRY\_GUINEA (116)  
#COUNTRY\_GUINEABISSAU (117)  
#COUNTRY\_GUYANA (118)  
#COUNTRY\_HAITI (119)  
#COUNTRY\_HOLYSEE (120)  
#COUNTRY\_HONDURAS (121)  
#COUNTRY\_HONGKONG (122)  
#COUNTRY\_INDIA (123)  
#COUNTRY\_INDONESIA (124)  
#COUNTRY\_IRAN (125)  
#COUNTRY\_IRAQ (126)  
#COUNTRY\_ISLEOFMAN (127)  
#COUNTRY\_ISRAEL (128)  
#COUNTRY\_JAMAICA (129)  
#COUNTRY\_JAPAN (130)  
#COUNTRY\_JERSEY (131)  
#COUNTRY\_JORDAN (132)  
#COUNTRY\_KAZAKHSTAN (133)  
#COUNTRY\_KENYA (134)  
#COUNTRY\_KIRIBATI (135)  
#COUNTRY\_NORTHKOREA (136)  
#COUNTRY\_SOUTHKOREA (137)  
#COUNTRY\_KUWAIT (138)  
#COUNTRY\_KYRGYZSTAN (139)  
#COUNTRY\_LAOS (140)  
#COUNTRY\_LATVIA (141)  
#COUNTRY\_LEBANON (142)  
#COUNTRY\_LESOTHO (143)  
#COUNTRY\_LIBERIA (144)  
#COUNTRY\_LIBYA (145)  
#COUNTRY\_MACAO (146)  
#COUNTRY\_MACEDONIA (147)  
#COUNTRY\_MADAGASCAR (148)  
#COUNTRY\_MALAWI (149)  
#COUNTRY\_MALAYSIA (150)  
#COUNTRY\_MALDIVES (151)

#COUNTRY\_MALI (152)  
#COUNTRY\_MARSHALLISLANDS (153)  
#COUNTRY\_MARTINIQUE (154)  
#COUNTRY\_MAURITANIA (155)  
#COUNTRY\_MAURITIUS (156)  
#COUNTRY\_MAYOTTE (157)  
#COUNTRY\_MEXICO (158)  
#COUNTRY\_MICRONESIA (159)  
#COUNTRY\_MOLDOVA (160)  
#COUNTRY\_MONGOLIA (161)  
#COUNTRY\_MONTENEGRO (162)  
#COUNTRY\_MONTSEERRAT (163)  
#COUNTRY\_MOROCCO (164)  
#COUNTRY\_MOZAMBIQUE (165)  
#COUNTRY\_MYANMAR (166)  
#COUNTRY\_NAMIBIA (167)  
#COUNTRY\_NAURU (168)  
#COUNTRY\_NEPAL (169)  
#COUNTRY\_NEWCALEDONIA (170)  
#COUNTRY\_NEWZEALAND (171)  
#COUNTRY\_NICARAGUA (172)  
#COUNTRY\_NIGER (173)  
#COUNTRY\_NIGERIA (174)  
#COUNTRY\_NIUE (175)  
#COUNTRY\_NORFOLKISLAND (176)  
#COUNTRY\_OMAN (177)  
#COUNTRY\_PAKISTAN (178)  
#COUNTRY\_PALAU (179)  
#COUNTRY\_PALESTINE (180)  
#COUNTRY\_PANAMA (181)  
#COUNTRY\_PAPUANEWGUINEA (182)  
#COUNTRY\_PARAGUAY (183)  
#COUNTRY\_PERU (184)  
#COUNTRY\_PHILIPPINES (185)  
#COUNTRY\_PITCAIRN (186)  
#COUNTRY\_PUERTORICO (187)  
#COUNTRY\_QATAR (188)  
#COUNTRY\_REUNION (189)  
#COUNTRY\_RWANDA (190)  
#COUNTRY\_SAINTBARTHELEMY (191)  
#COUNTRY\_SAINTHELENA (192)  
#COUNTRY\_SAINTKITTSANDNEVIS (193)  
#COUNTRY\_SAINTLUCIA (194)  
#COUNTRY\_SAINTVINCENT (195)  
#COUNTRY\_SAMOA (196)  
#COUNTRY\_SAOTOMEANDPRINCIPE (197)  
#COUNTRY\_SAUDIARABIA (198)

```
#COUNTRY_SENEGAL (199)
#COUNTRY_SERBIA (200)
#COUNTRY_SEYCHELLES (201)
#COUNTRY_SIERRALEONE (202)
#COUNTRY_SINGAPORE (203)
#COUNTRY_SOLOMONISLANDS (204)
#COUNTRY_SOMALIA (205)
#COUNTRY_SOUTHAFRICA (206)
#COUNTRY_SOUTHSUDAN (207)
#COUNTRY_SRILANKA (208)
#COUNTRY_SUDAN (209)
#COUNTRY_SURINAME (210)
#COUNTRY_SWAZILAND (211)
#COUNTRY_SYRIA (212)
#COUNTRY_TAIWAN (213)
#COUNTRY_TAJIKISTAN (214)
#COUNTRY_TANZANIA (215)
#COUNTRY_THAILAND (216)
#COUNTRY_TIMOR (217)
#COUNTRY_TOGO (218)
#COUNTRY_TONGA (219)
#COUNTRY_TRINIDADANDTOBAGO (220)
#COUNTRY_TUNISIA (221)
#COUNTRY_TURKMENISTAN (222)
#COUNTRY_TUVALU (223)
#COUNTRY_UGANDA (224)
#COUNTRY_UAE (225)
#COUNTRY_URUGUAY (226)
#COUNTRY_UZBEKISTAN (227)
#COUNTRY_VANUATU (228)
#COUNTRY_VENEZUELA (229)
#COUNTRY_VIETNAM (230)
#COUNTRY_YEMEN (231)
#COUNTRY_ZAMBIA (232)
```

**EINGABEN**

keine

**RÜCKGABEWERTE**

cntry      Ländereinstellung des aktuellen Benutzers

## 34.10 GetSystemLanguage

**BEZEICHNUNG**

GetSystemLanguage – ruft die aktuelle Benutzersprache ab (V5.0)

**ÜBERSICHT**

```
lang = GetSystemLanguage()
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die Standardsprache des aktuellen Benutzers abzurufen. Folgende Sprachen werden unterstützt:

```
#LANGUAGE_ENGLISH (0)
#LANGUAGE_GERMAN (1)
#LANGUAGE_DUTCH (2)
#LANGUAGE_ITALIAN (3)
#LANGUAGE_FRENCH (4)
#LANGUAGE_SPANISH (5)
#LANGUAGE_PORTUGUESE (6)
#LANGUAGE_SWEDISH (7)
#LANGUAGE_DANISH (8)
#LANGUAGE_FINNISH (9)
#LANGUAGE_NORWEGIAN (10)
#LANGUAGE_POLISH (11)
#LANGUAGE_HUNGARIAN (12)
#LANGUAGE_GREEK (13)
#LANGUAGE_CZECH (14)
#LANGUAGE_TURKISH (15)
#LANGUAGE_CROATIAN (16)
#LANGUAGE_RUSSIAN (17)
#LANGUAGE_UNKNOWN (18)
#LANGUAGE_ABKHAZIAN (19)
#LANGUAGE_A FAR (20)
#LANGUAGE_AFRIKAANS (21)
#LANGUAGE_AKAN (22)
#LANGUAGE_ALBANIAN (23)
#LANGUAGE_AMHARIC (24)
#LANGUAGE_ARABIC (25)
#LANGUAGE_ARAGONESE (26)
#LANGUAGE_ARMENIAN (27)
#LANGUAGE_ASSAMESE (28)
#LANGUAGE_AVARIC (29)
#LANGUAGE_AVESTAN (30)
#LANGUAGE_AYMARA (31)
#LANGUAGE_AZERBAIJANI (32)
#LANGUAGE_BAMBARA (33)
#LANGUAGE_BASHKIR (34)
#LANGUAGE_BASQUE (35)
#LANGUAGE_BELARUSIAN (36)
#LANGUAGE_BENGALI (37)
#LANGUAGE_BIHARI (38)
#LANGUAGE_BISLAMA (39)
#LANGUAGE_BOSNIAN (40)
#LANGUAGE_BRETON (41)
#LANGUAGE_BULGARIAN (42)
```

#LANGUAGE\_BURMESE (43)  
#LANGUAGE\_CATALAN (44)  
#LANGUAGE\_CHAMORRO (45)  
#LANGUAGE\_CHECHEN (46)  
#LANGUAGE\_CHICHEWA (47)  
#LANGUAGE\_CHINESE (48)  
#LANGUAGE\_CHUVASH (49)  
#LANGUAGE\_CORNISH (50)  
#LANGUAGE\_CORSICAN (51)  
#LANGUAGE\_CREE (52)  
#LANGUAGE\_DIVEHI (53)  
#LANGUAGE\_DZONGKHA (54)  
#LANGUAGE\_ESPERANTO (55)  
#LANGUAGE\_ESTONIAN (56)  
#LANGUAGE\_EWE (57)  
#LANGUAGE\_FAROESE (58)  
#LANGUAGE\_FIJIAN (59)  
#LANGUAGE\_FULAH (60)  
#LANGUAGE\_GALICIAN (61)  
#LANGUAGE\_GEORGIAN (62)  
#LANGUAGE\_GREENLANDIC (63)  
#LANGUAGE\_GUARANI (64)  
#LANGUAGE\_GUJARATI (65)  
#LANGUAGE\_HAITIAN (66)  
#LANGUAGE\_HAUSA (67)  
#LANGUAGE\_HEBREW (68)  
#LANGUAGE\_HERERO (69)  
#LANGUAGE\_HINDI (70)  
#LANGUAGE\_HIRIMOTU (71)  
#LANGUAGE\_INTERLINGUA (72)  
#LANGUAGE\_INDONESIAN (73)  
#LANGUAGE\_INTERLINGUE (74)  
#LANGUAGE\_IRISH (75)  
#LANGUAGE\_IGBO (76)  
#LANGUAGE\_INUPIAQ (77)  
#LANGUAGE\_IDO (78)  
#LANGUAGE\_ICELANDIC (79)  
#LANGUAGE\_INUKTITUT (80)  
#LANGUAGE\_JAPANESE (81)  
#LANGUAGE\_JAVANESE (82)  
#LANGUAGE\_KANNADA (83)  
#LANGUAGE\_KANURI (84)  
#LANGUAGE\_KASHMIRI (85)  
#LANGUAGE\_KAZAKH (86)  
#LANGUAGE\_CENTRALKHMER (87)  
#LANGUAGE\_KIKUYU (88)  
#LANGUAGE\_KINYARWANDA (89)

#LANGUAGE\_KIRGHIZ (90)  
#LANGUAGE\_KOMI (91)  
#LANGUAGE\_KONGO (92)  
#LANGUAGE\_KOREAN (93)  
#LANGUAGE\_KURDISH (94)  
#LANGUAGE\_KUANYAMA (95)  
#LANGUAGE\_LATIN (96)  
#LANGUAGE\_LUXEMBOURGISH (97)  
#LANGUAGE\_GANDA (98)  
#LANGUAGE\_LIMBURGAN (99)  
#LANGUAGE\_LINGALA (100)  
#LANGUAGE\_LAO (101)  
#LANGUAGE\_LITHUANIAN (102)  
#LANGUAGE\_LUBAKATANGA (103)  
#LANGUAGE\_LATVIAN (104)  
#LANGUAGE\_MANX (105)  
#LANGUAGE\_MACEDONIAN (106)  
#LANGUAGE\_MALAGASY (107)  
#LANGUAGE\_MALAY (108)  
#LANGUAGE\_MALAYALAM (109)  
#LANGUAGE\_MALTESE (110)  
#LANGUAGE\_MAORI (111)  
#LANGUAGE\_MARATHI (112)  
#LANGUAGE\_MARSHALLESE (113)  
#LANGUAGE\_MONGOLIAN (114)  
#LANGUAGE\_NAURU (115)  
#LANGUAGE\_NAVAJO (116)  
#LANGUAGE\_NORTHNDEBELE (117)  
#LANGUAGE\_NEPALI (118)  
#LANGUAGE\_NDONGA (119)  
#LANGUAGE\_NORWEGIANBOKMAL (120)  
#LANGUAGE\_NORWEGIANNYNORSK (121)  
#LANGUAGE\_SICHUANYI (122)  
#LANGUAGE\_SOUTHNDEBELE (123)  
#LANGUAGE\_OCCITAN (124)  
#LANGUAGE\_OJIBWA (125)  
#LANGUAGE\_CHURCHSLAVIC (126)  
#LANGUAGE\_OROMO (127)  
#LANGUAGE\_ORIYA (128)  
#LANGUAGE\_OSSETIAN (129)  
#LANGUAGE\_PANJABI (130)  
#LANGUAGE\_PALI (131)  
#LANGUAGE\_PERSIAN (132)  
#LANGUAGE\_PASHTO (133)  
#LANGUAGE\_QUECHUA (134)  
#LANGUAGE\_ROMANSH (135)  
#LANGUAGE\_RUNDI (136)

#LANGUAGE\_ROMANIAN (137)  
#LANGUAGE\_SANSKRIT (138)  
#LANGUAGE\_SARDINIAN (139)  
#LANGUAGE\_SINDHI (140)  
#LANGUAGE\_NORTHERNSAMI (141)  
#LANGUAGE\_SAMOAN (142)  
#LANGUAGE\_SANGO (143)  
#LANGUAGE\_SERBIAN (144)  
#LANGUAGE\_GAELIC (145)  
#LANGUAGE\_SHONA (146)  
#LANGUAGE\_SINHALA (147)  
#LANGUAGE\_SLOVAK (148)  
#LANGUAGE\_SLOVENIAN (149)  
#LANGUAGE\_SOMALI (150)  
#LANGUAGE\_SOUTHERNSOTHO (151)  
#LANGUAGE\_SUNDANESE (152)  
#LANGUAGE\_SWAHILI (153)  
#LANGUAGE\_SWATI (154)  
#LANGUAGE\_TAMIL (155)  
#LANGUAGE\_TELUGU (156)  
#LANGUAGE\_TAJIK (157)  
#LANGUAGE\_THAI (158)  
#LANGUAGE\_TIGRINYA (159)  
#LANGUAGE\_TIBETAN (160)  
#LANGUAGE\_TURKMEN (161)  
#LANGUAGE\_TAGALOG (162)  
#LANGUAGE\_TSWANA (163)  
#LANGUAGE\_TONGA (164)  
#LANGUAGE\_TSONGA (165)  
#LANGUAGE\_TATAR (166)  
#LANGUAGE\_TWI (167)  
#LANGUAGE\_TAHITIAN (168)  
#LANGUAGE\_UGHUR (169)  
#LANGUAGE\_UKRAINIAN (170)  
#LANGUAGE\_URDU (171)  
#LANGUAGE\_UZBEK (172)  
#LANGUAGE\_VENDA (173)  
#LANGUAGE\_VIETNAMESE (174)  
#LANGUAGE\_WALLOON (175)  
#LANGUAGE\_WELSH (176)  
#LANGUAGE\_WOLOF (177)  
#LANGUAGE\_WESTERNFRISIAN (178)  
#LANGUAGE\_XHOSA (179)  
#LANGUAGE\_YIDDISH (180)  
#LANGUAGE\_YORUBA (181)  
#LANGUAGE\_ZHUANG (182)  
#LANGUAGE\_ZULU (183)

**EINGABEN**

keine

**RÜCKGABEWERTE**

lang        Standardsprache des aktuellen Benutzers

## 34.11 OpenCatalog

**BEZEICHNUNG**

OpenCatalog – öffnet den lokalen Katalog

**ÜBERSICHT**

OpenCatalog(name\$, version)

**BESCHREIBUNG**

Dieser Befehl versucht dem in **name\$** angegebene Katalog mit der Sprache des Benutzers zu öffnen. Wenn der Katalog nicht in der Sprache des Benutzers vorhanden ist, wird dieser Befehl nicht scheitern, wenn Sie eine alternative Sprache beim Befehl **GetCatalogString()** angegeben haben. Diese wird verwendet werden, wenn kein Katalog in der Sprache des Benutzers vorhanden ist.

Standardmäßig sucht Hollywood nach Katalogen in einem Unterverzeichnis mit dem Namen "Catalogs" innerhalb des aktuellen Verzeichnis suchen. Wenn beispielsweise die Sprache des aktuellen Benutzers **#LANGUAGE\_GERMAN** ist und Sie versuchen "MyProgram.catalog" zu öffnen, wird Hollywood den Katalog an folgender Stelle suchen:

```
<current-directory>/Catalogs/deutsch/MyProgram.catalog
<current-directory>/Catalogs/german/MyProgram.catalog
```

Wenn die aktuelle Benutzersprache **#LANGUAGE\_FRENCH** ist, wird Hollywood in diesen Orten für den Katalog suchen:

```
<current-directory>/Catalogs/français/MyProgram.catalog
<current-directory>/Catalogs/french/MyProgram.catalog
```

Beachten Sie, dass Hollywood auf AmigaOS und kompatiblen Systemen auch in **Local:Catalogs** nach dem Katalog sucht.

Beachten Sie auch, dass es empfohlen wird, internationale Namen für das Unterverzeichnis der Sprache zu verwenden, d.h. "german" statt "deutsch" und "french" anstelle von "français". Die nativen Namen werden nur aus Kompatibilitätsgründen mit AmigaOS-basierten Systemen unterstützt.

Der Katalog in **name\$** kann entweder in dem von Commodore entwickelte Format **IFF CTLG** sein oder es kann in einem durch Hollywood definiertes plattformneutrales Format angegeben werden. Es wird empfohlen, das plattformneutrale Format von Hollywood zu verwenden, da **IFF CTLG** Unicode nicht unterstützt und einige andere Einschränkungen und potenzielle Kompatibilitätsprobleme hat. Siehe **Abschnitt 34.1 [Programme lokalisieren]**, **Seite 695**, für Details.

Dieser Befehl ist auch als Präprozessor-Anweisung verfügbar. Verwenden Sie **@CATALOG**, um Kataloge vom Präprozessor zu laden.

**EINGABEN**

name\$        Katalog, der geöffnet wird



**version** optional: Version, die der Katalog mindestens haben muss; wird sie weggelassen, wird dieser Befehl eine beliebige Version akzeptieren

**BEISPIEL**

```
OpenCatalog("Hollywood.catalog")
```

```
; dies ist Ihr Standard-Englisch Katalog
```

```
def$ = {}
```

```
def$[0]="Welcome to Hollywood!"
```

```
def$[1]="Written by Andreas Falkenhahn"
```

```
def$[2]="What do you want to do?"
```

```
; wenn der Hollywood.catalog nicht in der Benutzersprache
```

```
; verfügbar ist, wird die Englische Zeichenfolge benutzt
```

```
For k = 0 To 2
```

```
    c$[k]=GetCatalogString(k,def$[k])
```

```
Next
```

```
CloseCatalog()
```

Der obige Code öffnet "Hollywood.catalog" und gibt die ersten drei Einträge aus diesem Katalog aus.



## 35 Mathebibliothek

### 35.1 Abs

#### BEZEICHNUNG

Abs – gibt den Absolutwert zurück (V1.5)

#### ÜBERSICHT

```
result = Abs(val)
```

#### BESCHREIBUNG

Dieser Befehl gibt den Absolutwert von `val` zurück. Ist `val` positiv oder null, gibt dieser Befehl den gleichen Wert zurück. Ist `val` hingegen negativ, dann wird der Betrag zurückgegeben.

Siehe auch `Cast()` und `Sgn()`.

#### EINGABEN

`val`            Quellwert

#### RÜCKGABEWERTE

`result`        Absolutwert von `val`

#### BEISPIEL

```
a=Abs(124)
b=Abs(0)
c=Abs(-15.25854)
d=Abs(-2918)
```

In a wird 124, in b 0, in c 15.25854 und in d 2918 abgelegt.

### 35.2 ACos

#### BEZEICHNUNG

ACos – berechnet den Arkuskosinus (V2.0)

#### ÜBERSICHT

```
result = ACos(x)
```

#### BESCHREIBUNG

Berechnet den Arkuskosinus von `x`.

Siehe auch `ASin()`, `ATan()` und `ATan2()`.

#### EINGABEN

`x`            von deren Wert der Arkuskosinus berechnet werden soll

#### RÜCKGABEWERTE

`result`        den Arkuskosinus von `x`

## 35.3 Add

### BEZEICHNUNG

Add – addiert zwei Werte / VERALTET

### ÜBERSICHT

```
result = Add(value1, value2)
```

### BESCHREIBUNG

Addiert `value2` zu `value1` und gibt die Summe in `result` zurück. Dieser Befehl stammt noch aus der Anfangszeit von Hollywood. Sie können nun `result = value1 + value2` schreiben.

### EINGABEN

`value1`      Basiswert

`value2`      Zahl, die addiert werden soll

### RÜCKGABEWERTE

`result`      Ergebnis der Addition

### BEISPIEL

```
a=99
a=Add(a,1)
Print(a)
```

Das wird "100" auf dem Bildschirm ausgeben.

## 35.4 ASin

### BEZEICHNUNG

ASin – berechnet den Arkussinus (V2.0)

### ÜBERSICHT

```
result = ASin(x)
```

### BESCHREIBUNG

Berechnet den Arkussinus von `x`.

Siehe auch `ACos()`, `ATan()` und `ATan2()`.

### EINGABEN

`x`              von deren Wert der Arkussinus berechnet werden soll

### RÜCKGABEWERTE

`result`      den Arkussinus von `x`

## 35.5 ATan

### BEZEICHNUNG

ATan – berechnet den Arkustangens (V2.0)

### ÜBERSICHT

`result = ATan(x)`

### BESCHREIBUNG

Berechnet den Arkustangens von `x`.

Siehe auch `ACos()`, `ASin()` und `ATan2()`.

### EINGABEN

`x` von deren Wert der Arkustangens berechnet werden soll

### RÜCKGABEWERTE

`result` den Arkustangens von `x`

## 35.6 ATan2

### BEZEICHNUNG

ATan2 – berechnet den Arkustangens von `y/x` (V2.0)

### ÜBERSICHT

`result = ATan2(y, x)`

### BESCHREIBUNG

Berechnet den Arkustangens von `y/x`. Ist `x` gleich 0, gibt `ATan2()` 0 zurück.

Siehe auch `ACos()`, `ASin()` und `ATan()`.

### EINGABEN

`x` Zähler

`y` Nenner

### RÜCKGABEWERTE

`result` den Arkustangens von `y/x`

## 35.7 BitClear

### BEZEICHNUNG

BitClear – löscht ein Bit (V2.0)

### ÜBERSICHT

`n = BitClear(x, b)`

### BESCHREIBUNG

Löscht Bit Nummer `b` in `x`.

### EINGABEN

`x` Quellenwert

**b**                    Nummer des zu löschenden Bits (0-31)

### RÜCKGABEWERTE

**n**                    Ergebnis nach dem Löschen des Bits

### BEISPIEL

```
Print(BinStr(BitClear(Val("%11111111"), 2)))
```

Der Code löscht das Bit Nummer 2 von %11111111 und das Ergebnis ist %11111011

## 35.8 BitComplement

### BEZEICHNUNG

BitComplement – ergänzt und kehrt einen Wert um (V2.0)

### ÜBERSICHT

```
n = BitComplement(x)
```

### BESCHREIBUNG

Dieser Befehl invertiert alle Bits im Wert **x**. **x** wird als eine 32-Bit-Ganzzahl behandelt.

### EINGABEN

**x**                    Quellenwert

### RÜCKGABEWERTE

**n**                    umgekehrter und ergänzter Wert

### BEISPIEL

```
Print(BinStr(BitComplement(Val("%11110000"))))
```

Der obige Code wandelt den Wert %11110000 in eine 32-Bit Ganzzahl und kehrt sie um (%1111111111111111111111111111111100001111).

## 35.9 BitSet

### BEZEICHNUNG

BitSet – setzt ein Bit (V2.0)

### ÜBERSICHT

```
n = BitSet(x, b)
```

### BESCHREIBUNG

Setzt das Bit Nummer **b** im Wert **x** und gibt das Resultat in **n** zurück.

### EINGABEN

**x**                    Quellenwert

**b**                    Nummer des zu setzenden Bits (0-31)

### RÜCKGABEWERTE

**n**                    Ergebnis nach dem Setzen des Bits

**BEISPIEL**

```
Print(BinStr(BitSet(Val("%10111111"), 6)))
```

Der obige Code setzt das Bit Nummer 6 im Wert %10111111 und gibt als Resultat %11111111 zurück.

## 35.10 BitTest

**BEZEICHNUNG**

BitTest – testet, ob ein Bit gesetzt ist (V2.0)

**ÜBERSICHT**

```
bool = BitTest(x, b)
```

**BESCHREIBUNG**

Testet, ob Bit Nummer **b** in **x** gesetzt ist. Ist dies der Fall, wird **True** zurückgegeben, ansonsten **False**.

**EINGABEN**

<b>x</b>	Quellenwert
<b>b</b>	das zu testende Bit

**RÜCKGABEWERTE**

**bool**        **True** wenn das Bit gesetzt ist, sonst **False**

**BEISPIEL**

```
Print(BitTest(Val("%10101111"), 4))
```

Gibt **False** zurück, da Bit Nummer 4 in %10101111 nicht gesetzt ist.

## 35.11 BitXor

**BEZEICHNUNG**

BitXor – führt eine bitweise XOR zweier Werte durch (V2.0)

**ÜBERSICHT**

```
r = BitXor(a, b)
```

**BESCHREIBUNG**

Führt eine bitweise XOR-Operation an den beiden Werten **a** und **b** durch und gibt das Resultat in **r** zurück. Die Exklusiv-Oder-Operation wird jedes Bit in dem erhaltenen Wert nur setzen (1), wenn das entsprechende Bit in einem der Quellwerte gesetzt (1) ist. Wenn das Bit in beiden Quellwerten gesetzt (1) ist, wird es in dem erhaltenen Wert nicht eingestellt (0) werden.

**EINGABEN**

<b>a</b>	Quellenwert 1
<b>b</b>	Quellenwert 2

**RÜCKGABEWERTE**

**r** Ergebnis nach der Bitweise-Xor-Operation

**BEISPIEL**

```
Print(BinStr(BitXor(Val("%11010001"), Val("%10110010"))))
```

Führt Exklusiv-Oder auf die Werte %11010001 und %10110010 aus, was zu dem Wert %01100011 führt.

**35.12 Cast****BEZEICHNUNG**

Cast – wandelt eine Zahl in einen neuen Typ mit/ohne Vorzeichen um (V3.0)

**ÜBERSICHT**

```
result = Cast(val, sign, type)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um einen Wert zu einem anderen Typ zu konvertieren. Dies wird normalerweise nicht in Hollywood benötigt, weil Hollywood nur einen Variablentyp für Zahlen kennt (alle Zahlen in Hollywood sind als signierte 64-Bit-Fließkommawerte gespeichert, Hollywood unterscheidet intern nicht zwischen Byte, Kurz (short), Ganzzahl (integer) und Gleitkomma-Typen (float)). Sie erhalten in **result** nicht wirklich eine Variable vom angegebenen Typ, aber die Zahl wird nur in den Grenzen des Variablentyp bleiben.

Allerdings kann dieser Befehl ganz praktisch sein, wenn Sie das Vorzeichen umwandeln wollen. Vielleicht möchten Sie wissen, welche Nummer Sie erhalten, wenn 41234 zu einem signierten Short-Wert konvertiert werden soll.

**Cast()** akzeptiert drei Argumente: Das erste Argument **val** ist der Wert, der konvertiert werden soll, das zweite Argument **sign** gibt an, ob das Ergebnis ein Vorzeichen hat oder nicht (**True** = mit Vorzeichen oder **False** = ohne Vorzeichen). Das letzte Argument **type** gibt schließlich den Typ an. Dies kann **#INTEGER** (32-Bit), **#SHORT** (16 Bit) oder **#BYTE** (8-bit) sein.

Siehe auch **Abs()** und **Sgn()**.

**EINGABEN**

**val** Quellenwert  
**sign** **True** (mit Vorzeichen) oder **False** (ohne Vorzeichen)  
**type** neuer Typ für den Wert (**#INTEGER**, **#SHORT** oder **#BYTE**)

**RÜCKGABEWERTE**

**result** Ergebnis nach dem Wandeln

**BEISPIEL**

```
Print(Cast(41234, TRUE, #SHORT))
```

Der Code wandelt die Nummer 41234 in den Typ short (16-bit) mit Vorzeichen um und druckt das Ergebnis  $-(2^{16}-41234) = -24302$  aus.



### 35.13 Ceil

**BEZEICHNUNG**

Ceil – berechnet die ganzzahlige Obergrenze (V2.0)

**ÜBERSICHT**

`result = Ceil(x)`

**BESCHREIBUNG**

Berechnet die Obergrenze von `x`. Die Obergrenze eines Wertes ist die kleinste ganze Zahl, die größer oder gleich ist, z.B. die Obergrenze von 1,2 ist 2, die Obergrenze von -1,5 ist -1.

Siehe auch `Floor()` und `Round()`.

**EINGABEN**

`x`                    Wert, dessen Obergrenze berechnet werden soll

**RÜCKGABEWERTE**

`result`            Obergrenze von `x`

### 35.14 Cos

**BEZEICHNUNG**

Cos – berechnet den Kosinus (V1.5)

**ÜBERSICHT**

`result = Cos(x)`

**BESCHREIBUNG**

Berechnet den Kosinus des Winkels `x`.

Siehe auch `Sin()` und `Tan()`.

**EINGABEN**

`x`                    Winkel in Radian

**RÜCKGABEWERTE**

`result`            Kosinus von `x`

### 35.15 Deg

**BEZEICHNUNG**

Deg – konvertiert Radian in Grad (V2.0)

**ÜBERSICHT**

`result = Deg(x)`

**BESCHREIBUNG**

Konvertiert den in `x` angegebenen Radian in Grad.

Siehe auch `Rad()`.

**EINGABEN**

`x`            Wert als Radian, der in Grad konvertiert werden soll

**RÜCKGABEWERTE**

`result`       Grad von `x`

**35.16 Div****BEZEICHNUNG**

Div – dividiert einen Wert durch einen Faktor / VERALTET

**ÜBERSICHT**

```
result=Div(value1,value2)
```

**BESCHREIBUNG**

Dividiert `value1` durch `value2` und gibt das Resultat in `result` zurück.

Beachten Sie, dass obwohl die Division Fließkommagenauigkeit verwendet, `value2` nicht 0 sein darf. Wenn Sie in Fließkommazahlen durch Null dividieren müssen, verwenden Sie stattdessen `RawDiv()`. Siehe [Abschnitt 35.40 \[RawDiv\]](#), [Seite 734](#), für Details.

Dieser Befehl stammt noch aus der Anfangszeit von Hollywood. Sie können nun `result = value1 / value2` schreiben.

**EINGABEN**

`value1`       Zähler

`value2`       Nenner

**RÜCKGABEWERTE**

`result`       Ergebnis der Division

**BEISPIEL**

```
a=16
Div(a,4)
Print(a)
```

Dies wird "4" auf den Bildschirm schreiben.

**35.17 EndianSwap****BEZEICHNUNG**

EndianSwap – tauscht die Byte Reihenfolge von einem Wert (V6.0)

**ÜBERSICHT**

```
result = EndianSwap(val, bits)
```

**BESCHREIBUNG**

Dieser Befehl tauscht die Byte-Reihenfolge in `val`. Der zusätzliche Parameter `bits` gibt an, wie viele Bits berücksichtigt werden sollen und kann entweder 16 oder 32 sein.

**EINGABEN**

`val`            Eingabewert  
`bits`          Länge in Bit (16 oder 32)

**RÜCKGABEWERTE**

`result`        vertauschte Bytes

**BEISPIEL**

```
DebugPrint(HexStr(EndianSwap($ABCD, 16)))
```

Das druckt \$CDAB aus.

## 35.18 Exp

**BEZEICHNUNG**

Exp – berechnet das Exponential eines Wertes (V2.0)

**ÜBERSICHT**

`result = Exp(x)`

**BESCHREIBUNG**

Berechnet den exponentiellen Wert von  $x$  ( $e^x$ ).

**EINGABEN**

`x`            deren Wert exponentiell berechnet werden soll

**RÜCKGABEWERTE**

`result`        natürliches Exponential von  $x$

## 35.19 Floor

**BEZEICHNUNG**

Floor – berechnet die ganzzahlige Untergrenze eines Wertes (V2.0)

**ÜBERSICHT**

`result = Floor(x)`

**BESCHREIBUNG**

Berechnet die Untergrenze von  $x$ . Die Untergrenze eines Wertes ist die größte ganze Zahl, die kleiner oder gleich ist, z.B. die Untergrenze von 1,5 ist 1, von -1,5 ist -2.

Siehe auch `Ceil()` und `Round()`.

**EINGABEN**

`x`            Wert, dessen Untergrenze berechnet werden soll

**RÜCKGABEWERTE**

`result`        Untergrenze von  $x$

## 35.20 Frac

### BEZEICHNUNG

Frac – gibt den Nachkommateil einer Fließkommavariablen zurück (V1.5)

### ÜBERSICHT

```
result = Frac(val)
```

### BESCHREIBUNG

Dieser Befehl gibt den Nachkommateil einer Fließkommavariablen zurück.

Siehe auch `Int()`.

### EINGABEN

val            Quellwert

### RÜCKGABEWERTE

result       Nachkommateil von wert

### BEISPIEL

```
a = Frac(3.14156)
```

Die Variable a wird auf 0.14156 gesetzt.

## 35.21 FrExp

### BEZEICHNUNG

FrExp – berechnet die Mantisse und den Exponenten einer Fließkommazahl (V2.0)

### ÜBERSICHT

```
m, exp = FrExp(x)
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um von der Fließkommazahl `x` die Mantisse und den Exponenten zu extrahieren. Die Mantisse wird in `m`, der Exponent in `exp` abgelegt.

Siehe auch `LdExp()`.

### EINGABEN

x            Fließkommazahl

### RÜCKGABEWERTE

m            Mantisse von der Fließkommazahl

exp          Exponenten von der Fließkommazahl

## 35.22 Hypot

### BEZEICHNUNG

Hypot – berechnet die Hypotenuse (V5.0)

### ÜBERSICHT

```
h = Hypot(x, y)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die Hypotenuse eines rechtwinkligen Dreiecks zu berechnen. Sie geben die Länge der beiden Seiten in `x` und `y` an. Ein Aufruf des Befehls `Hypot()` ist das Gleiche wie die Quadratwurzel von  $x^2 + y^2$ .

**EINGABEN**

`x`            Länge der ersten Kathete (Seite a vom rechtwinkligen Dreieck)  
`y`            Länge der zweiten Kathete (Seite b vom rechtwinkligen Dreieck)

**RÜCKGABEWERTE**

`h`            Hypotenuse des rechtwinkligen Dreieck (Seite c)

### 35.23 Int

**BEZEICHNUNG**

`Int` – gibt von einer Fließkommavariablen die Stellen links des Kommas zurück (V1.5)

**ÜBERSICHT**

```
result = Int(val)
```

**BESCHREIBUNG**

Dieser Befehl gibt von einer Fließkommavariablen die Stellen links des Kommas zurück. Dieser Befehl stammt noch aus der Anfangszeit von Hollywood. Sie können nun `result = val1 \ val2` schreiben.

Siehe auch `Frac()`.

**EINGABEN**

`val`            Quellwert

**RÜCKGABEWERTE**

`result`        Ganzzahlteil von `val`

**BEISPIEL**

```
a = Int(4.5)
```

Dieser Aufruf gibt 4 zurück.

### 35.24 IsFinite

**BEZEICHNUNG**

`IsFinite` – prüft auf Endlichkeit (V9.0)

**ÜBERSICHT**

```
result = IsFinite(x)
```

**BESCHREIBUNG**

Dieser Befehl überprüft, ob `x` ein endlicher Wert ist. Ein endlicher Wert ist definiert als jeder Fließkommawert, der weder NaN noch unendlich ist.

Siehe [Abschnitt 35.26 \[IsNaN\]](#), [Seite 727](#), für Details.

Siehe [Abschnitt 35.25 \[IsInf\]](#), [Seite 726](#), für Details.

**EINGABEN**

`x`                    zu prüfender Wert

**RÜCKGABEWERTE**

`result`            True, wenn `x` ein endlicher Wert ist, sonst False

**BEISPIEL**

```
a=RawDiv(1,0) ; unendlich, nicht-unendlich
b=RawDiv(0,0) ; NaN, nicht-unendlich
c=RawDiv(5,2) ; 2.5, endlich
Print(IsFinite(a), IsFinite(b), IsFinite(c))
```

Dies wird "0 0 1" auf dem Bildschirm ausgegeben, da die ersten beiden Werte nicht endlich sind, während der letzte Wert endlich ist. Beachten Sie, dass wir hier `RawDiv()` verwenden müssen, da der Divisionsoperator sowie `Div()` keine Division durch Null zulassen.

## 35.25 IsInf

**BEZEICHNUNG**

IsInf – prüft auf Unendlichkeit (V9.0)

**ÜBERSICHT**

```
result = IsInf(x)
```

**BESCHREIBUNG**

Überprüft, ob `x` ein Unendlichkeitswert ist (positiv oder negativ). Positive und negative Unendlichkeitswerte werden bei der Division von 1/-1 durch Null im Fließkommabereich erzeugt.

Hollywood hat auch eine vordefinierte Konstante mit dem Namen `#INF`, die den Unendlichkeitswert enthält. Die bevorzugte Methode zur Überprüfung gegen Unendlichkeit ist jedoch die Verwendung vom Befehl `IsInf()`.

**EINGABEN**

`x`                    zu prüfender Wert

**RÜCKGABEWERTE**

`result`            True, wenn `x` ein Unendlichkeitswert ist, sonst False

**BEISPIEL**

```
a=RawDiv(1,0)
Print(IsInf(a))
```

Dies wird "1" auf dem Bildschirm ausgegeben, da die Division 1/0 einen Unendlichkeitswert ergibt. Beachten Sie, dass wir hier `RawDiv()` verwenden müssen, da der Divisionsoperator sowie `Div()` keine Division durch Null zulassen.

## 35.26 IsNan

### BEZEICHNUNG

IsNan – prüft, ob der Wert NaN ist (V9.0)

### ÜBERSICHT

```
result = IsNan(x)
```

### BESCHREIBUNG

Überprüft, ob `x` ein spezieller NaN-Wert ist (Not-a-Number/keine Zahl). NaN ist ein spezieller Rückgabewert für undefinierte Fließkommazahlen wie das Ergebnis von `0/0` oder die Quadratwurzel negativer Zahlen.

Beachten Sie, dass Sie nicht auf NaN testen sollten, indem Sie eine Zahl mit sich selbst vergleichen und erwarten, `False` zu erhalten. Dies funktioniert nicht auf allen Plattformen. Die Verwendung von `IsNan()` ist die einzige Methode, um zu überprüfen, ob ein Wert NaN ist.

Der Wert von NaN befindet sich ebenfalls in einer vordefinierten Konstante mit dem Namen `#NaN`. Aber aufgrund des Designs von Hollywoods Parser können Sie jedoch nur mit dem Befehl `GetConstant()` auf diesen Wert zugreifen. Die wörtliche Verwendung in einem Skript, d.h. außerhalb einer Zeichenkette, schlägt fehl.

### EINGABEN

`x` zu prüfender Wert

### RÜCKGABEWERTE

`result` True, wenn `x` ein NaN-Wert ist, sonst `False`

### BEISPIEL

```
a=RawDiv(0,0)
Print(IsNan(a))
```

Dadurch wird eine "1" auf dem Bildschirm ausgegeben, da das Ergebnis von `0/0` NaN ist. Beachten Sie, dass wir hier `RawDiv()` verwenden müssen, da der Divisionsoperator sowie `Div()` keine Division durch Null zulassen.

## 35.27 Ld

### BEZEICHNUNG

Ld – berechnet den Logarithmus zur Basis 2 (V1.5)

### ÜBERSICHT

```
result = Ld(val)
```

### BESCHREIBUNG

Dieser Befehl berechnet den Logarithmus zur Basis 2 von `val` und gibt ihn in `result` zurück.

Siehe auch `Ln()` und `Log()`.

### EINGABEN

`val` Quellwert

**RÜCKGABEWERTE**

`result`      Logarithmus zur Basis 2 von `val`

**BEISPIEL**

`a = Ld(8)`

Die Variable `a` erhält den Wert 3.

**35.28 LdExp****BEZEICHNUNG**

`LdExp` – berechnet die Fließkommazahl aus Mantisse und Exponent (V2.0)

**ÜBERSICHT**

`r = LdExp(m, exp)`

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um eine reelle Zahl aus der angegebenen Mantisse und Exponent zu berechnen. Er errechnet den Wert, der sich aus der Multiplikation von `m` und 2 hoch `exp` ergibt.

Siehe auch `FrExp()`.

**EINGABEN**

`m`              Mantissa

`exp`            Exponent

**RÜCKGABEWERTE**

`r`              Fließkommazahl

**35.29 Limit****BEZEICHNUNG**

`Limit` – begrenzt den Bereich einer Zahl (V2.0)

**ÜBERSICHT**

`n = Limit(x, low, high)`

**BESCHREIBUNG**

Begrenzt den Bereich von `x`. Wenn `x` größer oder gleich `low` und kleiner oder gleich `high` ist, wird der Wert von `x` zurückgegeben. Wenn `x` kleiner als `low` ist, so wird `low`, wenn `x` größer als `high` ist, dann wird `high` zurückgegeben.

Dieser Befehl stellt sicher, dass der Wert von `x` im Bereich der definierten Grenzen von `low` und `high` bleibt.

Siehe auch `Min()` und `Max()`.

**EINGABEN**

`x`              der zu überprüfender Wert



**RÜCKGABEWERTE**

`n` Ergebnis der Operation

**35.30 Ln****BEZEICHNUNG**

`Ln` – berechnet den natürlichen Logarithmus (zur Basis `e`) (V1.5)

**ÜBERSICHT**

```
result = Ln(val)
```

**BESCHREIBUNG**

Dieser Befehl berechnet den natürlichen Logarithmus von `val` und gibt ihn zurück (unter Verwendung der Basis `e`).

Siehe auch `Ld()` und `Log()`.

**EINGABEN**

`val` Quellwert

**RÜCKGABEWERTE**

`result` natürlicher Logarithmus von `val`

**35.31 Log****BEZEICHNUNG**

`Log` – berechnet den Logarithmus für eine beliebige Basis (V1.5)

**ÜBERSICHT**

```
result = Log(val, base)
```

**BESCHREIBUNG**

Dieser Befehl berechnet den Logarithmus von `val` zu jeder Basis und gibt das Ergebnis in `result` zurück.

Siehe auch `Ld()` und `Ln()`.

**EINGABEN**

`val` Quellwert

`base` gewünschte Basis des Logarithmus

**RÜCKGABEWERTE**

`result` Logarithmus von `val` zur Basis `base`

**BEISPIEL**

```
a = Log(100, 10)
```

Die Variable `a` erhält den Wert 2.

## 35.32 Max

### BEZEICHNUNG

Max – gibt den größten Wert zurück (V1.5)

### ÜBERSICHT

```
result = Max(a, b, ...)
```

### BESCHREIBUNG

Dieser Befehl vergleicht **a** und **b** und gibt den größeren der beiden Werte zurück.

Neu ab V2.0: Sie können nun eine beliebige Anzahl von Argumenten an diesen Befehl übergeben. Es wird immer der grösste aller Eingabewerte zurückgeben.

Siehe auch [Min\(\)](#) und [Limit\(\)](#).

### EINGABEN

**a**            Wert a

**b**            Wert b

**...**        eine beliebige Anzahl von zusätzlichen Werten

### RÜCKGABEWERTE

**result**      größter Wert

### BEISPIEL

```
a = Max(9, 10)
```

Die Variable a erhält den Wert 10.

## 35.33 Min

### BEZEICHNUNG

Min – gibt den kleinsten Wert zurück (V1.5)

### ÜBERSICHT

```
result = Min(a, b, ...)
```

### BESCHREIBUNG

Dieser Befehl vergleicht **a** und **b** und gibt den kleineren der beiden Werte zurück.

Neu ab V2.0: Sie können nun eine beliebige Anzahl von Argumenten an diesen Befehl übergeben. Es wird immer der kleinste aller Eingabewerte zurückgeben.

Siehe auch [Max\(\)](#) und [Limit\(\)](#).

### EINGABEN

**a**            Wert a

**b**            Wert b

**...**        eine beliebige Anzahl von zusätzlichen Werten

### RÜCKGABEWERTE

**result**      kleinster Wert

**BEISPIEL**

```
a = Min(9, 10)
```

Die Variable a erhält den Wert 9.

### 35.34 Mod

**BEZEICHNUNG**

Mod – berechnet den Rest (V1.5)

**ÜBERSICHT**

```
result = Mod(a, b)
```

**BESCHREIBUNG**

Dieser Befehl berechnet den Rest der Division **a** / **b**. Dieser Befehl stammt noch aus der Anfangszeit von Hollywood. Sie können nun `result = val1 % val2` schreiben.

**EINGABEN**

**a**            Zähler

**b**            Nenner

**RÜCKGABEWERTE**

**result**      Rest der Division

**BEISPIEL**

```
a = Mod(30, 4)
```

Das Resultat ist 2, da  $30 / 4 = 7$ , Rest 2.

### 35.35 Mul

**BEZEICHNUNG**

Mul – multipliziert zwei Werte / VERALTET

**ÜBERSICHT**

```
result=Mul(val1,val2)
```

**BESCHREIBUNG**

Multipliziert **val1** mit **val2** und gibt das Ergebnis in **result** zurück. Dieser Befehl stammt noch aus der Anfangszeit von Hollywood. Sie können nun `result = val1 * val2` schreiben.

**EINGABEN**

**val1**        Quellwert

**val2**        Multiplikator

**RÜCKGABEWERTE**

**result**      Ergebnis der Multiplikation

**BEISPIEL**

```
a=5
a=Mul(a,5)
Print(a)
```

Dies wird "25" auf den Bildschirm ausgeben.

**35.36 NearlyEqual****BEZEICHNUNG**

NearlyEqual – prüft Fließkommazahlen auf annähernde Gleichheit (V10.0)

**ÜBERSICHT**

```
result = NearlyEqual(x, y)
```

**BESCHREIBUNG**

Dieser Befehl vergleicht `x` und `y` und gibt `True` zurück, wenn sie nahezu gleich sind. Dieser Befehl ist nur für Fließkommazahlen sinnvoll. Der Vorteil von diesem Befehl gegenüber Hollywoods Gleichheitsoperator besteht darin, dass der Vergleich von Fließkommazahlen mit dem Gleichheitsoperator bei extrem minimalen Unterschieden auf Bitebene, z.B. verursacht durch (De-)Serialisierung. Aus diesem Grund wird empfohlen, Fließkommazahlen für mehr Zuverlässigkeit mit annähernder Gleichheit statt mit absoluter Gleichheit zu vergleichen.

**EINGABEN**

<code>x</code>	erster Operand für den Vergleich
<code>y</code>	zweiter Operand für den Vergleich

**RÜCKGABEWERTE**

<code>result</code>	<code>True</code> , wenn <code>x</code> und <code>y</code> annähernd gleich sind, sonst <code>False</code>
---------------------	--

**35.37 Pi****BEZEICHNUNG**

Pi – gibt den Wert von Pi zurück

**ÜBERSICHT**

```
result = Pi()
```

**BESCHREIBUNG**

Dieser Befehl gibt den Wert von Pi zurück.

Da Pi eine Konstante ist, ist es unnötig, diesen Befehl zu verwenden, um sie abzurufen. Stattdessen können Sie einfach Hollywoods integrierte Konstante `#PI` benutzen, um den Wert von Pi zu erhalten.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`result`      der Wert von `Pi`, entspricht der Konstante `#PI`

**BEISPIEL**

```
Print(Pi() = #PI)
```

Dies wird "1" ausgeben, da der Rückgabewert von `Pi()` derselbe ist wie `#PI`.

## 35.38 Pow

**BEZEICHNUNG**

`Pow` – berechnet  $x$  hoch  $y$  (V1.5)

**ÜBERSICHT**

```
result = Pow(x, y)
```

**BESCHREIBUNG**

Dieser Befehl berechnet die Potenz  $x$  hoch  $y$ . Dieser Befehl stammt noch aus der Anfangszeit von Hollywood. Sie können nun `result = val1 ^ val2` schreiben.

Siehe auch `Rt()` und `Sqrt()`.

**EINGABEN**

`x`              Basis

`y`              Exponent

**RÜCKGABEWERTE**

`result`      Ergebnis der Berechnung

**BEISPIEL**

```
a = Pow(2, 8)
```

Das Resultat ist 256.

## 35.39 Rad

**BEZEICHNUNG**

`Rad` – konvertiert Grad in Radian (V2.0)

**ÜBERSICHT**

```
result = Rad(x)
```

**BESCHREIBUNG**

Konvertiert den in `x` angegebenen Wert von Grad in Radian.

Siehe auch `Deg()`.

**EINGABEN**

`x`              Wert in Grad

**RÜCKGABEWERTE**

`result`      Radian von `x`

## 35.40 RawDiv

### BEZEICHNUNG

RawDiv – dividiert den Wert durch einen Faktor (V9.0)

### ÜBERSICHT

```
result = RawDiv(value1, value2)
```

### BESCHREIBUNG

Dividiert `value1` durch `value2` mit Fließkommagenauigkeit und gibt das Ergebnis zurück.

Dieser Befehl funktioniert genauso wie `Div()`, außer dass er auch eine Division durch Null zulässt, wodurch spezielle Werte wie NaN oder Unendlich generiert werden können.

Siehe [Abschnitt 35.26 \[IsNan\]](#), [Seite 727](#), für Details.

Siehe [Abschnitt 35.25 \[IsInf\]](#), [Seite 726](#), für Details.

### EINGABEN

`value1`     Zähler

`value2`     Nenner

### RÜCKGABEWERTE

`result`     Ergebnis der Division

### BEISPIEL

```
a=RawDiv(16,4)
```

```
Print(a)
```

Dadurch wird "4" auf dem Bildschirm ausgegeben.

## 35.41 Rnd

### BEZEICHNUNG

Rnd – generiert eine Zufallszahl

### ÜBERSICHT

```
result = Rnd(range)
```

### BESCHREIBUNG

Erzeugt eine zufällige Ganzzahl im Bereich von 0 bis `range` (exclusive, d.h. ohne `range`).

Siehe auch `RndF()` und `RndStrong()`.

### EINGABEN

`range`     obere Ganzzahlengrenze des Zufallsgenerators

### RÜCKGABEWERTE

`result`     eine zufällige Zahl

### BEISPIEL

```
num=Rnd(49)
```

Nun ja, ich kann nicht voraussagen welchen Wert `num` erhalten wird. Ich kann nur sagen, dass sie nicht größer als 48 und nicht kleiner als Null sein wird.

## 35.42 RndF

### BEZEICHNUNG

RndF – gibt eine zufällige Fließkommazahl zurück (V1.5)

### ÜBERSICHT

```
result = RndF()
```

### BESCHREIBUNG

Dieser Befehl gibt eine zufällige Fließkommazahl zwischen 0.0 und 1.0 zurück (exklusiv, d.h. ohne 1.0).

Beachten Sie, dass vor Hollywood 8.0 die obere Grenze von diesem Befehl mit einschließlich 1.0 dokumentiert wurde. Das war falsch. Der von RndF() zurückgegebene Wert ist garantiert kleiner als 1.0.

Siehe auch **Rnd()** und **RndStrong()**.

### EINGABEN

keine

### RÜCKGABEWERTE

**result** ein Zufallswert zwischen 0.0 und 1.0 (exklusiv, d.h. ohne 1.0)

### BEISPIEL

```
num = RndF
```

num erhält einen zufälligen Wert zwischen 0.0 und 1.0.

## 35.43 RndStrong

### BEZEICHNUNG

RndStrong – erzeugt eine kryptographisch sichere zufällige Zahl (V7.1)

### ÜBERSICHT

```
result = RndStrong(type, param)
```

### BESCHREIBUNG

Mit diesem Befehl können kryptographisch sichere Pseudozufallszahlen erzeugt werden. Die von RndStrong() zurückgegebenen Zahlen sind viel zufälliger als die, welche von **Rnd()** oder **RndF()** erzeugt werden. Deshalb sind beide nicht für kryptographisches geeignet.

RndStrong() kann in zwei verschiedenen Modi arbeiten: Wenn Sie **#INTEGER** in **type** übergeben, wird ein einzelner Ganzzahlenwert zurück gegeben, der nicht kleiner als 0 und nicht größer als die in **param** übergebenen Ganzzahl ist (aber er könnte gleich **param** sein). Wenn Sie **#STRING** in **type** übergeben, erzeugt RndStrong() eine Zeichenkette aus Zufallsbytes von **param**. D.h. wenn Sie **#STRING** übergeben, gibt **param** die gewünschte Länge der Zeichenkette an.

Seien Sie gewarnt, dass RndStrong() im Vergleich zu **Rnd()** und **RndF()** sehr langsam ist. Deshalb sollten Sie ihn nicht zu oft aufrufen, sondern die Ergebnisse zwischenspeichern, wenn Sie viele sehr zufällige Zahlen benötigen. Beispielsweise könnte man RndStrong() mit **type** auf **#STRING** und **param** auf 65536 aufrufen, um eine Zeichenkette mit 64kb

Zufallszahlen zu erzeugen. Sobald Sie diese verbraucht haben, können Sie sie den Befehl erneut aufrufen, um weitere Zufallszahlen zu erhalten.

Beachten Sie auch, dass auf AmigaOS 3.x und AROS `RndStrong()` derzeit auf `Rnd()` zurückgreift, da diese Betriebssysteme keine kryptographiefesten Zufallsgeneratoren anbieten.

Siehe auch `Rnd()` und `RndF()`.

#### EINGABEN

**type** Typ der zu erzeugenden Daten; kann entweder `#INTEGER` oder `#STRING` sein (siehe oben)

**param** größte annehmbare Ganzzahl, wenn bei **type** `#INTEGER` übergeben wurde, oder die Länge der zu erzeugenden Zeichenkette, wenn **type** `#STRING` ist.

#### RÜCKGABEWERTE

**result** Zufallszahl(en) entweder als Zeichenketten- oder Ganzzahlenwert

#### BEISPIEL

```
num=RndStrong(#INTEGER, 49)
```

Nun, ich kann nicht vorhersagen, welchen Wert `num` erhalten wird. Ich kann nur sagen, dass er nicht größer als 49 und nicht kleiner als Null ist.

## 35.44 Rol

#### BEZEICHNUNG

Rol – rotiert die Bits nach links (V3.0)

#### ÜBERSICHT

```
result = Rol(a, x[, length])
```

#### BESCHREIBUNG

Dieser Befehl rotiert von `x` Bits die Bits des Wertes `a` nach links. Bit Rotation bedeutet, dass die Bits nur innerhalb des Kreises `a` bewegt werden, das heißt Bits von der linken Seite werden sofort auf der rechten angehängt.

Beim optionale Argument `length` können Sie die Länge des Drehvorgangs angeben. Standardmäßig ist dies `#INTEGER` was bedeutet, dass `a` wird als 32-Bit-Ganzzahlenwert angesehen. Wenn Sie eine 16-Bit oder eine 8-Bit-Rotation möchten, müssen Sie jeweils `#SHORT` oder `#BYTE` verwenden.

Siehe auch `Ror()`.

#### EINGABEN

**a** Quellwert

**x** Anzahl Stellen (Bits), die rotiert werden

**length** optional: Bitlänge für diese Operation (Voreingestellt ist `#INTEGER` für 32-Bit Rotation); benutzen Sie `#SHORT` für 16-Bit und `#BYTE` für 8-Bit Rotation.

#### RÜCKGABEWERTE

**result** Wert, der rotiert wurde



**BEISPIEL**

```
r = Rol(Val("%10011110"), 4, #BYTE)
Print(BinStr(r, #BYTE))
```

Der obige Code dreht die binäre Zahl %10011110 4 Bits nach links und gibt das Ergebnis %11101001 aus.

**35.45 Ror****BEZEICHNUNG**

Ror – Rotiert die Bits nach rechts (V3.0)

**ÜBERSICHT**

```
result = Ror(a, x[, length])
```

**BESCHREIBUNG**

Dieser Befehl rotiert von **x** Bits die Bits des Wertes **a** nach rechts. Bit-Rotation bedeutet, dass die Bits nur innerhalb des Kreises **a** bewegt werden, das heißt Bits von der rechten Seite werden auf der linken angehängt.

Beim optionale Argument **length** können Sie die Länge des Drehvorgangs angeben. Standardmäßig ist dies **#INTEGER** was bedeutet, dass **a** wird als 32-Bit-Ganzzahlenwert angesehen. Wenn Sie eine 16-Bit oder eine 8-Bit-Rotation möchten, müssen Sie jeweils **#SHORT** und **#BYTE** verwenden.

Siehe auch **Rol()**.

**EINGABEN**

<b>a</b>	Quellwert
<b>x</b>	Anzahl Stellen (Bits), die rotiert werden
<b>length</b>	optional: Bitlänge für diese Operation (Voreingestellt ist <b>#INTEGER</b> für 32-Bit Rotation); benutzen Sie <b>#SHORT</b> für 16-Bit und <b>#BYTE</b> für 8-Bit Rotation.

**RÜCKGABEWERTE**

**result** Wert, der rotiert wurde

**BEISPIEL**

```
r = Ror(Val("%10011110"), 2, #BYTE)
Print(BinStr(r, #BYTE))
```

Dieser Code dreht die binäre Zahl %10011110 2 Bits nach rechts und gibt das Ergebnis von %10100111 aus.

**35.46 Round****BEZEICHNUNG**

Round – rundet eine Fließkommavariablen (V1.5)

**ÜBERSICHT**

```
result = Round(x)
```

**BESCHREIBUNG**

Dieser Befehl rundet die durch **x** angegebene Fließkommazahl und gibt die nächstliegende Ganzzahl zurück.

Siehe auch **Ceil()** und **Floor()**.

**EINGABEN**

**x** zu rundende Fließkommazahl

**RÜCKGABEWERTE**

**result** ganzzahliges Ergebnis

**BEISPIEL**

```
a = Round(3.7)
```

Dies gibt 4 zurück.

**35.47 Rt****BEZEICHNUNG**

Rt – berechnet eine Wurzel (V1.5)

**ÜBERSICHT**

```
result = Rt(x, y)
```

**BESCHREIBUNG**

Dieser Befehl berechnet die **y**-Wurzel des Werts **x** und gibt das Ergebnis zurück.

Siehe auch **Pow()** und **Sqrt()**.

**EINGABEN**

**x** Quellwert

**y** zu berechnende Wurzel

**RÜCKGABEWERTE**

**result** y-Wurzel von x

**BEISPIEL**

```
a = Rt(27, 3)
```

Die Variable a erhält den Wert 3.

**35.48 Sar****BEZEICHNUNG**

Sar – verschiebt Bits nach rechts (V3.0)

**ÜBERSICHT**

```
result = Sar(a, x[, bignum])
```

**BESCHREIBUNG**

Dieser Befehl verschiebt den Wert `a` `x` Bits nach rechts, Bitlöcher werden mit dem höchstwertigen Bit von `a` aufgefüllt (Dies wird als arithmetische Verschiebung bezeichnet). `a` wird vor der Verschiebung in eine 32-Bit-Ganzzahlvariable umgewandelt (es sei denn, `bignum` ist auf `True` gesetzt).

Ab Hollywood 9.0 gibt es ein optionales Argument `bignum`. Wenn dies auf `True` gesetzt ist, kann `Sar()` mit ganzen Zahlen arbeiten, die größer als  $2^{31}$  sind. Aber denken Sie daran, dass `Sar()` immer noch nicht den vollen 64-Bit-Ganzzahlbereich verwenden kann, da Hollywoods numerischer Typ eine 64-Bit-Fließkommazahl und daher auf ganze Zahlen im Bereich von `[-9007199254740992, 9007199254740992]` beschränkt ist.

Siehe auch `Shl()` und `Shr()`.

**EINGABEN**

<code>a</code>	Quellwert
<code>x</code>	Anzahl Stellen, um die der Wert geschoben werden soll
<code>bignum</code>	optional: ob 64-Bit-Ganzzahlen verwendet werden sollen oder nicht (Standardwert <code>False</code> ) (V9.0)

**RÜCKGABEWERTE**

<code>result</code>	Ergebnis als Ganzzahl
---------------------	-----------------------

**BEISPIEL**

```
a = Sar(-256, 3)
```

Dies wird -32 zurückgeben.

## 35.49 Sgn

**BEZEICHNUNG**

`Sgn` – ermittelt das Vorzeichen eines Wertes (V2.0)

**ÜBERSICHT**

```
sign = Sgn(x)
```

**BESCHREIBUNG**

Ermittelt das Vorzeichen von `x`. Ist `x` unter 0 wird -1, bei `x` gleich 0 wird 0 und bei `x` größer als 0 wird 1 zurückgegeben.

Siehe auch `Abs()` und `Cast()`.

**EINGABEN**

<code>x</code>	Wert
----------------	------

**RÜCKGABEWERTE**

<code>sign</code>	Vorzeichen von <code>x</code>
-------------------	-------------------------------

## 35.50 Shl

### BEZEICHNUNG

Shl – schiebt Bits nach links (V1.5)

### ÜBERSICHT

```
result = Shl(a, x[, bignum])
```

### BESCHREIBUNG

Dieser Befehl schiebt **a** um **x** Bits nach links, wobei die Löcher mit Nullbits aufgefüllt werden (dies wird als logische Verschiebung bezeichnet). **a** wird vor der Verschiebung in eine vorzeichenlose 32-Bit-Ganzzahlen-Variable umgewandelt (es sei denn, **bignum** ist auf **True** gesetzt).

Ab Hollywood 9.0 gibt es ein optionales Argument **bignum**. Wenn dies auf **True** gesetzt ist, kann **Shl()** mit ganzen Zahlen arbeiten, die größer als  $2^{31}$  sind. Aber denken Sie daran, dass **Shl()** immer noch nicht den vollen 64-Bit-Ganzzahlbereich verwenden kann, da Hollywoods numerischer Typ eine 64-Bit-Fließkommazahl und daher auf ganze Zahlen im Bereich von  $[-9007199254740992, 9007199254740992]$  beschränkt ist.

Siehe auch **Sar()** und **Shr()**.

### EINGABEN

<b>a</b>	Quellwert
<b>x</b>	Anzahl Stellen, um die der Wert geschoben werden soll
<b>bignum</b>	optional: ob 64-Bit-Ganzzahlen verwendet werden sollen oder nicht (Standardwert <b>False</b> ) (V9.0)

### RÜCKGABEWERTE

<b>result</b>	ganzzahliges Ergebnis
---------------	-----------------------

### BEISPIEL

```
a = Shl(256, 3)
```

Dies wird 2048 zurückgeben.

## 35.51 Shr

### BEZEICHNUNG

Shr – schiebt Bits nach rechts (V1.5)

### ÜBERSICHT

```
result = Shr(a, x[, bignum])
```

### BESCHREIBUNG

Dieser Befehl schiebt **a** um **x** Stellen nach rechts, wobei die Löcher mit Nullbits aufgefüllt werden (dies wird als logische Verschiebung bezeichnet). **a** wird vor der Verschiebung in eine vorzeichenlose 32-Bit-Ganzzahlen-Variable umgewandelt (es sei denn, **bignum** ist auf **True** gesetzt).

Ab Hollywood 9.0 gibt es ein optionales Argument **bignum**. Wenn dies auf **True** gesetzt ist, kann **Shr()** mit ganzen Zahlen arbeiten, die größer als  $2^{31}$  sind. Aber denken Sie

darán, dass `Shr()` immer noch nicht den vollen 64-Bit-Ganzzahlbereich verwenden kann, da Hollywoods numerischer Typ eine 64-Bit-Fließkommazahl und daher auf ganze Zahlen im Bereich von `[-9007199254740992, 9007199254740992]` beschränkt ist.

Siehe auch `Sar()` und `Shl()`.

#### EINGABEN

<code>a</code>	Quellwert
<code>x</code>	Anzahl Stellen, um die der Wert geschoben werden soll
<code>bignum</code>	optional: ob 64-Bit-Ganzzahlen verwendet werden sollen oder nicht (Standardwert <code>False</code> ) (V9.0)

#### RÜCKGABEWERTE

<code>result</code>	ganzzahliges Ergebnis
---------------------	-----------------------

#### BEISPIEL

```
a = Shr(256, 3)
```

Dies wird 32 zurückgeben.

## 35.52 Sin

#### BEZEICHNUNG

`Sin` – berechnet den Sinus (V1.5)

#### ÜBERSICHT

```
result = Sin(x)
```

#### BESCHREIBUNG

Berechnet den Sinus des Winkels `x`.

Siehe auch `Cos()` und `Tan()`.

#### EINGABEN

<code>x</code>	Winkel in Radian
----------------	------------------

#### RÜCKGABEWERTE

<code>result</code>	Sinus von <code>x</code>
---------------------	--------------------------

## 35.53 Sqrt

#### BEZEICHNUNG

`Sqrt` – berechnet die Quadratwurzel (V1.5)

#### ÜBERSICHT

```
result = Sqrt(x)
```

#### BESCHREIBUNG

Dieser Befehl berechnet die Quadratwurzel von `x` und gibt das Ergebnis in `result` zurück.

Siehe auch `Rt()` und `Pow()`.

**EINGABEN**

`x`            Quellwert

**RÜCKGABEWERTE**

`result`      Quadratwurzel von `x`

**BEISPIEL**

`a = Sqrt(64)`

Dies wird 8 zurückgeben.

## 35.54 Sub

**BEZEICHNUNG**

Sub – subtrahiert einen Wert von einem anderen / VERALTET

**ÜBERSICHT**

`result = Sub(val1,val2)`

**BESCHREIBUNG**

Subtrahiert `val2` von `val1` und gibt das Ergebnis in der Variable `result` zurück. Dieser Befehl stammt noch aus der Anfangszeit von Hollywood. Sie können nun `result = val1 - val2` schreiben.

**EINGABEN**

`val1`            Basiswert

`val2`            Wert, der abgezogen werden soll

**RÜCKGABEWERTE**

`result`      Ergebnis der Subtraktion

**BEISPIEL**

`a=1`

`a=Sub(a,1)`

`Print(a)`

Das Beispiel schreibt "0" auf den Bildschirm.

## 35.55 Tan

**BEZEICHNUNG**

Tan – berechnet den Tangens (V1.5)

**ÜBERSICHT**

`result = Tan(x)`

**BESCHREIBUNG**

Berechnet den Tangens des Winkels `x`.

`Sin()` und `Cos()`.

**EINGABEN**

`x`            Winkel in Radian

**RÜCKGABEWERTE**

`result`      Tangens von `x`

**35.56 Wrap****BEZEICHNUNG**

Wrap – bricht einen Wert um (V1.5)

**ÜBERSICHT**

`result = Wrap(x,low,high)`

**BESCHREIBUNG**

Das Ergebnis von Wrap ist `x`, umgebrochen an: `high`, falls `x` größer oder gleich `high` ist. `low`, falls `x` kleiner `low` ist. Ist der Wert kleiner als die `low`, wird `x-low+high` zurückgegeben. Ist der Wert größer oder gleich der `high`, wird `x-high+low` zurückgegeben.

**EINGABEN**

`x`            umzubrechender Wert

`low`          unterer Grenzwert

`high`        oberer Grenzwert

**RÜCKGABEWERTE**

`result`      Überlauf von `x`, `low` und `high`





## 36 Mauszeigerbibliothek

### 36.1 CreatePointer

#### BEZEICHNUNG

CreatePointer – erstellt einen neuen Mauszeiger (V4.0)

#### ÜBERSICHT

```
[id] = CreatePointer(id, type, ...)
[id] = CreatePointer(id, #SPRITE, srcid[, frame, spotx, spoty])
[id] = CreatePointer(id, #BRUSH, srcid[, spotx, spoty])
[id] = CreatePointer(id, #POINTER, ptrtype)
```

#### BESCHREIBUNG

Dieser Befehl erzeugt einen neuen Mauszeiger und weist die in `id` angegebene ID zu. Wenn Sie `Nil` als `id` übergeben, wird `CreatePointer()` automatisch eine ID wählen und zurückgeben. Der von diesem Befehl erstellte Mauszeiger kann später durch Aufruf von `SetPointer()` angezeigt werden. Mauszeiger können entweder aus einer Sprite- oder Pinselquelle erstellt werden oder Sie können einen vordefinierten Mauszeiger auswählen. Die Art des Aufrufs von `CreatePointer()` hängt vom Typ ab, den Sie als zweites Argument angeben.

Für die Typen `#SPRITE` und `#BRUSH` müssen Sie in `srcid` die ID des Objekts angeben, die als Quelle für die Zeigergrafiken verwendet werden soll. Der Mauszeiger, der von diesem Befehl erzeugt wird, ist unabhängig vom Quellobjekt, so dass Sie das Quellobjekt nach dem Aufruf von `CreatePointer()` löschen können.

Wenn Sie `#POINTER` als Typ angeben, müssen Sie ein zusätzliches Argument angeben, welches vordefinierte Mauszeigerbild Sie verwenden möchten. Derzeit kann dies `#STDPTR_SYSTEM` für den Standard-Systemmauszeiger und `#STDPTR_BUSY` für den Standard-Wartemauszeiger sein.

Mit den Argumenten `spotx` und `spoty` wird der aktive Zeigepunkt (Hotspot) innerhalb des Mauszeigers definiert. Der aktive Zeigepunkt ist der Mauszeigerpixel, welcher zum Klicken verwendet wird. Wenn das Mauszeigerbild ein Pfeil ist, dann ist der aktive Zeigepunkt normalerweise genau an der Spitze des Pfeils. Wenn Sie die `spotx` und `spoty` Argumente nicht angeben, wird `CreatePointer()` die Mitte des Bildes als aktiver Zeigepunkt verwenden.

Bitte beachten Sie, dass nicht alle Systeme Mauszeiger mit Echtfarben verarbeitet können. Wenn das System dies nicht unterstützt, wird Hollywood die Farben reduzieren. Außerdem werden Ihre Bilddaten möglicherweise skaliert, da einige Systeme Grenzen für die maximale Mauszeigergröße setzen.

Unter AmigaOS 3 unterstützt `CreatePointer()` auch Palettenpinsel und Sprites. Mauszeiger auf klassischer Amiga-Hardware sind immer palettenbasiert, da sie mit Hardware-Sprites implementiert sind. Wenn Sie also Palettenpinsel oder Sprites an `CreatePointer()` auf AmigaOS 3 übergeben, haben Sie die volle Kontrolle über die genauen Stifte, die vom Mauszeiger verwendet werden, was bequemer ist als die Verwendung von 32-Bit-Grafiken, da diese zuerst auf Palettengrafiken unter AmigaOS 3 abgebildet werden müssen und Sie keine Kontrolle über die Palettenstifte im neu zugewiesenen Pinsel oder Sprite haben.

**EINGABEN**

`id` ID für den Mauszeiger oder `Nil` für die automatische ID-Zuweisung  
`type` Typ, von welchem die Quelldaten zu nehmen sind  
`...` Weitere Argumente; hängen vom angegebenen Typ ab (siehe oben)

**RÜCKGABEWERTE**

`id` optional: ID des Mauszeigers; wird nur zurückgegeben, wenn Sie `Nil` in `id` weitergeben (siehe oben)

**BEISPIEL**

```
CreatePointer(1, #BRUSH, 2, 0, 0)
SetPointer(1)
```

Der obige Code erzeugt einen neuen Mauszeiger 1 vom Pinsel mit der ID 2. Der aktive Zeigepunkt befindet sich auf der Position 0:0 (d.h. die linke obere Ecke des Zeigers). Nach dem Erstellen des Mauszeigers wird er mit `SetPointer()` angezeigt.

## 36.2 FreePointer

**BEZEICHNUNG**

`FreePointer` – löscht den Mauszeiger aus dem Arbeitsspeicher (V4.0)

**ÜBERSICHT**

```
FreePointer(id)
```

**BESCHREIBUNG**

Dieser Befehl löscht den mit in `id` angegebenen Mauszeiger aus dem Arbeitsspeicher. Der Mauszeiger muss zuvor mit `CreatePointer()` erstellt worden sein. Bitte beachten Sie, dass der Mauszeiger momentan nicht aktiv sein darf. Sie dürfen nur Mauszeiger löschen, die derzeit nicht angezeigt werden.

**EINGABEN**

`id` ID des Mauszeigers der frei geben werden soll

## 36.3 HidePointer

**BEZEICHNUNG**

`HidePointer` – blendet den Mauszeiger im aktuellen Display aus

**ÜBERSICHT**

```
HidePointer()
```

**BESCHREIBUNG**

Mit diesem Befehl wird der Mauszeiger ausgeblendet. Verwenden Sie diesen Befehl mit Sorgfalt, weil Sie den Benutzer verwirren könnten. Verwenden Sie `ShowPointer()`, um den Zeiger zurück auf das Display zu bringen.

Bitte beachten Sie, dass jedes Display seine eigene Zeigereinstellung hat. So wird dieser Befehl nur den Mauszeiger im aktuellen Display ausblenden. Wenn der Benutzer ein anderes Display aktiviert, ist der Mauszeiger erneut sichtbar.

**EINGABEN**

keine

## 36.4 MovePointer

**BEZEICHNUNG**

MovePointer – bewegt den Mauszeiger

**ÜBERSICHT**

MovePointer(x, y)

**BESCHREIBUNG**

Dieser Befehl bewegt den Mauszeiger an die durch **x/y** angegebene Position. Benutzen Sie diesen Befehl mit Vorsicht, sie könnte den Benutzer verwirren.

**EINGABEN**

**x** gewünschte neue x-Position des Zeigers

**y** gewünschte neue y-Position des Zeigers

**BEISPIEL**

MovePointer(#CENTER, #CENTER)

Der obige Code verschiebt den Zeiger in die Mitte des Bildschirms.

## 36.5 SetPointer

**BEZEICHNUNG**

SetPointer – ändert den Mauszeiger des aktuellen Displays (V4.0)

**ÜBERSICHT**

SetPointer(id)

**BESCHREIBUNG**

Dieser Befehl zeigt den durch die **id** angegebenen Mauszeiger an. Der Mauszeiger muss zuvor mit **CreatePointer()** erstellt worden sein.

Bitte beachten Sie, dass jedes Display seine eigene Zeigereinstellung hat. Somit wird mit diesem Befehl nur der Mauszeiger auf dem aktuellen Display gesetzt. Wenn Sie den Mauszeiger aller Ihrer Displays ändern möchten, müssen Sie **SetPointer()** für jedes von ihnen aufrufen (nach jeder Aktivierung mit dem Befehl **SelectDisplay()**).

Hinweis: Dieser Befehl ist bereits seit Version 1.5 verfügbar, aber seine Funktionalität wurde in Version 4.0 komplett geändert. Der alte Befehl wird nicht mehr unterstützt.

**EINGABEN**

**id** ID des anzuzeigenden Mauszeigers

**BEISPIEL**

Siehe **Abschnitt 36.1 [CreatePointer]**, Seite 745.

## 36.6 ShowPointer

### BEZEICHNUNG

ShowPointer – zeigt den Mauszeiger im aktuellen Display

### ÜBERSICHT

ShowPointer()

### BESCHREIBUNG

Dieser Befehl bringt den Mauszeiger zurück, nachdem er mit dem Befehl `HidePointer()` ausgeblendet wurde.

### EINGABEN

keine

## 37 Menübibliothek

### 37.1 CreateMenu

#### BEZEICHNUNG

CreateMenu – erstellt eine Menüliste (V6.0)

#### ÜBERSICHT

```
[id] = CreateMenu(id, table)
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um eine Menüleiste zu erstellen, die später an einem oder mehreren Displays zugewiesen werden kann. Dies geschieht bei einem bereits vorhandenen Display mit dem Befehl `SetDisplayAttributes()`. Über den `Menu`-Tag mit der Präprozessor-Anweisung `@DISPLAY` oder dem Befehl `CreateDisplay()` kann die Menüleiste einem neuen Display zugeteilt werden.

Sie müssen eine ID oder `Nil` für die neue Menüleiste übergeben. Wenn Sie `Nil` angeben, wird `CreateMenu()` automatisch eine freie ID für Sie auswählen.

Sie müssen auch eine Tabelle übergeben, die die aktuelle Menübaumdefinition für diesen Befehl enthält. Menüs werden als Baumstruktur definiert, die von einer Haupttabelle zusammengesetzt ist, welche verschiedene Untertabellen enthält. Siehe [Abschnitt 37.8 \[MENU\]](#), [Seite 754](#), für eine detaillierte Beschreibung der Menübaumtabellen.

Dieser Befehl ist auch als Präprozessor vorhanden: Verwenden Sie `@MENU` um Menüleisten beim Start zu erstellen!

#### EINGABEN

<code>id</code>	Identifikator für die Menüleiste oder <code>Nil</code> für die automatische ID-Zuweisung
<code>table</code>	Definition des Menübaumes

#### RÜCKGABEWERTE

<code>id</code>	optional: Identifikator des neuen Menüs; wird nur zurückgegeben, wenn <code>Nil</code> als Argument 1 übergeben wurde (siehe oben)
-----------------	--

#### BEISPIEL

```
CreateMenu(1, {
  {"File", {
    {"New", ID = "new"},
    {"Open...", ID = "open"},
    {""},
    {"Close", ID = "close", Flags = #MENUITEM_DISABLED},
    {""},
    {"Save", Flags = #MENUITEM_DISABLED, Hotkey = "S"},
    {"Compress", ID = "cmp", Flags = #MENUITEM_TOGGLE},
    {""},
    {"Export image...", {
      {"JPEG...", ID = "jpeg"},
      {"PNG...", ID = "png"},
    }
  }
})
```

```

        {"BMP...", ID = "bmp"}}},
{" "},
{"Dump state", ID = "dump"},
{" "},
{"Quit", ID = "quit", Hotkey = "Q"}}},

{"Edit", {
    {"Cut", ID = "cut"},
    {"Copy", ID = "copy"},
    {"Paste", ID = "paste"}}},

{"?", {
    {"About...", ID = "about"}}}
})

```

```
SetDisplayAttributes({Menu = 1})
```

Der obige Code erstellt eine Menüleiste und fügt es in das aktuelle Display ein.

## 37.2 DeselectMenuItem

### BEZEICHNUNG

DeselectMenuItem – wählt ein Toggle-Menüpunkt ab (V6.0)

### ÜBERSICHT

```
DeselectMenuItem(id, item$[, detached])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen Toggle-Menüpunkt abzuwählen. Das optionale Argument **detached** gibt an, ob eine an ein Display angehängte Menüleiste oder eine losgelöste Menüleiste verwendet werden soll. Wenn **detached False** ist (was auch die Voreinstellung ist), wird die Menüleiste des durch **id** angegebenen Displays verwendet. Wenn **detached True** ist, wird die durch **id** angegebene Menüleiste verwendet. Mit anderen Worten: Wenn Sie **detached** auf **True** setzen, müssen Sie in **id** den Identifikator einer Menüleiste übergeben; andernfalls müssen Sie den Identifikator eines Displays in **id** übergeben.

Beachten Sie, dass beim Setzen von **detached** auf **True** Ihre Operation niemals Auswirkungen auf Menüleisten hat, die mit einem Display verbunden sind. Das Setzen von **detached** auf **True** wird normalerweise nur bei Menüleisten verwendet, die als Popup-Menüs mit dem Befehl **PopupMenu()** angezeigt werden. Es ist unmöglich, Display-Menüleisten anzusprechen, wenn **detached** auf **True** gesetzt wird, da eine einzelne Menüleiste an mehrere Displays angehängt werden kann.

Unter AmigaOS 4 können Sie für **id** auch den Sonderwert 0 übergeben. In diesem Fall wird das Kontextmenü des Docky verwendet.

### EINGABEN

<b>id</b>	ID eines Displays, welches die Menüliste zugewiesen wurde oder 0 (siehe oben)
-----------	---

**item\$**      Identifikator des Menüpunktes innerhalb der Menüleiste

**detached**   optional: **False**, wenn **id** ein Display angibt; **True**, wenn es ein Menüleistenobjekt angibt (voreingestellt ist **False**) (V10.0)

### 37.3 DisableMenuItem

#### BEZEICHNUNG

DisableMenuItem – deaktiviert einen Menüpunkt (V6.0)

#### ÜBERSICHT

DisableMenuItem(id, item\$[, detached])

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen Menüpunkt zu deaktivieren. Das optionale Argument **detached** gibt an, ob eine an ein Display angehängte Menüleiste oder eine losgelöste Menüleiste verwendet werden soll. Wenn **detached False** ist (was auch die Voreinstellung ist), wird die Menüleiste des durch **id** angegebenen Displays verwendet. Wenn **detached True** ist, wird die durch **id** angegebene Menüleiste verwendet. Mit anderen Worten: Wenn Sie **detached** auf **True** setzen, müssen Sie in **id** den Identifikator einer Menüleiste übergeben; andernfalls müssen Sie den Identifikator eines Displays in **id** übergeben.

Beachten Sie, dass beim Setzen von **detached** auf **True** Ihre Operation niemals Auswirkungen auf Menüleisten hat, die mit einem Display verbunden sind. Das Setzen von **detached** auf **True** wird normalerweise nur bei Menüleisten verwendet, die als Popup-Menüs mit dem Befehl **PopupMenu()** angezeigt werden. Es ist unmöglich, Display-Menüleisten anzusprechen, wenn **detached** auf **True** gesetzt wird, da eine einzelne Menüleiste an mehrere Displays angehängt werden kann.

Unter AmigaOS 4 können Sie für **id** auch den Sonderwert 0 übergeben. In diesem Fall wird das Kontextmenü des Docky verwendet.

#### EINGABEN

**id**            ID eines Displays, welches die Menüleiste zugewiesen wurde oder 0 (siehe oben)

**item\$**      Identifikator des Menüpunktes innerhalb der Menüleiste

**detached**   optional: **False**, wenn **id** ein Display angibt; **True**, wenn es ein Menüleistenobjekt angibt (voreingestellt ist **False**) (V10.0)

### 37.4 EnableMenuItem

#### BEZEICHNUNG

EnableMenuItem – aktiviert einen Menüpunkt (V6.0)

#### ÜBERSICHT

EnableMenuItem(id, item\$[, detached])

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um einen Menüpunkt zu aktivieren. Das optionale Argument **detached** gibt an, ob eine an ein Display angehängte Menüleiste oder eine losgelöste Menüleiste verwendet werden soll. Wenn **detached False** ist (was auch die Voreinstellung ist), wird die Menüleiste des durch **id** angegebenen Displays verwendet. Wenn **detached True** ist, wird die durch **id** angegebene Menüleiste verwendet. Mit anderen Worten: Wenn Sie **detached** auf **True** setzen, müssen Sie in **id** den Identifikator einer Menüleiste übergeben; andernfalls müssen Sie den Identifikator eines Displays in **id** übergeben.

Beachten Sie, dass beim Setzen von **detached** auf **True** Ihre Operation niemals Auswirkungen auf Menüleisten hat, die mit einem Display verbunden sind. Das Setzen von **detached** auf **True** wird normalerweise nur bei Menüleisten verwendet, die als Popup-Menüs mit dem Befehl **PopupMenu()** angezeigt werden. Es ist unmöglich, Display-Menüleisten anzusprechen, wenn **detached** auf **True** gesetzt wird, da eine einzelne Menüleiste an mehrere Displays angehängt werden kann.

Unter AmigaOS 4 können Sie für **id** auch den Sonderwert 0 übergeben. In diesem Fall wird das Kontextmenü des Docky verwendet.

**EINGABEN**

<b>id</b>	ID eines Displays, welches die Menüleiste zugewiesen wurde oder 0 (siehe oben)
<b>item\$</b>	Identifikator des Menüpunktes innerhalb der Menüleiste
<b>detached</b>	optional: <b>False</b> , wenn <b>id</b> ein Display angibt; <b>True</b> , wenn es ein Menüleistenobjekt angibt (voreingestellt ist <b>False</b> ) (V10.0)

**37.5 FreeMenu****BEZEICHNUNG**

FreeMenu – löscht die Menüleiste aus dem Speicher (V6.0)

**ÜBERSICHT**

**FreeMenu(id)**

**BESCHREIBUNG**

Dieser Befehl kann zum Löschen der angegebenen Menüleiste aus dem Speicher verwendet werden. Bitte beachten Sie, dass nur die Menüleisten, die nicht mehr an ein Display zugewiesen sind, gelöscht werden können. Um eine Menüleiste von einem Display zu lösen, rufen Sie den Befehl **SetDisplayAttributes()** auf und übergeben dem Display über den **Menu**-Tag den besonderen Wert -1.

**EINGABEN**

<b>id</b>	ID eines Displays, dessen Menüleiste gelöscht wird
-----------	--



## 37.6 IsMenuItemDisabled

### BEZEICHNUNG

IsMenuItemDisabled – überprüft, ob ein Menüpunkt deaktiviert ist (V6.0)

### ÜBERSICHT

```
result = IsMenuItemDisabled(id, item$[, detached])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um zu überprüfen, ob ein Menüpunkt deaktiviert ist oder nicht. Das optionale Argument **detached** gibt an, ob eine an ein Display angehängte Menüleiste oder eine losgelöste Menüleiste verwendet werden soll. Wenn **detached False** ist (was auch die Voreinstellung ist), wird die Menüleiste des durch **id** angegebenen Displays verwendet. Wenn **detached True** ist, wird die durch **id** angegebene Menüleiste verwendet. Mit anderen Worten: Wenn Sie **detached** auf **True** setzen, müssen Sie in **id** den Identifikator einer Menüleiste übergeben; andernfalls müssen Sie den Identifikator eines Displays in **id** übergeben.

Beachten Sie, dass beim Setzen von **detached** auf **True** Ihre Operation niemals Auswirkungen auf Menüleisten hat, die mit einem Display verbunden sind. Das Setzen von **detached** auf **True** wird normalerweise nur bei Menüleisten verwendet, die als Popup-Menüs mit dem Befehl **PopupMenu()** angezeigt werden. Es ist unmöglich, Display-Menüleisten anzusprechen, wenn **detached** auf **True** gesetzt wird, da eine einzelne Menüleiste an mehrere Displays angehängt werden kann.

Unter AmigaOS 4 können Sie für **id** auch den Sonderwert 0 übergeben. In diesem Fall wird das Kontextmenü des Docky verwendet.

### EINGABEN

<b>id</b>	ID eines Displays, welches die Menüleiste zugewiesen wurde oder 0 (siehe oben)
<b>item\$</b>	Identifikator des Menüpunktes innerhalb der Menüleiste
<b>detached</b>	optional: <b>False</b> , wenn <b>id</b> ein Display angibt; <b>True</b> , wenn es ein Menüleistenobjekt angibt (voreingestellt ist <b>False</b> ) (V10.0)

### RÜCKGABEWERTE

**result**      **True** wenn der Menüpunkt deaktiviert ist, andernfalls **False**

## 37.7 IsMenuItemSelected

### BEZEICHNUNG

IsMenuItemSelected – überprüft, ob ein Menüpunkt ausgewählt ist (V6.0)

### ÜBERSICHT

```
result = IsMenuItemSelected(id, item$[, detached])
```

### BESCHREIBUNG

Mit diesem Befehl kann überprüft werden, ob ein Toggle- oder Radio-Menüpunkt ausgewählt ist oder nicht. Das optionale Argument **detached** gibt an, ob eine an ein Display angehängte Menüleiste oder eine losgelöste Menüleiste verwendet werden soll. Wenn

**detached** **False** ist (was auch die Voreinstellung ist), wird die Menüleiste des durch **id** angegebenen Displays verwendet. Wenn **detached** **True** ist, wird die durch **id** angegebene Menüleiste verwendet. Mit anderen Worten: Wenn Sie **detached** auf **True** setzen, müssen Sie in **id** den Identifikator einer Menüleiste übergeben; andernfalls müssen Sie den Identifikator eines Displays in **id** übergeben.

Beachten Sie, dass beim Setzen von **detached** auf **True** Ihre Operation niemals Auswirkungen auf Menüleisten hat, die mit einem Display verbunden sind. Das Setzen von **detached** auf **True** wird normalerweise nur bei Menüleisten verwendet, die als Popup-Menüs mit dem Befehl **PopupMenu()** angezeigt werden. Es ist unmöglich, Display-Menüleisten anzusprechen, wenn **detached** auf **True** gesetzt wird, da eine einzelne Menüleiste an mehrere Displays angehängt werden kann.

Unter AmigaOS 4 können Sie für **id** auch den Sonderwert 0 übergeben. In diesem Fall wird das Kontextmenü des Docky verwendet.

#### EINGABEN

<b>id</b>	ID eines Displays, welches die Menüleiste zugewiesen wurde oder 0 (siehe oben)
<b>item\$</b>	Identifikator des Menüpunktes innerhalb der Menüleiste
<b>detached</b>	optional: <b>False</b> , wenn <b>id</b> ein Display angibt; <b>True</b> , wenn es ein Menüleistenobjekt angibt (voreingestellt ist <b>False</b> ) (V10.0)

#### RÜCKGABEWERTE

<b>result</b>	<b>True</b> wenn der Menüpunkt ausgewählt ist, andernfalls <b>False</b>
---------------	---

## 37.8 MENU

### BEZEICHNUNG

**MENU** – erstellt eine Menüleiste (V6.0)

### ÜBERSICHT

@MENU **id**, **table**

### BESCHREIBUNG

Diese Präprozessor-Anweisung kann verwendet werden, um eine Menüleiste zu erstellen, die später an ein oder mehreren Displays zugewiesen werden kann. Dies geschieht bei einem bereits vorhandenen Display mit dem Befehl **SetDisplayAttributes()**. Über den **Menu**-Tag mit der Präprozessor-Anweisung **@DISPLAY** oder dem Befehl **CreateDisplay()** kann die Menüleiste einem neuen Display zugeteilt werden.

Sie müssen dieser Präprozessor-Anweisung eine ID für die Menüleiste sowie die tatsächlichen Menüdefinition übergeben. Menüs werden als Baumstruktur definiert, die eine Haupttabelle mit verschiedenen Untertabellen enthält. Es gibt zwei verschiedene Arten von Untertabellen:

1. Menü-Tabellen: Diese Tabellen enthalten eine Überschrift für das Menü im Tabellenelement Index 0 und eine Liste der einzelnen Menüpunkte im Tabellenelement Index 1. Die Liste der einzelnen Menüpunkte ist natürlich eine andere untergeordnete Tabelle, die von einer anderen Zahl Tabellen zusammengesetzt ist, um jeden Menüpunkt zu beschreiben (siehe unten). Die Tabelle muss mit einer Reihe

von Menütabellen gestartet werden, da jeder Menüpunkt ein übergeordnetes Menü benötigt. Diese übergeordneten Menüs werden in den Menütabellen beschrieben. Es ist auch möglich, ein Untermenü unter einer Anzahl von Menüelementen einzufügen.

2. Menüpunkt-Tabellen: Eine Menüpunktuntertabelle ist eine Tabelle, die ein einzelner Menüpunkt beschreibt. Der Name des Menüpunkt, der im Menü angezeigt werden soll, steht bei Tabellenindex 0. Wenn Sie eine leere Zeichenkette ("" ) bei Tabellenindex 0 einsetzen, wird Hollywood einen horizontalen Trennbalken anstelle eines wählbaren Menüeintrages einfügen. Mit diesen Trennbalken können Sie verwandte Menüpunkte zusammenfassen. Es darf kein Element am Tabellenindex 1 für Menüpunkttabellen sein. Stattdessen können Sie eine Reihe von anderen Tags verwenden, um Dinge wie Typ, Tastaturkürzel und Identifikator zu konfigurieren. Siehe unten für eine Liste von möglichen Tags.

Die folgenden Tags können Sie für Menüpunkttabellen verwenden:

**ID:** Hier können Sie eine Zeichenfolge angeben, die diesen Menüpunkt identifiziert. Diese Zeichenkette wird an Ihren Callback-Ereignis-Handler übergeben werden, so dass Sie wissen, welchen Menüpunkt durch den Benutzer ausgewählt wurde. Die hier angegebene ID ist auch erforderlich, wenn Sie Befehle wie `DisableMenuItem()` oder `SelectMenuItem()` nutzen wollen, um manuell den Zustand der Menüpunkte zu ändern.

**Flags:** Mit diesem Tag können Sie einen oder mehrere Flags für diesen Menüpunkt setzen:

**#MENUITEM\_TOGGLE:**

Wenn dieser Flag gesetzt ist, wird dieser Menüpunkt als Toggle-Menüpunkt erstellt werden. Toggle-Menüpunkt haben zwei verschiedene Zustände (An- und Abwahl) und der Fenstermanager gibt ihren aktuellen Zustand in der Regel mit einem Häkchen an. Sie können manuell die Wechselfunktion eines Menüpunktes ändern, indem sie die Befehle `SelectMenuItem()` und `DeselectMenuItem()` aufrufen oder durch den `#MENUITEM_SELECTED` Flags (siehe unten). Der Zustand des Toggle-Menüpunktes kann mit dem Befehl `IsMenuItemSelected()` überprüft werden.

**#MENUITEM\_RADIO:**

Setzen Sie diesen Flag, um den Menüpunkt zu einem Teil einer Menüpunktgruppe zu machen. Alle benachbarten Menüpunkte, die `#MENUITEM_RADIO` gesetzt haben, werden in die gleiche Menüpunktgruppe aufgenommen. Alle Menüpunkte innerhalb einer Menüpunktgruppe schließen sich gegenseitig aus, d.h. es kann jeweils nur ein Menüpunkt einer Menüpunktgruppe aktiv sein. Sie können den Zustand eines Menüpunktgruppen-Eintrags manuell ändern, indem Sie den Befehl `SelectMenuItem()` aufrufen oder den Flag `#MENUITEM_SELECTED` setzen (siehe unten). Der Status einer Menüpunktgruppe kann durch den Befehl `IsMenuItemSelected()` überprüft werden. Da Menüpunktgruppen immer ein aktives Element benötigen,

ist es nicht möglich, `DeselectMenuItem()` auf einem Menüpunktgruppen-Eintrag aufzurufen. Wenn Sie einen Menüpunktgruppen-Eintrag deselektieren möchten, müssen Sie mit `SelectMenuItem()` einen anderen Menüpunktgruppen-Eintrag auswählen und dann wird der zuvor ausgewählte Menüpunktgruppen-Eintrag automatisch deselektiert. (V7.1)

#### `#MENUITEM_SELECTED:`

Wenn Sie den Flag `#MENUITEM_TOGGLE` oder `#MENUITEM_RADIO` zum erstellen eines Toggle- oder Radio-Menüpunktes festgelegt haben, können Sie diesen Flag setzen, um das Menüelement in den ausgewählten Status zu versetzen. Siehe oben für weitere Informationen zu den Menüoptionen für Toggle- und Radio-Menüpunkte.

#### `#MENUITEM_DISABLED:`

Wenn dieser Flag gesetzt ist, wird der Menüpunkt grau dargestellt, so dass der Benutzer ihn nicht auszuwählen kann. Sie können auch einen Menüpunkt manuell deaktivieren, indem Sie den Befehl `DisableMenuItem()` aufrufen. Einen deaktivierten Menüpunkt wird wieder aktiviert, in dem Sie den Befehl `EnableMenuItem()` verwenden. Der deaktivierte Zustand eines Menüpunktes kann mit dem Befehl `IsMenuItemDisabled()` überprüft werden.

**Hotkey:** Dieser Tag kann auf ein Zeichen gesetzt werden, das für diesen Menüpunkt als Tastenkombination benutzt wird. Für die beste Cross-Plattform-Kompatibilität sollte dieser Tag auf ein Zeichen gesetzt werden, die nur ein Zeichen enthält, z.B. "Q" für Programm beenden. Einige Plattformen unterstützen auch benutzerdefinierte Tastaturkürzel wie "Strg + F1", aber in diesem Fall müssen Sie die Tastenkombination auf eigene Faust implementieren, da der Fenstermanager diese speziellen Tastaturkürzel nicht unterstützt. Wenn Sie ein Zeichen der Zeichenkette übergeben, wird jedoch der automatische Tastaturkürzel auf allen Plattformen funktionieren.

Menüleisten können auch mit dem Befehl `CreateMenu()` während der Laufzeit erstellt werden. Siehe [Abschnitt 37.1 \[CreateMenu\]](#), [Seite 749](#), für Details.

Sie können ein Menü mit dem Befehl `SetDisplayAttributes()`, mit dem `Menu`-Tag in der Präprozessor-Anweisung `@DISPLAY` oder dem Befehl `CreateDisplay()` einem Display zuweisen. Um eine Menüleiste von einem Display zu lösen, setzen Sie beim `Menu`-Tag vom Befehl `SetDisplayAttributes()` den besonderen Wert -1.

Um informiert zu werden, wenn der Benutzer Elemente aus dem Menü auswählt, müssen Sie den Ereignis-Handler `OnMenuSelect` installieren. Dies kann mit dem Befehl `InstallEventHandler()` erfolgen. Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), [Seite 483](#), für Details.

Bitte beachten Sie, dass Menüleisten Displays im Vollbildmodus nicht unterstützen. Sie werden nur dann funktionieren, wenn Ihr Display im Fenstermodus läuft.

Beachten Sie, dass es unter Android normalerweise nicht möglich ist, Menüelemente in der Stammebene des Optionsmenüs der Aktionsleiste zu platzieren, da Menüelemente

auf Desktopsystemen immer Mitglieder bestimmter Stammgruppen sein müssen (z.B. "Datei", "Bearbeiten", "Ansicht" usw.). Es gibt keine Menüelemente außerhalb solcher Stammgruppen. Wenn Sie Menüleisten unter Android verwenden, repliziert Hollywood natürlich das Verhalten des Desktop-Menüs, indem Sie für diese Stammgruppen eigene Untermenüs erstellen. Das bedeutet jedoch, dass der Benutzer mindestens zweimal auf tippen muss, um ein Menüelement auszuwählen, da sich in der Stammebene keine Menüelemente befinden. Diese werden stattdessen immer in Untermenüs angezeigt. Wenn Sie nicht möchten, dass Hollywood diese Untermenüs erstellt, sondern alle Elemente in der Stammebene platziert, setzen Sie den Tag `SingleMenu` mit der Präprozessor-Anweisung `@DISPLAY` auf `True`. Dies ist besonders nützlich, wenn es nur wenige Menüpunkte gibt und es nicht sinnvoll ist, sie in Untermenüs zu platzieren.

Beachten Sie auch, dass Menüleisten derzeit nicht von Linux und iOS unterstützt werden.

## EINGABEN

`id`            einen Wert, der verwendet wird, um diese Menüleiste zu identifizieren

`table`        Definition des Menübaumes (siehe oben)

## BEISPIEL

```
@MENU 1, {
    {"File", {
        {"New", ID = "new"},
        {"Open...", ID = "open"},
        {""},
        {"Close", ID = "close", Flags = #MENUITEM_DISABLED},
        {""},
        {"Save", Flags = #MENUITEM_DISABLED, Hotkey = "S"},
        {"Compress", ID = "cmp", Flags = #MENUITEM_TOGGLE},
        {""},
        {"Export image...", {
            {"JPEG...", ID = "jpeg"},
            {"PNG...", ID = "png"},
            {"BMP...", ID = "bmp"}}},
        {""},
        {"Dump state", ID = "dump"},
        {""},
        {"Quit", ID = "quit", Hotkey = "Q"}}},

    {"Edit", {
        {"Cut", ID = "cut"},
        {"Copy", ID = "copy"},
        {"Paste", ID = "paste"}}},

    {"?", {
        {"About...", ID = "about"}}}
}

@DISPLAY {Menu = 1}
```

Der obige Code erzeugt eine Menüleiste, die dann dem Standarddisplay zugewiesen wird.

## 37.9 PopupMenu

### BEZEICHNUNG

PopupMenu – zeigt ein Popup-Menü an (V10.0)

### ÜBERSICHT

PopupMenu(id[, x, y])

### BESCHREIBUNG

Dieser Befehl zeigt die durch `id` angegebene Menüleiste als Popup-Menü an. Popup-Menüs sind auch als Kontextmenüs bekannt und werden typischerweise angezeigt, wenn der Benutzer die rechte Maustaste in einem bestimmten Bereich des Displays drückt. Die an `PopupMenu()` übergebene Menüleiste muss entweder mit `CreateMenu()` oder `@MENU` erstellt worden sein und darf nur aus einer einzigen Leiste bestehen. Der Titel der Menüleiste wird ignoriert.

Mit den optionalen Argumenten `x` und `y` können Sie die gewünschte Position des Popup-Menüs angeben. Beachten Sie, dass dies in Koordinaten relativ zur oberen linken Ecke des Bildschirms übergeben werden muss, d.h. wenn Sie 0 für `x` und `y` übergeben, erscheint das Popup-Menü in der linken oberen Ecke des Bildschirms. Wenn Sie die Argumente `x` und `y` weglassen, wird das Popup-Menü an der Position des Mauszeigers angezeigt.

`PopupMenu()` blockiert die Ausführung des Skripts, bis der Benutzer einen Menüpunkt ausgewählt oder das Popup-Menü geschlossen hat. Genau wie normale Menüereignisse werden Popup-Menüereignisse über den Ereignishandler `OnMenuSelect` an Ihr Skript gesendet. Siehe [Abschnitt 26.13 \[InstallEventHandler\]](#), [Seite 483](#), für Details.

### EINGABEN

<code>id</code>	Identifikator einer Menüleiste
<code>x</code>	optional: gewünschte x-Position für das Popup-Menü
<code>y</code>	optional: gewünschte y-Position für das Popup-Menü

### BEISPIEL

```
CreateMenu(1, [{"Unused", {
  {"Cut", ID = "cut"},
  {"Copy", ID = "copy"},
  {"Paste", ID = "paste"},
  {""},
  {"Undo", ID = "undo"},
  {"Redo", ID = "redo"}
}]})
PopupMenu(1)
```

Der obige Code definiert und zeigt ein Popup-Menü.

## 37.10 SelectMenuItem

### BEZEICHNUNG

SelectMenuItem – wählt einen Toggle- oder Radio-Menüpunkt aus (V6.0)

### ÜBERSICHT

```
SelectMenuItem(id, item$[, detached])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen Toggle- oder Radio-Menüpunkt auszuwählen. Das optionale Argument **detached** gibt an, ob eine an ein Display angehängte Menüleiste oder eine losgelöste Menüleiste verwendet werden soll. Wenn **detached False** ist (was auch die Voreinstellung ist), wird die Menüleiste des durch **id** angegebenen Displays verwendet. Wenn **detached True** ist, wird die durch **id** angegebene Menüleiste verwendet. Mit anderen Worten: Wenn Sie **detached** auf **True** setzen, müssen Sie in **id** den Identifikator einer Menüleiste übergeben; andernfalls müssen Sie den Identifikator eines Displays in **id** übergeben.

Beachten Sie, dass beim Setzen von **detached** auf **True** Ihre Operation niemals Auswirkungen auf Menüleisten hat, die mit einem Display verbunden sind. Das Setzen von **detached** auf **True** wird normalerweise nur bei Menüleisten verwendet, die als Popup-Menüs mit dem Befehl **PopupMenu()** angezeigt werden. Es ist unmöglich, Display-Menüleisten anzusprechen, wenn **detached** auf **True** gesetzt wird, da eine einzelne Menüleiste an mehrere Displays angehängt werden kann.

Unter AmigaOS 4 können Sie für **id** auch den Sonderwert 0 übergeben. In diesem Fall wird das Kontextmenü des Docky verwendet.

Bitte beachten Sie auch, dass Sie die ID eines Displays an diesen Befehl übergeben müssen. Es ist nicht ausreichend, die ID einer Menüleiste zu übergeben, da einzelne Menüleisten an mehreren Displays zugewiesen werden können. Wenn Sie das Kontextmenü eines Docky auf AmigaOS4 ansprechen wollen, übergeben Sie 0.

### EINGABEN

<b>id</b>	ID eines Displays, welches die Menüleiste zugewiesen wurde oder 0 (siehe oben)
<b>item\$</b>	Identifikator des Menüpunktes innerhalb der Menüleiste
<b>detached</b>	optional: <b>False</b> , wenn <b>id</b> ein Display angibt; <b>True</b> , wenn es ein Menüleistenobjekt angibt (voreingestellt ist <b>False</b> ) (V10.0)





## 38 Mobile-Geräte-Bibliothek

### 38.1 CallJavaMethod

#### BEZEICHNUNG

CallJavaMethod – ruft die Methode der Java-Aktivität auf (V8.0)

#### ÜBERSICHT

```
[ret] = CallJavaMethod(name$, t, type1, value1, type2, value2, ...)
```

#### PLATTFORMEN

Nur Android

#### BESCHREIBUNG

Dies ist ein leistungsstarker Befehl, mit der Sie direkt in den Java-Code von Hollywoods Android-Aktivitäten zugreifen können. Der Java-Code kann dann auf die gesamte Android-API zugreifen, um Ihre App mit benutzerdefinierten Funktionen zu verbessern, die in Hollywood nicht verfügbar sind.

Sie müssen den Namen der aufzurufenden Methode im Argument **name\$** übergeben. Beachten Sie, dass Java eine von Groß- und Kleinschreibung abhängige Sprache ist, daher muss der Methodenname, den Sie in **name\$** übergeben, genau mit seiner Definition im Java-Code übereinstimmen.

Optional können Sie im zweiten Argument eine Tabelle übergeben. Diese Tabelle unterstützt derzeit die folgenden Tags:

**Static:** Setzen Sie diese Option auf **True**, wenn die Methode, die Sie aufrufen möchten, statisch ist. Standardmäßig erwartet **CallJavaMethod()**, dass die Methode nicht statisch ist.

#### ReturnType:

Mit diesem Tag konfigurieren Sie den Rückgabedatentyp der Methode, welche Sie in **name\$** übergeben haben. Dies muss eine der folgenden vordefinierten Konstanten sein:

**#BYTE:** Javas Datentyp **byte**, eine signierte 8-Bit-Größe.

**#SHORT:** Javas Datentyp **short**, eine signierte 16-Bit-Größe.

**#INTEGER:** Javas Datentyp **int**, eine signierte 32-Bit-Größe.

**#FLOAT:** Javas Datentyp **float**, eine 32-Bit-Fließkommazahl.

**#DOUBLE:** Javas Datentyp **double**, eine 64-Bit-Fließkommazahl.

**#BOOLEAN:** Javas Datentyp **boolean**, ein boolescher Wert (**True** oder **False**).

**#STRING:** Javas Datentyp **String**, eine Zeichenkette

**#VOID:** Kein Rückgabewert.

Der Standardwert von diesem Tag ist **#VOID**, d.h. die Methode gibt keinen Wert zurück.

#### **ReturnArray:**

Wenn dieser Tag auf **True** gesetzt ist, wird erwartet, dass die in **name\$** übergebene Methode ein Array des in **ReturnType** angegebenen Datentyps zurückgibt. Wenn **ReturnArray** auf **True** gesetzt ist, darf **ReturnType** nicht auf **#VOID** gesetzt sein. Der Standardwert ist **False**.

Nach dem optionalen Tabellenargument akzeptiert **CallJavaMethod()** eine unbegrenzte Anzahl von **type**- und **value**-Paaren. Diese Paare können verwendet werden, um Parameter an die in **name\$** angegebene Methode zu übergeben. Für jeden **type** muss direkt ein entsprechendes **value**-Argument vorhanden sein.

Die folgenden vordefinierten Konstanten werden derzeit für das Argument **type** unterstützt:

- #BYTE:**     Javas Datentyp **byte**, eine signierte 8-Bit-Größe.
- #SHORT:**    Javas Datentyp **short**, eine signierte 16-Bit-Größe.
- #INTEGER:**    Javas Datentyp **int**, eine signierte 32-Bit-Größe.
- #FLOAT:**     Javas Datentyp **float**, eine 32-Bit-Fließkommazahl.
- #DOUBLE:**    Javas Datentyp **double**, eine 64-Bit-Fließkommazahl.
- #BOOLEAN:**    Javas Datentyp **boolean**, ein boolescher Wert (**True** oder **False**).
- #STRING:**    Javas Datentyp **String**, eine Zeichenkette
- #VOID:**     Kein Rückgabewert.

Der Wert, der jedem Argument **type** folgt, muss dem für diesen Parameter angegebenen Typ entsprechen, z.B. wenn Sie in einem Argument **type** **#STRING** übergeben, muss nach dem Argument **#STRING** eine Zeichenkette folgen.

Hier ist ein Beispiel für eine Java-Methode:

```
public int littleTest(String s, int v) {
    Log.v("Test", "Got data: " + s + " " + v);
    return 50;
}
```

Beachten Sie, dass es wichtig ist, die Methode mit dem Schlüsselwort **public** zu definieren, da auf sie außerhalb ihrer Klasse zugegriffen wird. Um die Java-Methode **littleTest()** von Ihrem Hollywood-Skript mit **CallJavaMethod()** aufzurufen, müssen Sie den folgenden Code verwenden:

```
r = CallJavaMethod("littleTest", {ReturnType = #INTEGER},
    #STRING, "Hello Java!", #INTEGER, 10)
```

Da die Java-Methode als Ganzzahlwert deklariert wurde und die Implementierung auf der Java-Seite 50 ergibt, wird die Hollywood-Variable **r** auf 50 gesetzt, sobald **CallJavaMethod()** beendet wird.

Beachten Sie, dass diese Funktion nur in Verbindung mit dem Hollywood APK Compiler verwendet werden kann, da der Hollywood-Player Ihnen nicht erlaubt, benutzerdefinierten Code in seine Aktivität einzufügen. Dies wird nur vom Hollywood APK Compiler unterstützt.

Die Java-Methoden, die Sie definieren, sind Teil einer Unterklasse von Hollywoods Android-Activity. Sie können also jede der **Activity**-Methoden direkt aus den von **CallJavaMethod()** aufgerufenen Methoden aufrufen. Beachten Sie jedoch, dass von **CallJavaMethod()** ausgeführte Java-Methoden nicht im Haupt-UI-Thread, sondern im VM-Thread von Hollywood ausgeführt werden. Wenn Sie also auf Android-APIs zugreifen müssen, die nur über den UI-Thread aufgerufen werden können (wie die meisten **View**-bezogenen APIs), müssen Sie zuerst vom Hollywood-Thread an den Haupt-Thread delegieren. Andernfalls funktioniert der Code nicht.

#### EINGABEN

<b>name\$</b>	Name der aufzurufenden Methode
<b>t</b>	optional: Tabelle mit weiteren Optionen (siehe oben)
<b>type1</b>	optional: Typ des ersten Parameters, der an die Methode übergeben wird (mögliche Werte siehe oben)
<b>value1</b>	optional: Wert des ersten Parameters
<b>...</b>	optional: unbegrenzte Anzahl von weiteren Methodenparametern

#### RÜCKGABEWERTE

<b>ret</b>	optional: falls <b>ReturnType</b> nicht <b>#VOID</b> ist, ist das der von der Java-Methode zurückgegebene Wert
------------	--

## 38.2 GetAsset

#### BEZEICHNUNG

GetAsset – erhält Zugriff auf Android-Asset (V6.1)

#### ÜBERSICHT

```
handle$ = GetAsset(f$)
```

#### BESCHREIBUNG

Dieser Befehl kann benutzt werden, um auf ein Asset zuzugreifen, welches vom Hollywood APK Compiler in eine APK-Datei gebunden wurde. Die Zeichenkette, die von diesem Befehl zurückgegeben wird, kann anschließend allen Hollywood-Befehlen, die mit Dateien arbeiten, übergeben werden, z.B. **LoadBrush()** oder **OpenMusic()**. Beachten Sie allerdings, dass Assets nur gelesen werden können. Es ist nicht möglich, die Daten von Assets zu verändern. Ein solcher Versuch erzeugt einen Fehler.

Beachten Sie außerdem, dass bei Assetnamen zwischen Groß- und Kleinschreibung unterschieden wird, da Android auf Linux basiert. Daher muss der Name, den Sie diesem Befehl übergeben, genau mit dem Namen übereinstimmen, mit dem das Asset mittels des Hollywood APK Compiler in die APK-Datei gebunden wurde oder es wird ein "Datei nicht gefunden"-Fehler erzeugt.

Aus Gründen der Bequemlichkeit wird `GetAsset()` auch von allen anderen Hollywood-Versionen unterstützt, aber in diesem Fall gibt der Befehl einfach nur die Zeichenkette zurück, die ihm übergeben wurde, wenn er außerhalb von Hollywood APK Compiler erstellten APKs verwendet wird.

#### EINGABEN

`f$`            Name des Assets, auf das zugegriffen werden soll

#### RÜCKGABEWERTE

`handle$`      Zeichenkette, die den Zugriff auf das Asset über alle Hollywood-Befehle ermöglicht, die mit Dateien arbeiten

#### BEISPIEL

```
LoadBrush(1, GetAsset("test.png"))
```

Der obige Code lädt das Asset "test.png" als Pinsel mit der Nummer 1. Damit dieser Code funktioniert, müssen Sie eine Datei namens "test.png" (in genau dieser Schreibweise!) in Ihre APK-Datei mit dem Hollywood APK Compiler binden.

## 38.3 HideKeyboard

#### BEZEICHNUNG

HideKeyboard – blendet die Softwaretastatur aus (V5.0)

#### ÜBERSICHT

```
HideKeyboard()
```

#### PLATTFORMEN

Nur für Mobile Plattformen

#### BESCHREIBUNG

Mit diesem Befehl kann die Softwaretastatur des Host-Systems auf Mobilgeräten ausgeblendet werden. Sobald die Software-Tastatur ausgeblendet ist, wird der Benutzer keine `OnKeyDown`- oder `OnKeyUp`-Ereignisse mehr auslösen können.

Um die Software-Tastatur wieder anzuzeigen, verwenden Sie den Befehl `ShowKeyboard()`.

#### EINGABEN

keine

## 38.4 PerformSelector

#### BEZEICHNUNG

PerformSelector – führt einen Selektor aus (V7.0)

#### ÜBERSICHT

```
[ret] = PerformSelector(s$[, ...])
```

#### PLATTFORMEN

Nur iOS

## BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen Selektor im AppDelegate Ihrer Anwendung auszuführen, d.h. er erlaubt nativen Code mit Ihrem Hollywood-Skript aufzurufen. Der Name des Selektors muss als Zeichenkette in `s$` angegeben werden.

Der in `s$` angegebene Selektor wird dann mit einem `NSMutableArray` als einziges Argument ausgeführt. Innerhalb dieses Feldes enthält Index 1 den `lua_State` und Index 2 beinhaltet einen Zeiger auf eine `hwPluginAPI-Struktur`, die es Ihnen erlaubt, auf alle öffentlichen APIs und insbesondere die Lua-VM zuzugreifen. Die Indizes 3 und 4 enthalten den `UIViewController` und `UIView`. Bei der Rückgabe müssen Sie den Index 0 auf eine `NSValue` setzen, die einen `int` enthält, der den Rückgabecode von Ihrer Funktion angibt. Sehen Sie bitte in der Hollywood SDK-Dokumentation für weitere Details nach (vor allem die Kapitel zum Schreiben von Bibliotheken-Plugins).

Sie müssen den gewünschten Selektor im AppDelegate Ihrer Anwendung in nativem Code implementieren. So könnte ein benutzerdefinierter Selektor in Objective-C aussehen:

```
- (void)MyTestSelector:(NSMutableArray *) args
{
    // notwendige Daten von Hollywood holen
    lua_State *L = (lua_State *)
    [[(NSValue *) [args objectAtIndex:1]] pointerValue];
    hwPluginAPI *hwcl = (hwPluginAPI *)
    [[(NSValue *) [args objectAtIndex:2]] pointerValue];

    // es gibt einen Rückgabewert (Zeichenkette)
    int retval = 1;

    // Zeichenkette im Stackindex 2 ausgeben
    printf("%s\n", hwcl->LuaBase->luaL_checklstring(L, 2, NULL));

    // Rückgabestring auf Stack legen
    hwcl->LuaBase->lua_pushstring(L, "Test return value");

    // Rückgabewert setzen
    [args replaceObjectAtIndex:0 withObject:[NSValue value:&retval
    withObjectType:@encode(int*)]];
}
```

Dieser Selektor gibt das Zeichenketten-Argument aus, welches `PerformSelector()` als Argument 2 übergeben wird. Es wird dann die Zeichenfolge "Test return value" an das Hollywood-Skript zurückgegeben. Sie können diesen Selektor aus Ihrem Hollywood-Skript wie folgt ausführen:

```
DebugPrint(PerformSelector("MyTestSelector", "Test"))
```

Dieser Code übergibt die Zeichenfolge "Test" an die Methode `MyTestSelector`. Der Befehl `DebugPrint()` wird "Test return value" ausgegeben, weil das der Rückgabewert von `MyTestSelector` ist.

Denken Sie daran, dass Ihre Selektor-Funktion nicht auf dem Main-Thread (UI) laufen wird, sondern auf dem VM-Thread von Hollywood. Wenn Sie also UIKit-Funktionen

(oder andere OS-Funktionen, die nur vom Main-Thread ausgeführt werden dürfen) benutzen möchten, dürfen Sie nicht vergessen, den entsprechenden Code zuerst an den Main-Thread zu delegieren.

#### EINGABEN

`s$`            Name des auszuführenden Selektors

#### RÜCKGABEWERTE

`ret`            optional: Rückgabewerte Ihrer Selektorfunktion

## 38.5 ShowKeyboard

#### BEZEICHNUNG

ShowKeyboard – zeigt die Softwaretastatur (V5.0)

#### ÜBERSICHT

ShowKeyboard()

#### PLATTFORMEN

Nur für Mobile Plattform

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die Software-Tastatur des Host-Systems auf mobilen Geräten anzuzeigen. Sobald die Software-Tastatur sichtbar ist, wird der Benutzer Text eingeben können, der dann in Form von `OnKeyDown`- und `OnKeyUp`-Ereignissen an Ihr Skript gesendet werden kann. Sie sollten daher die entsprechenden Ereignis-Handler mit `InstallEventHandler()` installieren, bevor Sie `ShowKeyboard()` aufrufen. Zum Ausblenden der Software-Tastatur verwenden Sie den Befehl `HideKeyboard()`.

#### EINGABEN

keine

## 38.6 ShowToast

#### BEZEICHNUNG

ShowToast – zeigt eine kurze Nachricht (V5.3)

#### ÜBERSICHT

ShowToast(`s$`[, `x`, `y`, `long`])

#### PLATTFORMEN

Nur für Mobile Plattform

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um eine kurze Nachricht (ein sogenannter "Toast") zu zeigen, die nach einer bestimmten Zeit automatisch verschwindet. Sie müssen diese Nachricht im Argument `s$` übergeben. Mit den optionalen Argumenten können Sie die gewünschte Position ( `x/y` ) der Nachricht auf dem Bildschirm angeben und die Präsentationszeit in `long` festlegen (lang (`True`) oder kurz (`False`)). Sie können auch Hollywoods spezielle Koordinatenkonstanten in den Argumenten `x` und `y` verwenden.

**EINGABEN**

<code>s\$</code>	Nachricht, die gezeigt wird
<code>x</code>	optional: x-Position für die Nachricht
<code>y</code>	optional: y-Position für die Nachricht
<code>long</code>	optional: ob die Darstellungsdauer lang oder kurz sein soll (standardmäßig <code>False</code> , das heißt eine kurze Darstellungsdauer)

**BEISPIEL**

```
ShowToast("Hello World!", #CENTER, #CENTER)
```

Der obige Code zeigt die Meldung "Hello World!" in der Mitte des Bildschirms an und verbirgt sie automatisch nach einer kurzen Zeitspanne.

## 38.7 Vibrate

**BEZEICHNUNG**

Vibrate – vibriert das Gerät (V8.0)

**ÜBERSICHT**

```
Vibrate(ms)
```

**PLATTFORMEN**

Nur Android

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um das Gerät für die durch `ms` festgelegte Dauer zu vibrieren. Die Dauer muss in Millisekunden angegeben werden.

**EINGABEN**

<code>ms</code>	Dauer in Millisekunden der Vibration
-----------------	--------------------------------------

**BEISPIEL**

```
Vibrate(1000)
```

Der obige Code vibriert das Gerät eine Sekunde lang.





## 39 Netzbibliothek

### 39.1 CloseConnection

#### BEZEICHNUNG

CloseConnection – trennt die Verbindung zum Server (V5.0)

#### ÜBERSICHT

CloseConnection(id)

#### BESCHREIBUNG

Dieser Befehl trennt sich von dem in `id` angegebenen Server und schließt die Verbindung. Die Verbindung, die Sie hier angeben müssen, muss durch den Befehl `OpenConnection()` erstellt worden sein.

#### EINGABEN

`id` ID der Verbindung, die getrennt werden soll

### 39.2 CloseServer

#### BEZEICHNUNG

CloseServer – fährt einen vorhandenen Server runter (V5.0)

#### ÜBERSICHT

CloseServer(id)

#### BESCHREIBUNG

Mit diesem Befehl wird der in `id` angegebene Server heruntergefahren. Der Server, den Sie hier übergeben, muss mit dem Befehl `CreateServer()` erstellt worden sein.

Es ist wichtig, dass Sie alle Clients mit dem Befehl `CloseConnection()` von Ihrem Server trennen, bevor Sie `CloseServer()` aufrufen.

#### EINGABEN

`id` Identifikator des Servers, der heruntergefahren wird

### 39.3 CloseUDPObject

#### BEZEICHNUNG

CloseUDPObject – schließt ein vorhandenes UDP-Objekt (V5.0)

#### ÜBERSICHT

CloseUDPObject(id)

#### BESCHREIBUNG

Dieser Befehl schließt das in `id` angegebene UDP-Objekt. Dieses UDP-Objekt muss zuvor mit dem Befehl `CreateUDPObject()` erstellt worden sein.

#### EINGABEN

`id` Identifikator des UDP-Objekts, welches geschlossen wird

## 39.4 CreateServer

### BEZEICHNUNG

CreateServer – öffnet einen neuen Server (V5.0)

### ÜBERSICHT

```
[id] = CreateServer(id[, port, ip$, backlog, protocol])
```

### BESCHREIBUNG

Mit diesem Befehl kann ein neuer Server eingerichtet werden, der bereit ist, eingehende Verbindungen auf dem lokalen Host aufzunehmen. Über das optionale Argument **port** kann festgelegt werden, auf welchem Port der Server geöffnet werden soll. Wenn Sie dieses Argument nicht angeben, wird **CreateServer()** automatisch einen freien Port auswählen und Sie können später den Befehl **GetLocalPort()** verwenden, um die Portnummer des Servers herauszufinden. Im ersten Argument **id** müssen Sie eine Kennung übergeben, die benötigt wird, um später auf diesen Server zu verweisen. Alternativ können Sie **Nil** in **id** übergeben. In diesem Fall wird **CreateServer()** automatisch eine ID auswählen und ihnen zurückgeben.

Mit dem optionalen Argument **ip\$** kann eine lokale IP-Adresse angegeben werden, an die der Server gebunden sein soll. Der Standardwert ist **"\*"**. Dies bedeutet, dass der Server Verbindungen vom gesamten Netzwerk akzeptiert. Sie können auch **"127.0.0.1"** (oder **"::1"** in IPv6) angeben, um nur Verbindungen vom lokalen Host zuzulassen. Beachten Sie, dass das Übergeben von **"\*"**, das auch die Standardeinstellung ist, bei einigen Konfigurationen die Firewall auslösen kann.

Das optionale Argument **backlog** kann verwendet werden, um die maximale Anzahl von Clientverbindungen anzugeben, die der Warteschlange hinzugefügt werden können. Der Standardwert ist 32.

Ab Hollywood 8.0 gibt es optional ein neues Argument **protocol**, mit dem Sie das Internetprotokoll angeben können, das beim Erstellen des Servers verwendet werden soll. Dies kann eine der folgenden speziellen Konstanten sein:

- #IPV4:** Verwendet die Internetprotokollversion 4 (IPv4). IPv4-Adressen sind auf 32 Bit beschränkt und werden unter Verwendung von vier durch drei Punkte getrennten Zahlen dargestellt, z.B. 127.0.0.1.
- #IPV6:** Verwendet die Internetprotokollversion 6 (IPv6). IPv6-Adressen verwenden 128 Bits und werden durch acht Gruppen von vier Hexadezimalziffern dargestellt, z.B. 2001:0db8:85a3:0000:0000:8a2e:0370:7334. Beachten Sie, dass **#IPV6** auf AmigaOS und kompatiblen Systemen derzeit nicht unterstützt wird.
- #IPAUTO:** Lassen Sie das Host-Betriebssystem das zu verwendende Internetprotokoll bestimmen. Anschließend können Sie mit dem Befehl **GetLocalProtocol()** ermitteln, welches Internetprotokoll das Host-Betriebssystem für diesen Server ausgewählt hat. Siehe [Abschnitt 39.14 \[GetLocalProtocol\]](#), [Seite 785](#), für Details.

Das Argument **protocol** verwendet standardmäßig den Standardprotokolltyp, der mit **SetNetworkProtocol()** festgelegt wird. Standardmäßig ist dies aus historischen

Gründen und aus Gründen der Portabilität **#IPV4**. @see{SetNetworkProtocol, SetNetworkProtocol}

Sobald der Server erfolgreich eingerichtet wurde, müssen Sie den Befehl **InstallEventHandler()** verwenden, um die Ereignisse **OnConnect**, **OnDisconnect** und **OnReceiveData** zu installieren. Diese Ereignisse informieren Sie, wenn ein neuer Client versucht, eine Verbindung zu Ihrem Server herzustellen oder wenn ein Client neue Daten (d.h. Befehle, die Sie behandeln müssen) an Ihren Server sendet.

#### EINGABEN

<b>id</b>	Identifikator des neuen Server oder <b>Nil</b> für die <b>automatische ID-Auswahl</b>
<b>port</b>	optional: Port, auf dem dieser Server geöffnet oder 0 für die automatische Portwahl (voreingestellt ist 0)
<b>ip\$</b>	optional: lokale IP-Adresse, an die der Server gebunden werden soll (Standardeinstellung ist "")
<b>backlog</b>	optional: maximale Anzahl von Verbindungen, die in die Warteschlange gestellt werden können (Standardeinstellung ist 32)
<b>protocol</b>	optional: Zu verwendendes Internetprotokoll (mögliche Werte siehe oben); der Standardwert ist der mit <b>SetNetworkProtocol()</b> festgelegter Protokolltyp. (V8.0)

#### RÜCKGABEWERTE

<b>id</b>	optional: Identifikator des Servers; wird nur zurückgegeben werden, wenn Sie <b>Nil</b> als <b>id</b> angegeben haben (siehe oben)
-----------	--

## 39.5 CreateUDPObject

#### BEZEICHNUNG

CreateUDPObject – erstellt ein neues UDP-Objekt (V5.0)

#### ÜBERSICHT

```
[id] = CreateUDPObject(id[, port, ip$, mode, protocol])
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um ein neues UDP-Objekt zu erstellen, das Daten empfangen und Daten an andere Netzwerkteilnehmer senden kann. Das optionale Argument **port** kann verwendet werden, um festzulegen, an welchem lokalen Port das UDP-Objekt erstellt werden soll. Wenn Sie dieses Argument nicht angeben, wird **CreateUDPObject()** automatisch einen freien Port auswählen und Sie können später den Befehl **GetLocalPort()** verwenden, um die Portnummer des UDP-Objekts herauszufinden. Im ersten Argument **id** müssen Sie eine ID übergeben, die benötigt wird, um später auf dieses UDP-Objekt zu verweisen. Alternativ können Sie **Nil** in **id** übergeben. In diesem Fall wird **CreateUDPObject()** automatisch eine ID auswählen und ihnen zurückgeben.

Ab Hollywood 8.0 gibt es ein neues Argument **mode**, mit dem Sie den Typ des UDP-Objekts angeben können, das für Sie erstellt werden soll. Dies kann einer der folgenden vordefinierten Werte sein:

**#UDPSERVER:**

Erstellt ein UDP-Serverobjekt. In diesem Fall kann mit dem optionalen Argument `ip$` eine lokale IP-Adresse angegeben werden, an die das UDP-Objekt gebunden werden soll. Der Standardwert ist `"*"`. Dies bedeutet, dass der Server Verbindungen vom gesamten Netzwerk akzeptiert. Sie können auch `"127.0.0.1"` (oder `::1` in IPv6) angeben, um nur Verbindungen vom lokalen Host zuzulassen. Beachten Sie, dass das Übergeben von `"*"`, das auch die Standardeinstellung ist, bei einigen Konfigurationen die Firewall auslösen kann. `#UDPSERVER` ist der Standardmodus für `CreateUDPObject()`.

**#UDPCLIENT:**

Erstellt ein Client-UDP-Objekt. In diesem Fall müssen die optionalen Argumente `ip$` und `port` angegeben werden, um den Server anzugeben, mit dem der Client verbunden ist. Wenn Ihr UDP-Objekt immer Daten an denselben Server sendet, wird empfohlen, es als `#UDPCLIENT`-Objekt zu erstellen, da `SendUDPData()` dann schneller ist. (V8.0)

**#UDPNONE:**

Erstellen Sie ein nicht verbundenes UDP-Objekt. In diesem Fall werden die optionalen Argumente `ip$` und `port` ignoriert. Nicht verbundene UDP-Objekte sind nützlich, wenn Sie Daten an verschiedene Server senden müssen, ohne selbst Daten empfangen zu können. (V8.0)

Darüber hinaus führt Hollywood 8.0 ein optionales neues Argument `protocol` ein, mit dem Sie das Internetprotokoll angeben können, das vom UDP-Objekt verwendet werden soll. Dies kann eine der folgenden speziellen Konstanten sein:

- #IPv4:** Verwendet die Internetprotokollversion 4 (IPv4). IPv4-Adressen sind auf 32 Bit beschränkt und werden unter Verwendung von vier durch drei Punkte getrennten Zahlen dargestellt, z.B. 127.0.0.1.
- #IPv6:** Verwendet die Internetprotokollversion 6 (IPv6). IPv6-Adressen verwenden 128 Bits und werden durch acht Gruppen von vier Hexadezimalziffern dargestellt, z.B. 2001:0db8:85a3:0000:0000:8a2e:0370:7334. Beachten Sie, dass `#IPv6` auf AmigaOS und kompatiblen Systemen derzeit nicht unterstützt wird.
- #IPAUTO:** Lassen Sie das Host-Betriebssystem das zu verwendende Internetprotokoll bestimmen. Anschließend können Sie mit dem Befehl `GetLocalProtocol()` ermitteln, welches Internetprotokoll das Host-Betriebssystem für diesen Server ausgewählt hat. Siehe [Abschnitt 39.14 \[GetLocalProtocol\]](#), [Seite 785](#), für Details.

Das Argument `protocol` verwendet standardmäßig den Standardprotokolltyp, der mit `SetNetworkProtocol()` festgelegt wird. Standardmäßig ist dies aus historischen Gründen und aus Gründen der Portabilität `#IPv4`. Siehe [Abschnitt 39.23 \[SetNetworkProtocol\]](#), [Seite 795](#), für Details.

Sobald das UDP-Objekt erstellt ist, können Sie mit den Befehlen `SendUDPData()`, `ReceiveUDPData()` und dem Ereignis-Handler `OnReceiveUDPData` mit anderen Systemen im Netzwerk zu kommunizieren.

Bitte beachten Sie, dass UDP ein unzuverlässiges Übertragungsprotokoll ist. Es ist schneller als das TCP-Protokoll, aber Daten können unvollständig oder gar nicht ankommen. Somit ist es nur für Zwecke geeignet, die nicht von der Integrität der übertragenen Daten abhängt.

#### EINGABEN

<code>id</code>	Identifikator für das neue UDP-Objekt oder <code>Nil</code> für die automatische ID-Auswahl
<code>port</code>	optional: Port für dieses UDP-Objekt oder 0 für die automatische Portwahl (voreingestellt ist 0); muss für <code>#UDPCLIENT</code> gesetzt sein
<code>ip\$</code>	optional: IP-Adresse, an die der Server gebunden werden soll (für <code>#UDPSERVER</code> ) oder um ein UDP-Objekt zu verbinden (für <code>#UDPCLIENT</code> ) (Standardeinstellung ist "*" für <code>#UDPSERVER</code> ); muss für <code>#UDPCLIENT</code> gesetzt sein
<code>mode</code>	optional: gewünschter Modus für dieses UDP-Objekt (mögliche Werte siehe oben) (V8.0)
<code>protocol</code>	optional: zu verwendendes Internetprotokoll (mögliche Werte siehe oben); der Standardwert ist der mit <code>SetNetworkProtocol()</code> festgelegte Protokolltyp. (V8.0)

#### RÜCKGABEWERTE

<code>id</code>	optional: Identifikator des UDP-Objekts; wird nur zurückgegeben werden, wenn Sie <code>Nil</code> als <code>id</code> angegeben haben (siehe oben)
-----------------	--

## 39.6 DownloadFile

#### BEZEICHNUNG

`DownloadFile` – lädt Dateien über HTTP-, FTP- oder ein anderes Protokoll herunter (V5.0)

#### ÜBERSICHT

```
data$, count = DownloadFile(url$[, options, func, userdata])
```

#### BESCHREIBUNG

Mit diesem Befehl können Sie bequem eine Datei von einem beliebigen Netzwerk-Server herunterladen. Standardmäßig werden HTTP- und FTP-Server genutzt, aber Hollywood-Plugins können weitere Protokolle unterstützen. Sie müssen die URL der Datei im Argument `url$` übergeben. `DownloadFile()` lädt dann die Datei herunter und gibt sie als Zeichenkette in `data$` zurück. Die Speicherung binärer Daten in Zeichenketten ist möglich, weil die Zeichenketten von Hollywood nicht auf druckbare Zeichen beschränkt sind. Stattdessen können sie auch Steuerzeichen und das Nullzeichen enthalten. Der zweite Rückgabewert `count` gibt die Größe der heruntergeladenen Datei in Byte an.

Alternativ können Sie auch den Tag `File` im optionalen Tabellenargument `options` auf einen Dateinamen setzen. In diesem Fall wird die heruntergeladene Datei nicht als Zeichenkette zurückgegeben, sondern als die im Tag `File` angegebene Datei gespeichert. Dies wird für größere Dateien empfohlen, da Zeichenketten offensichtlich im Speicher

gespeichert werden. Das Herunterladen einer großen Datei in eine Zeichenkette ist daher im Allgemeinen keine gute Idee, da dies viel Speicher benötigt.

Die URL, die in `url$` übergeben wird, muss mit einem Protokollpräfix beginnen wie `http://` oder `ftp://` und darf keine ESC-Zeichen enthalten. Die ESC-Zeichenumwandlung wird durch `DownloadFile()` durchgeführt, so stellen Sie sicher, dass Sie nur unverschlüsselte URLs übergeben. Das bedeutet, dass z.B. die URL `"http://www.mysite.net/cool%20file.html"` nicht funktionieren wird. Sie müssen eine URL ohne ESC-Zeichen angeben, womit die korrekte Version wäre: `"http://www.mysite.net/cool file.html"`. Wenn Sie eine bereits übergebene URL übergeben wollen, müssen Sie den Tag `Encoded` auf `True` setzen (siehe unten). In diesem Fall wird `DownloadFile()` keine weiteren ESC-Zeichenumwandlungen auf Ihre URL anwenden.

`DownloadFile()` unterstützt auch die Authentifizierung für die HTTP- und FTP-Protokolle. In diesem Fall müssen Benutzername und Passwort nach der Protokollkennung im Formular `username:password` übergeben werden, gefolgt von einem @-Zeichen und dem Server. Hier ist ein Beispiel für den Benutzer "joe" und das Passwort "secret": `http://joe:secret@www.test.net/private/files.lha`. Beachten Sie, dass HTTP-Authentifizierungsunterstützung vor Hollywood 6.0 nicht verfügbar war. Beim Herunterladen von einem FTP-Server, wird "anonymous" als Standard-Benutzername und "anonymous@anonymous.org" als das Standard-Passwort verwendet. Wenn Sie ein anderes Anmeldekonto verwenden möchten, müssen Sie die Benutzername/Passwort-Kombination in der URL übergeben, z.B: `ftp://joe:secret@ftp.test.net/pub/files.lha`.

Die an diesen Befehl übergebene URL kann auch eine Portnummer enthalten. Wenn Sie eine Portnummer angeben möchten, müssen Sie sie hinter dem Hostnamen setzen und mit einem Doppelpunkt trennen. Hier ist ein Beispiel für die Verwendung von Port 1234: `http://www.test.com:1234/test/image.jpg`. Wenn kein Port angegeben wird, verwendet `DownloadFile()` Port 80 für HTTP-Server und Port 21 für FTP-Server.

Mit dem zweiten Argument `options` können weitere Optionen für den Download angegeben werden. Es ist eine Tabelle, die die folgende Felder erkennt:

**File:** Wenn Sie einen Dateinamen in diesem Tabellenfeld angeben, lädt `DownloadFile()` die heruntergeladenen Daten direkt in diese Datei, anstatt sie als Zeichenkette zurückzugeben. Dies ist für sehr große Dateien auf der einen Seite, aber es ist auch nützlich für andere Dateien, weil es Ihnen die Mühe erspart, die Daten in Zeichenketten manuell in eine Datei zu speichern und danach die Zeichenfolge auf `Nil` zu setzen. Wenn Sie also die heruntergeladene Zeichenkette in einer Datei speichern möchten, ist es effizienter, dieses Feld zu verwenden. Wenn Sie es nicht verwenden, lesen Sie bitte unten die wichtigen Informationen fürs Herunterladen von Dateien in Zeichenketten.

**TransferMode:**

Dieser Tag wird nur beim Herunterladen von Dateien von einer FTP-Quelle unterstützt. In diesem Fall können Sie mit diesem Tag angeben, ob `DownloadFile()` die Datei im ASCII oder im Binärmodus übertragen soll. Im ASCII-Modus geben Sie hier `#FTPASCII` an. Verwenden Sie für

den Binärmodus `#FTPBINARY`. Der voreingestellte Übertragungsmodus ist `#FTPBINARY`.

**Proxy:** Dieser Tag wird nur beim Herunterladen von Dateien von einer HTTP-Quelle unterstützt. In diesem Fall können Sie hier einen Server angeben, der als Proxy-Server für die Verbindung fungieren soll.

**Fail404:** Dieser Tag legt fest, ob `DownloadFile()` mit einem Fehler "file not found" fehlschlagen soll, wenn Sie eine URL übergeben, die auf eine nicht vorhandene Datei verweist. Wenn Sie eine nicht vorhandene Datei anfordern, generiert HTTP-Server normalerweise eine spezielle HTML-Seite mit einem Fehler "404 - Datei nicht gefunden" und sendet diese an Sie. Sie erhalten also immer eine Datei, selbst wenn Sie eine nicht vorhandene Datei anfordern. Wenn Sie dieses Verhalten nicht möchten, setzen Sie diesen Tag auf `True`. In diesem Fall wird `DownloadFile()` fehlschlagen, wenn eine ungültige Datei angefordert wird und Sie erhalten keine 404-Fehlerseite. Standardmäßig ist dieser Tag auf `False` gesetzt, was bedeutet, dass eine Fehlerseite generiert wird. Dieser Tag wird nur für das Herunterladen von Dateien von einer HTTP-Quelle unterstützt.

**SilentFail:** Wenn Sie diesen Tag auf `True` setzen, wird `DownloadFile()` niemals einen Fehler auslösen, sondern einfach still eine Fehlermeldung im ersten Rückgabewert `data$` und -1 im zweiten Rückgabewert `count` zurückgeben, um anzuzeigen, dass ein Fehler aufgetreten ist. Wenn dieser Tag auf `False` gesetzt ist, lädt `DownloadFile()` einen Systemfehler für alle auftretenden Fehler. Standardwert ist `False`.

**Redirect:** Gibt an, ob der Webserver Sie zu einer neuen URL umleiten darf. Diese Vorgabe ist `True`, was bedeutet, dass die Umleitung erlaubt ist. Dieser Tag wird nur beim Herunterladen von Dateien von einer HTTP-Quelle unterstützt.

**Post:** Dieser Tag wird nur unterstützt, wenn mit einem HTTP-Server gearbeitet wird. Wenn dieser Tag angegeben wird, sendet `DownloadFile()` eine POST-Anforderung an den HTTP-Server anstelle einer GET-Anforderung. Eine POST-Anforderung hat den Vorteil, dass Sie zusätzliche Daten an Ihre Anforderung anhängen können. So wird es häufig für die Übermittlung der Inhalte von Web-Formularen oder für das Hochladen von Dateien über HTTP verwendet. Die Daten, die an die POST-Anforderung angehängt werden sollen, müssen in diesem Tag als Zeichenkette angegeben werden. Sie können den Datentyp mit dem Tag `PostType` einstellen (siehe unten).

**PostType:** Dieser Tag wird nur behandelt, wenn der Tag `Post` ebenfalls angegeben wurde. Wenn dies der Fall ist, gibt `PostType` den Datentyp innerhalb des Tags `Post` an. Der Typ muss als MIME-Inhaltstyp übergeben werden. Dieser Tag ist standardmäßig auf "application/x-www-form-urlencoded" gesetzt. Dies ist der MIME-Typ, der für die Übermittlung des Inhalts von Webformularen an Perl (CGI)-Skripten verwendet wird.

**UserAgent:**

Mit diesem Tag können Sie den User Agent ändern, den `DownloadFile()` an den Zielsever sendet. Dies ist nützlich bei Servern, die die Zusammenarbeit mit unbekannten Benutzern verweigern. Standardmäßig sendet `DownloadFile()` "Hollywood" in das User-Agent-Feld von HTTP-Anfragen. Dieser Tag wird nur für das Herunterladen von Dateien einer HTTP-Quelle unterstützt. (V5.2)

**CustomHeaders:**

Mit diesem Tag können Sie eine Reihe von benutzerdefinierten Headern angeben, die bei der Anforderung an den HTTP-Server gesendet werden sollen. Dies kann für einige Feinabstimmungen für gewisse Server nützlich sein. Beachten Sie, dass die einzelnen Header-Elemente durch einen Wagenrücklauf und einen Zeilenvorschub terminiert werden müssen. Dieser Tag wird nur unterstützt, wenn das HTTP-Protokoll verwendet wird. (V6.0)

**Encoded:** Setzen Sie diesen Tag auf `True`, wenn die URL, die Sie an diesen Befehl übergeben haben, bereits korrekt mit ESC-Zeichen versehen wurde. Wenn dieser Tag auf `True` gesetzt ist, wird `DownloadFile()` keine ESC-codierten Zeichen formatieren. Stattdessen wird erwartet, dass Sie eine URL übergeben, die bereits korrekt mit ESC-Zeichen versehen wurde, somit sie direkt für Serveranforderungen ohne zusätzliche ESC-Zeichenumwandlungen verwendet werden kann. (V6.1)

**Protocol:**

Mit diesem Tag kann das Internetprotokoll angegeben werden, das beim Öffnen der Verbindung verwendet werden soll. Dies kann eine der folgenden speziellen Konstanten sein:

**#IPV4:** Verwendet die Internetprotokollversion 4 (IPv4).

**#IPV6:** Verwendet die Internetprotokollversion 6 (IPv6). Beachten Sie, dass **#IPV6** auf AmigaOS und kompatiblen Systemen derzeit nicht unterstützt wird.

**#IPAUTO:** Lassen Sie das Host-Betriebssystem das zu verwendende Internetprotokoll bestimmen.

Dieser Tag verwendet standardmäßig den Standardprotokolltyp, der mit `SetNetworkProtocol()` festgelegt wird. Standardmäßig ist dies aus historischen Gründen und aus Gründen der Portabilität **#IPV4**. Siehe [Abschnitt 39.23 \[SetNetworkProtocol\]](#), [Seite 795](#), für Details. (V8.0)

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Netzwerkadapter angeben, die zum Herstellen der angegebenen Verbindung benutzt werden sollten. Dies muss auf eine Zeichenkette gesetzt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V8.0)

**SSL:** Setzen Sie diesen Tag auf `True`, um eine Verbindung über TLS/SSL-Verschlüsselung anzufordern. Beachten Sie, dass die Einstellung dieses Tags



bei Verwendung des integrierten Netzwerkadapters von Hollywood keine Auswirkungen hat, da dieser keine TLS/SSL-Verbindungen unterstützt. Möglicherweise gibt es jedoch einen Netzwerkadapter, der von einem Plugin bereitgestellt wird, das TLS/SSL unterstützt. Wenn Sie diesen Tag auf **True** setzen, leitet Hollywood Ihren Wunsch nach einer TLS/SSL-Verbindung an den Netzwerkadapter weiter, der vom Plugin bereitgestellt wird. Beachten Sie jedoch, dass Sie diesen Tag normalerweise nicht setzen müssen, wenn das Schema der URL bereits eine SSL-Verbindung mit einem Präfix wie "https://" oder "ftps://" angibt. (V8.0)

**Async:** Wenn dies auf **True** gesetzt ist, arbeitet `DownloadFile()` im asynchronen Modus. Das bedeutet, dass der Befehl sofort zurückkehrt und falls Ihr Skript während des Vorgangs etwas anderes tun muss, Ihnen ein asynchroner Operationshandler übergibt. Sie können dann diesen asynchronen Operationshandler verwenden, um den Vorgang abzuschließen, indem Sie wiederholt `ContinueAsyncOperation()` aufrufen, bis **True** zurückgegeben wird. Dies ist sehr nützlich, z.B. für die Anzeige eines Fortschrittsbalken oder ähnlichem. Indem Sie `DownloadFile()` in den asynchronen Modus versetzen, ist es Ihrem Skript leicht möglich, während der Verarbeitung des Vorgangs etwas anderes zu tun. Siehe [Abschnitt 19.4 \[ContinueAsyncOperation\]](#), [Seite 234](#), für Details. Voreingestellt ist **False**. (V9.0)

**Verbose:** Dieser Tag kann auf **True** gesetzt werden, um detaillierte Protokollinformationen über die Verbindung und die Protokollinteraktion mit dem Server anzufordern. Dies wird derzeit nur von Hollywood-Plugins verwendet. Wenn Sie also den internen Netzwerkadapter von Hollywood verwenden, hat das Setzen dieses Tags auf **True** keine Auswirkung. Plugins können jedoch erweiterte Verbindungsinformationen bereitstellen, wenn dieser Tag auf **True** gesetzt wurde. Die Voreinstellung ist **False**. (V9.0)

**FileAdapter:** Dieser Tag wird nur verwendet, wenn auch der Tag **File** gesetzt ist. In diesem Fall können Sie mit **FileAdapter** einen oder mehrere Dateiadapter angeben, die die angegebene Datei speichern sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Zeichenkette setzen, die den Namen eines oder mehrerer Adapter enthält. Voreingestellt ist **default**. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V10.0)

**UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Datei- und Netzwerkdapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Der optionale Parameter **func** kann verwendet werden, um eine Callback-Funktion zu übergeben, die von Zeit zu Zeit von `DownloadFile()` aufgerufen wird, so dass Sie beispielsweise eine Fortschrittsanzeige aktualisieren können. Die Callback-Funktion, die Sie hier angeben, wird mit einem einzigen Tabellenargument aufgerufen. Hier ist ei-

ne Übersicht der Tabellenfelder, die initialisiert werden, bevor `DownloadFile()` Ihre Callback-Funktion ausführt:

**Action:** `#DOWNLOADFILE_STATUS`

**Count:** Enthält die Anzahl der Bytes, die bereits heruntergeladen wurden.

**Total:** Enthält die Größe der herunterzuladenden Datei.

**UserData:**

Enthält den Wert, den Sie im Argument `userdata` übergeben haben.

Die Callback-Funktion vom Typ `#DOWNLOADFILE_STATUS` sollte normalerweise `False` zurückgeben. Wenn sie `True` zurückgibt, wird der Downloadvorgang abgebrochen.

schließlich gibt es ein viertes optionales Argument mit dem Namen `userdata`. Der Wert, den Sie hier angeben, wird bei jedem Aufruf an Ihre Callback-Funktion übergeben. Dies ist sinnvoll, wenn Sie vermeiden wollen, mit globalen Variablen zu arbeiten. Mit dem Argument `userdata` können Sie einfach Daten an Ihre Callback-Funktion übergeben. Sie können einen Wert eines beliebigen Typs in `userdata` verwenden. Zahlen, Zeichenketten, Tabellen und sogar Funktionen können als Benutzerdaten übergeben werden.

Wenn Sie eine Zeichenfolge herunterladen, können Sie den Befehl `StringToFile()`-Kürzelfunktion verwenden, um die von `DownloadFile()` zurückgegebene Zeichenkette in eine Datei zu konvertieren. Alternativ können Sie den Befehl `DefineVirtualFileFromString()` verwenden, um eine virtuelle Datei aus einer Zeichenkettenquelle zu erstellen. Dies kann z.B. nützlich sein, wenn Sie eine Bilddatei herunterladen, die Sie mit dem Befehl `LoadBrush()` in Hollywood laden möchten. Mit `DefineVirtualFileFromString()` können Sie diese Datei direkt in Hollywood laden, ohne sie vorher in eine temporären Datei speichern zu müssen.

**Wichtiger Hinweis:** Stellen Sie sicher, dass Sie die zurückgegebene Zeichenkette `data$` auf `Nil` setzen, wenn Sie sie nicht mehr benötigen. Durch das setzen auf `Nil` signalisieren Sie dem Speicherbereiniger von Hollywood, dass er den belegten Arbeitsspeicher durch diese Zeichenkette wieder freigeben kann. Dies ist besonders für große Dateien wichtig. Wenn Sie eine große Datei herunterladen, auf der Festplatte speichern und die Zeichenkette nicht auf `Nil` setzen, werden Sie eine Menge Arbeitsspeicher verschwenden.

## EINGABEN

**url\$** URL der Datei, die runtergeladen werden soll

**options** optional: eine Tabelle mit weiteren Optionen für das Runterladen

**func** optional: eine Callback-Funktion, die von Zeit zu Zeit aufgerufen wird

**userdata** optional: Benutzerdaten, welche an die Callback-Funktion übergeben werden

## RÜCKGABEWERTE

**data\$** die Daten, welche aus dem Netzwerkpuffer gelesen wurden, oder eine leere Zeichenfolge, wenn der Tag `File` in der Optionstabelle angegeben wurde

**count** Anzahl der erfolgreich übertragenen Bytes

## BEISPIEL

```
DownloadFile("http://www.airsoftsoftwair.de/images/products/" ..
```

```
"hollywood/47_shot1.jpg", {File = "47_shot1.jpg"})
```

Der obige Code lädt die angegebene Datei herunter und speichert sie als "47\_shot1.jpg" in das aktuelle Verzeichnis.

```
DownloadFile("http://www.<your server>.com/cgi-bin/formmailer.cgi",
    {Post = "sender=Hollywood&mail=me@hollywood-mal.de" ..
    "&message=Hello from Hollywood!"}))
```

Der obige Code zeigt an, dass ein CGI-Skript mit `DownloadFile()` aufgerufen wird. Die im Tag `Post` angegebenen Daten werden mit der POST-Methode an den HTTP-Server übergeben.

```
DownloadFile("https://www.hollywood-mal.com/index.html", {
    File = "index.html",
    CustomHeaders = "Accept-Encoding: gzip, deflate\r\n"})
```

Der obige Code lädt die angegebene Datei herunter und sendet einen benutzerdefinierten Header, um dem Server mitzuteilen, dass er die Datei auch als gzip oder flate komprimierte Datei senden kann.

```
@REQUIRE "hurl"
```

```
...
```

```
DownloadFile("https://www.hollywood-mal.com/index.html", {
    File = "index.html", Adapter = "hurl"})
```

Der obige Code lädt eine Datei mit dem HTTPS-Protokoll herunter. Da Hollywood SSL/TLS standardmäßig nicht unterstützt, verwendet dieser Code das hURL-Plugin für den Vorgang, da hURL SSL/TLS bereitstellt. hURL wird durch Übergabe von `hurl` im Tag `Adapter` aktiviert.

## 39.7 GetConnectionIP

### BEZEICHNUNG

`GetConnectionIP` – gibt die IP-Adresse der Gegenseite zurück (V5.0)

### ÜBERSICHT

```
ip$ = GetConnectionIP(id[, type])
```

### BESCHREIBUNG

Dieser Befehl gibt die IP-Adresse des in `id` angegebenen Verbindungsobjekts zurück. Dies kann entweder die Bezeichnung einer Server-Verbindung sein, die durch einen Aufruf von `OpenConnection()` zustande kam, den Identifikator einer Client-Verbindung sein, die sich durch die Ereignisse `OnConnect` und `OnReceiveData` des Befehls `InstallEventHandler()` ergeben haben oder es kann der Identifikator eines UDP-Objekts sein, das von `CreateUDPObject()` erstellt wird. Die IP-Adresse der Gegenseite wird von diesem Befehl als Zeichenkette zurückgegeben.

Das optionale Argument `type` gibt den Typ des in Argument 1 übergebenen Netzwerkobjekts an. Die folgenden Typen werden derzeit von diesem Befehl unterstützt:

**#NETWORKCONNECTION:**

Ermittelt die IP einer Verbindung, die durch einen Aufruf von `OpenConnection()` zustande kam oder die IP einer Clientverbindung, die durch Überwachen der Ereignisse `OnConnect` und `OnReceiveData` abgerufen wird, die mit dem Befehl `InstallEventHandler()` installiert wurden.

**#NETWORKUDP:**

Ermittelt die IP-Adresse eines UDP-Objekts, das mit dem Befehl `CreateUDPObject()` erstellt wurde.

Wenn Sie das optionale Argument `type` nicht angeben, ist standardmäßig `#NETWORKCONNECTION` voreingestellt.

**EINGABEN**

<code>id</code>	Identifikator des Verbindungsobjekts
<code>type</code>	optional: Typ des Netzwerkobjekts, das abgefragt werden soll (Standardeinstellung ist <code>#NETWORKCONNECTION</code> ) (V8.0)

**RÜCKGABEWERTE**

<code>ip\$</code>	IP-Adresse der Gegenseite der Verbindung als Zeichenkette
-------------------	---

**BEISPIEL**

```
OpenConnection(1, "www.airsoftsoftwair.de", 80)
DebugPrint(GetConnectionIP(1), GetConnectionPort(80))
CloseConnection(1)
```

Der obige Code verbindet sich mit dem Port 80 von `www.airsoftsoftwair.de` und erhält dann die IP-Adresse dieses Servers.

## 39.8 GetConnectionPort

**BEZEICHNUNG**

`GetConnectionPort` – gibt die Portnummer der Gegenseite zurück (V5.0)

**ÜBERSICHT**

```
port = GetConnectionPort(id[, type])
```

**BESCHREIBUNG**

Dieser Befehl gibt die Portnummer des in `id` angegebenen Verbindungsobjekts zurück. Dies kann entweder der Identifikator einer Serververbindung sein, die durch einen Aufruf von `OpenConnection()` zustande kam, der Identifikator einer Client-Verbindung, die sich durch die Ereignisse `OnConnect` und `OnReceiveData` des Befehls `InstallEventHandler()` ergeben haben oder es kann der Identifikator eines UDP-Objekts sein, das von `CreateUDPObject()` erstellt wurde.

Das optionale Argument `type` gibt den Typ des in Argument 1 übergebenen Netzwerkobjekts an. Die folgenden Typen werden derzeit von diesem Befehl unterstützt:

**#NETWORKCONNECTION:**

Ermittelt den Port einer Verbindung, die durch einen Aufruf von `OpenConnection()` zustande kam oder die IP einer Clientverbindung, die

durch Überwachen der Ereignisse `OnConnect` und `OnReceiveData` abgerufen wird, die mit dem Befehl `InstallEventHandler()` installiert wurden.

**#NETWORKUDP:**

Ermittelt den Port eines UDP-Objekts, das mit dem Befehl `CreateUDPObject()` erstellt wurde.

Wenn Sie das optionale Argument `type` nicht angeben, ist standardmäßig `#NETWORKCONNECTION` voreingestellt.

## EINGABEN

`id` Identifikator des Verbindungsobjekts

`type` optional: Typ des Netzwerkobjekts, das abgefragt werden soll (Standardeinstellung ist `#NETWORKCONNECTION`) (V8.0)

## RÜCKGABEWERTE

`port` Portnummer der Gegenseite der Verbindung

## BEISPIEL

Siehe [Abschnitt 39.7 \[GetConnectionIP\]](#), Seite 779.

# 39.9 GetConnectionProtocol

## BEZEICHNUNG

`GetConnectionProtocol` – ermittelt das Protokoll der Gegenseite (V8.0)

## ÜBERSICHT

```
protocol = GetConnectionProtocol(id[, type])
```

## BESCHREIBUNG

Dieser Befehl gibt das Internetprotokoll des in `id` angegebenen Verbindungsobjekts zurück. Dies kann entweder der Identifikator einer Serververbindung sein, die durch einen Aufruf von `OpenConnection()` zustande kam, der Identifikator einer Client-Verbindung, die sich durch die Ereignisse `OnConnect` und `OnReceiveData` des Befehls `InstallEventHandler()` ergeben haben oder es kann der Identifikator eines UDP-Objekts sein, das von `CreateUDPObject()` erstellt wird.

Das optionale Argument `type` gibt den Typ des in Argument 1 übergebenen Netzwerkobjekts an. Die folgenden Typen werden derzeit von diesem Befehl unterstützt:

**#NETWORKCONNECTION:**

Ermittelt die IP einer Verbindung, die durch einen Aufruf von `OpenConnection()` zustande kam oder die IP einer Clientverbindung, die durch Überwachen der Ereignisse `OnConnect` und `OnReceiveData` abgerufen wird, die mit dem Befehl `InstallEventHandler()` installiert wurden.

**#NETWORKUDP:**

Ermittelt die IP-Adresse eines UDP-Objekts, das mit dem Befehl `CreateUDPObject()` erstellt wurde.

Wenn Sie das optionale Argument **type** nicht angeben, ist standardmäßig **#NETWORKCONNECTION** voreingestellt.

Der Rückgabewert in **protocol** ist eine der folgenden vordefinierten Konstanten:

**#IPV4:** Verwendet die Internetprotokollversion 4 (IPv4). IPv4-Adressen sind auf 32 Bit beschränkt und werden unter Verwendung von vier durch drei Punkte getrennten Zahlen dargestellt, z.B. 127.0.0.1.

**#IPV6:** Verwendet die Internetprotokollversion 6 (IPv6). IPv6-Adressen verwenden 128 Bits und werden durch acht Gruppen von vier Hexadezimalziffern dargestellt, z.B. 2001:0db8:85a3:0000:0000:8a2e:0370:7334. Beachten Sie, dass **#IPV6** auf AmigaOS und kompatiblen Systemen derzeit nicht unterstützt wird.

**#IPAUTO:** Das Host-Betriebssystem bestimmte das zu verwendende Internetprotokoll.

**#IPUNKNOWN:**  
Das Netzwerkobjekt verwendet ein unbekanntes Protokoll.

#### EINGABEN

**id** Identifikator des Verbindungsobjekts

**type** optional: Typ des Netzwerkobjekts, das abgefragt werden soll (Standardeinstellung ist **#NETWORKCONNECTION**)

#### RÜCKGABEWERTE

**protocol** Protokoll der Gegenseite der Verbindung

## 39.10 GetHostName

#### BEZEICHNUNG

GetHostName – gibt den Standard-Hostname des Computers zurück (V5.0)

#### ÜBERSICHT

**host\$** = GetHostName()

#### BESCHREIBUNG

Dieser Befehl gibt den Standard-Hostname des Computers zurück, auf der Hollywood gerade läuft.

#### EINGABEN

keine

#### RÜCKGABEWERTE

**host\$** Standard-Hostname des Computers

## 39.11 GetLocalInterfaces

### BEZEICHNUNG

GetLocalInterfaces – gibt die lokalen Netzwerkschnittstellen zurück (V9.0)

### ÜBERSICHT

```
t = GetLocalInterfaces([linklocal])
```

### BESCHREIBUNG

Dieser Befehl gibt eine Liste aller derzeit verfügbaren Netzwerkschnittstellen zurück. Auf diese Weise können Sie bequem die lokale IP-Adresse eines Systems ermitteln. Wenn das optionale Argument `linklocal` auf `True` gesetzt ist, werden auch Link-Local-Adressen eingeschlossen.

`GetLocalInterfaces()` gibt eine Tabelle zurück, die eine Reihe von Untertabellen enthält, die jeweils eine lokale Schnittstelle beschreiben. Die folgenden Felder werden in jeder Untertabelle initialisiert:

**Name:** Der Name der Schnittstelle.

**Address:** Die Adresse der Schnittstelle. Abhängig von der Einstellung des Tags `Protocol` (siehe unten) kann dies entweder eine IPv4- oder eine IPv6-Adresse sein.

**Protocol:**

Das Protokoll der Schnittstelle. Dies wird eine der folgenden vordefinierten Konstanten sein:

**#IPv4:** Internetprotokollversion 4 (IPv4). IPv4-Adressen sind auf 32 Bit beschränkt und werden unter Verwendung von vier durch drei Punkte getrennten Zahlen dargestellt, z.B. 127.0.0.1.

**#IPv6:** Internetprotokollversion 6 (IPv6). IPv6-Adressen verwenden 128 Bits und werden durch acht Gruppen von vier Hexadezimalziffern dargestellt, z.B. 2001:0db8:85a3:0000:0000:8a2e:0370:7334. Beachten Sie, dass **#IPv6** auf AmigaOS und kompatiblen Systemen derzeit nicht unterstützt wird.

### EINGABEN

`linklocal`

optional: `True`, um Link-Local-Adressen in die Rückgabetabelle aufzunehmen, `False`, um sie auszuschließen (Standardwert ist `False`)

### RÜCKGABEWERTE

`t` Tabelle mit einer Liste aller lokalen Schnittstellen

### BEISPIEL

```
t = GetLocalInterfaces()
For Local k = 0 To ListItems(t) - 1
    NPrint(t[k].Name, t[k].Address, t[k].Protocol)
Next
```

Der obige Code gibt Informationen über alle verfügbaren Netzwerkschnittstellen aus.

## 39.12 GetLocalIP

### BEZEICHNUNG

GetLocalIP – gibt die IP-Adresse der lokalen Seite zurück (V5.0)

### ÜBERSICHT

```
ip$ = GetLocalIP(id[, type])
```

### BESCHREIBUNG

Dieser Befehl gibt die IP-Adresse auf der lokalen Seite des in `id` angegebenen Netzwerkobjekts zurück. Das optionale Argument `type` gibt den Typ des in Argument 1 übergebenen Netzwerkobjekts an. Folgende Typen werden zur Zeit von diesem Befehl unterstützt:

#### #NETWORKCONNECTION:

Durch Abfragen der lokalen IP einer Verbindung, die durch den Aufruf des Befehls `OpenConnection()` zustande kam oder die lokale IP einer Client-Verbindung, die sich durch die Ereignisse `OnConnect` und `OnReceiveData` des Befehls `InstallEventHandler()` ergeben haben.

#### #NETWORKSERVER:

Durch Abfragen der lokalen IP eines Servers, der mit dem Befehl `CreateServer()` erstellt wurde.

#### #NETWORKUDP:

Abfrage der lokalen IP eines UDP-Objekts, das mit dem Befehl `CreateUDPObject()` erstellt wurde.

Wenn Sie das optionale Argument `type` weglassen, wird standardmäßig `#NETWORKCONNECTION` eingegeben.

### EINGABEN

<code>id</code>	Netzwerkobjekt
<code>type</code>	optional: Typ des Netzwerkobjekts vom Argument 1 (voreingestellt ist <code>#NETWORKCONNECTION</code> )

### RÜCKGABEWERTE

<code>ip\$</code>	IP-Adresse der lokalen Seite der Verbindung
-------------------	---

## 39.13 GetLocalPort

### BEZEICHNUNG

GetLocalPort – gibt die Portnummer der lokalen Seite zurück (V5.0)

### ÜBERSICHT

```
port = GetLocalPort(id[, type])
```

### BESCHREIBUNG

Dieser Befehl gibt die Portnummer auf der lokalen Seite des in `id` angegebenen Netzwerkobjekts zurück. Das optionale Argument `type` gibt den Typ des in Argument 1



übergebenen Netzwerkobjekts an. Folgende Typen werden zur Zeit von diesem Befehl unterstützt:

**#NETWORKCONNECTION:**

Durch Abfrage des lokalen Ports, welcher mit dem Befehl `OpenConnection()` zustande kam oder dem lokalen Port einer Clientverbindung, die sich durch die Ereignisse `OnConnect` und `OnReceiveData` des Befehls `InstallEventHandler()` ergeben haben.

**#NETWORKSERVER:**

Abfrage des lokalen Ports eines Servers, der mit dem Befehl `CreateServer()` erstellt wurde.

**#NETWORKUDP:**

Abfrage des lokalen Ports eines UDP-Objekts, das mit dem Befehl `CreateUDPObject()` erstellt wurde.

Wenn Sie das optionale Argument `type` weglassen, wird standardmäßig `#NETWORKCONNECTION` eingegeben.

**EINGABEN**

<code>id</code>	Netzwerkobjekt
<code>type</code>	optional: Typ des Netzwerkobjekts vom Argument 1 (voreingestellt ist <code>#NETWORKCONNECTION</code> )

**RÜCKGABEWERTE**

<code>port</code>	Portnummer der lokalen Seite der Verbindung
-------------------	---

## 39.14 GetLocalProtocol

**BEZEICHNUNG**

`GetLocalProtocol` – ermittelt das Protokoll der lokalen Seite (V8.0)

**ÜBERSICHT**

```
protocol = GetLocalProtocol(id[, type])
```

**BESCHREIBUNG**

Dieser Befehl gibt das Internetprotokoll auf der lokalen Seite des in `id` angegebenen Netzwerkobjekts zurück. Das optionale Argument `type` gibt den Typ des in Argument `id` übergebenen Netzwerkobjekts an. Die folgenden Typen werden derzeit von diesem Befehl unterstützt:

**#NETWORKCONNECTION:**

Durch Abfragen der lokalen IP einer Verbindung, die durch den Aufruf des Befehls `OpenConnection()` zustande kam oder die lokale IP einer Client-Verbindung, die sich durch die Ereignisse `OnConnect` und `OnReceiveData` des Befehls `InstallEventHandler()` ergeben haben.

**#NETWORKSERVER:**

Durch Abfragen der lokalen IP eines Servers, der mit dem Befehl `CreateServer()` erstellt wurde.

**#NETWORKUDP:**

Abfrage der lokalen IP eines UDP-Objekts, das mit dem Befehl `CreateUDPObject()` erstellt wurde.

Wenn Sie das optionale Argument `type` weglassen, wird standardmäßig `#NETWORKCONNECTION` eingegeben.

Der Rückgabewert in `protocol` ist eine der folgenden vordefinierten Konstanten:

**#IPV4:** Verwendet die Internetprotokollversion 4 (IPv4). IPv4-Adressen sind auf 32 Bit beschränkt und werden unter Verwendung von vier durch drei Punkte getrennten Zahlen dargestellt, z.B. 127.0.0.1.

**#IPV6:** Verwendet die Internetprotokollversion 6 (IPv6). IPv6-Adressen verwenden 128 Bits und werden durch acht Gruppen von vier Hexadezimalziffern dargestellt, z.B. 2001:0db8:85a3:0000:0000:8a2e:0370:7334. Beachten Sie, dass **#IPV6** auf AmigaOS und kompatiblen Systemen derzeit nicht unterstützt wird.

**#IPAUTO:** Das Host-Betriebssystem bestimmte das zu verwendende Internetprotokoll.

**#IPUNKNOWN:**

Das Netzwerkobjekt verwendet ein unbekanntes Protokoll.

**EINGABEN**

`id` Identifikator des Netzwerkobjekt

`type` optional: Typ des Netzwerkobjekts, das abgefragt werden soll (Standardeinstellung ist `#NETWORKCONNECTION`) (V8.0)

**RÜCKGABEWERTE**

Protokoll der lokalen Seite der Verbindung (mögliche Rückgabewerte siehe oben)

## 39.15 GetMACAddress

**BEZEICHNUNG**

GetMACAddress – gibt die MAC-Adresse des Host-Systems zurück (V7.0)

**ÜBERSICHT**

`addr$ = GetMACAddress()`

**BESCHREIBUNG**

Dieser Befehl gibt die MAC-Adresse des Host-Systems als 6 Oktette zurück, die durch Doppelpunkte getrennt sind, z.B. 12:34:56:78:9A:BC.

Wenn die MAC-Adresse nicht ermittelt werden kann, wird "Unknown" zurückgegeben.

Beachten Sie, dass Roadshow unter AmigaOS 3 der einzige TCP/IP-Stack ist, der das Ermitteln der MAC-Adresse unterstützt.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`addr$` MAC-Adresse des Host-System oder "Unknown"

## 39.16 IsOnline

### BEZEICHNUNG

IsOnline – prüft, ob eine Internetverbindung verfügbar ist (V5.0)

### ÜBERSICHT

```
bool = IsOnline()
```

### BESCHREIBUNG

Mit diesem Befehl können Sie überprüfen, ob eine Internetverbindung verfügbar ist oder nicht. Er wird **True** zurückgeben, wenn die Verbindung zum Internet möglich ist, andernfalls **False**.

### EINGABEN

keine

### RÜCKGABEWERTE

**bool**            Gibt an, ob Internet verfügbar ist oder nicht

## 39.17 OpenConnection

### BEZEICHNUNG

OpenConnection – stellt eine Verbindung zu einem Server her (V5.0)

### ÜBERSICHT

```
[id] = OpenConnection(id, server$, port[, table])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um eine neue Verbindung zu dem in **server\$** angegebenen Server herzustellen. Dies kann entweder ein Hostname oder direkt eine IP-Adresse sein. Das dritte Argument **port** gibt die Portnummer an, über die **OpenConnection()** versuchen soll, eine Verbindung herzustellen. Im ersten Argument **id** müssen Sie einen Identifikator übergeben, der benötigt wird, um später auf diese Verbindung zu verweisen. Alternativ können Sie auch **Nil** als erstes Argument übergeben. In diesem Fall wird **OpenConnection()** automatisch eine ID auswählen und an Sie zurückgeben.

Sobald die Verbindung erfolgreich hergestellt wurde, können Sie die Befehle **SendData()** und **ReceiveData()** verwenden, um mit dem Server zu kommunizieren. Wenn Sie fertig sind, sollten Sie den Befehl **CloseConnection()** aufrufen, um die Verbindung zum Server zu trennen.

Bitte beachten Sie, dass Schemapräfixe wie "http://" oder "ftp://" nicht Teil eines Servernamens sind. Diese geben nur das Protokoll an, das für die Kommunikation mit dem Server verwendet wird. Wenn Sie also eine Verbindung mit **http://www.airsoftsoftwair.de** erstellen wollen, müssen Sie als Servername "www.airsoftsoftwair.de" und Port 80 als Portnummer angeben, da 80 der Standard-HTTP-Port ist. Siehe unten für ein Beispiel. Eine Ausnahme von dieser Regel kann der Fall sein, wenn Sie den Tag **Adapter** verwenden, um mit einem Netzwerkadapter die Verbindung herzustellen. In diesem Fall werden Sie möglicherweise vom Netzwerkadapter aufgefordert, einen Schemanamen wie "http://" oder "ftp://"

anzugeben. Dieser hängt jedoch wirklich vom Netzwerkadapter ab. Der integrierte Adapter von Hollywood unterstützt keine Schemapräfixe und erwartet, dass Sie die IP-Adresse oder den Hostnamen direkt angeben.

Ab Hollywood 8.0 akzeptiert `OpenConnection()` ein optionales Tabellenargument `table`, mit dem weitere Optionen angegeben werden können. Die folgenden Tags werden derzeit vom optionalen Tabellenargument erkannt:

**Protocol:**

Mit diesem Tag kann das Internetprotokoll angegeben werden, das beim Öffnen der Verbindung verwendet werden soll. Dies kann eine der folgenden speziellen Konstanten sein:

**#IPV4:** Verwendet die Internetprotokollversion 4 (IPv4). IPv4-Adressen sind auf 32 Bit beschränkt und werden unter Verwendung von vier durch drei Punkte getrennten Zahlen dargestellt, z.B. 127.0.0.1.

**#IPV6:** Verwendet die Internetprotokollversion 6 (IPv6). IPv6-Adressen verwenden 128 Bits und werden durch acht Gruppen von vier Hexadezimalziffern dargestellt, z.B. 2001:0db8:85a3:0000:0000:8a2e:0370:7334. Beachten Sie, dass **#IPV6** auf AmigaOS und kompatiblen Systemen derzeit nicht unterstützt wird.

**#IPAUTO:** Lassen Sie das Host-Betriebssystem das zu verwendende Internetprotokoll bestimmen. Sie können dann `GetConnectionProtocol()` verwenden, um herauszufinden, welches Internetprotokoll das Host-Betriebssystem für diesen Server ausgewählt hat. Siehe [Abschnitt 39.9 \[GetConnectionProtocol\]](#), [Seite 781](#), für Details.

Der Tag `Protocol` verwendet standardmäßig den Standardprotokolltyp, der mit `SetNetworkProtocol()` festgelegt wird. Standardmäßig ist dies aus historischen Gründen und aus Gründen der Portabilität **#IPV4**. Siehe [Abschnitt 39.23 \[SetNetworkProtocol\]](#), [Seite 795](#), für Details. (V8.0)

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Netzwerkadapter angeben, die zum Herstellen der angegebenen Verbindung benutzt werden sollten. Dies muss auf eine Zeichenkette gesetzt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V8.0)

**SSL:** Setzen Sie diesen Tag auf `True`, um eine Verbindung über TLS/SSL-Verschlüsselung anzufordern. Beachten Sie, dass die Einstellung dieses Tags bei Verwendung des integrierten Netzwerkadapters von Hollywood keine Auswirkungen hat, da dieser keine TLS/SSL-Verbindungen unterstützt. Möglicherweise gibt es jedoch einen Netzwerkadapter, der von einem Plugin bereitgestellt wird, das TLS/SSL unterstützt. Wenn Sie diesen Tag auf `True` setzen, leitet Hollywood Ihren Wunsch nach einer TLS/SSL-Verbindung an den Netzwerkadapter weiter, der vom Plugin bereitgestellt wird. (V8.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Datei- und Netzwerkdapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

**EINGABEN**

<code>id</code>	Identifikator für die neue Verbindung oder <code>Nil</code> für die <a href="#">automatische ID-Auswahl</a>
<code>server\$</code>	Server, mit dem Sie sich verbinden wollen
<code>port</code>	Portnummer für die Verbindung
<code>table</code>	optional: Tabellenargument mit weiteren Optionen (siehe oben) (V8.0)

**RÜCKGABEWERTE**

<code>id</code>	optional: Identifikator der Verbindung; wird nur zurückgegeben werden, wenn Sie <code>Nil</code> als <code>id</code> angegeben haben (siehe oben)
-----------------	---

**BEISPIEL**

```
OpenConnection(1, "www.airsoftsoftwair.de", 80)
SendData(1, "GET http://www.airsoftsoftwair.de/index.html " ..
           "HTTP/1.0\r\n\r\n")
a$ = ReceiveData(1, #RECEIVEALL)
Print(a$)
CloseConnection(1)
```

Der oben stehende Code verbindet sich mit <http://www.airsoftsoftwair.de> und lädt die Index-HTML-Seite herunter.

## 39.18 ReceiveData

**BEZEICHNUNG**

`ReceiveData` – empfängt Daten über das Netzwerk (V5.0)

**ÜBERSICHT**

```
data$, count, done = ReceiveData(id, mode, ...)
data$, count, done = ReceiveData(id, #RECEIVEBYTES, maxbytes[, callback,
                                userdata])
data$, count, done = ReceiveData(id, #RECEIVEALL[, untilterm, callback,
                                userdata])
data$, count, done = ReceiveData(id, #RECEIVELINE[, callback, userdata])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um Daten von einem Server oder einem Client zu empfangen. Wenn Sie Daten von einem Server empfangen möchten, müssen Sie eine ID übergeben, welche beim Befehl `OpenConnection()` angegeben oder mit ihm erstellt wurde. Wenn es ein Server ist und Daten von einem Ihrer Clients abrufen möchten, müssen

Sie die Netzwerkennung des jeweiligen Clients übergeben. Sie können die Identifikatoren Ihrer Clients abrufen, indem Sie das Ereignisse `OnConnect` benutzen, welches Sie mit dem Befehl `InstallEventHandler()` installiert haben.

Das zweite Argument legt fest, wie viele Daten Sie vom Absender erhalten möchten. Derzeit werden folgende Modi unterstützt:

#### **#RECEIVEBYTES:**

Empfangen Sie alle verfügbaren Daten, aber nicht mehr als die angegebene Anzahl von Bytes. Wenn Sie diesen Modus verwenden, müssen Sie die maximale Anzahl von Bytes, die Sie empfangen möchten, im dritten Argument `maxbytes` übergeben. `ReceiveData()` erhält dann nie mehr Bytes als angegeben. Es kann jedoch vorkommen, dass weniger Bytes zurückgegeben werden, falls nicht genügend Daten vom Absender verfügbar sind. Die Anzahl der übermittelten Bytes werden im zweiten Rückgabewert `count` enthalten sein.

#### **#RECEIVEALL:**

Erhalten Sie alle aktuell verfügbaren Daten. Wenn Sie das optionale Argument `untilterm` auf `True` setzen, wird `ReceiveData()` nichts zurückgegeben, bevor der Absender die Verbindung beendet, so dass Sie alle Daten empfangen können, die ein Absender mit nur einem einzigen Aufruf anbieten kann. Wenn `untilterm` auf `False` gesetzt ist, gibt `ReceiveData()` nur die aktuell verfügbaren Daten zurück. Er wird nicht warten, bis weitere Daten ankommen, aber er wird Ihnen sagen, ob mehr Daten zum Abrufen vorhanden sein werden (im dritten Rückgabewert `done`). Die Voreinstellung für `untilterm` ist `True`, d.h. `ReceiveData()` liest Daten, bis der Absender die Verbindung beendet.

#### **#RECEIVELINE:**

Empfangen Sie eine einzelne Textzeile vom Absender. Dieser Modus darf nur bei nicht binären Daten verwendet werden. Der Wagenrücklauf und das Zeilenumbruchzeichen werden nicht in die zurückgegebene Zeichenkette `data$` aufgenommen. Sie werden aus dem Netzwerkpuffer gelesen, aber `ReceiveData()` nicht zurückgegeben, wenn Sie `#RECEIVELINE` verwenden.

`ReceiveData()` gibt drei Werte zurück: Der erste Rückgabewert `data$` ist eine Zeichenkette, die die Daten enthält, die vom Netzwerk empfangen wurden, oder eine leere Zeichenfolge, wenn Sie eine Callback-Funktion verwenden, um die Daten zu behandeln oder keine Daten gelesen werden konnten. Bitte beachten Sie, dass obwohl die Daten gelesen werden und als Zeichenkette zurückgegeben werden, ist sie nicht auf Text beschränkt. Sie kann auch binäre Daten enthalten, da die Zeichenketten von Hollywood Steuerzeichen und das Nullzeichen gut verarbeiten können. Der zweite Rückgabewert `count` gibt an, wie viele Bytes aus dem Netzwerkpuffer gelesen werden konnten. Ist dies 0, stehen derzeit keine Daten zur Verfügung. Der dritte Rückgabewert `done` ist nur sinnvoll, wenn der Übertragungsmodus `#RECEIVEALL` mit `untilterm` auf `False` gesetzt wurde. In diesem Fall gibt der dritte Rückgabewert an, wenn mehr Daten im Netzwerkpuffer vorhanden sind. Wenn es mehr Daten zu lesen gibt, dann ist `done False`, sonst ist es `True`.

Ab Hollywood 6.0 gibt es ein optionales Argument `callback`, mit dem Sie eine Callback-Funktion übergeben können, die die vom Server gelesenen Daten bearbeiten soll. Dies kann nützlich sein, wenn Sie große Datenmengen empfangen müssen, die nicht effizient in

einer Zeichenkette gespeichert werden können. Die Callback-Funktion könnte einfach die empfangenen Daten in eine Datei schreiben. Beachten Sie, dass `ReceiveData()` immer eine leere Zeichenfolge in `data$` zurückgibt, wenn Sie eine Callback-Funktion angeben. Die Callback-Funktion wird mit einem einzigen Argument aufgerufen: Eine Tabelle, die mehr Informationen enthält. Hier eine Übersicht der Tabellenfelder, die initialisiert werden, bevor `ReceiveData()` Ihre Callback-Funktion ausführt:

<b>Action:</b>	<code>#RECEIVEDATA_PACKET</code>
<b>Data:</b>	Die Daten, die vom Server empfangen wurden. Beachten Sie, dass dieses Feld binäre Daten enthalten kann.
<b>Count:</b>	Enthält die Anzahl der Bytes von <code>Data</code> .
<b>Total:</b>	Enthält die Gesamtanzahl von Bytes, die bereits empfangen wurden.
<b>UserData:</b>	Enthält den Wert, den Sie im Argument <code>userdata</code> übergeben haben.

Die Callback-Funktion des Typs `#RECEIVEDATA_PACKET` sollte normalerweise `False` zurückgeben. Wenn sie `True` zurückgibt, wird `ReceiveData()` seine Operationen abbrechen.

Schließlich gibt es noch das optionale Argument `userdata`. Der Wert, den Sie hier angeben, wird bei jedem Aufruf an Ihre Callback-Funktion übergeben. Dies ist sinnvoll, wenn Sie vermeiden wollen, mit globalen Variablen zu arbeiten. Mit dem Argument `userdata` können Sie einfach Daten an Ihre Callback-Funktion übergeben. Sie können einen Wert eines beliebigen Typs verwenden. Zahlen, Zeichenketten, Tabellen und sogar Funktionen können übergeben werden.

## EINGABEN

<code>id</code>	Identifikator des Senders
<code>mode</code>	der gewünschte Transfermodus; siehe oben für eine Liste der aktuell unterstützten Transfermodi
<code>...</code>	weitere Argumente; hängen vom angegebenen Übertragungsmodus ab; siehe oben

## RÜCKGABEWERTE

<code>data\$</code>	Daten, welche aus dem Netzwerkpuffer gelesen wurden oder eine leere Zeichenfolge, wenn eine Callback-Funktion angegeben ist
<code>count</code>	Anzahl der erfolgreich übertragenen Bytes
<code>done</code>	ob es mehr Daten im Netzwerkpuffer gibt oder nicht (nur wenn <code>#RECEIVEALL</code> zusammen mit <code>untilterm</code> auf <code>False</code> gesetzt ist)

## BEISPIEL

Siehe [Abschnitt 39.17 \[OpenConnection\]](#), Seite 787.

## 39.19 ReceiveUDPData

### BEZEICHNUNG

ReceiveUDPData – empfängt Daten über das UDP-Protokoll (V5.0)

### ÜBERSICHT

```
data$, ip$, port = ReceiveUDPData(id[, size])
```

### BESCHREIBUNG

Dieser Befehl kann zum Empfangen von Daten aus dem durch `id` angegebenen UDP-Objekt verwendet werden. Dieses UDP-Objekt muss zuvor mit dem Befehl `CreateUDPObject()` erstellt worden sein. Das optionale Argument `size` kann verwendet werden, um die maximale Anzahl der zu empfangenden Bytes festzulegen. Standardmäßig ist dieser Wert auf 8192 Bytes eingestellt, was auch die maximale Anzahl von Bytes ist, die `ReceiveUDPData()` verarbeiten kann. Sie können also `size` auf Werte unter 8192 Byte setzen, aber nicht auf mehr.

`ReceiveUDPData()` gibt drei Werte zurück: Der erste Rückgabewert `data$` ist eine Zeichenkette, die die vom UDP-Objekt empfangenen Daten enthält. Der zweite Rückgabewert `ip$` enthält die IP-Adresse des Absenders und der dritte Rückgabewert `port` die Portnummer des Absenders.

Beachten Sie, dass das mit `SetNetworkTimeout()` festgelegte globaler Netzwerk-Timeout derzeit von `ReceiveUDPData()` ignoriert wird.

### EINGABEN

<code>id</code>	Identifikator des UDP-Objekts
<code>size</code>	optional: maximale Anzahl zu empfangender Bytes (voreingestellt sind 8192 Byte)

### RÜCKGABEWERTE

<code>data\$</code>	die Daten, die aus dem Netzwerk gelesen wurden
<code>ip\$</code>	IP-Adresse des Senders
<code>port</code>	Portnummer des Senders

## 39.20 ResolveHostName

### BEZEICHNUNG

ResolveHostName – konvertiert den Hostnamen in eine IP-Adresse (V8.0)

### ÜBERSICHT

```
t = ResolveHostName(host$)
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den in `host$` angegebenen Hostnamen in eine IP-Adresse umzuwandeln. Im Gegensatz zu dem ähnlichen Befehl `ToIP()` gibt `ResolveHostName()` alle Informationen zurück, die in einer Tabelle abgerufen werden können. Dies ist besonders nützlich, wenn Sie Informationen zu den verfügbaren Internetprotokollen für den Hostnamen benötigen. Somit kann dieser Befehl mehrere



IP-Adressen für den angegebenen Hostnamen zurückgeben, z.B. eine IPv4- und eine IPv6-Adresse.

`ResolveHostName()` gibt eine Tabelle zurück, die eine Anzahl von Untertabellen enthält; eine für jede erfolgreich konvertierte IP-Adresse. Für jede dieser Untertabellen werden die folgenden Felder initialisiert:

**Address:** Die IP-Adresse des angegebenen Hostnamens im Format des jeweiligen Internetprotokolls (siehe unten).

**Protocol:**

Das von dieser IP-Adresse verwendete Internetprotokoll. Dies kann einer der folgenden speziellen Werte sein:

**#IPv4:** Internetprotokollversion 4 (IPv4). IPv4-Adressen sind auf 32 Bit beschränkt und werden unter Verwendung von vier durch drei Punkte getrennten Zahlen dargestellt, z.B. 127.0.0.1.

**#IPv6:** Internetprotokollversion 6 (IPv6). IPv6-Adressen verwenden 128 Bits und werden durch acht Gruppen von vier Hexadezimalziffern dargestellt, z.B. 2001:0db8:85a3:0000:0000:8a2e:0370:7334. Beachten Sie, dass **#IPv6** auf AmigaOS und kompatiblen Systemen derzeit nicht unterstützt wird.

**#IPUNKNOWN:**

Die IP-Adresse verwendet ein unbekanntes Protokoll.

## EINGABEN

`host$` Hostname, der konvertiert werden soll

## RÜCKGABEWERTE

`t` eine Tabelle mit allen Informationen (siehe oben)

## BEISPIEL

```
t = ResolveHostName("www.airsoftsoftwair.de")
For Local k = 0 To ListItems(t) - 1
    Print(t[k].address, t[k].protocol)
Next
```

Bei einem Windows 10-System konvertiert der obige Code zwei IP-Adressen für `www.airsoftsoftwair.de`: Eine IPv4- und eine IPv6-Adresse.

## 39.21 SendData

### BEZEICHNUNG

`SendData` – sendet Daten über das Netzwerk (V5.0)

### ÜBERSICHT

```
count = SendData(id, data$)
```

### BESCHREIBUNG

Mit diesem Befehl können Daten an einen Server oder einen Client gesendet werden. Wenn Sie Daten an einen Server senden möchten, müssen Sie eine ID des Befehls

`OpenConnection()` an diesen Befehl übergeben. Wenn Sie hingegen ein Server sind und Daten an einen Ihrer Clients senden möchten, müssen Sie den Identifikator des jeweiligen Clients übergeben. Sie erhalten die Identifikatoren Ihrer Clients, indem Sie den Ereignis-Handler `OnConnect` überprüfen, den Sie mit dem Befehl `InstallEventHandler()` aufrufen können.

Das zweite Argument `data$` ist eine Zeichenkette, die die Daten enthält, die an den Empfänger gesendet werden sollen. Beachten Sie, dass dieses Argument zwar eine Zeichenkette ist, aber nicht auf Text beschränkt ist. Sie können auch Rohdaten mit diesem Befehl senden, da die Zeichenketten von Hollywood sowohl Zeichencodes als auch das spezielle Nullzeichen problemlos verarbeiten können.

`SendData()` gibt die Anzahl der Bytes in `count` zurück, die erfolgreich an den Empfänger übertragen wurden. Dies kann weniger als die Anzahl der Bytes in `data$` sein.

#### EINGABEN

`id` ID des Empfängers  
`data$` Zeichenkette, welche die zu sendenden Daten enthält

#### RÜCKGABEWERTE

`count` Anzahl der erfolgreich übertragenen Bytes

#### BEISPIEL

Siehe [Abschnitt 39.17 \[OpenConnection\]](#), Seite 787.

## 39.22 SendUDPData

#### BEZEICHNUNG

`SendUDPData` – sendet Daten mit dem UDP-Protokoll (V5.0)

#### ÜBERSICHT

```
count = SendUDPData(id, data$[, ip$, port])
```

#### BESCHREIBUNG

Dieser Befehl sendet die in `data$` angegebenen Daten an den in `ip$` und `port` angegebenen Empfänger. Die Daten werden über das im ersten Argument `id` angegebene UDP-Objekt gesendet. Dieses UDP-Objekt muss zuvor mit dem `CreateUDPObject()` erstellt worden sein. Aus Leistungsgründen müssen Sie eine IP-Adresse direkt an diesen Befehl übergeben. Das Übergeben eines Hostnamens wird nicht unterstützt, da er zuerst konvertiert werden müsste, was zu viel Zeit in Anspruch nehmen würde. `SendUDPData()` gibt die Anzahl der erfolgreich übertragenen Bytes zurück. Dies kann weniger als die Anzahl der Bytes in `data$` sein.

Beachten Sie, dass das Argument `data$` zwar eine Zeichenkette ist, aber nicht auf Text beschränkt ist. Mit diesem Befehl können auch Binärdaten gesendet werden, da Zeichenketten von Hollywood sowohl die Zeichencodes als auch das spezielle Nullzeichen problemlos verarbeiten kann.

Ab Hollywood 8.0 sind die Argumente `ip$` und `port` nicht erforderlich, wenn das UDP-Objekt vom Typ `#UDPCIENT` erstellt wurde. In diesem Fall ist das UDP-Objekt bereits verbunden und Sie müssen `ip$` und `port` nicht übergeben. Sie werden für UDP-Objekte

vom Typ `#UDPCLIENT` ignoriert. Siehe [Abschnitt 39.5 \[CreateUDPObject\]](#), Seite 771, für Details.

Beachten Sie, dass das mit `SetNetworkTimeout()` festgelegte globale Netzwerk-Timeout von `SendUDPData()` ignoriert wird.

#### EINGABEN

<code>id</code>	Identifikator des UDP-Objekts
<code>data\$</code>	Zeichenkette, welche die zu sendenden Daten enthält
<code>ip\$</code>	optional: IP-Adresse des Empfängers; muss für UDP-Objekte vom Typ <code>#UDPSERVER</code> und <code>#UDPNONE</code> angegeben werden
<code>port</code>	optional: Portnummer des Empfängers; muss für UDP-Objekte vom Typ <code>#UDPSERVER</code> und <code>#UDPNONE</code> angegeben werden

#### RÜCKGABEWERTE

<code>count</code>	Anzahl der erfolgreich übertragenen Bytes
--------------------	---

## 39.23 SetNetworkProtocol

#### BEZEICHNUNG

`SetNetworkProtocol` – setzt des Standard-Netzwerkprotokoll (V8.0)

#### ÜBERSICHT

`SetNetworkProtocol(protocol)`

#### BESCHREIBUNG

Mit diesem Befehl kann das Internetprotokoll für Befehle wie `OpenConnection()`, `CreateServer()`, `CreateUDPObject()` und `DownloadFile()` gesetzt werden, wenn kein explizites Protokoll angefordert wurde.

Sie müssen das gewünschte Standard-Internetprotokoll im Argument `protocol` übergeben. Dies muss einer der folgenden speziellen Werte sein:

- #IPV4:** Verwendet die Internetprotokollversion 4 (IPv4). IPv4-Adressen sind auf 32 Bit beschränkt und werden unter Verwendung von vier durch drei Punkte getrennten Zahlen dargestellt, z.B. 127.0.0.1.
- #IPV6:** Verwendet die Internetprotokollversion 6 (IPv6). IPv6-Adressen verwenden 128 Bits und werden durch acht Gruppen von vier Hexadezimalziffern dargestellt, z.B. 2001:0db8:85a3:0000:0000:8a2e:0370:7334. Beachten Sie, dass **#IPV6** auf AmigaOS und kompatiblen Systemen derzeit nicht unterstützt wird.
- #IPAUTO:** Lassen Sie das Host-Betriebssystem das zu verwendende Internetprotokoll bestimmen.

Aus historischen Gründen und aus Gründen der Portabilität ist **#IPV4** das Standard-Netzwerkprotokoll, das Hollywood verwendet. Wenn Sie diese Voreinstellung ändern möchten, rufen Sie diesen Befehl auf.

**EINGABEN**

`protocol` das gewünschte neue Standard-Internetprotokoll

**39.24 SetNetworkTimeout****BEZEICHNUNG**

`SetNetworkTimeout` – stellt das globale Zeitlimit für Netzwerkbefehle ein (V5.0)

**ÜBERSICHT**

`SetNetworkTimeout(ms)`

**BESCHREIBUNG**

Mit diesem Befehl können Sie ein globales Zeitlimite für alle Befehle der Netzwerkbibliothek von Hollywood definieren. Standardmäßig ist dieses Zeitlimit auf 10000 Millisekunden (10 Sekunden) eingestellt. Dies bedeutet, dass die Netzwerkbibliothek die Verbindung automatisch beendet, wenn ein Server länger als 10 Sekunden zum Antworten benötigt. Das Definieren eines solchen Zeitlimits ist sehr wichtig, um sicherzustellen, dass Netzwerkbefehle Ihr gesamtes Skript nicht blockieren können, nur weil ein Server ausfällt. So sollte hier immer ein vernünftiges Zeitlimit eingestellt werden.

Normalerweise sollte dieser Befehl nicht verwendet werden. In seltenen Fällen kann es jedoch sinnvoll sein, das globale Zeitlimit zu ändern. Wenn Sie dies tun müssen, übergeben Sie einfach den gewünschten Wert in Millisekunden an diesen Befehl.

**EINGABEN**

`ms` neues gewünschtes globales Zeitlimit in Millisekunden (voreingestellt sind 10000 Millisekunden)

**39.25 ToHostName****BEZEICHNUNG**

`ToHostName` – konvertiert eine IP-Adresse in einen Hostname (V5.0)

**ÜBERSICHT**

`host$ = ToHostName(ip$)`

**BESCHREIBUNG**

Mit diesem Befehl kann der Hostname der angegebenen IP-Adresse ermittelt werden. Die IP-Adresse muss als Zeichenkette in der Form von vier durch Punkte getrennten Zahlen angegeben werden: "10.20.30.40". Um die IP-Adresse von einem Hostnamen abzurufen, verwenden Sie den Befehl `ToIP()`.

**EINGABEN**

`ip$` IP-Adresse

**RÜCKGABEWERTE**

`host$` der Hostname der angegebenen IP-Adresse

## 39.26 ToIP

### BEZEICHNUNG

ToIP – konvertiert einen Hostname in eine IP-Adresse (V5.0)

### ÜBERSICHT

```
ip$ = ToIP(host$[, protocol])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die IP-Adresse des angegebenen Hostnamens zu erhalten. Die IP-Adresse des Hosts wird als Zeichenkette zurückgegeben. Um den Host einer IP-Adresse zu erhalten, verwenden Sie denn Befehl `ToHostName()`.

Ab Hollywood 8.0 gibt es optional ein neues Argument `protocol`, mit dem Sie das Internetprotokoll angeben können, das für die resultierende IP-Adresse in `ip$` verwendet werden soll. Dies kann eine der folgenden speziellen Konstanten sein:

- #IPV4:** Verwendet die Internetprotokollversion 4 (IPv4). IPv4-Adressen sind auf 32 Bit beschränkt und werden unter Verwendung von vier durch drei Punkte getrennten Zahlen dargestellt, z.B. 127.0.0.1.
- #IPV6:** Verwendet die Internetprotokollversion 6 (IPv6). IPv6-Adressen verwenden 128 Bits und werden durch acht Gruppen von vier Hexadezimalziffern dargestellt, z.B. 2001:0db8:85a3:0000:0000:8a2e:0370:7334. Beachten Sie, dass **#IPV6** auf AmigaOS und kompatiblen Systemen derzeit nicht unterstützt wird.
- #IPAUTO:** Lassen Sie das Host-Betriebssystem das zu verwendende Internetprotokoll bestimmen.

Das Argument `protocol` verwendet standardmäßig den Standardprotokolltyp, der mit dem Befehl `SetNetworkProtocol()` festgelegt wird. Standardmäßig ist dies aus historischen Gründen und aus Gründen der Portabilität **#IPV4**. Siehe [Abschnitt 39.23 \[SetNetworkProtocol\]](#), [Seite 795](#), für Details.

Um einen Hostnamen mit erweiterten Funktionen zu konvertieren, werfen Sie einen Blick auf den Befehl `ResolveHostName()`. Siehe [Abschnitt 39.20 \[ResolveHostName\]](#), [Seite 792](#), für Details.

### EINGABEN

<code>host\$</code>	Hostname
<code>protocol</code>	optional: zu verwendendes Internetprotokoll (mögliche Werte siehe oben); der Standardwert ist der mit <code>SetNetworkProtocol()</code> festgelegte Protokolltyp. (V8.0)

### RÜCKGABEWERTE

<code>ip\$</code>	IP-Adresse des angegebenen Hosts
-------------------	----------------------------------

## 39.27 UploadFile

### BEZEICHNUNG

UploadFile – lädt die Datei auf einen Server hoch (V5.0)

## ÜBERSICHT

```
s$, len = UploadFile(url$, options[, func, userdata])
```

## BESCHREIBUNG

Mit diesem Befehl können Sie eine Datei auf einen Netzwerkservers hochladen. Standardmäßig unterstützt `UploadFile()` die Protokolle FTP und HTTP, aber Plugins können zusätzliche Protokolle nutzen. Vor Hollywood 6.0 unterstützte `UploadFile()` nur das hochladen auf FTP-Server. Aber ab Version 6.0 wird das hochladen auf HTTP-Server ebenfalls unterstützt. Da das hochladen auf HTTP- und FTP-Server zwei völlig unterschiedliche Mechanismen zum Senden der Daten an den empfangenden Server verwenden, sind die Verfahren zum hochladen einer Datei unter Verwendung von FTP und HTTP ziemlich unterschiedlich.

Wenn Sie eine Datei mit `UploadFile()` auf einen FTP-Server hochladen möchten, müssen Sie als erstes Argument die URL übergeben, auf der die Datei auf dem FTP-Server gespeichert werden soll. Diese URL muss mit dem Präfix `ftp://` beginnen und sie muss eine vollständige Pfadangabe (d.h. den Zieldateinamen) enthalten. Die URL darf keine Escape-Zeichen enthalten. Die ESC-Zeichenumwandlung wird durch `UploadFile()` durchgeführt werden, so stellen Sie sicher, dass Sie nur unverschlüsselte URLs übergeben. D.h. URL wie z.B. `"ftp://ftp.site.net/my%20file.zip"` funktionieren nicht. Sie müssen eine URL ohne ESC-Zeichen angeben, womit die korrekte Version wäre: `"ftp://ftp.site.net/my file.zip"`. Wenn Sie eine bereits mit ESC-Zeichen codierte URL übergeben möchten, müssen Sie den Tag `Encoded` auf `True` setzen (siehe unten). In diesem Fall wird `UploadFile()` keine weitere ESC-Zeichenumwandlung auf Ihre URL verwenden.

Die Datei, die auf den FTP-Server hochgeladen werden soll, muss entweder im Tabellenelement `File` oder `Data` angegeben werden (siehe unten).

Wenn Sie eine Datei auf einen HTTP-Server hochladen möchten, müssen Sie die URL eines PHP- oder CGI-Skripts übergeben, das das Hochladen abwickelt. Beachten Sie, dass `UploadFile()` nur das Hochladen mit der HTTP-POST-Methode unterstützt. Das Hochladen Upload über HTTP-PUT wird nicht unterstützt. Zusätzlich zur URL eines PHP- oder CGI-Skripts müssen Sie auch die Parameter angeben, die an dieses Skript übergeben werden sollen. Diese Parameter werden im Tabellenelement `FormData` übergeben (siehe unten). Die Dateien, die hochgeladen werden sollen, müssen auch als Parameter im Tabellenelement `FormData` übergeben werden.

Sie können auch einen Benutzernamen und ein Passwort angeben, die für die Anmeldung am HTTP- oder FTP-Server verwendet werden sollen. Bei Verwendung vom FTP-Hochladen wird `"anonymous"` als Standardbenutzername und `"anonymous@anonymous.org"` als Standardkennwort verwendet. Wenn Sie ein anderes Benutzerkonto verwenden möchten, müssen Sie das Benutzername-/Passwortpaar in der URL übergeben. Hier ist eine Beispiel-URL für den Nutzernamen `"johndoe"` und dem Passwort `"topsecret"`: `ftp://johndoe:topsecret@ftp.test.net/pub/files.lha` für das FTP-Hochladen oder `http://johndoe:topsecret@www.test.com/private/upload.php` für das HTTP-Hochladen.

Die an diesen Befehl übergebene URL kann auch eine Portnummer enthalten. Wenn Sie eine Portnummer angeben möchten, müssen Sie sie nach dem Hostnamen setzen und mit einem Doppelpunkt trennen. Hier ist ein Beispiel für die Verwendung von Port

1234: ftp://ftp.test.net:1234/test/image.jpg. Wenn kein Port angegeben wird, verwendet `UploadFile()` Port 21.

Das zweite Argument `options` ist eine Tabelle, die mehrere Optionen erkennt. Bei der Verwendung vom FTP-Hochladen können die hochzuladenden Daten entweder mit dem Tabellenfeld `File` oder mit `Data` angegeben werden. Bei der Verwendung vom HTTP-Hochladen müssen die hochzuladenden Daten mit dem Feld `FormData` angegeben werden. Hier ist eine Übersicht aller Felder, die derzeit erkannt werden:

**File:** Für das FTP-Hochladen muss die Datei in diesem Tag angegeben werden. Wenn Sie Daten aus einer Zeichenkettenquelle hochladen möchten, müssen Sie den Tag `Data` verwenden. Bei jedem Aufruf von `UploadFile()` für das FTP-Hochladen müssen Sie entweder `File` oder `Data` angeben. Sie dürfen diesen Tag nicht fürs HTTP-Hochladen benutzen. Verwenden Sie dafür `FormData` (siehe unten).

**Data:** Für das FTP-Hochladen können Sie bei diesem Tag eine Zeichenkette angeben, die hochgeladen wird. Die Zeichenkette ist nicht nur auf Text beschränkt, sondern kann auch binäre Daten enthalten. Wenn Sie Daten aus einer Datei hochladen möchten, müssen Sie stattdessen den Tag `File` verwenden (siehe oben). Sie müssen bei jedem Aufruf von `UploadFile()` fürs FTP-Hochladen entweder `Data` oder `File` angeben. Sie dürfen diesen Tag nicht fürs HTTP-Hochladen benutzen. Verwenden Sie `FormData` für das HTTP-Hochladen (siehe unten).

**TransferMode:**

Mit diesem Tag können Sie festlegen, ob `UploadFile()` die Datei im ASCII- oder im Binärmodus übertragen soll. Für den ASCII-Modus geben Sie hier `#FTPASCII` an. Verwenden Sie für den Binärmodus `#FTPBINARY`. Der voreingestellte Übertragungsmodus ist `#FTPBINARY`. Dieser Tag wird nur beim FTP-Hochladen unterstützt.

**SilentFail:**

Wenn Sie diesen Tag auf `True` setzen, wird `UploadFile()` niemals einen Fehler auslösen, sondern einfach still abbrechen und eine Fehlermeldung im ersten Rückgabewert `s$` sowie `-1` im zweiten Rückgabewert `len` zurückgeben, um anzuzeigen, dass ein Fehler aufgetreten ist. Wenn es auf `False` gesetzt ist, wird `UploadFile()` einen Systemfehler für alle auftretenden Fehler melden. Standardwert ist `False`.

**FormData:**

Dieser Tag wird für das HTTP-Hochladen benötigt. Er erlaubt Ihnen, eine Tabelle von Parametern anzugeben, die an das PHP- oder CGI-Skript weitergegeben werden soll, welches das Hochladen verarbeitet. Die Datei oder die Daten, die hochgeladen werden sollen, müssen ebenfalls in dieser Tabelle übergeben werden. Sie müssen eine Tabelle mit Tabellen an dieses Argument übergeben. Jede Untertabelle beschreibt einen einzelnen Skriptparameter. `UploadFile()` verwendet diese Tabelle mit Tabellen, um Multipart/Form-Data zu erstellen, die dann an den HTTP-Server über den POST-Anforderungstyp gesendet wird. Folgende Tabellenelemente können in jeder Untertabelle verwendet werden:

**Name:** Dies muss auf den Namen des Parameters eingestellt sein. Dieses Tabellenelement muss immer vorhanden sein.

**Data:** Die Daten, die als Parameterwert übergeben werden sollen. Diese muss mit einer Zeichenkette geschehen, die auch binäre Daten enthalten kann, so dass es möglich ist, die in diesem Tabellenelement hochzuladenden Dateidaten zu übergeben. Alternativ können Sie auch das Tabellenelement **File** verwenden, um Daten aus einer Dateiquelle hochzuladen. Beachten Sie, dass Sie für jede Untertabelle den Tag **Data** oder **File** angeben müssen. Wenn Sie **Data** zum Hochladen von Dateien anstelle von Formulardaten verwenden möchten, müssen Sie auch die Tags **MIMEType** und **FileAlias** angeben (siehe unten).

**File:** Wenn Sie den Tag **Data** nicht setzen, müssen Sie dieses Tabellenelement auf einen Dateinamen setzen, dessen Inhalt als Teil des übergebenen Parameters in **Name** hochgeladen werden soll. Alternativ können Sie auch das Tabellenelement **Data** verwenden, um Daten aus einer Zeichenkettenquelle hochzuladen. Beachten Sie, dass Sie für jede Untertabelle entweder den Tag **File** oder **Data** angeben müssen.

**MIMEType:**

Mit diesem Tag können Sie den MIME-Typ der hochzuladenden Daten einstellen. Dies sollte angegeben werden, wann immer die Parameteruntertabelle eine Datei hochladen möchte. Sie darf nicht angegeben werden, wenn die Parameteruntertabelle lediglich einfache Formulardaten (d. h. Klartext) an den Server übergibt. Dieser Tag wird standardmäßig auf "application/octet-stream" eingestellt, wenn der Tag **File** gesetzt wurde. Wenn der Tag **Data** gesetzt wurde, gibt es keine Vorgabe für den Tag **MIMEType**, da der Tag **Data** auch einfache Formulardaten enthalten könnte. Wenn Sie also den Tag **Data** zum Hochladen von Dateidaten verwenden möchten, müssen Sie **MIMEType** immer explizit auf den MIME-Typ der Daten setzen.

**FileAlias:**

Wenn die Parameteruntertabelle eine Datei hochladen möchte, kann mit diesem Tag ein Name für diese Datei festgelegt werden. Dies ist in der Regel nur erforderlich, wenn die Dateidaten mit dem Tag **Data** angegeben werden. Bei Verwendung des Tags **File** benutzt **UploadFile()** einfach den Namen dieser Datei und Sie müssen **FileAlias** nicht verwenden, obwohl es benutzt werden kann, um den in **File** angegebenen Dateinamen zu überschreiben.

Es ist absolut erlaubt, mehr als eine Datei auf einmal hochzuladen. Sie können beliebig viele Untertabellen mit dem Tabellenelement **FormData** ver-



wenden. Die resultierende HTML-Seite, die vom PHP- oder CGI-Skript nach dem Upload erzeugt wird, wird als Zeichenkette von `UploadFile()` zurückgegeben. (V6.0)

**CustomHeaders:**

Mit diesem Tag können Sie eine Reihe von benutzerdefinierten Headern angeben, die an den HTTP-Server gesendet werden sollen. Dies kann für einige Feinabstimmungen bei einigen Servern nützlich sein. Beachten Sie, dass die einzelnen Header-Elemente durch einen Wagenrücklauf und einen Zeilenvorschub terminiert werden müssen. Dieser Tag wird nur unterstützt, wenn das HTTP-Protokoll verwendet wird. (V6.0)

**Encoded:** Setzen Sie diesen Tag auf `True`, wenn die URL, die Sie an diesen Befehl übergeben haben, bereits korrekt mit ESC-Zeichen versehen wurde. Wenn dieser Tag auf `True` gesetzt ist, wird `UploadFile()` keine Zeichen codieren. Stattdessen wird erwartet, dass Sie eine URL übergeben, die bereits korrekt mit ESC-Zeichen versehen wurde, somit sie direkt für Serveranforderungen ohne zusätzliche ESC-Zeichenumwandlungen verwendet werden kann. (V6.1)

**Protocol:**

Mit diesem Tag kann das Internetprotokoll angegeben werden, das beim Öffnen der Verbindung verwendet werden soll. Dies kann eine der folgenden speziellen Konstanten sein:

**#IPV4:** Verwendet die Internetprotokollversion 4 (IPv4).

**#IPV6:** Verwendet die Internetprotokollversion 6 (IPv6). Beachten Sie, dass **#IPV6** auf AmigaOS und kompatiblen Systemen derzeit nicht unterstützt wird.

**#IPAUTO:** Lassen Sie das Host-Betriebssystem das zu verwendende Internetprotokoll bestimmen.

Dieser Tag verwendet standardmäßig den Standardprotokolltyp, der mit `SetNetworkProtocol()` festgelegt wird. Standardmäßig ist dies aus historischen Gründen und aus Gründen der Portabilität **#IPV4**. Siehe [Abschnitt 39.23 \[SetNetworkProtocol\]](#), [Seite 795](#), für Details. (V8.0)

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Netzwerkadapter angeben, die zum Herstellen der angegebenen Verbindung benutzt werden sollten. Dies muss auf eine Zeichenkette gesetzt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V8.0)

**SSL:** Setzen Sie diesen Tag auf `True`, um eine Verbindung über TLS/SSL-Verschlüsselung anzufordern. Beachten Sie, dass die Einstellung dieses Tags bei Verwendung des integrierten Netzwerkadapters von Hollywood keine Auswirkungen hat, da dieser keine TLS/SSL-Verbindungen unterstützt. Möglicherweise gibt es jedoch einen Netzwerkadapter, der von einem Plugin bereitgestellt wird, das TLS/SSL unterstützt. Wenn Sie diesen Tag auf `True` setzen, leitet Hollywood Ihren Wunsch nach einer TLS/SSL-Verbindung an

den Netzwerkadapter weiter, der vom Plugin bereitgestellt wird. Beachten Sie jedoch, dass Sie diesen Tag normalerweise nicht setzen müssen, wenn das Schema der URL bereits eine SSL-Verbindung mit einem Präfix wie "https://" oder "ftps://" angibt. (V8.0)

**Async:** Wenn dies auf **True** gesetzt ist, arbeitet **UploadFile()** im asynchronen Modus. Das bedeutet, dass der Befehl sofort zurückkehrt und falls Ihr Skript während des Vorgangs etwas anderes tun muss, Ihnen ein asynchroner Operationshandler übergibt. Sie können dann diesen asynchronen Operationshandler verwenden, um den Vorgang abzuschließen, indem Sie wiederholt **ContinueAsyncOperation()** aufrufen, bis **True** zurückgegeben wird. Dies ist sehr nützlich, z.B. für die Anzeige eines Fortschrittsbalken oder ähnlichem. Indem Sie **UploadFile()** in den asynchronen Modus versetzen, ist es Ihrem Skript leicht möglich, während der Verarbeitung des Vorgangs etwas anderes zu tun. Siehe **Abschnitt 19.4 [ContinueAsyncOperation]**, **Seite 234**, für Details. Voreingestellt ist **False**. (V9.0)

**Verbose:** Dieser Tag kann auf **True** gesetzt werden, um detaillierte Protokollinformationen über die Verbindung und die Protokollinteraktion mit dem Server anzufordern. Dies wird derzeit nur von Hollywood-Plugins verwendet. Wenn Sie also den internen Netzwerkadapter von Hollywood verwenden, hat das Setzen diesen Tags auf **True** keine Auswirkung. Plugins können jedoch erweiterte Verbindungsinformationen bereitstellen, wenn dieser Tag auf **True** gesetzt wurde. Die Voreinstellung ist **False**. (V9.0)

**FileAdapter:**

Dieser Tag wird nur verwendet, wenn auch der Tag **File** gesetzt ist. In diesem Fall können Sie mit **FileAdapter** einen oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Zeichenkette setzen, die den Namen eines oder mehrerer Adapter enthält. Voreingestellt ist **default**. Siehe **Abschnitt 7.9 [Lade- und Adaptermodule]**, **Seite 96**, für Details. (V10.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Datei- und Netzwerkadapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe **Abschnitt 7.10 [Benutzer-Tags]**, **Seite 100**, für Details. (V10.0)

Der optionale Parameter **func** kann verwendet werden, um eine Callback-Funktion zu übergeben, die von Zeit zu Zeit von **UploadFile()** aufgerufen wird, so dass Sie beispielsweise eine Fortschrittsanzeige aktualisieren können. Die Callback-Funktion, wird mit einem einzigen Argument aufgerufen: Eine Tabelle, die mehr Informationen enthält. Hier ist eine Übersicht der Tabellenfelder, die initialisiert werden, bevor **UploadFile()** Ihre Callback-Funktion ausführt:

**Action:** **#UPLOADFILE\_STATUS**

**Count:** Enthält die Anzahl der bereits hochgeladenen Bytes.

**Total:** Enthält die Größe der hochzuladenden Datei.

**UserData:**

Enthält den Wert, den Sie im Argument `userdata` übergeben haben.

Die Callback-Funktion vom Typ `#UPLOADFILE_STATUS` sollte normalerweise `False` zurückgeben. Wenn sie `True` zurückgibt, wird der Uploadvorgang abgebrochen.

Beachten Sie, dass beim Verwenden von `UploadFile()` für das HTTP-Hochladen das PHP- oder CGI-Skript, das für das Hochladen verwendet wird, auch eine resultierende HTML-Seite generiert, die normalerweise vom Browser nach Abschluss des Hochladens angezeigt wird. Diese HTML-Seite wird von `UploadFile()` als Zeichenkette `s$` zurückgegeben. Der zweite Rückgabewert `len` beschreibt die Länge dieser HTML-Seite in Bytes. Da `UploadFile()` diese resultierende HTML-Seite vom Server herunterladen muss, wird Ihre Callback-Funktion aufgerufen, während `UploadFile()` die Antwort des Servers erhält, damit Sie den Fortschritt überwachen können. Die Tabelle, die an Ihre Callback-Funktion übergeben wird, wird in diesem Fall wie folgt initialisiert:

**Action:** `#UPLOADFILE_RESPONSE`

**Count:** Enthält die Anzahl der bereits heruntergeladenen Bytes.

**Total:** Enthält die Größe der herunterzuladenden Datei.

**UserData:**

Enthält den Wert, den Sie im Argument `userdata` übergeben haben.

Die Callback-Funktion vom Typ `#UPLOADFILE_RESPONSE` sollte normalerweise `False` zurückgeben. Wenn es `True` zurückgibt, wird der Downloadvorgang abgebrochen.

Schließlich gibt es ein viertes optionales Argument mit dem Namen `userdata`. Der Wert, den Sie hier angeben, wird bei jedem Aufruf an Ihre Callback-Funktion übergeben. Dies ist sinnvoll, wenn Sie vermeiden wollen, mit globalen Variablen zu arbeiten. Mit dem Argument `userdata` können Sie einfach Daten an Ihre Callback-Funktion übergeben. Sie können einen Wert eines beliebigen Typs angeben. Zahlen, Zeichenketten, Tabellen und sogar Funktionen können als Benutzerdaten weitergegeben werden.

**EINGABEN**

<b>url\$</b>	FTP-URL für die Zieldatei oder URL eines PHP- oder CGI-Skripts auf einem HTTP-Server
<b>options</b>	eine Tabelle mit der zu hochladenden Datei/Daten sowie weitere Optionen
<b>func</b>	optional: eine Callback-Funktion, die von Zeit zu Zeit aufgerufen wird
<b>userdata</b>	optional: Benutzerdaten, welche an die Callback-Funktion übergeben werden

**RÜCKGABEWERTE**

<b>s\$</b>	optional: die resultierende HTML-Seite, die durch das PHP- oder CGI-Skript erzeugt wird; Beachten Sie, dass diese nur beim HTTP-Hochladen übergeben wird
<b>len</b>	optional: die Länge der resultierenden HTML-Seite, die durch das PHP- oder CGI-Skript erzeugt wird; Beachten Sie, dass diese nur bei HTTP-Hochladen übergeben wird

**BEISPIEL**

```
UploadFile("ftp://ftp.test.net/pub/image.jpg", {File = "image.jpg"})
```

Der obige Code lädt die Datei "image.jpg" auf den in Argument 1 angegebenen FTP-Server hoch.

```
UploadFile("http://www.test.com/upload.php", {FormData = {
    {Name = "uploadername", Data = "John Doe"},
    {Name = "uploaderemail", Data = "john@doe.com"},
    {Name = "description", Data = "My profile picture"},
    {Name = "imagefile", File = "image.jpg", MIMETYPE = "image/jpeg"}}})
```

Der obige Code lädt die Datei "image.jpg" auf einen HTTP-Server hoch. Zusätzlich werden die Parameter `uploadername`, `uploaderemail` und `description` an das PHP-Skript weitergegeben.

```
@REQUIRE "hurl"
```

```
...
```

```
UploadFile("ftp://ftp.test.net/pub/image.jpg",
    {File = "image.jpg", SSL = True, Adapter = "hurl"})
```

Der obige Code lädt eine Datei mit explizitem FTPS hoch. Da Hollywood SSL/TLS standardmäßig nicht unterstützt, verwendet dieser Code das hURL-Plugin für den Vorgang, da hURL SSL/TLS unterstützt. hURL wird aktiviert, indem `hurl` im Tag `Adapter` übergeben wird.

```
@REQUIRE "hurl"
```

```
...
```

```
UploadFile("ftps://ftp.test.net/pub/image.jpg",
    {File = "image.jpg", Adapter = "hurl"})
```

Der obige Code lädt eine Datei mit implizitem FTPS hoch. Da Hollywood SSL/TLS standardmäßig nicht unterstützt, verwendet dieser Code das hURL-Plugin für den Vorgang, da hURL SSL/TLS unterstützt. hURL wird aktiviert, indem `hurl` im Tag `Adapter` übergeben wird.

## 40 Objektbibliothek

### 40.1 Übersicht

Diese Bibliothek bietet abstrakte Befehle, um mit Hollywood-Objekten umzugehen. Hollywood-Objekte sind alle Objekte, die von Hollywood erstellt und verwaltet werden, z.B. Pinsel, Anims, Hintergrundbilder, Videos etc. Diese Objekte werden automatisch geschlossen und aus dem Speicher gelöscht, wenn Hollywood beendet wird. Es empfiehlt sich jedoch, dass Sie nicht mehr benötigte Objekte löschen, um unnötigen Speicherverbrauch zu vermeiden.

Hollywood-Objekte werden entweder über numerische Identifikatoren oder über Handler angesprochen, die von allen Objekterstellungsbefehlen zurückgegeben werden, wenn Sie den speziellen Wert `Nil` als numerischer Identifikator übergeben. Siehe [Abschnitt 7.8 \[Automatische ID-Zuweisung\]](#), Seite 96, für Details. Wenn Sie numerische Identifikatoren verwenden und einen bereits existierenden numerischen Identifikator an eine Objekterstellungsfunktion übergeben, wird das vorhandene Objekt automatisch gelöscht.

Objektbibliotheksbefehle wie `GetAttribute()`, `GetObjectData()` oder `SetObjectData()` erfordern, dass Sie einen Objekttyp zusammen mit dem Identifikator vom Objekt übergeben. Folgende Objekttypkonstanten werden derzeit erkannt:

- #ANIM** Ein Animations-Objekt, das mit `@ANIM` oder `LoadAnim()` erstellt wurde. Siehe [Abschnitt 17.2 \[ANIM\]](#), Seite 185, für Details.
- #ANIMSTREAM** Ein Animations-Objekt, welches mit `BeginAnimStream()` erstellt wurde. Siehe [Abschnitt 17.3 \[BeginAnimStream\]](#), Seite 189, für Details.
- #ASYNCDRAW** Ein asynchrones Zeichnungsobjekt, das mit `PlayAnim()`, den Verschiebungsobjektbefehlen oder durch Übergangseffekten erstellt wurde.
- #ASYNCOBJ** Ein asynchroner Operationshandler, der von Befehlen wie `CopyFile()` oder `DownloadFile()` erstellt wird.
- #BGPIC** Ein Hintergrundbild-Objekt, welches mit `@BGPIC`, `LoadBGPic()` und dergleichen erstellt wurde. Siehe [Abschnitt 30.2 \[BGPIC\]](#), Seite 619, für Details.
- #BRUSH** Ein Pinsel-Objekt, das z. B. mit `@BRUSH`, `LoadBrush()` erstellt wurde. Siehe [Abschnitt 43.6 \[BRUSH\]](#), Seite 908, für Details.
- #CLIENT** Ein Client-Objekt, welches mit `OpenConnection()` erstellt oder an Ihren Callback vom Ereignis-Handler `OnConnect` übergeben wurde.
- #CLIPREGION** Ein Clip-Regionen-Objekt, das mit `CreateClipRegion()` erstellt wurde. Siehe [Abschnitt 28.8 \[CreateClipRegion\]](#), Seite 574, für Details.
- #CONSOLEWINDOW** Ein Konsolenfensterobjekt, welches mit `CreateConsoleWindow()` erstellt wurde. Siehe [Abschnitt 33.10 \[CreateConsoleWindow\]](#), Seite 658, für Details.

**#DIRECTORY**

Ein mit dem Befehl `OpenDirectory()` erstelltes Verzeichnis-Objekt. Siehe [Abschnitt 20.47 \[OpenDirectory\]](#), Seite 289, für Details.

**#DISPLAY** Ein Display-Objekt, welches mit `@DISPLAY` oder `CreateDisplay()` erstellt wurde. Siehe [Abschnitt 24.8 \[DISPLAY\]](#), Seite 370, für Details.

**#FILE** Ein Datei-Objekt, das mit `@FILE` oder `OpenFile()` erstellt wurde. Siehe [Abschnitt 20.18 \[FILE\]](#), Seite 262, für Details.

**#FONT** Ein mit `@FONT` oder `OpenFont()` erstelltes Schriftarten-Objekt. Siehe [Abschnitt 52.11 \[FONT\]](#), Seite 1147, für Details.

**#ICON** Ein durch `@ICON`, `LoadIcon()` und dergleichen erstelltes Piktogrammobjekt. Siehe [Abschnitt 42.6 \[ICON\]](#), Seite 888, für Details.

**#INTERVAL**

Ein Intervall-Objekt, welches mit `SetInterval()` erstellt wurde. Siehe [Abschnitt 26.26 \[SetInterval\]](#), Seite 515, für Details.

**#LAYER** Ein Hollywood-Ebenen-Objekt, das mit einem Grafikbefehl wie z.B. `DisplayBrush()` erstellt wurde.

**#MEMORY** Ein mit `AllocMem()` oder dergleichen Befehl erstelltes Speicherblock-Objekt. Siehe [Abschnitt 48.1 \[AllocMem\]](#), Seite 1061, für Details.

**#MENU** Ein Menu-Objekt, welches mit `@MENU` oder `CreateMenu()` erstellt wurde. Siehe [Abschnitt 37.8 \[MENU\]](#), Seite 754, für Details.

**#MOVELIST**

Ein Bewegungsliste-Objekt, das mit `AddMove()` erstellt wurde. Siehe [Abschnitt 25.2 \[AddMove\]](#), Seite 407, für Details.

**#MUSIC** Ein mit `@MUSIC`, `OpenMusic()` und dergleichen erstelltes Music-Objekt. Siehe [Abschnitt 47.28 \[MUSIC\]](#), Seite 1043, für Details.

**#PALETTE:**

Ein Palettenobjekt, welches mit `@PALETTE`, `LoadPalette()` oder `CreatePalette()` erstellt wurde. Siehe [Abschnitt 41.17 \[PALETTE\]](#), Seite 854, für Details.

**#POINTER** Ein Mauszeiger-Objekt, welches mit `CreatePointer()` erstellt wurde. Siehe [Abschnitt 36.1 \[CreatePointer\]](#), Seite 745, für Details.

**#SAMPLE** Ein Sample-Objekt, das mit `@SAMPLE`, `LoadSample()` und dergleichen erstellt wurde. Siehe [Abschnitt 47.39 \[SAMPLE\]](#), Seite 1051, für Details.

**#SERIAL** Ein serielles Verbindungsobjekt, das von `OpenSerialPort` erstellt wurde. Siehe [Abschnitt 46.11 \[OpenSerialPort\]](#), Seite 1010, für Details.

**#SERVER** Ein mit `CreateServer()` erstelltes Server-Objekt. Siehe [Abschnitt 39.4 \[CreateServer\]](#), Seite 770, für Details.

**#SPRITE** Ein Sprite-Objekt, das mit `@SPRITE` oder `LoadSprite()` erstellt wurde. Siehe [Abschnitt 49.13 \[SPRITE\]](#), Seite 1085, für Details.

**#TEXTOBJECT**

Ein Text-Objekt, das mit `CreateTextObject()` erstellt wurde. Siehe [Abschnitt 52.8 \[CreateTextObject\]](#), Seite 1143, für Details.

**#TIMEOUT** Ein mit `SetTimeout()` erstelltes Timeout-Objekt. Siehe [Abschnitt 26.27 \[SetTimeout\]](#), Seite 517, für Details.

**#TIMER** Ein Zeitmess-Objekt, welches mit `StartTimer()` erstellt wurde. Siehe [Abschnitt 21.17 \[StartTimer\]](#), Seite 332, für Details.

**#UDPOBJECT**

Ein UDP-Objekt, das mit `CreateUDPObject()` erstellt wurde. Siehe [Abschnitt 39.5 \[CreateUDPObject\]](#), Seite 771, für Details.

**#VECTORPATH**

Ein mit dem Befehl `StartPath()` erstelltes Vektorpfad-Objekt. Siehe [Abschnitt 53.36 \[StartPath\]](#), Seite 1206, für Details.

**#VIDEO** Ein Video-Objekt, welches mit `@VIDEO` oder `OpenVideo()` erstellt wurde. Siehe [Abschnitt 55.17 \[VIDEO\]](#), Seite 1247, für Details.

## 40.2 ClearObjectData

**BEZEICHNUNG**

`ClearObjectData` – löscht private Daten in einem Objekt (V5.0)

**ÜBERSICHT**

`ClearObjectData(type, id[, key$])`

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um private Daten von einem Objekt zu entfernen. Sie müssen den Typ und die ID des Objekts übergeben, dessen private Daten Sie löschen möchten. Wenn das optionale Argument `key$` angegeben wird, entfernt `ClearObjectData()` nur Daten, welche mit diesem Schlüssel zugeordnet sind. Wenn `key$` weggelassen wird, wird `ClearObjectData()` die privaten Daten aller Schlüssel in diesem Objekt entfernen.

Siehe [Abschnitt 40.1 \[Objekttypen\]](#), Seite 805, für eine Liste aller Objekttypen.

**EINGABEN**

<code>type</code>	Typ des Objekts
<code>id</code>	ID des Objekts
<code>key\$</code>	optional: Schlüssel, die von dem Objekt entfernt werden sollen; wenn dieses Argument weggelassen wird, werden alle Schlüssel von diesem Objekt entfernt.

## 40.3 CopyObjectData

**BEZEICHNUNG**

`CopyObjectData` – kopiert private Daten zwischen Objekten (V5.0)

**ÜBERSICHT**

```
CopyObjectData(srctype, srcid, dsttype, dstid[, overwrite])
```

**BESCHREIBUNG**

Dieser Befehl kopiert alle privaten Daten, die dem in `srctype` und `srcid` angegebenen Objekt zugeordnet sind, in das von `dsttype` und `dstid` angegebene Objekt. Das optionale Argument `overwrite` gibt an, ob `CopyObjectData()` die Schlüssel im Zielobjekt überschreiben soll, falls sie denselben Namen tragen wie der Schlüssel im Quellobjekt. Standardmäßig ist dies auf `True` gesetzt, womit vorhandene Schlüssel im Zielobjekt durch die Schlüssel des Quellobjekts bei gleichem Namen ersetzt werden. Wenn Sie dieses Verhalten nicht möchten, setzen Sie `overwrite` auf `False`.

Siehe [Abschnitt 40.1 \[Objekttypen\]](#), [Seite 805](#), für eine Liste aller Objekttypen.

**EINGABEN**

<code>srctype</code>	Type vom verwendeten Quellobjekt
<code>srcid</code>	ID des verwendeten Quellobjekts
<code>dsttype</code>	Type vom verwendeten Zielobjekt
<code>dstid</code>	ID des verwendeten Zielobjekts
<code>overwrite</code>	optional: gibt an, ob oder ob nicht vorhandene Schlüssel im Zielobjekt überschrieben werden soll (Standardwert ist <code>True</code> )

**BEISPIEL**

```
SetObjectData(#BRUSH, 1, "name", "mybrush")
CopyObjectData(#BRUSH, 1, #BRUSH, 2)
DebugPrint(GetObjectData(#BRUSH, 2, "name"))
```

Der obige Code kopiert alle Daten von Pinsel 1 nach Pinsel 2. Der Aufruf von `DebugPrint()` wird dann "mybrush" ausgeben.

## 40.4 GetAttribute

**BEZEICHNUNG**

`GetAttribute` – holt Informationen über ein Objekt

**ÜBERSICHT**

```
info = GetAttribute(obj, id, attr[, param, param2])
```

**BESCHREIBUNG**

Dieser Befehl kann benutzt werden, um Informationen über ein Hollywood-Objekt zu erhalten.

Dieser Befehl kann verwendet werden, um Eigenschaften von allen verschiedenen Hollywood-Objekten abzurufen. Sie können beispielsweise die Bemaßungen eines Pinsels oder die Länge der Sounddatei abfragen. Nachfolgend finden Sie eine vollständige Liste der Objekttypen und deren Attribute.



Folgende Attribute können für **#ANIM** abgefragt werden:

**#ATTRWIDTH:**

Liefert die Breite der Animation.

**#ATTRHEIGHT:**

Liefert die Höhe der Animation.

**#ATTRTRANSPARENTCOLOR:**

Liefert die transparente Farbe der Animation oder **#NOTRANSARENCY**.

**#ATTRNUMFRAMES:**

Liefert die Anzahl der Einzelbilder in dieser Animation. (V2.0)

**#ATTRHASMASK:**

Gibt **True** zurück, wenn die Animation eine Maske hat. (V2.0)

**#ATTRHASALPHA:**

Gibt **True** zurück, wenn die Animation einen Alphakanal hat. (V4.5)

**#ATTRFRAMEDELAY:**

Gibt die Zeit in Millisekunden an, die der Animationsplayer nach einem Einzelbild warten soll. Sie müssen für diese Abfrage auch noch die Einzelbildnummer im Argument **param** übergeben. Einzelbilder werden ab 1 gezählt. Wenn Sie das Argument **param** weglassen, wird das erste Einzelbild verwendet. Bitte beachten Sie, dass nicht alle Animationsformate Verzögerungen der Einzelbilder unterstützen und dass die Informationen nur für Einzelbilder verfügbar sind, die bereits geladen sind; Wenn Sie ein statistisches Einzelbild einer plattenbasierten Animation abfragen, könnte es sein, dass Sie eine Null als Rückgabewert erhalten, da dieses Einzelbild noch nicht geladen wurde. (V4.5)

**#ATTRCOUNT:**

Gibt zurück, wie viele Animationen sich derzeit im Arbeitsspeicher befinden. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)

**#ATTRLOADER:**

Gibt den Namen des Lademoduls an das verwendet wurde um diese Animation zu laden. (V6.0)

**#ATTRDEPTH:**

Gibt die Farbtiefe des im Argument **param** angegebenen Einzelbildes zurück. Einzelbilder werden ab 1 gezählt. Wird das Argument **param** weggelassen, wird das erste Einzelbild verwendet. Wenn die Farbtiefe kleiner oder gleich 8 ist, handelt es sich um eine Palettenanimation. (V9.0)

**#ATTRPALETTE:**

Gibt die Palette des im Argument **param** angegebenen Einzelbildes zurück. Einzelbilder werden ab 1 gezählt. Wird das Argument **param** weggelassen, wird das erste Einzelbild verwendet. Die Palette des Einzelbildes wird als Tabelle zurückgegeben und enthält so viele Elemente, wie Stifte in der Palette vorhanden sind. Die einzelnen Stifte werden als **RGB-Farben** zurückgegeben. Wenn das Einzelbild keine Palette hat, wird eine leere Tabelle zurückgegeben. (V9.0)

**#ATTRTRANSPARENTPEN:**

Gibt den Stift zurück, der in der Palette des durch **param** angegebenen Einzelbildes transparent ist. Einzelbilder werden ab 1 gezählt. Wird das Argument **param** weggelassen, wird das erste Einzelbild verwendet. Wenn kein transparenter Stift vorhanden ist oder das Einzelbild keine Palette hat, wird **#NOPEN** zurückgegeben. (V9.0)

**#ATTRTYPE:**

Gibt den Typ der Animation zurück. Dies wird entweder auf **#ANIMTYPE\_RASTER** für eine Raster-Animation oder **#ANIMTYPE\_VECTOR** für eine **Vektor-Animation** gesetzt. (V9.0)

**#ATTRFORMAT:**

Gibt den Namen des Animationsformats als Zeichenkette zurück. (V10.0)

Folgende Attribute können für **#ANIMSTREAM** abgefragt werden:

**#ATTRCOUNT:**

Gibt zurück, wie viele Animations-Objekt derzeit im Speicher sind. Nützlich, um die von Ihrem Skript verwendeten Ressourcen zu verfolgen. (V5.0)

Folgende Attribute können für **#ASYNCDRAW** abgefragt werden:

**#ATTRTYPE:**

Gibt den Typ des asynchronen Zeichnungsobjekts an; wird **#ADF\_FX**, **#ADF\_MOVEOBJECT** oder **#ADF\_ANIM** sein. (V4.5)

**#ATTRNUMFRAMES:**

Gibt die Anzahl der Einzelbilder dieses asynchronen Zeichnungsobjekts an; Bitte beachten Sie, dass wenn Sie diesen Wert als Basis für eine Schleife über **AsyncDrawFrame()** verwenden, müssen Sie eine zusätzliche Schleife hinzufügen, da der letzte Aufruf von **AsyncDrawFrame()** nicht als Einzelbild zählt; Siehe **Abschnitt 19.1 [AsyncDrawFrame]**, **Seite 231**, für Details. (V4.5)

**#ATTRCURFRAME:**

Gibt das Einzelbild zurück, welches derzeit auf dem Bildschirm in diesem asynchron Zeichnungsobjekt angezeigt wird. (V4.5)

**#ATTRCOUNT:**

Gibt zurück wie viele asynchrone Zeichnungsobjekte sich derzeit im Arbeitsspeicher befinden. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)

Folgende Attribute können für **#ASYNCOBJ** abgefragt werden:

**#ATTRCOUNT:**

Gibt zurück, wie viele Handler für asynchrone Operationen sich derzeit im Speicher befinden. Nützlich, um den Speicherverbrauch zu verfolgen. (V9.0)

Folgende Attribute können für **#BGPic** abgefragt werden:

**#ATTRWIDTH:**

Liefert die Breite des BGPic.

**#ATTRHEIGHT:**

Liefert die Höhe des BGPic.

**#ATTRTRANSPARENTCOLOR:**

Gibt die transparente Farbe des BGPic oder **#NOTTRANSPARENCY** zurück.

**#ATTRLAYERS:**

Gibt die Anzahl der Ebenen zurück, welche dem BGPic zugewiesen sind.  
(V1.5)

**#ATTRHASMASK:**

Gibt **True** zurück, wenn BGPic eine Maske enthält. (V2.0)

**#ATTRHASALPHA:**

Gibt **True** zurück, wenn BGPic einen Alphakanal enthält. (V4.5)

**#ATTRCLIPREGION:**

Gibt die ID der Clip-Region zurück, die derzeit aktiv im BGPic ist oder -1, wenn es keine aktive Clip-Region gibt. (V4.5)

**#ATTRCOUNT:**

Gibt zurück, wie viele BGPics sich derzeit im Arbeitsspeicher befinden. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)

**#ATTRTYPE:**

Gibt den Typ des BGPic zurück. Dies ist entweder **#IMAGETYPE\_RASTER** für eine Raster-BGPic oder **#IMAGETYPE\_VECTOR** für ein **Vektor-BGPic**. (V5.0)

**#ATTRLOADER:**

Gibt den Namen des Lademoduls zurück, das verwendet wurde, um das BGPic zu laden. (V6.0)

**#ATTRSPRITES:**

Gibt die Anzahl der derzeit auf diesem BGPic sichtbaren Sprites zurück.  
(V7.0)

**#ATTRDEPTH:**

Gibt die Farbtiefe des BGPic zurück. Wenn dieser Wert kleiner oder gleich 8 ist, ist das BGPic ein Paletten-BGPic. (V9.0)

**#ATTRPALETTE:**

Gibt die Palette des BGPic als Tabelle zurück. Die Tabelle enthält so viele Elemente, wie Stifte in der Palette vorhanden sind. Die einzelnen Stifte werden als **RGB-Farben** zurückgegeben. Wenn das BGPic keine Palette hat, wird eine leere Tabelle zurückgegeben. (V9.0)

**#ATTRTRANSPARENTPEN:**

Gibt den Stift zurück, der in der Palette des BGPic transparent ist. Wenn kein transparenter Stift vorhanden ist oder das BGPic keine Palette hat, wird **#NOPEN** zurückgegeben. (V9.0)

**#ATTRCYCLE:**

Wenn das BGPic eine Palette mit definierten Farbwechselbereichen hat, gibt dieses Attribut eine Tabelle mit allen definierten Farbwechselbereichen zurück. In diesem Fall enthält die von **#ATTRCYCLE** zurückgegebene Tabelle eine Reihe von Untertabellen, in denen die folgenden Felder initialisiert sind:

- Low:** Der Stiftindex, der den Anfang des Farbwechselbereichs markiert.
- High:** Der Stiftindex, der das Ende des Farbwechselbereichs markiert.
- Rate:** Die gewünschte Geschwindigkeit des Farbwechseleffekts. Ein Wert von 16384 bedeutet 60 Bilder pro Sekunde. Alle anderen Geschwindigkeiten werden linear von dieser Basis aus skaliert, z.B. ein Wert von 8192 bedeutet 30 Bilder pro Sekunde und so weiter.
- Reverse:** Wenn dieser Tag auf **True** gesetzt ist, sollten die Farben in umgekehrter Reihenfolge durchlaufen werden.
- Active:** Wenn dieser Tag auf **True** gesetzt ist, wird dieser Farbwechselbereich als aktiv markiert.

(V9.0)

**#ATTRFORMAT:**

Gibt den Namen des Bildformats als Zeichenkette zurück. (V10.0)

Folgende Attribute können für **#BRUSH** abgefragt werden:

**#ATTRWIDTH:**

Liefert die Breite des Pinsels.

**#ATTRHEIGHT:**

Liefert die Höhe des Pinsels.

**#ATTRTRANSPARENTCOLOR:**

Gibt die transparente Farbe des Pinsels oder **#NOTTRANSPARENCY** zurück.

**#ATTRHASMASK:**

Gibt **True** zurück, wenn der Pinsel eine Maske enthält. (V2.0)

**#ATTRHASALPHA:**

Gibt **True** zurück, wenn der Pinsel einen Alphakanal enthält. (V2.0)

**#ATTRCOUNT:**

Gibt zurück, wie viele Pinsel sich derzeit im Arbeitsspeicher befinden. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)

**#ATTRTYPE:**

Gibt den Typ des Pinsels zurück. Dies wird entweder **#IMAGETYPE\_RASTER** für einen Rasterpinsel oder **#IMAGETYPE\_VECTOR** für einen **Vectorpinsel** sein. (V5.0)

**#ATTRHARDWARE:**

Gibt **True** zurück, wenn der angegebene Pinsel ein Hardware-Pinsel ist. Siehe [Abschnitt 43.37 \[Hardware-Pinsel\]](#), [Seite 940](#), für Details. (V5.0)

**#ATTRDISPLAY:**

Gibt die ID des zu diesem Pinsel gehörenden Displays zurück, wenn dieser Pinsel ein displayabhängiger Hardware-Pinsel ist. Ansonsten wird -1 zurückgegeben. Siehe [Abschnitt 43.37 \[Hardware-Pinsel\]](#), [Seite 940](#), für Details. (V6.0)

**#ATTRLOADER:**

Gibt den Namen des Lademoduls zurück, das verwendet wurde um den Pinsel zu laden. (V6.0)

**#ATTRDEPTH:**

Gibt die Farbtiefe des Pinsels zurück. Wenn dieser Wert kleiner oder gleich 8 ist, handelt es sich um einen Palettenpinsel. (V9.0)

**#ATTRPALETTE:**

Gibt die Palette des Pinsels als Tabelle zurück. Die Tabelle enthält so viele Elemente, wie Stifte in der Palette vorhanden sind. Die einzelnen Stifte werden als **RGB-Farben** zurückgegeben. Wenn der Pinsel keine Palette hat, wird eine leere Tabelle zurückgegeben. (V9.0)

**#ATTRTRANSPARENTPEN:**

Gibt den Stift zurück, der in der Palette des Pinsels transparent ist. Wenn kein transparenter Stift vorhanden ist oder der Pinsel keine Palette hat, wird **#NOPEN** zurückgegeben. (V9.0)

**#ATTRCYCLE:**

Wenn der Pinsel eine Palette mit definierten Farbwechselbereichen hat, gibt dieses Attribut eine Tabelle mit allen definierten Farbwechselbereichen zurück. In diesem Fall enthält die von **#ATTRCYCLE** zurückgegebene Tabelle eine Reihe von Untertabellen, in denen die folgenden Felder initialisiert sind:

**Low:** Der Stiftindex, der den Anfang des Farbwechselbereichs markiert.

**High:** Der Stiftindex, der das Ende des Farbwechselbereichs markiert.

**Rate:** Die gewünschte Geschwindigkeit des Farbwechseleffekts. Ein Wert von 16384 bedeutet 60 Bilder pro Sekunde. Alle anderen Geschwindigkeiten werden linear von dieser Basis aus skaliert, z.B. ein Wert von 8192 bedeutet 30 Bilder pro Sekunde und so weiter.

**Reverse:** Wenn dieser Tag auf **True** gesetzt ist, sollten die Farben in umgekehrter Reihenfolge durchlaufen werden.

**Active:** Wenn dieser Tag auf **True** gesetzt ist, wird dieser Farbwechselbereich als aktiv markiert.

(V9.0)

**#ATTRFORMAT:**

Gibt den Namen des Bildformats als Zeichenkette zurück. (V10.0)

Folgende Attribute können für **#CLIENT** abgefragt werden:

**#ATTRCOUNT:**

Gibt zurück, wie viele Netzwerkclientobjekte derzeit im Speicher sind. Nützlich, um die von Ihrem Skript verwendeten Ressourcen zu verfolgen. (V5.0)

**#ATTRADAPTER:**

Gibt den Namen des Adapters zurück, der zum Öffnen dieser Verbindung verwendet wurde oder **inbuilt**, wenn die Verbindung mit dem integrierten Netzwerkadapters von Hollywood geöffnet wurde. (V8.0)

Folgende Attribute können für **#CLIPREGION** abgefragt werden:

**#ATTRXPOS:**

Gibt die x-Position der Clip-Region zurück. (V3.0)

**#ATTRYPOS:**

Gibt die y-Position der Clip-Region zurück. (V3.0)

**#ATTRWIDTH:**

Gibt die Breite der Clip-Region zurück. (V3.0)

**#ATTRHEIGHT:**

Gibt die Höhe der Clip-Region zurück. (V3.0)

**#ATTRCOUNT:**

Gibt zurück, wie viele Clip-Regionen sich derzeit im Arbeitsspeicher befinden. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)

Folgende Attribute können für **#CONSOLEWINDOW** abgefragt werden:

**#ATTRCOUNT:**

Gibt zurück, wie viele Konsolenfenster sich derzeit im Speicher befinden. Nützlich für die Überwachung des Speicherverbrauchs. (V10.0)

Folgende Attribute können für **#DIRECTORY** abgefragt werden:

**#ATTRCOUNT:**

Gibt zurück, wie viele Verzeichnisse derzeit für Zugriffe offen sind. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)

**#ATTRADAPTER:**

Gibt den Namen des Adaptermoduls zurück, der verwendet wurde, um das Verzeichnis zu öffnen oder **inbuilt**, wenn das Verzeichnis mit einem eingebauten Handler von Hollywood geöffnet wurde. (V6.0)

**#ATTRFORMAT:**

Gibt den Namen des Verzeichnisformats als Zeichenkette zurück. Dies wird normalerweise nur festgelegt, wenn das Verzeichnis von einem Plugin verwaltet wird, da Verzeichnisse standardmäßig kein "Format" haben, Plugins jedoch möglicherweise Adapter implementieren, die ZIP-Dateien usw. auf Verzeichnisse abbilden. In diesem Fall können Sie mit **#ATTRFORMAT** das Format der Verzeichnisquelle herauszufinden, z.B. "zip archive". (V10.0)

Folgende Attribute können für **#DISPLAY** abgefragt werden:

**#ATTRWIDTH:**

Gibt die Breite des Displays zurück; dieser Wert beinhaltet nicht die Breite des Fensterrahmens.

**#ATTRHEIGHT:**

Gibt die Höhe des Displays zurück; dieser Wert beinhaltet nicht die Höhe des Fensterrahmens.

**#ATTRMAXWIDTH:**

Gibt die maximal mögliche Breite für dieses Display zurück (das heißt aktuelle Desktop-Auflösung minus Fensterrahmenbreite).

**#ATTRMAXHEIGHT:**

Gibt die maximal mögliche Höhe für dieses Display zurück (das heißt aktuelle Desktop-Auflösung minus Fensterrahmenhöhe).

**#ATTRBGPIC:**

Gibt das BGPic zurück, welches diesem Display zugewiesen wurde. (V1.5)

**#ATTRLAYERS:**

Gibt die Anzahl der Ebenen zurück, die zu diesem BGPic gehören. (V1.5)

**#ATTRCURSORX:**

Gibt die aktuelle x-Position des Cursors zurück. (V2.5)

**#ATTRCURSORY:**

Gibt die aktuelle y-Position des Cursors zurück. (V2.5)

**#ATTRXPOS:**

Gibt die x-Position des Displays auf dem Bildschirm zurück. (V3.0)

**#ATTRYPOS:**

Gibt die y-Position des Displays auf dem Bildschirm zurück. (V3.0)

**#ATTRBORDERLEFT:**

Gibt die Breite des linken Bildschirmrandes oder 0 zurück, wenn der Bildschirm randlos ist. (V3.0)

**#ATTRBORDERRIGHT:**

Gibt die Breite des rechten Bildschirmrandes oder 0 zurück, wenn der Bildschirm randlos ist. (V3.0)

**#ATTRBORDERTOP:**

Gibt die Höhe des oberen Bildschirmrandes oder 0 zurück, wenn der Bildschirm randlos ist. (V3.0)

**#ATTRBORDERBOTTOM:**

Gibt die Höhe des unteren Bildschirmrandes oder 0 zurück, wenn der Bildschirm randlos ist. (V3.0)

**#ATTRHOSTWIDTH:**

Gibt die Breite des Bildschirms zurück, auf dem das Display geöffnet ist (in der Regel der Desktop-Bildschirm). (V3.0)

**#ATTRHOSTHEIGHT:**

Gibt die Höhe des Bildschirms zurück, auf dem das Display geöffnet ist (in der Regel der Desktop-Bildschirm). (V3.0)

**#ATTRFONTASCENDER, #ATTRFONTDESCENDER, #ATTRFONTNAME, #ATTRFONTSIZE, #ATTRFONTSCALABLE, #ATTRFONTAA, #ATTRFONTDEPTH, #ATTRFONTPALETTE, #ATTRFONTTRANSPARENTPEN, #ATTRFONTEENGINE, #ATTRFONTTYPE, #ATTRFONTCHARMAP, #ATTRFONTLOADER, #ATTRFONTHEIGHT, #ATTRFONTFORMAT:**

Liefert Informationen über die Schriftart, welche derzeit ausgewählt ist und angezeigt wird; bitte lesen Sie unten in **#TEXTOBJECT** für Infos über diese Attribute. (V3.1)

**#ATTRPOINTER:**

Gibt den Identifikator des Mauszeigers zurück, der derzeit diesem Display zugeordnet ist. (V4.5)

**#ATTRSTATE:**

Gibt den aktuellen Status von diesem Display zurück; kann entweder **#DISPSTATE\_OPEN** (wenn das Display geöffnet ist), **#DISPSTATE\_CLOSED** (Display ist derzeit geschlossen) oder **#DISPSTATE\_MINIMIZED** (Display ist zur Zeit minimiert) sein. (V4.5)

**#ATTRACTIVE:**

Gibt zurück, ob dieses Display gerade aktiv ist; es kann nur ein Display zu einem Zeitpunkt aktiv sein. (V4.5)

**#ATTRMODE:**

Gibt den aktuellen Display-Modus zurück; dies kann entweder **#DISPMODE\_WINDOWED** für Fenster- oder **#DISPMODE\_FULLSCREEN** für Vollbildmodus sein. Beachten Sie, dass dies eine globale Einstellung ist. Sie können bei der Abfrage von **#ATTRMODE** 0 in **id** für das Display weitergeben. (V4.5)

**#ATTRSCALEMODE:**

Gibt den Skalierungsmodus zurück, der derzeit aktiv ist. Kann entweder **#SCALEMODE\_NONE**, **#SCALEMODE\_AUTO**, oder **#SCALEMODE\_LAYER** sein. (V4.5)

**#ATTRSCALEWIDTH:**

Gibt die aktuell eingestellte Skalierungsbreite für dieses Display zurück. Wenn keine Skalierung aktiv ist, wird dieses Attribut die gleiche Breite wie **#ATTRWIDTH** haben. (V4.5)

**#ATTRSCALEHEIGHT:**

Gibt die aktuell eingestellte Skalierungshöhe für dieses Display zurück. Wenn keine Skalierung aktiv ist, wird dieses Attribut die gleiche Höhe wie **#ATTRHEIGHT** zurückgeben. (V4.5)



**#ATTRBORDERLESS:**

Gibt zurück, ob das Display randlos ist oder nicht. (V4.5)

**#ATTRSIZEABLE:**

Gibt zurück, ob das Display veränderbar ist oder nicht. (V4.5)

**#ATTRFIXED:**

Gibt zurück, ob das Display fixiert ist oder nicht. (V4.5)

**#ATTRNOHIDE:**

Gibt zurück, ob das Display durch den Benutzer als Piktogramm dargestellt wird oder nicht. (V4.5)

**#ATTRNOMODESWITCH:**

Gibt **True** zurück, wenn die Modusumschaltung über die Tastenkombination **CMD+RETURN** (Amiga/macOS) oder **ALT+RETURN** (Windows) für dieses Display deaktiviert wurde. (V4.5)

**#ATTRTITLE:**

Gibt die Titelzeichenkette vom Display zurück. (V4.5)

**#ATTRMARGINLEFT, #ATTRMARGINRIGHT:**

Gibt die aktuellen Randeinstellungen für dieses Display unter Verwendung von **SetMargins()** zurück. (V4.5)

**#ATTRDOUBLEBUFFER:**

Gibt **True** zurück, wenn das angegebene Display doppelt gepuffert ist, **False** wenn nicht. (V4.5)

**#ATTROUTPUTDEVICE:**

Dieses Attribut gibt drei Werte zurück, die Informationen über die aktuellen Ausgabegeräte enthält. Der erste Rückgabewert kann entweder **#DISPLAY**, **#BGPIC**, **#BRUSH**, **#ANIM**, oder **#DOUBLEBUFFER** sein. Der zweite Rückgabewert gibt die entsprechende ID an mit dem Typ, der im ersten Rückgabewert angegeben wurde. Der dritte Rückgabewert wird schließlich nur von Typen wie **#BRUSH**, **#ANIM** und **#BGPIC** verwendet (für **#BGPIC** wird es nur verwendet, wenn **SelectBGPic()** mit den Modi **#SELMODE\_NORMAL** oder **#SELMODE\_COMBO** aufgerufen wurde). In diesem Fall gibt es die Grafiken Pinsel/Animation/BGPic an, die derzeit ausgewählt sind: Dies kann entweder **#MASK**, **#ALPHACHANNEL** oder **#BRUSH** sein. Beachten Sie, dass der dritte Rückgabewert auch dann auf **#BRUSH** gesetzt wird, wenn **#ANIM/#BGPIC** durch den ersten Rückgabewert zurückgegeben wird. Wenn **#BRUSH** als dritter Rückgabewert zurückgegeben wird, bedeutet dies, dass der Farbkanal des/der Pinsels/Animation derzeit ausgewählt ist. Wenn **SelectBGPic()** in **#SELMODE\_LAYERS** ist, wird der erste und der dritte Rückgabewert **#BGPIC** sein. Wenn **SelectBGPic()** in einem anderen Modus ist, wird der erste Rückgabewert **#BGPIC**, aber der dritte Rückgabewert wird entweder **#BRUSH**, **#MASK**, oder **#ALPHACHANNEL** sein. (V4.5)

**#ATTRCOUNT:**

Gibt zurück, wie viele Displays derzeit im Arbeitsspeicher sind. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)

**#ATTRMASKMODE:**

Gibt den aktuellen Maskenmodus zurück. Dies ist eine globale Einstellung, welche an kein bestimmtes Display gebunden ist. Siehe [Abschnitt 43.73 \[SetMaskMode\]](#), [Seite 976](#), für Details. (V4.5)

**#ATTRALPHAINTENSITY:**

Gibt die aktuelle Alpha-Intensität zurück. Dies ist eine globale Einstellung, welche an kein bestimmtes Display gebunden ist. Siehe [Abschnitt 43.67 \[SetAlphaIntensity\]](#), [Seite 972](#), für Details. (V4.5)

**#ATTRLAYERSON:**

Gibt **True** zurück, wenn im angegebenen Display Ebenen aktiviert sind, **False** wenn dies nicht der Fall ist. (V5.0)

**#ATTRORIENTATION:**

Gibt die aktuelle Ausrichtung des mobilen Geräts zurück, welches mit Hollywood läuft. Der Rückgabewert wird eines der folgenden Konstanten sein:

**#ORIENTATION\_PORTRAIT**  
**#ORIENTATION\_LANDSCAPE**  
**#ORIENTATION\_PORTRAITREV**  
**#ORIENTATION\_LANDSCAPEREV**

Bitte beachten Sie, dass dieser Tag nur in der mobilen Version von Hollywood unterstützt wird. In der Desktop-Version wird immer **#ORIENTATION\_NONE** zurückgegeben. (V5.0)

**#ATTRPUBSCREEN:**

Gibt den öffentlichen Bildschirm vom Display zurück, der zur Zeit geöffnet ist. Dies wird nur auf AmigaOS kompatiblen Betriebssysteme unterstützt. (V5.2)

**#ATTRDENSITY:**

Gibt die logische Dichte des Displays zurück. Der Rückgabewert wird eine der folgenden vordefinierten Dichtewerte haben:

**#DENSITY\_LOW**  
**#DENSITY\_MEDIUM**  
**#DENSITY\_HIGH**

Bitte beachten Sie, dass dieser Tag nur in der mobilen Version von Hollywood unterstützt wird. In der Desktop-Version wird immer **#DENSITY\_NONE** zurückgegeben. (V5.3)

**#ATTRXDPI:**

Gibt die genauen physikalischen Pixel pro Zoll auf der X-Achse des Bildschirms zurück. Bitte beachten Sie, dass dieser Tag auf älteren Android-Geräten einen falschen Wert zurückgeben kann. (V5.3)

**#ATTRYDPI:**

Gibt die genauen physikalischen Pixel pro Zoll auf der Y-Achse des Bildschirms zurück. Bitte beachten Sie, dass dieser Tag auf älteren Android-Geräten einen falschen Wert zurückgeben kann. (V5.3)

**#ATTRMENU:**

Gibt die ID der Menüleiste zurück, die diesem Display zugewiesen ist oder -1 wenn keine zugewiesen wurde. (V6.0)

**#ATTRMONITOR:**

Gibt die Monitornummer zurück, auf dem das Display geöffnet wurde. Monitore werden ab 1 gezählt (das ist der primäre Monitor). (V6.0)

**#ATTRHOSTMONITORS:**

Gibt die Gesamtanzahl der Monitore zurück, die derzeit an dem System zur Verfügung stehen. Sie können ihre Abmessungen und erweiterten Desktop-Positionen mit `GetMonitors()` abfragen. (V6.0)

**#ATTRXSERVER:**

Gibt den Namen des X-Servers zurück, der an diesem Bildschirm angeschlossen ist. Dies wird nur unter Linux unterstützt. (V6.0)

**#ATTRADAPTER:**

Gibt den Namen des derzeit verwendeten Display-Adaptermoduls zurück. Wenn Hollywoods Display-Adaptermodul verwendet wird, wird `inbuilt` zurückgegeben. (V6.0)

**#ATTRMAXIMIZED:**

Gibt `True` zurück, wenn das Display aktuell maximiert ist, sonst `False`. (V7.0)

**#ATTRRAWWIDTH:**

Gibt die rohe physikalische Breite des Displays zurück, unabhängig vom derzeit aktivem Skalierungssystem. Benutzen Sie dieses Attribut mit Vorsicht, da es mit den Skalierungssystemen in Konflikt geraten kann, weil sie immer vorgeben, dass Hollywood in einer anderen Auflösung läuft. (V7.0)

**#ATTRRAWHEIGHT:**

Gibt die rohe physikalische Breite des Displays zurück, unabhängig vom derzeit aktivem Skalierungssystem. Benutzen Sie dieses Attribut mit Vorsicht, da es mit den Skalierungssystemen in Konflikt geraten kann, weil sie immer vorgeben, dass Hollywood in einer anderen Auflösung läuft. (V7.0)

**#ATTRHOSTTITLEBARHEIGHT:**

Gibt die Höhe der Titelleiste des Host-Bildschirms zurück. Beachten Sie, dass nicht alle Systeme eine Titelleiste besitzen, vor allem Windows hat keine. In diesem Fall wird 0 zurückgegeben. (V7.0)

**#ATTRHOSTTASKBAR:**

Gibt Informationen über die Taskleiste unter Windows zurück. Dieses Feld gibt 5 Werte zurück: Die ersten beiden Werte beschreiben die x- und y-Position der Taskleiste auf dem Host-Bildschirm, Rückgabewerte drei und vier enthalten die Dimensionen der Taskleiste. Der fünfte und letzte Rückgabewert ist ein Boolescher Wert, ob die Taskleiste aktuell sichtbar ist oder nicht. Dieses Feld wird derzeit nur unter Windows unterstützt. (V7.0)

**#ATTRSPRITES:**

Gibt die Anzahl der aktuell auf diesem Display aktiven Sprites zurück. (V7.0)

**#ATTRHOSTSCALEX:**

Gibt den Skalierungskoeffizienten auf der x-Achse des Monitors zurück. Normalerweise ist dies 1, aber für hochauflösende Displays (z. B. Retina Macs oder 4K Monitore unter Windows) kann dies größer als 1 sein. Beachten Sie, dass dieser Wert unter Windows immer 1 ist, es sei denn, Sie aktivieren die DPI-Awareness explizit, indem Sie den Tag `DPIAware` in der Präprozessor-Anweisung `@OPTIONS` auf `True` setzen. (V7.0)

**#ATTRHOSTSCALEY:**

Gibt den Skalierungskoeffizienten auf der y-Achse des Monitors zurück. Normalerweise ist dies 1, aber für hochauflösende Displays (z. B. Retina Macs oder 4K Monitore unter Windows) kann dies größer als 1 sein. Beachten Sie, dass dieser Wert unter Windows immer 1 ist, es sei denn, Sie aktivieren die DPI-Awareness explizit, indem Sie den Tag `DPIAware` in der Präprozessor-Anweisung `@OPTIONS` auf `True` setzen. (V7.0)

**#ATTRHOSTSCALE:**

Gibt den globalen Skalierungskoeffizienten des Monitors des Displays zurück. Normalerweise ist dies 1, für hochauflösende Displays (z. B. Retina Macs oder 4K-Monitore unter Windows) kann dies jedoch größer als 1 sein. Beachten Sie, dass dieser Wert unter Windows immer 1 ist, es sei denn, Sie aktivieren die DPI-Awareness explizit, indem Sie den Tag `DPIAware` in der Präprozessor-Anweisung `@OPTIONS` auf `True` setzen. (V8.0)

**#ATTRIMMERSIVEMODE:**

Gibt den immersiven Modus zurück, der vom Display verwendet wird. Der Rückgabewert ist eine der folgenden speziellen Konstanten:

```
#IMMERSIVE_NONE  
#IMMERSIVE_NORMAL  
#IMMERSIVE_LEANBACK  
#IMMERSIVE_STICKY
```

Siehe [Abschnitt 24.8 \[DISPLAY\]](#), [Seite 370](#), für Details. (V9.0)

**#ATTRSYSTEMBARS:**

Gibt `True` zurück, wenn die Systemleisten aktuell sichtbar sind, ansonsten `False`. Dies wird derzeit nur auf Android unterstützt. Beachten Sie, dass die Systemleisten immer nur dann unsichtbar sein können, wenn sich ein Display im immersiven Modus befindet. Siehe [Abschnitt 24.8 \[DISPLAY\]](#), [Seite 370](#), für Details. (V9.0)

**#ATTRDEPTH:**

Gibt die Farbtiefe des Displays zurück. Wenn dieser Wert kleiner oder gleich 8 ist, handelt es sich um ein Display im Palettenmodus. Siehe [Abschnitt 24.16 \[Palettenmodus-Displays\]](#), [Seite 393](#), für Details. (V9.0)

**#ATTRPALETTE:**

Gibt die Palette des Displays als Tabelle zurück. Die Tabelle enthält so viele Elemente, wie Stifte in der Palette vorhanden sind. Die einzelnen Stifte werden als `RGB-Farben` zurückgegeben. Wenn das Display keine Palette hat, wird eine leere Tabelle zurückgegeben. (V9.0)

**#ATTRTRANSPARENTPEN:**

Gibt den Stift zurück, der in der Palette des Pinsels transparent ist. Wenn kein transparenter Stift vorhanden ist oder der Pinsel keine Palette hat, wird #NOPEN zurückgegeben. (V9.0)

**#ATTRPALETTEMODE:**

Gibt den aktuellen Palettenmodus zurück, der mit `SetPaletteMode()` eingestellt wurde. Siehe [Abschnitt 41.31 \[SetPaletteMode\]](#), [Seite 868](#), für Details. (V9.0)

**#ATTRDITHERMODE:**

Gibt den aktuellen Dither-Modus zurück, der mit `SetDitherMode()` eingestellt wurde. Siehe [Abschnitt 41.26 \[SetDitherMode\]](#), [Seite 863](#), für Details. (V9.0)

**#ATTRPEN:**

Gibt den aktuellen Zeichnungsstift zurück, der mit `SetDrawPen()` gesetzt wurde. Siehe [Abschnitt 41.27 \[SetDrawPen\]](#), [Seite 864](#), für Details. (V9.0)

**#ATTRSHADOWPEN:**

Gibt den aktuellen Schattenstift zurück, der mit `SetShadowPen()` gesetzt wurde. Siehe [Abschnitt 41.35 \[SetShadowPen\]](#), [Seite 872](#), für Details. (V9.0)

**#ATTRBORDERPEN:**

Gibt den aktuellen Rahmenstift zurück, der mit `SetBorderPen()` eingestellt wurde. Siehe [Abschnitt 41.22 \[SetBorderPen\]](#), [Seite 860](#), für Details. (V9.0)

**#ATTRBULLETPEN:**

Gibt den aktuellen Aufzählungszeichnungsstift zurück, der mit `SetBulletPen()` festgelegt wurde. Siehe [Abschnitt 41.23 \[SetBulletPen\]](#), [Seite 860](#), für Details. (V9.0)

**#ATTRSCALESWITCH:**

Gibt zurück, ob dieses Display sich selbst auf die aktuelle Auflösung des Monitors skaliert, wenn Sie das Tastaturkürzel `CMD+RETURN` (`LALT+RETURN` unter Windows) drücken oder `#DISPMODE_MODESWITCH` mit `ChangeDisplayMode()` verwenden. (V9.0)

**#ATTRINTERPOLATE:**

Wenn ein Skalierungssystem aktiv ist, gibt dies zurück, ob interpolierte Skalierung verwendet wird oder nicht. (V9.0)

Folgende Attribute können für `#EVENTHANDLER` abgefragt werden:

**#ATTRFUNCTION:**

Gibt die Callback-Funktion des angegebenen Ereignis-Handlers zurück. Beachten Sie, dass Sie den Namen des Ereignis-Handlers als Zeichenfolge übergeben müssen, so wie Sie es beim Befehl `InstallEventHandler()` tun würden. (V4.5)

**#ATTRUSERDATA:**

Gibt die Benutzerdaten des angegebenen Ereignis-Handlers zurück. Beachten Sie, dass Sie den Namen des Ereignis-Handlers als Zeichenfolge übergeben müssen, so wie Sie es beim Befehl `InstallEventHandler()` tun würden. (V4.5)

Folgende Attribute können für **#FILE** abgefragt werden:

**#ATTRMODE:**

Gibt den Modus zurück, in dem diese Datei geöffnet wurde. Kann entweder **#MODE\_READ**, **#MODE\_WRITE** oder **#MODE\_READWRITE** sein. (V4.5)

**#ATTRCOUNT:**

Gibt an, wie viele Dateien zurzeit geöffnet sind. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)

**#ATTRADAPTER:**

Gibt den Namen des verwendeten Adaptermoduls zurück, der diese Datei geöffnet hat oder **inbuilt**, wenn diese Datei mit Hollywoods eingebautem Datei-Handler geöffnet wurde. (V6.0)

**#ATTRENCODING:**

Gibt den Zeichensatz zurück, welcher für die Datei mit `OpenFile()` oder `SetFileEncoding()` eingestellt wurde. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für eine Liste der verfügbaren Zeichensätze. (V9.0)

**#ATTRFORMAT:**

Gibt das Dateiformat als Zeichenkette zurück. Dies wird normalerweise nur festgelegt, wenn die Datei von einem Plugin verarbeitet wird, da Hollywoods Standard-Dateiverwaltung keine Formaterkennung bietet. Plugins implementieren jedoch möglicherweise Adapter, die es Ihnen ermöglichen, Dateien aus ZIP-Archiven usw. zu öffnen. In diesem Fall können Sie mit **#ATTRFORMAT** das Format der Dateiquelle herausfinden, z.B. "zip archive". (V10.0)

Folgende Attribute können für **#FONT** abgefragt werden:

**#ATTRFONTASCENDER**, **#ATTRFONTDESCENDER**, **#ATTRFONTNAME**, **#ATTRFONTSIZE**,  
**#ATTRFONTSCALABLE**, **#ATTRFONTAA**, **#ATTRFONTDEPTH**, **#ATTRFONTPALETTE**,  
**#ATTRFONTTRANSPARENTPEN**, **#ATTRFONTEENGINE**, **#ATTRFONTTYPE**, **#ATTRFONTCHARMAP**:  
 Für mehr Informationen über die Bedeutung dieser Attribute sehen Sie bitte unter **#TEXTOBJECT** nach. (V4.5)

**#ATTRCOUNT:**

Gibt an, wie viele Schriftarten derzeit im Arbeitsspeicher vorhanden sind. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)

Folgende Attribute können für **#ICON** abgefragt werden:

**#ATTRNUMENTRIES:**

Gibt die Anzahl der Bilder im Piktogramm zurück. (V8.0)

**#ATTRSTANDARD:**

Gibt den Index des Standardbilds im Piktogramm oder 0 zurück, falls das Piktogramm kein Standardbild enthält. (V8.0)

**#ATTRWIDTH:**

Gibt die Breite des Bildes an dem Index zurück, der im optionalen Argument **param** angegeben ist. Indizes beginnen bei 1. (V8.0)

**#ATTRHEIGHT:**

Gibt die Höhe des Bildes an dem Index zurück, der im optionalen Argument **param** angegeben ist. Indizes beginnen bei 1. (V8.0)

**#ATTRNUMFRAMES:**

Gibt die Anzahl der Einzelbilder des Bildes an dem Index zurück, der im optionalen Argument **param** angegeben ist. Dies kann entweder 1 oder 2 sein, abhängig davon, ob das Bild nur einen normalen Zustand oder einen normalen und einen ausgewählten Zustand hat. Indizes beginnen bei 1. (V8.0)

**#ATTRCOUNT:**

Gibt zurück, wie viele Piktogramme sich gerade im Speicher befinden. Nützlich zum Nachverfolgen des Speicherverbrauchs. (V8.0)

**#ATTRDEPTH:**

Gibt die Farbtiefe des Bildes an dem im Argument **param** angegebenen Index zurück. Indizes beginnen bei 1. Wenn **param2** auf **True** gesetzt ist, wird das ausgewählte Bild abgefragt, bei **False** (ist Voreingestellt) wird das normale Bild abgefragt. Wenn die Farbtiefe kleiner oder gleich 8 ist, ist das Bild ein Palettenbild. (V9.0)

**#ATTRPALETTE:**

Gibt die Palette des Bildes an dem im Argument **param** angegebenen Index zurück. Indizes beginnen bei 1. Wenn **param2** auf **True** gesetzt ist, wird das ausgewählte Bild abgefragt, bei **False** (ist Voreingestellt) wird das normale Bild abgefragt. Die Palette des Bildes wird als Tabelle zurückgegeben und enthält so viele Elemente, wie Stifte in der Palette vorhanden sind. Die einzelnen Stifte werden als **RGB-Farben** zurückgegeben. Wenn das Bild keine Palette hat, wird eine leere Tabelle zurückgegeben. (V9.0)

**#ATTRTRANSPARENTPEN:**

Gibt den Stift zurück, der in der Palette des Bildes an dem im Argument **param** angegebenen Index transparent ist. Indizes beginnen bei 1. Wenn **param2** auf **True** gesetzt ist, wird das ausgewählte Bild abgefragt, bei **False** (ebenfalls die Voreinstellung) wird das normale Bild abgefragt. Wenn kein transparenter Stift vorhanden ist oder das Bild keine Palette hat, wird **#NOPEN** zurückgegeben. (V9.0)

**#ATTRFORMAT:**

Gibt den Namen des Piktogramm-Format als Zeichenkette zurück. (V10.0)

Folgende Attribute können für **#INTERVAL** abgefragt werden:

**#ATTRDURATION:**

Liefert die Häufigkeit dieses Intervallobjekts in Millisekunden. (V4.5)

**#ATTRFUNCTION:**

Gibt die Callback-Funktion des angegebenen Intervallobjekts zurück. (V4.5)

**#ATTRUSERDATA:**

Gibt die zugehörigen Benutzerdaten mit diesem Intervallobjekt zurück. (V4.5)

Folgende Attribute können für **#LAYER** abgefragt werden:

**#ATTRTYPE:**

Gibt den Typ der Ebene zurück (z.B. **#PRINT**). (V1.5)

**#ATTRXPOS:**

Liefert die x-Position der Ebene auf dem Display. (V1.5)

**#ATTRYPOS:**

Liefert die y-Position der Ebene auf dem Display. (V1.5)

**#ATTRWIDTH:**

Gibt die Breite der Ebene zurück. (V1.5)

**#ATTRHEIGHT:**

Gibt die Höhe der Ebene zurück. (V1.5)

**#ATTRVISIBLE:**

Gibt **True** zurück, wenn die Ebene derzeit sichtbar ist, **False** wenn sie ausgeblendet ist. (V1.5)

**#ATTRLAYERID:**

Gibt die ID der angegebenen Ebene zurück; Das ist nur sinnvoll, wenn Sie einen Ebenennamen anstelle einer ID als Quelle angeben haben. (V2.0)

**#ATTRNUMFRAMES:**

Gibt die Anzahl der Einzelbilder dieser Ebene zurück; funktioniert nur bei der Verwendung mit Ebenen des Typs **#ANIM** (V2.0) oder **#VIDEO** (V6.0).

**#ATTRCURFRAME:**

Liefert das Einzelbild, welches momentan angezeigt wird; funktioniert nur mit Ebenen des Typs **#ANIM**. Beachten Sie, dass im Gegensatz zu den meisten anderen Befehlen hier Einzelbilder ab 0 gezählt werden, so dass Sie für das erste Einzelbild 0 erhalten, nicht 1. (V2.0)

**#ATTRTEXT, #ATTRFONTASCENDER, #ATTRFONTDESCENDER, #ATTRFONTNAME, #ATTRFONTSIZE, #ATTRFONTSCALABLE, #ATTRFONTAA, #ATTRFONTDEPTH, #ATTRFONTPALETTE, #ATTRFONTTRANSPARENTPEN, #ATTRFONTEENGINE, #ATTRFONTTYPE, #ATTRFONTCHARMAP, #ATTRFONTLOADER, #ATTRFONTHEIGHT, #ATTRFONTFORMAT:**

Diese Attribute können nur mit Ebenen des Typs **#PRINT** und **#TEXTOUT** verwendet werden; um mehr darüber zu erfahren lesen Sie den Punkt **#TEXTOBJECT** weiter unten nach. (V4.0)



**#ATTRFRAMEDELAY:**

Gibt die Zeit in Millisekunden zurück, die der Animationsplayer nach Anzeige des aktuellen Einzelbildes warten soll; dies funktioniert nur mit dem Ebenen des Typs **#ANIM**. (V4.5)

**#ATTRCOUNT:**

Gibt an, wie viele Ebenen sich derzeit im Arbeitsspeicher befinden. Nützlich für das Verfolgen des Speicherverbrauchs. Beachten Sie, dass dies die Summe aller Ebenen von allen BGPics zurückgibt. Wenn Sie die Anzahl der Ebenen des aktuellen BGPic abfragen wollen, verwenden Sie **#ATTRLAYERS** mit dem Typ **#BGPIC**. (V4.5)

**#ATTRRAWXPOS, #ATTRRAWYPOS, #ATTRRAWWIDTH, #ATTRRAWHEIGHT:**

Diese vier Attribute können verwendet werden, um die reale Position und Größe einer Ebene rauszufinden. Der Unterschied zwischen diesen und den Standardattributen **#ATTRXPOS**, **#ATTRWIDTH** usw. besteht darin, dass die Standardattribute immer die Position und Größe der grundlegenden, nicht transformierten Ebene zurückgeben. Die Standardattribute berücksichtigen auch keine Unter-/Überhänge. Trotzdem sollten Sie wann immer möglich mit den Standardattributen arbeiten, da die Attribute **#ATTRRAWxxx** auf einem niedrigen Niveau im Ebenensystem arbeiten und von zukünftigen Änderungen im Ebenensystem betroffen sein könnten. (V4.7)

**#ATTRZPOS:**

Liefert die z-Position der Ebene. Siehe [Abschnitt 25.53 \[SetLayerZPos\]](#), [Seite 466](#), für Details. (V5.1)

**#ATTRDURATION:**

Gibt die Gesamtdauer der Video-Ebene in Millisekunden zurück. Dies wird nur von Ebenen des Typs **#VIDEO** unterstützt. (V6.0)

**#ATTRPOSITION:**

Gibt die aktuelle Position einer abgespielten oder pausierten Video-Ebene in Millisekunden zurück. Dies wird nur von Ebenen des Typs **#VIDEO** unterstützt. (V6.0)

**#ATTRFORMAT:**

Gibt den Quellformatnamen der Video-Ebene als Zeichenkette zurück. Dies wird nur für Ebene des Typs **#VIDEO** unterstützt. (V6.0)

**#ATTRCANSEEK:**

Gibt zurück ob [SeekLayer\(\)](#) auf dieser Video-Ebene verwendet werden kann oder nicht. Dies wird nur von Ebenen des Typs **#VIDEO** unterstützt. (V6.0)

**#ATTRPLAYING:**

Gibt **True** zurück, wenn diese Video-Ebene gerade abgespielt wird. Dies wird nur von Ebenen des Typs **#VIDEO** unterstützt. (V6.0)

**#ATTRPAUSED:**

Gibt **True** zurück, wenn dieses Video zurzeit pausiert. Dies wird nur von Ebenen des Typs **#VIDEO** unterstützt. (V6.0)

**#ATTRID:** Liefert die ID des Quellobjekts dieser Ebene. Dies gilt nur für die Ebenentypen **#ANIM**, **#BRUSH**, **#BRUSHPART**, **#BGPICPART**, **#TEXTOBJECT**, **#VECTORPATH** und **#VIDEO**. (V6.0)

**#ATTRDEPTH:**  
Gibt die Farbtiefe der Ebene zurück. Wenn dieser Wert kleiner oder gleich 8 ist, ist die Ebene eine Palettenebene. (V9.0)

**#ATTRPALETTE:**  
Gibt die Palette der Ebene als Tabelle zurück. Die Tabelle enthält so viele Elemente, wie Stifte in der Palette vorhanden sind. Die einzelnen Stifte werden als **RGB-Farben** zurückgegeben. Wenn die Ebene keine Palette hat, wird eine leere Tabelle zurückgegeben. (V9.0)

**#ATTRTRANSPARENTPEN:**  
Gibt den Stift zurück, der in der Palette der Ebene transparent ist. Wenn kein transparenter Stift vorhanden ist oder die Ebene keine Palette hat, wird **#NOPEN** zurückgegeben. (V9.0)

**#ATTRBUTTON:**  
Wenn die Ebene mit einer Schaltfläche verbunden ist, wird die ID dieser Schaltfläche zurückgegeben, andernfalls **Nil**. (V9.1)

**#ATTRGROUP:**  
Gibt den Namen der Gruppe zurück, der diese Ebene zugeordnet ist, oder eine leere Zeichenkette, wenn die Ebene keiner Gruppe zugeordnet ist. Siehe **Abschnitt 25.16 [GroupLayer]**, **Seite 418**, für Details. (V10.0)

Bitte beachten Sie, dass sich die Positions- und Größenwerte immer auf die Ebene im ursprünglichen, nicht transformierten Zustand beziehen. Wenn Sie eine Ebene drehen oder skalieren, erhalten Sie die ursprünglichen Dimensionen durch **#ATTRWIDTH** und **#ATTRHEIGHT**.

Folgende Attribute können für **#MEMORY** abgefragt werden:

**#ATTRSIZE:**  
Gibt die Größe des angegebenen Speicherblocks zurück. (V4.5)

**#ATTRCOUNT:**  
Gibt an, wie viele Speicherblöcke es zurzeit im Arbeitsspeicher gibt. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)

Folgende Attribute können für **#MENU** abgefragt werden:

**#ATTRCOUNT:**  
Gibt an, wie viele Menüleisten zurzeit verfügbar sind. Nützlich für das Verfolgen des Speicherverbrauchs. (V6.0)

Folgende Attribute können für **#MOVELIST** abgefragt werden:

**#ATTRCOUNT:**

Gibt zurück, wie viele Bewegungsliste-Objekte aktuell im Speicher sind. Nützlich, um die von Ihrem Skript verwendeten Ressourcen zu verfolgen. (V5.0)

Folgende Attribute können für **#MUSIC** abgefragt werden:

**#ATTRTYPE:**

Gibt das Format der Musik-Rohdaten zurück; Dieses Attribut wird **#MONO8**, **#MONO16**, **#STEREO8** oder **#STEREO16** zurückgeben. Funktioniert nicht bei Protracker-Modulen. (V2.0)

**#ATTRDURATION:**

Gibt die Musikdauer in Millisekunden zurück. Dies wird nicht bei Protracker-Modulen unterstützt. Wenn Sie mit **#ATTRDURATION** bei Protracker-Module anfragen, wird -1 zurückgegeben. (V2.0)

**#ATTRPITCH:**

Gibt die Wiedergabe-Tonhöhe (Frequenz) der Musik in Hertz zurück; nicht mit Protracker-Modulen möglich. (V2.0)

**#ATTRPOSITION:**

Gibt die Position der Musik in Millisekunden zurück. (V2.0)

**#ATTRFORMAT:**

Gibt das Musikformat als Zeichenkette zurück. (V2.0)

**#ATTRBITRATE:**

Liefert die Bitrate der Musik; wenn die derzeit abgespielte Musik eine variable Bitrate verwendet; nicht möglich bei Protracker-Modulen. (V2.0)

**#ATTRCOUNT:**

Gibt an, wie viele Musikstücke sich derzeit im Arbeitsspeicher befinden. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)

**#ATTRCANSEEK:**

Gibt zurück ob **SeekMusic()** bei diesem Musikobjekt verwendet werden kann oder nicht. (V5.0)

**#ATTRNUMSUBSONGS:**

Gibt die Anzahl der Subsongs zurück, die Sie mit **PlaySubsong()** abspielen können. Wenn es 1 ist, dann ist dort nur ein Song als Musikobjekt vorhanden. (V5.3)

**#ATTRCURSUBSONG:**

Gibt die Anzahl der aktuell abgespielten Subsongs zurück. (V5.3)

**#ATTRPLAYING:**

Gibt **True** zurück, wenn diese Musik gerade abgespielt wird. (V6.0)

**#ATTRPAUSED:**

Gibt **True** zurück, wenn diese Musik momentan angehalten wurde. (V6.0)

**#ATTRLOADER:**

Gibt den Namen des verwendeten Lademoduls zurück, mit dem die Musik geladen wurde. (V6.0)

Folgende Attribute können für **#PALETTE** abgefragt werden:

**#ATTRPALETTE:**

Gibt die Stifte der Palette als Tabelle zurück. Die Tabelle enthält so viele Elemente, wie Stifte in der Palette vorhanden sind. Die einzelnen Stifte werden als **RGB-Farben** zurückgegeben. (V9.0)

**#ATTRTRANSPARENTPEN:**

Gibt den Stift zurück, der in der Palette der Ebene transparent ist. Wenn kein transparenter Stift vorhanden ist oder die Ebene keine Palette hat, wird **#NOPEN** zurückgegeben. (V9.0)

**#ATTRDEPTH:**

Gibt die Tiefe der Palette zurück. Dies ist immer ein Wert zwischen 1 (= 2 Farben) und 8 (= 256 Farben). (V9.0)

**#ATTRCYCLE:**

Wenn es sich um eine Palette handelt, in der Farbwechselbereiche definiert sind, gibt dieses Attribut eine Tabelle mit allen definierten Farbwechselbereichen zurück. In diesem Fall enthält die von **#ATTRCYCLE** zurückgegebene Tabelle eine Reihe von Untertabellen, in denen die folgenden Felder initialisiert sind:

- Low:** Der Stiftindex, der den Anfang des Farbwechselbereichs markiert.
- High:** Der Stiftindex, der das Ende des Farbwechselbereichs markiert.
- Rate:** Die gewünschte Geschwindigkeit des Farbwechseleffekts. Ein Wert von 16384 bedeutet 60 Bilder pro Sekunde. Alle anderen Geschwindigkeiten werden linear von dieser Basis aus skaliert, z.B. ein Wert von 8192 bedeutet 30 Bilder pro Sekunde und so weiter.
- Reverse:** Wenn dieser Tag auf **True** gesetzt ist, sollten die Farben in umgekehrter Reihenfolge durchlaufen werden.
- Active:** Wenn dieser Tag auf **True** gesetzt ist, wird dieser Farbwechselbereich als aktiv markiert.

(V9.0)

**#ATTRLOADER:**

Gibt den Namen des Laders zurück, der zum Laden dieser Palette verwendet wurde. (V9.0)

**#ATTRCOUNT:**

Gibt zurück, wie viele Palettenobjekte sich derzeit im Speicher befinden. Nützlich zur Überwachung des Speicherverbrauchs. (V9.0)

Folgende Attribute können für **#POINTER** abgefragt werden:

**#ATTRWIDTH:**

Gibt die Breite des Mauszeigers zurück. (V4.5)

**#ATTRHEIGHT:**

Gibt die Höhe des Mauszeigers zurück. (V4.5)

**#ATTRTYPE:**

Gibt den Typ dieses Mauszeigers zurück; es kann **#STDPTR\_CUSTOM**, **#STDPTR\_SYSTEM** oder **#STDPTR\_BUSY** sein; Siehe [Abschnitt 36.1 \[CreatePointer\]](#), Seite 745, für Details. (V4.5)

**#ATTRCOUNT:**

Gibt an, wie viele Mauszeiger sich derzeit im Arbeitsspeicher befinden. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)

Folgende Attribute können für **#SAMPLE** abgefragt werden:

**#ATTRTYPE:**

Liefert das Format der Sample-Rohdaten; Dieses Attribut wird **#MONO8**, **#MONO16**, **#STEREO8** oder **#STEREO16** zurückgeben. (V2.0)

**#ATTRDURATION:**

Gibt die Dauer des Samples in Millisekunden zurück. (V2.0)

**#ATTRPITCH:**

Gibt die Tonhöhe (Frequenz) des Samples in Hertz zurück. (V2.0)

**#ATTRPOSITION:**

Gibt an, wie lange das Sample gespielt wurde und wie viele Male es wiederholt wurde. Die zurückgegebene Position wird ein Wert in Millisekunden (1000 Millisekunden = 1 Sekunde) sein. Dieser Wert wird jedes Mal zurückgesetzt, wenn das Sample wiederholt wird, so dass der zurückgegebene Positionswert niemals die Samplelänge überschreitet. Der zweite Rückgabewert gibt an, wie viele Male das Sample gespielt wurde. Es wird jedes Mal erhöht, wenn das Sample wiederholt wird. Wenn Sie die gesamte Spielzeit in Millisekunden ermitteln müssen, multiplizieren Sie einfach den zweiten Rückgabewert minus 1 mit der Sample-Dauer (benutzen Sie **#ATTRDURATION**) und fügen den ersten Rückgabewert hinzu. (V2.0)

**#ATTRCOUNT:**

Gibt an, wie viele Samples sich derzeit im Arbeitsspeicher befinden. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)

**#ATTRNUMFRAMES:**

Gibt die Anzahl der PCM-Daten im Sample zurück. (V5.0)

**#ATTRPLAYING:**

Liefert **True** zurück, wenn dieses Sample gerade abgespielt wird. (V6.0)

**#ATTRLOADER:**

Gibt den Namen des Lademoduls zurück, das verwendet wurde, um die Musik zu laden. (V6.0)

**#ATTRFORMAT:**

Gibt den Namen des Tonformats als Zeichenkette zurück. (V10.0)

Folgende Attribute können für **#SERIAL** abgefragt werden:

**#ATTRCOUNT:**

Gibt zurück, wie viele Objekte der seriellen Verbindung gerade im Speicher sind. Nützlich, um die von Ihrem Skript verwendeten Ressourcen zu verfolgen. (V8.0)

Folgende Attribute können für **#SERVER** abgefragt werden:

**#ATTRCOUNT:**

Gibt zurück, wie viele Netzwerkserverobjekte aktuell im Speicher sind. Nützlich, um die von Ihrem Skript verwendeten Ressourcen zu verfolgen. (V5.0)

Folgende Attribute können für **#SPRITE** abgefragt werden:

**#ATTRWIDTH:**

Gibt die Breite des Sprites zurück. (V2.0)

**#ATTRHEIGHT:**

Gibt die Höhe des Sprites zurück. (V2.0)

**#ATTRTRANSPARENTCOLOR:**

Gibt die transparente Farbe des Sprites oder **#NOTTRANSPARENCY** zurück. (V2.0)

**#ATTRHASMASK:**

Gibt **True** zurück, wenn das Sprite eine Maske hat. (V2.0)

**#ATTRHASALPHA:**

Gibt **True** zurück, wenn der Sprite einen Alphakanal hat. (V2.0)

**#ATTRNUMFRAMES:**

Gibt die Anzahl der Einzelbilder in diesem Sprite zurück. (V2.0)

**#ATTRCURFRAME:**

Gibt das momentan angezeigte Einzelbilder zurück. (V2.0)

**#ATTRONSCREEN:**

Gibt **True** zurück, wenn der angegebene Sprite derzeit auf dem Bildschirm vorhanden ist. (V2.5)

**#ATTRXPOS:**

Liefert die x-Position des Sprites auf dem Bildschirm. (V2.5)

**#ATTRYPOS:**

Liefert die y-Position des Sprites auf dem Bildschirm. (V2.5)

**#ATTRCOUNT:**

Gibt an, wie viele Sprites sich derzeit im Arbeitsspeicher befinden. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)

**#ATTRZPOS:**

Enthält die z-Position des Sprites. Siehe [Abschnitt 49.12 \[SetSpriteZPos\]](#), [Seite 1084](#), für Details. (V7.0)

**#ATTRDEPTH:**

Gibt die Farbtiefe des im Argument **param** angegebenen Einzelbildes zurück. Einzelbilder werden ab 1 gezählt. Wird das Argument **param** weggelassen, wird das erste Einzelbild verwendet. Wenn die Tiefe kleiner oder gleich 8 ist, ist der Sprite ein Paletten-Sprite. (V9.0)

**#ATTRPALETTE:**

Gibt die Palette des im Argument **param** angegebenen Einzelbildes zurück. Einzelbilder werden ab 1 gezählt. Wird das Argument **param** weggelassen, wird das erste Einzelbild verwendet. Die Palette des Einzelbildes wird als Tabelle zurückgegeben und enthält so viele Elemente, wie Stifte in der Palette vorhanden sind. Die einzelnen Stifte werden als **RGB-Farben** zurückgegeben. Wenn das Einzelbild keine Palette hat, wird eine leere Tabelle zurückgegeben. (V9.0)

**#ATTRTRANSPARENTPEN:**

Gibt den Stift zurück, der in der Palette des durch **param** angegebenen Einzelbildes transparent ist. Einzelbilder werden ab 1 gezählt. Wird das Argument **param** weggelassen, wird das erste Einzelbild verwendet. Wenn kein transparenter Stift vorhanden ist oder das Einzelbild keine Palette hat, wird **#NOPEN** zurückgegeben. (V9.0)

Folgende Attribute können für **#TEXTOBJECT** abgefragt werden:

**#ATTRWIDTH:**

Gibt die Breite des Textobjekts zurück.

**#ATTRHEIGHT:**

Gibt die Höhe des Textobjekts zurück.

**#ATTRFONTASCENDER:**

Gibt die Oberlänge der aktuellen Schriftart in Pixel zurück. Die Oberlänge einer Schriftart ist der maximale Buchstabenumfang von der Grundlinie bis zur obersten Linie (k-Linie). Oberlänge + Unterlänge ist immer gleich der Pixelhöhe der Schriftart. (V3.1)

**#ATTRFONTDESCENDER:**

Gibt die Unterlänge der aktuellen Schriftart in Pixeln zurück. Die Unterlänge einer Schriftart ist der maximale Buchstabenumfang von der Grundlinie bis zur untersten Linie (p-Linie). Oberlänge + Unterlänge ist immer gleich der Pixelhöhe der Schriftart. (V3.1)

**#ATTRFONTNAME:**

Gibt den Namen der aktuell ausgewählten Schriftart zurück. (V3.1)

- #ATTRFONTSIZE:**  
Gibt die Größe der aktuell ausgewählten Schriftart zurück. (V3.1)
- #ATTRFONTSCALABLE:**  
Gibt **True** zurück, wenn die Schriftart eine skalierbare Vektorschriftart ist. (V3.1)
- #ATTRFONTAA:**  
Gibt **True** zurück, wenn die Schriftart Antialiasing kann. (V3.1)
- #ATTRTEXT:**  
Gibt die Textzeichenfolge dieses Textobjekts zurück. (V4.0)
- #ATTRCOUNT:**  
Gibt an, wie viele Textobjekte sich derzeit im Arbeitsspeicher befinden. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)
- #ATTRDEPTH:**  
Gibt die Farbtiefe des Textobjekts zurück. Wenn dieser Wert kleiner oder gleich 8 ist, ist das Textobjekt ein Palettentext-Objekt. (V9.0)
- #ATTRPALETTE:**  
Gibt die Palette des Textobjekts als Tabelle zurück. Die Tabelle enthält so viele Elemente, wie Stifte in der Palette vorhanden sind. Die einzelnen Stifte werden als **RGB-Farben** zurückgegeben. Wenn das Textobjekt keine Palette hat, wird eine leere Tabelle zurückgegeben. (V9.0)
- #ATTRTRANSPARENTPEN:**  
Gibt den Stift zurück, der in der Palette des Textobjekts transparent ist. Wenn kein transparenter Stift vorhanden ist oder das Textobjekt keine Palette hat, wird **#NOPEN** zurückgegeben. (V9.0)
- #ATTRTYPE:**  
Gibt den Typ des Textobjekts zurück. Dieser wird entweder auf **#IMAGETYPE\_RASTER** für ein Rastertextobjekt oder **#IMAGETYPE\_VECTOR** für ein Vektortextobjekt gesetzt. (V10.0)
- #ATTRFONTDEPTH:**  
Wenn die Schriftart eine Amiga-Farbschrift ist, gibt dieses Attribut ihre Farbtiefe zurück. (V9.0)
- #ATTRFONTPALETTE:**  
Wenn die Schriftart eine Amiga-Farbschrift ist, gibt dieses Attribut seine Palette als Tabelle zurück. Die Tabelle enthält so viele Elemente, wie Stifte in der Palette vorhanden sind. Die einzelnen Stifte werden als **RGB-Farben** zurückgegeben. (V9.0)
- #ATTRFONTTRANSPARENTPEN:**  
Wenn die Schriftart eine Amiga-Farbschrift ist, gibt dieses Attribut den Stift zurück, der in der Palette der Schriftart transparent ist. (V9.0)
- #ATTRFONTEENGINE:**  
Gibt das Schriftartenmodul zurück, welches zum Öffnen dieser Schriftart verwendet wurde. Dies ist entweder **#FONTEENGINE\_INBUILT**, **#FONTEENGINE\_**



NATIVE oder #FONTENGINE\_PLUGIN. Siehe [Abschnitt 52.32 \[SetFont\]](#), [Seite 1163](#), für Details. (V9.0)

**#ATTRFONTTYPE:**

Gibt den Schrifttyp zurück. Dies kann einer der folgenden Typen sein:

**#FONTTYPE\_BITMAP:**

Eine Amiga-Bitmap-Schriftart.

**#FONTTYPE\_COLOR:**

Eine Amiga-Farbschrift.

**#FONTTYPE\_VECTOR:**

Eine Vektorschrift, z.B. im TrueType- oder OpenType-Format.

**#FONTTYPE\_BRUSH:**

Eine benutzerdefinierte Schrift, die mit dem Befehl [CreateFont\(\)](#) aus einer Pinselquelle erstellt wurde. Siehe [Abschnitt 52.7 \[CreateFont\]](#), [Seite 1141](#), für Details. (V10.0)

(V9.0)

**#ATTRFONTCHARMAP:**

Gibt die von der Schriftart verwendete Zeichentabelle zurück. Dies wird nur von Schriftarten unterstützt, die vom integrierten Schriftartenmodul verwaltet werden, d.h. die Schriftart muss mit #FONTENGINE\_INBUILT geöffnet worden sein. Siehe [Abschnitt 52.16 \[GetCharMaps\]](#), [Seite 1151](#), für Details. (V9.0)

**#ATTRFONTLOADER:**

Gibt den Namen des Laders zurück, der zum Laden dieser Schriftart verwendet wurde. (V10.0)

**#ATTRFONTHEIGHT:**

Gibt die Pixelhöhe der Schriftart zurück. Dies ist oft dasselbe wie #ATTRFONTSIZE, jedoch nicht, wenn die Schriftart im Punktmodus geöffnet wurde oder wenn das zugrunde liegende Text-Modul die Schriftgröße als etwas anderes als die Schrifthöhe interpretiert. In jedem Fall ist #ATTRFONTHEIGHT immer dasselbe wie #ATTRFONTASCENDER + #ATTRFONTDESCENDER.

**#ATTRFONTFORMAT:**

Gibt den Namen des Schriftformats als Zeichenkette zurück. (V10.0)

**#ATTRADJUSTX:**

Beim Zeichnen von Textobjekten mit [DisplayTextObject\(\)](#) positioniert Hollywood diese so, dass sie aussehen, als wären sie mit [TextOut\(\)](#) gezeichnet worden, was bedeutet, dass sie gegebenenfalls nach links und oben versetzt sein könnten, falls Teile einiger Zeichen so gestaltet sind, dass sie im Bereich vorheriger Zeichen erscheinen. Dies ist häufig bei Zeichen wie "j" der Fall. Sie können die Anzahl der horizontalen Pixel abfragen, die Hollywood das Textobjekt versetzt, indem Sie diesen Tag abfragen. Die Anpassung von Textobjekten kann deaktiviert werden, indem [NoAdjust](#) beim Aufruf von [CreateTextObject\(\)](#) auf [True](#) gesetzt wird. (V10.0)

**#ATTRADJUSTY:**

Dies tut dasselbe wie **#ATTRADJUSTX** (siehe oben), gibt aber die vertikalen Anpassungspixel für dieses Textobjekt zurück. (V10.0)

Folgende Attribute können für **#TIMEOUT** abgefragt werden:

**#ATTRDURATION:**

Gibt die Timeout-Dauer dieses Timeout-Objekts in Millisekunden zurück. (V4.5)

**#ATTRFUNCTION:**

Gibt die Callback-Funktion des angegebenen Timeout-Objekts zurück. (V4.5)

**#ATTRUSERDATA:**

Gibt die dazugehörigen Benutzerdaten des Timeout-Objekts zurück. (V4.5)

Folgende Attribute können für **#TIMER** abgefragt werden:

**#ATTRCOUNT:**

Gibt zurück, wie viele Timer-Objekte gerade im Speicher sind. Nützlich, um die von Ihrem Skript verwendeten Ressourcen zu verfolgen. (V5.0)

**#ATTRELAPSE:**

Gibt den Schwellenwert für das Ablaufen des Timers zurück. (V9.0)

Folgende Attribute können für **#UDPOBJECT** abgefragt werden:

**#ATTRCOUNT:**

Gibt zurück, wie viele UDP-Objekte derzeit im Speicher sind. Nützlich, um die von Ihrem Skript verwendeten Ressourcen zu verfolgen. (V5.0)

Folgende Attribute können für **#VECTORPATH** abgefragt werden:

**#ATTRCOUNT:**

Gibt an, wie viele Vektorpfadobjekte es zurzeit im Arbeitsspeicher gibt. Nützlich für das Verfolgen des Speicherverbrauchs. (V4.5)

Folgende Attribute können für **#VIDEO** abgefragt werden:

**#ATTRWIDTH:**

Gibt die Breite des Videos zurück. (V5.0)

**#ATTRHEIGHT:**

Gibt die Höhe des Videos zurück. (V5.0)

**#ATTRDURATION:**

Gibt die Gesamtdauer des Videos in Millisekunden zurück. (V5.0)

**#ATTRPOSITION:**

Gibt die aktuelle Position eines abgespielten oder pausierten Videos in Millisekunden zurück. (V5.0)

**#ATTRFORMAT:**

Gibt den Namen des Videoformats als Zeichenfolge zurück. (V5.0)

**#ATTRNUMFRAMES:**

Gibt die Anzahl der Einzelbilder dieses Videos zurück. Bitte beachten Sie, dass dies oft eine Annäherung ist, weil es zu viel Zeit in Anspruch nehmen würde, um eine genaue Berechnung aller Einzelbilder in einem Video-Stream zu machen. Dies kann auch 0 zurückgeben, wenn der Videocodex die Einzelbildberechnung nicht unterstützt. (V5.0)

**#ATTRCOUNT:**

Gibt an, wie viele Videos es derzeit im Arbeitsspeicher gibt. Nützlich für das Verfolgen des Speicherverbrauchs. (V5.0)

**#ATTRCANSEEK:**

Gibt zurück ob `SeekVideo()` auf diesem Videoobjekt verwendet werden kann oder nicht. (V5.0)

**#ATTRDRIVER:**

Gibt den für dieses Video verwendeten Treiber zurück. Siehe [Abschnitt 55.4 \[ForceVideoDriver\]](#), [Seite 1237](#), für Details. Dies ist seit Hollywood 6.0 veraltet. Benutze `#ATTRLOADER` stattdessen. (V6.1)

**#ATTRPLAYING:**

Gibt `True` zurück, wenn dieses Video gerade abgespielt wird. (V6.0)

**#ATTRPAUSED:**

Gibt `True` zurück, wenn dieses Video momentan angehalten wird. (V6.0)

**#ATTRSCALEWIDTH:**

Gibt die für das Video eingestellte aktuelle Skalierungsbreite zurück `SetVideoSize()`. (V6.0)

**#ATTRSCALEHEIGHT:**

Gibt die für das Video eingestellte aktuelle Skalierungshöhe zurück `SetVideoSize()`. (V6.0)

**#ATTRSCALEMODE:**

Gibt den für das Video eingestellten aktuellen Skalierungsmodus zurück `SetVideoSize()`. (V6.0)

**#ATTRLOADER:**

Gibt den Namen des Ladeprogramms zurück, welches für das Laden des Videos verwendet wurde. (V6.0)

**EINGABEN**

<code>obj</code>	Art des Abfrageobjekts (siehe Liste oben)
<code>id</code>	ID des Objekts
<code>attr</code>	welche Informationen zurückgegeben werden sollen
<code>param</code>	optional: zusätzlicher Parameter, der für einige Attribute erforderlich ist (siehe oben)

**RÜCKGABEWERTE**

info            die gewünschten Informationen

**BEISPIEL**

```
width = GetAttribute(#DISPLAY, 0, #ATTRWIDTH)
```

Der obige Code fragt das Display nach der aktuellen Breite ab. Da es nur ein Display gibt, müssen Sie keine ID angeben.

## 40.5 GetObjectData

**BEZEICHNUNG**

GetObjectData – ruft private Daten aus einem Objekt ab (V5.0)

**ÜBERSICHT**

```
data = GetObjectData(type, id, key$)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um private Daten von einem Objekt abzurufen, die unter Verwendung des Befehls **SetObjectData()** gespeichert wurde. Geben Sie einfach den Typ und die ID des Objekts sowie den Schlüssel an, unter denen die Daten gespeichert wurden und dieser Befehl wird die entsprechenden Daten zurückgeben. Siehe [Abschnitt 40.1 \[Objekttypen\]](#), [Seite 805](#), für eine Liste aller Objekttypen.

**EINGABEN**

type            Art des Objekts  
id              ID des Objekts  
key\$            Schlüssel, unter dem die Daten gespeichert wurden

**RÜCKGABEWERTE**

data            Daten, die unter dem angegebenen Schlüssel gespeichert wurden

**BEISPIEL**

Siehe [Abschnitt 40.10 \[SetObjectData\]](#), [Seite 839](#).

## 40.6 GetObjects

**BEZEICHNUNG**

GetObjects – gibt alle Objekte des angegebenen Typs zurück (V5.1)

**ÜBERSICHT**

```
table, count = GetObjects(type)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um eine Liste aller Objekte des angegebenen Typs abzurufen, die derzeit im Arbeitsspeicher sind. Dieser Befehl gibt eine Tabelle mit den Identifikatoren der Objekte sowie die Gesamtzahl der Objekte der angegebenen Art zurück.

Siehe [Abschnitt 40.1 \[Objekttypen\]](#), [Seite 805](#), für eine Liste aller Objekttypen.

**EINGABEN**

`type`        gibt die Objekte von diesem Typ zurück

**RÜCKGABEWERTE**

`table`        Eine Tabelle mit Identifikatoren aller Objekte des angegebenen Typs

`count`        Anzahl der Elemente in der Tabelle

**BEISPIEL**

```
t, c = GetObjects(#BRUSH)
For Local k = 0 To c - 1 Do DebugPrint(t[k])
```

Der obige Code listet alle Objekte des Typs `#BRUSH` auf, die derzeit im Arbeitsspeicher sind.

## 40.7 GetObjectType

**BEZEICHNUNG**

`GetObjectType` – ruft den Typ eines Objekts ab (V5.0)

**ÜBERSICHT**

```
type = GetObjectType(handle)
```

**BESCHREIBUNG**

Dieser Befehl gibt den Typ des angegebenen Objekts zurück. Dieses Objekt, welches diesem Befehl übergeben wird, muss mit der automatischen ID-Zuweisung angelegt worden sein. Siehe [Automatische ID-Zuweisung](#) für mehr Informationen.

Siehe [Abschnitt 40.1 \[Objekttypen\]](#), [Seite 805](#), für eine Liste aller Objekttypen.

**EINGABEN**

`handle`        Objekt, dessen Typ Sie abrufen möchten

**RÜCKGABEWERTE**

`type`        Typ des Objekts

**BEISPIEL**

```
my_anim = LoadAnim(nil, "test.gif")
my_brush = LoadBrush(nil, "test.png")
DebugPrint(GetObjectType(my_anim), GetObjectType(my_brush))
```

Der obige Code gibt die Werte der Konstanten `#ANIM` und `#BRUSH` aus.

## 40.8 HaveObject

**BEZEICHNUNG**

`HaveObject` – prüft, ob ein bestimmtes Objekt verfügbar ist (V5.2)

**ÜBERSICHT**

```
r = HaveObject(type, id)
```

**BESCHREIBUNG**

Mit diesem Befehl können Sie überprüfen, ob ein bestimmtes Objekt existiert und bereits geladen wurde. Geben Sie einfach den Typ und die ID des Objekts bei diesem Befehl ein und es wird **True** oder **False** zurückgegeben, abhängig davon, ob das Objekt im Arbeitsspeicher ist oder nicht. Siehe [Abschnitt 40.1 \[Objekttypen\]](#), [Seite 805](#), für eine Liste aller Objekttypen.

**EINGABEN**

**type**            Typ des zu prüfenden Objekts

**id**             ID des zu prüfenden Objekts

**RÜCKGABEWERTE**

**r**                **True** wenn das Objekt vorhanden ist, andernfalls **False**

**BEISPIEL**

```
DebugPrint(HaveObject(#BRUSH, 1))
```

Der obige Code gibt **True** zurück, wenn der Pinsel Nummer 1 im Arbeitsspeicher ist, andernfalls **False**.

## 40.9 HaveObjectData

**BEZEICHNUNG**

HaveObjectData – prüft, ob Daten in einem Objekt vorhanden sind (V6.1)

**ÜBERSICHT**

```
b = HaveObjectData(type, id, key$)
```

**BESCHREIBUNG**

Mit diesem Befehl können Sie überprüfen, ob private Daten mit dem Befehl [SetObjectData\(\)](#) in einem Objekt unter dem angegebenen Schlüssel gespeichert wurden. Geben Sie einfach den Typ und die ID des Objekts sowie den Schlüssel in **key\$** an. Dieser Befehl prüft nun, ob es Daten für den Schlüssel gibt (**True**) oder nicht (**False**).

Siehe [Abschnitt 40.1 \[Object types\]](#), [Seite 805](#), für eine Liste aller Objekttypen.

**EINGABEN**

**type**            Art des Objekts

**id**             ID des Objekts

**key\$**           Schlüssel zur Abfrage

**RÜCKGABEWERTE**

**b**                **True**, wenn der Schlüssel Daten hat, andernfalls **False**

## 40.10 SetObjectData

### BEZEICHNUNG

SetObjectData – speichert private Daten in einem Objekt (V5.0)

### ÜBERSICHT

SetObjectData(type, id, key\$, value)

### BESCHREIBUNG

Mit diesem Befehl können Sie beliebige private Daten einem Objekt zuordnen. Sie müssen den Typ und die ID des Objekts sowie einen Schlüssel, unter dem die Daten im Objekt gespeichert werden sollen, angeben. **value** kann jede Art von Hollywood-Datenwert sein wie eine Zeichenfolge, eine Tabelle, eine Zahl oder sogar eine Funktion. Wenn der Schlüssel in **key\$** angegeben ist und bereits innerhalb des Objekts verwendet wird, werden die alten Daten durch die neuen ersetzt.

Um später auf die Daten zuzugreifen, können Sie den Befehl **GetObjectData()** benutzen.

Siehe [Abschnitt 40.1 \[Objekttypen\]](#), [Seite 805](#), für eine Liste aller Objekttypen.

### EINGABEN

type	Typ des zu verwendenden Objekts
id	ID des zu verwendenden Objekts
key\$	Schlüssel, unter dem die Daten gespeichert werden sollen
value	Daten, welche gespeichert werden

### BEISPIEL

```
SetObjectData(#BRUSH, 1, "brushgroup", "A")  
d$ = GetObjectData(#BRUSH, 1, "brushgroup")
```

Der obige Code speichert den Wert "A" in Pinsel 1 unter dem Schlüssel "brushgroup" und ruft ihn dann wieder ab. **d\$** wird auf "A" gesetzt.





## 41 Palettenbibliothek

### 41.1 Übersicht

Diese Bibliothek bietet Befehle für den Umgang mit Paletten. Wie der Name schon sagt, ist eine Palette eine Menge von Farben. Die minimale Anzahl von Farben in einer Palette ist 2 und die maximale beträgt 256. Die einzelnen Farben in einer Palette werden als Stifte bezeichnet. Diese Stifte werden durch ihre Indizes innerhalb der Palette beginnend mit 0 adressiert. Somit ist der erste Stift in einer Palette Stift 0, der zweite Stift 1 und so weiter. In einer Palette kann sich ein Stift befinden, der als transparenter Stift gekennzeichnet ist. Pixel, die diesen Stift verwenden, z.B. in einem Palettenpinsel, erscheinen dann transparent. In der Regel ist Stift 0 der transparente Stift.

Die Anzahl der Farben in einer Palette wird als Farbtiefe bezeichnet und immer als Exponent einer 2er-Potenz ausgedrückt. Folgende Farbtiefen stehen zur Verfügung:

1-bit:	2 Farben.
2-bit:	4 Farben.
3-bit:	8 Farben.
4-bit:	16 Farben.
5-bit:	32 Farben.
6-bit:	64 Farben.
7-bit:	128 Farben.
8-bit:	256 Farben.

Aufgrund der begrenzten Anzahl von Farben, die in einer Palette gespeichert werden können, werden Palettengrafiken heutzutage nicht mehr häufig verwendet. Allerdings hat die Verwendung von Palettengrafiken in bestimmten Situationen einige Vorteile gegenüber von RGB-Grafiken, wie zum Beispiel:

- Palettengrafiken machen es sehr einfach, bestimmte Effekte wie Farbwechsel oder Überblendungen zu implementieren. Aufgrund der begrenzten Anzahl von Farben, die geändert werden müssen, können diese Effekte in sehr wenigen CPU-Zyklen berechnet werden.
- Der Speicherverbrauch ist viel geringer als bei Verwendung von RGB-Grafiken. Bei 32-Bit-RGB-Grafiken benötigt ein einzelnes Pixel 4 Byte Speicher, während im Palettenmodus ein einzelnes Pixel nur 1 Byte Speicher benötigt. Daher benötigt ein 1920x1080-Bild im 32-Bit-Modus etwa 8 Megabyte, im Palettenmodus jedoch nur 2 Megabyte Speicher.
- Da Palettengrafiken weniger Speicherplatz benötigen als RGB-Grafiken, können sie auch besser komprimiert werden, wenn sie auf der Festplatte gespeichert werden, z.B. als PNG-Bilder. Das Speichern von Bildern, die nicht mehr als 256 Farben als RGB-Pixel in PNG verwenden, führt zu einer erheblich größeren Bilddatei als das Speichern als PNG, die eine Palette verwenden. Aus diesem Grund sollten Sie Bilder, die nicht viele Farben beinhalten, als Palettenbilder speichern, wenn Ihnen die Dateigröße wichtig ist. Befehle wie `SaveBrush()` unterstützen auch Palettenbilder.

Natürlich muss eine Palette immer an ein anderes Objekt angehängt werden, das die eigentlichen Pixeldaten liefert, die mit den Farben aus der Palette gezeichnet werden sollen. In Hollywood unterstützen die folgenden Objekttypen Paletten:

- Animationen
- Hintergrundbilder (BGPics)
- Pinsel
- Displays
- Sprites

Für die meisten Objekttypen können Sie einfach den Tag `LoadPalette` in Befehlen wie `LoadBrush()` auf `True` setzen, damit Hollywood einen Palettenpinsel für Sie erstellt. Sie können auch RGB-Pinsel in Paletten-Pinsel umwandeln, indem Sie Befehle wie `QuantizeBrush()` oder `RemapBrush()` verwenden. Sobald Sie ein Palettenobjekt haben, können Sie dessen Palette mit dem Befehl `SetPalette()` ändern.

Besondere Vorsicht ist geboten, wenn ein Display in den Palettenmodus versetzt wird. Dies hat mehrere Auswirkungen, die Sie beachten müssen, um ein Display im Palettenmodus optimal nutzen zu können. Siehe [Abschnitt 24.16 \[Palettenmodus-Displays\]](#), [Seite 393](#), für Details.

## 41.2 ContrastPalette

### BEZEICHNUNG

`ContrastPalette` – erhöht oder reduziert den Palettenkontrast (V9.0)

### ÜBERSICHT

`ContrastPalette(id, inc[, repeat])`

### BESCHREIBUNG

Mit diesem Befehl können Sie den Farbkontrast in der angegebenen Palette erhöhen oder reduzieren. Wenn das Argument `inc` auf `True` gesetzt ist, wird der Kontrast verbessert, bei `False` wird der Kontrast reduziert. Mit dem optionalen Argument `repeat` kann der Effekt mehrmals auf die Palette angewendet werden. Dies ist nützlich, wenn Sie schärfere Kontraste erstellen möchten.

### EINGABEN

<code>id</code>	Palette, die geändert wird
<code>inc</code>	<code>True</code> , um den Kontrast zu erhöhen; <code>False</code> , um den Kontrast zu reduzieren
<code>repeat</code>	optional: Gibt an, wie oft der Kontrastvorgang wiederholt werden soll (voreingestellt ist 1, was bedeutet, dass der Effekt nur einmal ausgeführt wird)

## 41.3 CopyPalette

### BEZEICHNUNG

`CopyPalette` – kopiert eine Palette (V9.0)

### ÜBERSICHT

`[id] = CopyPalette(source, dest)`

**BESCHREIBUNG**

Dieser Befehl erstellt eine Kopie von der in **source** angegebene Palette und gibt sie in **dest** zurück. Die neue Palette ist unabhängig von der alten Palette, so dass Sie die Quellpalette nach dem Kopieren löschen können.

Wenn Sie **Nil** als **dest** angeben, wird automatisch eine freie ID für diese Palette ausgewählt und in **id** zurückgegeben. Andernfalls wird die neue Palette die in **dest** angegebene ID verwenden.

**EINGABEN**

**source**      Identifikator der zu kopierenden Palette

**dest**        ID der neu erstellten Palette oder **Nil** für die automatische ID-Zuweisung

**RÜCKGABEWERTE**

**id**            optional: Identifikator der neuen Palette; wird nur zurückgegeben werden, wenn Sie **Nil** als Argument in **dest** angegeben haben (siehe oben)

**BEISPIEL**

```
CopyPalette(1, 10)
FreePalette(1)
```

Der obige Code erstellt eine neue Palette 10, die dieselben Farbdaten wie Palette 1 enthält. Dann wird die Palette 1 gelöscht, da sie nicht mehr benötigt wird.

## 41.4 CopyPens

**BEZEICHNUNG**

CopyPens – kopiert Stifte von einer Palette in eine andere (V9.0)

**ÜBERSICHT**

```
CopyPens(srcid, dstid, srcidx, n[, dstidx])
```

**BESCHREIBUNG**

Dieser Befehl kopiert **n**-Stifte aus der von **srcid** angegebenen Palette in die durch **dstid** angegebene Palette. Die Stifte werden aus dem Index **srcidx** der Quellpalette gelesen und in den Index **dstidx** der Zielpalette kopiert. Wenn **dstidx** weggelassen wird, wird der in **srcidx** angegebene Index als Zielindex verwendet.

Beachten Sie, dass es erlaubt ist, denselben Paletten-Identifikator für **srcid** und **dstid** zu verwenden. In diesem Fall können die Stifte innerhalb eines einzelnen Palettenobjekts verschoben werden. Auch überlappende Stifte werden unterstützt.

**EINGABEN**

**srcid**        Quellpalette

**dstid**        Zielpalette; kann mit der Quelle identisch sein

**srcidx**      Index des ersten zu kopierenden Stifts (ab 0)

**n**            Anzahl der zu kopierenden Stifte

**dstidx**      optional: Index zum Kopieren von Stiften in die Zielpalette; voreingestellt ist **srcidx**

**BEISPIEL**

```
CopyPens(1, 2, 0, 32)
```

Der obige Code kopiert die ersten 32 Stifte von Palette 1 in die Palette 2.

## 41.5 CreatePalette

**BEZEICHNUNG**

CreatePalette – erstellt eine neue Palette (V9.0)

**ÜBERSICHT**

```
[id] = CreatePalette(id[, data, t])
```

**BESCHREIBUNG**

Dieser Befehl erstellt eine neue Palette und weist ihr den Identifikator `id` zu. Das Argument `data` kann entweder eine Tabelle mit einer Anzahl von Farben sein, die zur Initialisierung der Stifte der Palette verwendet werden sollen, oder Sie können `data` auf einen von Hollywoods vordefinierten Palettentypen setzen. Siehe unten für alle vordefinierten Palettentypen, die von Hollywood unterstützt werden. Wenn Sie `Nil` im Argument `id` übergeben, wählt `CreatePalette()` automatisch einen Identifikator für die neue Palette aus und gibt ihn an Sie zurück.

Die folgenden vordefinierten Palettentypen werden von Hollywood unterstützt:

**#PALETTE\_MONOCHROME:**

Zweifarbige Schwarz-Weiß-Palette.

**#PALETTE\_GRAY4:**

4-Farben-Graustufen-Palette.

**#PALETTE\_GRAY8:**

8-Farben-Graustufen-Palette.

**#PALETTE\_GRAY16:**

16-Farben-Graustufen-Palette.

**#PALETTE\_GRAY32:**

32-Farben-Graustufen-Palette.

**#PALETTE\_GRAY64:**

64-Farben-Graustufen-Palette.

**#PALETTE\_GRAY128:**

128-Farben-Graustufen-Palette.

**#PALETTE\_GRAY256:**

256-Farben-Graustufen-Palette.

**#PALETTE\_CGA:**

Standard-CGA-Palette (16 Farben).

**#PALETTE\_OCS:**

Standard-OCS-Palette (32 Farben).

```
#PALETTE_EGA:
    Standard-EGA-Palette (64 Farben).

#PALETTE_AGA:
    Standard-AGA-Palette (256 Farben).

#PALETTE_WORKBENCH:
    Standard-Palette der klassischen Amiga Workbench (256 Farben).

#PALETTE_MACINTOSH:
    Standardmäßige klassische Macintosh-Palette (256 Farben).

#PALETTE_WINDOWS:
    Standardmäßige klassische Windows-Palette (256 Farben).

#PALETTE_DEFAULT:
    Gleich wie #PALETTE_AGA. Wenn Sie das Argument data weglassen, wird
    CreatePalette() die neue Palette mit den Stiften aus #PALETTE_DEFAULT
    initialisieren.
```

Wenn Sie im Argument **data** eine Tabelle mit Farben übergeben, stellen Sie sicher, dass alle Farben als **RGB-Werte** übergeben werden. Beachten Sie, dass die Tabelle auch ein spärliches Array sein kann, bei dem nur die Stifte initialisiert sind, die Sie tatsächlich benötigen. Stifte, die nicht in der Tabelle **data** enthalten sind, werden auf Schwarz initialisiert. Ein Beispiel finden Sie weiter unten.

Mit dem optionalen Tabellenargument **t** können weitere Optionen angegeben werden. Folgende Optionen können Sie verwenden:

**Depth:** Die gewünschte Farbtiefe für die Palette. Diese muss zwischen 1 (= 2 Farben) und 8 (= 256 Farben) liegen. Der Standardwert ist 8. Wenn **Depth** mehr Farben angibt, als Sie in der Tabelle im Parameter **data** übergeben, werden die verbleibenden Farben auf Schwarz initialisiert. Dieser Tag wird nur verwendet, wenn Sie eine Tabelle im Argument **data** übergeben. Wenn Sie in **data** einen vordefinierten Palettentyp übergeben, hat die Farbtiefe des vordefinierten Palettentyps Vorrang vor der hier angegebenen Farbtiefe.

**TransparentPen:** Mit diesem Tag kann der Stift angegeben werden, der in der Palette transparent sein soll. Der Standardwert ist **#NOPEN**, was bedeutet, dass kein Stift transparent sein wird.

**Cycle:** Mit diesem Tag können mehrere Farbbereiche definiert werden, die durchlaufen werden können. Wenn diese Option gesetzt ist, müssen Sie eine Tabelle mit Untertabellen an **Cycle** übergeben, wobei jede Untertabelle eine Konfiguration eines Farbzykluseffekts beschreibt. Jede Untertabelle unterstützt die folgenden Tags:

**Low:** Der Stiftindex, der den Beginn des Farbbereichs markiert.

**High:** Der Stiftindex, der das Ende des Farbbereichs markiert.

**Rate:** Die gewünschte Geschwindigkeit des Farbwechseleffekts. Ein Wert von 16384 gibt 60 Bilder pro Sekunde an. Alle anderen

Geschwindigkeiten skalieren linear von dieser Basis, z.B. bedeutet ein Wert von 8192 30 Bilder pro Sekunde usw.

**Reverse:** Wenn dieser Tag auf **True** gesetzt ist, werden die Farben in umgekehrter Reihenfolge durchlaufen. Der Standardwert ist **False**.

**Active:** Wenn dieser Tag auf **False** gesetzt ist, wird der Farbbereich als inaktiv markiert. Voreingestellt ist **True**.

## EINGABEN

**id** ID für die neue Palette oder **Nil** für die automatische ID-Zuweisung

**data** optional: entweder einer der vordefinierten Palettentypen (siehe oben) oder eine Tabelle mit einem Array von Farben (Voreingestellt ist **#PALETTE\_DEFAULT**)

**t** optional: Tabelle mit weiteren Optionen (siehe oben)

## RÜCKGABEWERTE

**id** optional: Identifikator der Palette; wird nur zurückgegeben, wenn Sie beim Argument **id** **Nil** angegeben haben.

## BEISPIEL

```
CreatePalette(1, {#RED, #GREEN, #BLUE}, {Depth = 2})
```

Mit dem obigen Code wird eine Palette mit vier Farben erstellt, die auf Rot, Grün, Blau und Schwarz initialisiert sind.

```
CreatePalette(1, {[0] = #RED, [127] = #BLUE, [255] = #GREEN})
```

Der obige Code erstellt eine neue Palette mit 256 Farben und initialisiert Stift 0 auf Rot, Stift 127 auf Blau und den letzten Stift auf Grün. Alle anderen Stifte werden auf Schwarz initialisiert.

```
CreatePalette(1)
```

Erzeugt eine neue Palette und initialisiert ihre Farben auf die von **#PALETTE\_DEFAULT**.

```
CreatePalette(1, #PALETTE_CGA)
```

Erstellt eine neue Palette mit den CGA-Farben.

## 41.6 CyclePalette

### BEZEICHNUNG

CyclePalette – durchläuft die Farben der Palette (V9.0)

### ÜBERSICHT

```
CyclePalette(id, start, end[, repeat])
```

### BESCHREIBUNG

Dieser Befehl wechselt die Palettenfarben zwischen dem durch **start** und dem durch **end** angegebenen Stifte. Wenn **end** größer als **start** ist, werden alle Stifte, die mit dem Index

**start** beginnen, nach rechts verschoben und enden bei dem durch **end** angegebenen Stift. Wenn **start** größer als **end** ist, werden die Stifte in umgekehrter Reihenfolge durchlaufen, d.h. sie werden nach links verschoben und enden bei dem durch **start** angegebenen Stift. Mit dem Argument **repeat** kann angegeben werden, wie oft der Zyklus wiederholt werden soll. Der Standardwert ist 1, was bedeutet, dass die Farben nur einmal durchlaufen werden.

### EINGABEN

**id**            Identifikator der Palette, deren Stifte durchlaufen werden sollen

**start**        Startstift des Bereichs, der durchlaufen wird

**end**           Endstift des Bereichs, der durchlaufen wird

**repeat**       optional: Anzahl der Wiederholungen des Durchlaufs (Voreingestellt ist 1)

### BEISPIEL

```
@DISPLAY {Palette = #PALETTE_CGA}
SetFont(#SANS, 96)
SetPaletteMode(#PALETTEMODE_PEN)
t$ = "Hollywood"
For Local k = 0 To StrLen(t$) - 1
    SetDrawPen(k + 2)
    Print(MidStr(t$, k, 1))
Next
ExtractPalette(1, #BGPIC, 1)
StartTimer(1)
Repeat
    CyclePalette(1, 2, 10)
    SetPalette(1)
    WaitTimer(1, 80)
Forever
```

Der obige Code gibt die einzelnen Zeichen der Zeichenkette "Hollywood" in verschiedenen Farben aus und wechselt dann deren Farben.

## 41.7 ExtractPalette

### BEZEICHNUNG

ExtractPalette – extrahiert die Palette aus dem Objekt (V9.0)

### ÜBERSICHT

```
[id] = ExtractPalette(id, srctype, srcid[, frame])
```

### BESCHREIBUNG

Dieser Befehl extrahiert die Palette aus dem durch **srctype** und **srcid** angegebenen Objekt, erstellt eine neue Palette und weist dieser neuen Palette den Identifikator **id** zu. wenn Sie **Nil** als **id** verwenden, wird **ExtractPalette()** automatisch eine freie ID für Sie auswählen. Andernfalls verwendet die neue Palette den in **id** angegebenen Identifikator.

Die folgenden Objekttypen können in **src**type übergeben werden:

- #ANIM:** Extrahiert die Palette aus einem Einzelbild der Animation. Wenn Sie **src**type auf **#ANIM** setzen, müssen Sie auch das Einzelbild der Animation übergeben, dessen Palette im Parameter **frame** extrahiert werden soll. Einzelbilder werden ab 1 gezählt. Dies ist auch der Standardwert für **frame**.
- #BGPIC:** Extrahiert die Palette aus einem Hintergrundbild.
- #BRUSH:** Extrahiert die Palette aus einem Pinsel.
- #FONT:** Extrahiert die Palette aus einer farbigen Schriftart.
- #SPRITE:** Extrahiert die Palette aus einem Einzelbild des Sprites. Wenn Sie **src**type auf **#SPRITE** setzen, müssen Sie auch das Einzelbild des Sprites, dessen Palette extrahiert werden soll, im Parameter **frame** übergeben. Einzelbilder werden ab 1 gezählt, was auch der Standardwert für **frame** ist.

#### EINGABEN

- id** ID für die neue Palette oder **Nil** für die automatische ID-Zuweisung
- src**type Objekttyp, der als Quelle verwendet werden soll (siehe oben)
- src**id ID des zu verwendenden Quellobjekts
- frame** optional: Nummer des Einzelbildes, wenn der Objekttyp **#ANIM** oder **#SPRITE** ist (voreingestellt ist 1)

#### RÜCKGABEWERTE

- id** optional: Identifikator der Palette; wird nur zurückgegeben, wenn Sie beim Argument **id** **Nil** angegeben haben.

#### BEISPIEL

```
ExtractPalette(1, #BRUSH, 10)
```

Der obige Code extrahiert die Palette aus Pinsel 10 und speichert sie als Palettenobjekt 1.

## 41.8 FreePalette

#### BEZEICHNUNG

FreePalette – löscht eine Palette (V9.0)

#### ÜBERSICHT

```
FreePalette(id)
```

#### BESCHREIBUNG

Dieser Befehl löscht die durch **id** angegebene Palette. Um den Speicherverbrauch zu reduzieren, sollten Sie Paletten löschen, wenn Sie sie nicht mehr benötigen.

#### EINGABEN

- id** Identifikator der zu löschenden Palette



## 41.9 GammaPalette

### BEZEICHNUNG

GammaPalette – verändert die Gammawerte der Palette (V9.0)

### ÜBERSICHT

```
GammaPalette(id, red, green, blue)
```

### BESCHREIBUNG

Mit diesem Befehl können die Gammawerte der Farbkanäle der angegebenen Palette verändert werden. Für jeden Farbkanal müssen Sie einen Fließkommawert übergeben, der die gewünschte Gammaveränderung angibt. Ein Wert von 1,0 bedeutet keine Änderung, ein Wert kleiner als 1,0 verdunkelt den Kanal, ein Wert größer als 1,0 hellt den Kanal auf.

### EINGABEN

<code>id</code>	Palette, dessen Gammawerte verändert werden sollen
<code>red</code>	Gammaveränderung für den roten Kanal
<code>green</code>	Gammaveränderung für den grünen Kanal
<code>blue</code>	Gammaveränderung für den blauen Kanal

### BEISPIEL

```
GammaPalette(1, 1.5, 1.0, 0.5)
```

Der obige Code hellt den roten Kanal auf und verdunkelt den blauen Kanal, während der grüne Farbkanal unverändert bleibt.

## 41.10 GetBestPen

### BEZEICHNUNG

GetBestPen – ermittelt den besten Stift für die Farbe (V9.0)

### ÜBERSICHT

```
pen = GetBestPen(id, color)
```

### BESCHREIBUNG

Dieser Befehl sucht in der in `id` angegebenen Palette nach einem Stift, dessen Farbe am ehesten mit der im Argument `color` angegebenen Farbe übereinstimmt, und gibt diesen Stift zurück. Das Argument `color` muss ein **RGB-Farbe** sein.

### EINGABEN

<code>id</code>	Identifikator der Palette
<code>color</code>	<b>RGB-Farbe</b> , um den passendsten Stift zu finden

### RÜCKGABEWERTE

<code>pen</code>	Stift, der der angegebenen Farbe am ehesten entspricht
------------------	--

### BEISPIEL

```
SetDrawPen(GetBestPen(1, #RED))
```

Der obige Code legt den Stift fest, der dem Rot als Zeichnungsstift am ähnlichsten ist.

## 41.11 GetFreePen

### BEZEICHNUNG

GetFreePen – gibt unbenutzten Stift zurück (V10.0)

### ÜBERSICHT

```
pen = GetFreePen([t])
```

### BESCHREIBUNG

Dieser Befehl versucht, einen nicht verwendeten Stift im aktuell aktiven Palettenbild zu finden und gibt ihn zurück. Wenn alle Stifte verwendet werden, wird -1 zurückgegeben. Standardmäßig durchsucht dieser Befehl die Palettenpixeldaten des aktuellen Displays, sodass er nur funktioniert, wenn es sich bei dem aktuellen Display um ein Display im Palettenmodus handelt. Siehe [Abschnitt 24.16 \[Palettenmodus-Displays\]](#), [Seite 393](#), für Details. Wenn Sie nicht möchten, dass `GetFreePen()` das aktuelle Display verwendet, können Sie das aktive Palettenbild mit dem Befehl `SelectPalette()` auswählen. Beachten Sie jedoch, dass Sie ein Palettenobjekt auswählen müssen, an dem Pixeldaten zugeordnet sind, z.B. einen Palettenpinsel oder ein Paletten-BGPic. Die bloße Auswahl eines Palettenobjekts funktioniert nicht, da freie Stifte natürlich nur ermittelt werden können, wenn Pixeldaten vorhanden sind.

Sie können auch das optionale Tabellenargument `t` verwenden, um das Quellpalettenobjekt anzugeben. Das Tabellenargument unterstützt genau dieselben Argumente wie das optionale Tabellenargument von `GetPen()`. Siehe [Abschnitt 41.13 \[GetPen\]](#), [Seite 851](#), für Details.

### EINGABEN

`t` optional: Tabelle zur Angabe weiterer Optionen (siehe oben)

### RÜCKGABEWERTE

`pen` Index der unbenutzten Stifte oder -1, wenn alle Stifte benutzt werden

## 41.12 GetPalettePen

### BEZEICHNUNG

GetPalettePen – gibt die Stiftfarbe aus der Palette zurück (V9.0)

### ÜBERSICHT

```
color = GetPalettePen(id, pen)
```

### BESCHREIBUNG

Dieser Befehl ruft die Farbe des durch `pen` angegebenen Stifts aus der durch `id` angegebenen Palette ab. Die Farbe wird als **RGB-Farbe** zurückgegeben.

### EINGABEN

`id` ID der zu verwendenden Palette

`pen` Stift, dessen Farbe Sie erhalten möchten (beginnend bei 0)

### RÜCKGABEWERTE

`color` Farbe des Stiftes, angegeben als **RGB-Farbe**

**BEISPIEL**

```
color = GetPalettePen(1, 0)
```

Der Code gibt die Farbe des ersten Stifts in Palette 1 zurück.

## 41.13 GetPen

**BEZEICHNUNG**

GetPen – gibt die Stiftfarbe aus der aktuellen Palette zurück (V9.0)

**ÜBERSICHT**

```
color = GetPen(pen[, t])
```

**BESCHREIBUNG**

Dieser Befehl ruft die Farbe des durch **pen** angegebenen Stifts aus der aktuell aktiven Palette ab. Standardmäßig ist die Palette des aktuellen Displays die aktive Palette, aber natürlich nur, wenn das aktuelle Display ein Palettenmodus-Display ist. Siehe [Abschnitt 24.16 \[Palettenmodus-Displays\]](#), [Seite 393](#), für Details. Mit dem Befehl [SelectPalette\(\)](#) kann eine Palette zur aktuellen Palette werden.

Alternativ können Sie auch **GetPen()** verwenden, um die Stiftfarbe von einem anderen Palettenobjekt abzurufen. Dazu müssen Sie das optionale Tabellenargument **t** an **SetPen()** übergeben und die Tags **Type** und **ID** angeben. Ein Beispiel finden Sie weiter unten.

Die folgenden Tags werden vom optionalen Tabellenargument **t** unterstützt:

**Type:** Setzen Sie dies auf den Typ des Objekts, dessen Palette Sie abfragen möchten. Dies kann einer der folgenden Objekttypen sein:

```
#ANIM
#BGPIC
#BRUSH
#DISPLAY
#LAYER
#PALETTE
#SPRITE
```

Beachten Sie, dass Sie bei der Verwendung der Typen **#ANIM** oder **#SPRITE** auch den Tag **Frame** (siehe unten) setzen müssen, um das Einzelbild anzugeben, dessen Palette Sie abfragen möchten. Wenn Sie **#LAYER** verwenden und die angegebene Ebene eine Animationsebene ist, müssen Sie auch den Tag **Frame** setzen.

**Type** ist standardmäßig der Typ der aktuell aktiven Palette, die mit dem Befehl [SelectPalette\(\)](#) gewählt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.

**ID:** Setzen Sie diesen Tag auf den Identifikator des Objekts, dessen Palette Sie abfragen möchten. Voreingestellt ist die ID der aktuell aktiven Palette, die mit [SelectPalette\(\)](#) gesetzt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.

**Frame:** Wenn der Typ eine Animation, ein Sprite oder eine Animationsebene ist, müssen Sie diesen Tag setzen, um das Einzelbild anzugeben, dessen Palette Sie abfragen möchten. Einzelbilder werden von 1 an gezählt. Bei Verwendung mit Animationen und Sprites wird standardmäßig 1 und mit Animationsebenen das aktuelle Einzelbild verwendet.

#### EINGABEN

**pen** Stift, den Sie erhalten möchten (beginnend bei 0)  
**t** optional: Tabelle zum Festlegen weiterer Optionen (siehe oben)

#### RÜCKGABEWERTE

**color** Farbe des Stifts, angegeben als **RGB-Farbe**

#### BEISPIEL

```
@DISPLAY {Palette = #PALETTE_MONOCHROME}
color0 = GetPen(0)
color1 = GetPen(1)
```

Der obige Code erzeugt ein monochromes Palettenmodus-Display und gibt die Farben der ersten beiden Stifte zurück. **color0** wird schwarz und **color1** wird weiß sein.

```
color = GetPen(4, {Type = #BRUSH, ID = 2})
```

Der Code gibt die Farbe von Stift 4 in Pinsel 2 zurück.

## 41.14 InvertPalette

#### BEZEICHNUNG

InvertPalette – invertiert die Palettenfarben (V9.0)

#### ÜBERSICHT

```
InvertPalette(id)
```

#### BESCHREIBUNG

Dieser Befehl invertiert alle Farben in der durch **id** angegebenen Palette, d.h. alle Farben werden durch ihre Komplemente ersetzt (Weiß wird zu Schwarz, Blau wird zu Gelb usw.).

#### EINGABEN

**id** Palette, die invertiert wird

#### BEISPIEL

```
InvertPalette(1)
```

Der obige Code invertiert die Farben der Palette 1.

## 41.15 LoadPalette

#### BEZEICHNUNG

LoadPalette – lädt eine Palette (V9.0)

## ÜBERSICHT

```
[id] = LoadPalette(id, filename$[, table])
```

## BESCHREIBUNG

Dieser Befehl lädt die durch `filename$` angegebene Palette in den Speicher und weist ihr den Identifikator `id` zu. Wenn Sie in `id` `Nil` übergeben, wählt `LoadPalette()` automatisch eine ID aus und gibt sie zurück.

Die in `filename$` angegebene Palette kann entweder im IFF-ILBM-Palettenformat vorliegen, wie es von Deluxe Paint festgelegt wurde, oder `filename$` kann alternativ auch eine normale Bilddatei sein, die eine Palette enthält. In diesem Fall extrahiert `LoadPalette()` einfach die Palette aus der Bilddatei.

Das dritte Argument ist optional. In dieser Tabelle können weitere Optionen für den Ladevorgang festgelegt werden. Die folgenden Felder der Tabelle können verwendet werden:

**Loader:** Mit diesem Tag können Sie ein oder mehrere Formatlademodule angeben, die diese Palette laden sollen. Dies muss als eine Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Lademodulen enthält. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details.

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen sollen. Dies muss als eine Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Adaptermodulen enthält. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details.

**UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Dieser Befehl gibt es auch als Präprozessor: Verwenden Sie `@PALETTE`, um Paletten vorzuladen.

## EINGABEN

<code>id</code>	ID für die Palette oder <code>Nil</code> für die automatische ID-Zuweisung
<code>filename\$</code>	Datei, welche geladen wird
<code>table</code>	optional: weitere Optionen (siehe oben)

## RÜCKGABEWERTE

<code>id</code>	optional: Identifikator der Palette; wird nur zurückgegeben, wenn Sie beim Argument <code>id</code> <code>Nil</code> angegeben haben.
-----------------	---

## BEISPIEL

```
LoadPalette(1, "DPaint32.pal")
```

Dieser Code lädt "DPaint32.pal" als Palette 1.

## 41.16 ModulatePalette

### BEZEICHNUNG

ModulatePalette – ändert Helligkeit, Sättigung und Farbton der Palette (V9.0)

### ÜBERSICHT

```
ModulatePalette(id, brightness, saturation, hue)
```

### BESCHREIBUNG

Mit diesem Befehl können Sie die Einstellungen für Helligkeit, Sättigung und Farbton der Farben in einer Palette ändern. Für jede Einstellung müssen Sie einen Fließkommawert übergeben, der die gewünschte Änderung beschreibt. Ein Wert von 1,0 bedeutet keine Änderung. Ein Wert kleiner als 1,0 verringert die Helligkeit/Sättigung/den Farbton, während ein Wert größer als 1,0 diese erhöht.

### EINGABEN

<code>id</code>	Identifikator der Palette
<code>brightness</code>	gewünschte Helligkeitskorrektur
<code>saturation</code>	gewünschte Sättigungskorrektur
<code>hue</code>	gewünschte Farbtonkorrektur

### BEISPIEL

```
ModulatePalette(1, 1.0, 2.0, 1.0)
```

Der obige Code erhöht die Sättigung, während Helligkeit und Farbton unberührt bleiben. Das Ergebnis ist eine Palette mit hervorgehobenen Farben, genau wie in einem Cartoon.

## 41.17 PALETTE

### BEZEICHNUNG

PALETTE – definiert eine Palette für die spätere Verwendung (V9.0)

### ÜBERSICHT

```
@PALETTE id[, filename$][, table]
```

### BESCHREIBUNG

Mit dieser Präprozessor-Anweisung kann eine Palette für die spätere Verwendung definiert werden. Die Palette kann entweder aus einer Datei geladen werden, es kann sich um eine der vordefinierten Paletten Hollywoods handeln oder Sie können Ihre eigene Palette definieren, indem Sie eine Reihe von Farben angeben.

Wenn Sie das Argument `filename$` übergeben, wird die Palette aus dieser Datei geladen. Die in `filename$` angegebene Datei kann entweder im von Deluxe Paint festgelegten IFF ILBM-Palettenformat vorliegen, oder alternativ kann `filename$` auch eine normale

Bilddatei sein, die eine Palette enthält. In diesem Fall extrahiert @PALETTE einfach die Palette aus der Bilddatei. Beachten Sie jedoch, dass Hollywood die gesamte Bilddatei mit Ihrem Applet oder Ihrem Programm verknüpft, wenn Sie eine vollständige Bilddatei an @PALETTE übergeben und Ihr Skript in ein Applet oder ein Programm kompilieren, wodurch sich die Größe der Ausgabedatei erhöht. Daher wird empfohlen, nur Paletten-dateien im IFF-ILBM-Format mit @PALETTE zu verwenden, da diese sehr klein sind, da sie keine Bilddaten enthalten.

Wenn Sie das Argument `filename$` nicht übergeben, müssen Sie im optionalen Tabellenargument entweder den Tag `Type` oder `Colors` setzen, um die Palette zu definieren (siehe unten). Folgende Tags können Sie mit dem optionalen Tabellenargument verwenden:

- Type:** Setzen Sie diesen Tag, wenn Sie eine der vordefinierten Paletten von Hollywood verwenden möchten. In diesem Fall müssen Sie `Type` auf den Identifikator der gewünschten eingebauten Palette setzen. Siehe [Abschnitt 41.36 \[SetStandardPalette\]](#), [Seite 872](#), für eine Liste der integrierten Paletten. Wenn Sie diesen Tag setzen, dürfen Sie `filename$` oder `Colors` nicht übergeben.
- Colors:** Mit diesem Tag kann eine benutzerdefinierte Palette definiert werden. Wenn Sie `Colors` festlegen, dürfen Sie `filename$` oder `Type` nicht übergeben. Um eine benutzerdefinierte Palette zu definieren, setzen Sie `Colors` auf eine Tabelle mit den gewünschten Farben für die Palette. Die Palettenfarbtiefe wird automatisch aus der Anzahl der Farben in der Tabelle berechnet. Alternativ können Sie auch den Tag `Depth` setzen, um die Palettenfarbtiefe zu definieren (siehe unten).
- Depth:** Die gewünschte Farbtiefe für die Palette. Diese muss zwischen 1 (= 2 Farben) und 8 (= 256 Farben) liegen. Voreingestellt ist 8. Dies muss nur angegeben werden, wenn auch der Tag `Colors` gesetzt wird. Wenn `Depth` mehr Farben angibt, als Sie in der Tabelle im Tag `Colors` übergeben haben, werden die verbleibenden Farben auf Schwarz initialisiert.
- TransparentPen:** Mit diesem Tag kann der Stift angegeben werden, der in der Palette transparent sein soll. Der Standardwert ist `#NOPEN`, was bedeutet, dass kein Stift transparent gemacht wird. Dies darf nur verwendet werden, wenn auch der Tag `Type` oder `Colors` übergeben wird.
- Link:** Setzen Sie diesen Tag auf `False`, wenn Sie `filename$` beim Kompilieren Ihres Skripts nicht mit Ihrem Programm/Applet verknüpft haben möchten. Dieser Tag ist standardmäßig auf `True` gesetzt. Dies bedeutet, dass die Palette mit Ihrem Programm/Applet verknüpft wird, wenn sich Hollywood im Kompilierungsmodus befindet. Wenn Sie diesen Tag verwenden, müssen Sie auch `filename$` übergeben.
- Loader:** Mit diesem Tag können Sie ein oder mehrere Formatlademodule angeben, die diese Palette laden sollen. Dies muss als eine Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Lademodulen enthält. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. Wenn Sie diesen Tag verwenden, müssen Sie auch `filename$` übergeben.

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen sollen. Dies muss als eine Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Adaptermodulen enthält. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. Wenn Sie diesen Tag verwenden, müssen Sie auch `filename$` übergeben.

**UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Um Paletten zur Laufzeit zu laden oder zu erstellen, sehen Sie sich die Befehle `LoadPalette()` und `CreatePalette()` an.

## EINGABEN

`id` ein Wert, der zur Identifizierung dieser Palette später im Code verwendet wird

`filename$` die Palettendatei, die Sie laden möchten

`table` optional: Tabelle mit weiteren Optionen (siehe oben)

## BEISPIEL

```
@PALETTE 1, "DPaint32.pal"
```

Lädt "DPaint32.pal" als Palette 1 vor.

```
@PALETTE 1, {Colors = {#WHITE, #BLACK, $7777CC, $BBBBBB}}
```

Definiert eine benutzerdefinierte Palette als Palette 1.

```
@PALETTE 1, {Type = #PALETTE_CGA}
```

Definiert die Standard-CGA-Palette als Palette 1.

## 41.18 PaletteToGray

### BEZEICHNUNG

PaletteToGray – wandelt die Palette in Graustufen um (V9.0)

### ÜBERSICHT

PaletteToGray(id)

### BESCHREIBUNG

Dieser Befehl wandelt alle Farben in der durch `id` angegebenen Palette in Grau um.

### EINGABEN

`id` Identifikator der zu konvertierenden Palette



**BEISPIEL**

```
PaletteToGray(1)
```

Wandelt alle Farben in Palette 1 in Grau um.

**41.19 ReadPen****BEZEICHNUNG**

ReadPen – liest einen Stift aus dem Palettenobjekt (V9.0)

**ÜBERSICHT**

```
pen = ReadPen(x, y[, t])
```

**BESCHREIBUNG**

Dieser Befehl liest den Stift an der durch **x** und **y** angegebenen Position aus dem aktuell aktiven Palettenobjekt. Standardmäßig ist das aktuelle Display das aktive Palettenobjekt, aber natürlich nur, wenn das aktuelle Display ein Palettenmodus-Display ist. Siehe [Abschnitt 24.16 \[Palettenmodus-Displays\]](#), [Seite 393](#), für Details. Sie können das aktive Palettenobjekt mit dem Befehl [SelectPalette\(\)](#) festlegen.

Alternativ können Sie auch [ReadPen\(\)](#) verwenden, um einen Stift aus einem anderen Palettenobjekt zu lesen. Dazu müssen Sie das optionale Tabellenargument **t** an [ReadPen\(\)](#) übergeben und die Tags **Type** und **ID** angeben. Siehe unten für ein Beispiel.

Die folgenden Tags werden vom optionalen Tabellenargument **t** unterstützt:

**Type:** Setzen Sie dies auf den Typ des Objekts, dessen Palette Sie abfragen möchten. Dies kann einer der folgenden Objekttypen sein:

```
#ANIM
#BGPIC
#BRUSH
#DISPLAY
#LAYER
#PALETTE
#SPRITE
```

Beachten Sie, dass Sie bei Verwendung der Typen **#ANIM** oder **#SPRITE** auch den Tag **Frame** (siehe unten) setzen müssen, um das Einzelbild anzugeben, dessen Pixeldaten verwendet werden sollen. Wenn Sie **#LAYER** verwenden und die angegebene Ebene eine Animationsebene ist, müssen Sie auch den Tag **Frame** setzen.

**Type** ist standardmäßig der Typ der aktuell aktiven Palette, die mit dem Befehl [SelectPalette\(\)](#) gewählt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.

**ID:** Setzen Sie diesen Tag auf den Identifikator des Objekts, dessen Pixeldaten verwendet werden sollen. Die Standardeinstellung ist die ID des aktuell aktiven Palettenobjekts, der mit [SelectPalette\(\)](#) festgelegt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.

**Frame:** Wenn der Typ eine Animation, ein Sprite oder eine Animationsebene ist, müssen Sie diesen Tag setzen, um das Einzelbild anzugeben, dessen Pixeldaten

ten verwendet werden sollen. Die Einzelbilder werden von 1 an gezählt. Bei Verwendung mit Animationen und Sprites wird standardmäßig 1 und mit Animationsebenen das aktuelle Einzelbild verwendet.

#### EINGABEN

**x**            Position x, von der gelesen wird  
**y**            Position y, von der gelesen wird  
**t**            optional: Tabelle mit weiteren Optionen (siehe oben)

#### RÜCKGABEWERTE

**pen**            Stift an der angegebenen Position

#### BEISPIEL

```
@DISPLAY {Palette = #PALETTE_MONOCHROME}
pen = ReadPen(0, 0)
```

Der obige Code liest den Stift in der oberen linken Ecke des Displays. Dies ist 0, da der Display-Hintergrund standardmäßig mit Stift 0 gefüllt wird.

```
pen = ReadPen(0, 0, {Type = #BRUSH, ID = 2})
```

Der Code liest den Stift in der oberen linken Ecke vom Pinsel 2.

## 41.20 SavePalette

#### BEZEICHNUNG

SavePalette – speichert die Palette in einer Datei (V9.0)

#### ÜBERSICHT

```
SavePalette(id, f$, t))
```

#### BESCHREIBUNG

Dieser Befehl speichert die von **id** angegebene Palette in der durch **f\$** angegebenen Datei. Die Palette wird im von Deluxe Paint festgelegten IFF ILBM-Palettenformat gespeichert.

Ab Hollywood 10.0 akzeptiert **SavePalette()** ein optionales Tabellenargument, mit dem Sie zusätzliche Argumente an den Befehl übergeben können. Die folgenden Tags werden derzeit vom optionalen Tabellenargument unterstützt:

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, die die angegebene Datei speichern sollen. Dies muss als eine Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen enthält. Standardmäßig wird der mit **SetDefaultAdapter()** eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), Seite 96, für Details. (V10.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die

die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Verwenden Sie den Befehl `LoadPalette()` oder die Präprozessor-Anweisung `@PALETTE`, um eine Palette wieder in Hollywood zu laden.

#### EINGABEN

<code>id</code>	Identifikator der zu speichernden Palette
<code>f\$</code>	Dateiname, in welche die Palette gespeichert wird
<code>t</code>	optional: Tabelle mit weiteren Optionen (siehe oben) (V10.0)

## 41.21 SelectPalette

#### BEZEICHNUNG

`SelectPalette` – setzt das aktive Palettenobjekt (V9.0)

#### ÜBERSICHT

`SelectPalette(type, id)`

#### BESCHREIBUNG

Mit diesem Befehl können Sie das Palettenobjekt setzen, welches von Befehlen wie `SetPen()`, `GetPen()` und `SetPalette()` standardmäßig verwendet werden soll. Die Palette, die standardmäßig von diesem Befehl verwendet wird, wird auch als aktive Palette bezeichnet.

Sie müssen den Typ und den Identifikator des Objekts übergeben, dessen Palette zum aktiven Objekt gemacht werden soll. Die folgenden Objekttypen können an das Argument `type` übergeben werden:

```
#ANIM
#BGPIC
#BRUSH
#DISPLAY
#LAYER
#PALETTE
#SPRITE
```

Beachten Sie, dass `SelectPalette()` weder überprüft, ob das übergebene Objekt vorhanden ist, noch ob es eine Palette hat. Aus Performance-Gründen wird dies erst überprüft, wenn Sie einen Befehl aufrufen, der tatsächlich versucht, auf die aktuell aktive Palette zuzugreifen. Somit ist es auch möglich, eine Palette zu aktivieren, die noch nicht existiert.

Standardmäßig ist die Palette des aktuellen Displays die aktive. Wenn das aktuelle Display keine Palette hat und Sie einen Befehl aufrufen, der versucht darauf zuzugreifen, tritt ein Fehler auf. Siehe [Abschnitt 24.16 \[Palettenmodus-Displays\]](#), [Seite 393](#), für Details.

Beachten Sie, dass `EndSelect()` niemals für `SelectPalette()` aufgerufen werden darf. Verwechseln Sie `SelectPalette()` nicht mit Befehlen wie `SelectBrush()` oder `SelectAlphaChannel()`, bei denen Sie `EndSelect()` aufrufen müssen, wenn Sie fertig sind mit ihnen. Dies darf für `SelectPalette()` nicht erfolgen, da nur die Standardpalette

für Befehle wie `SetPen()` festgelegt wird. Sie dürfen also niemals `EndSelect()` für `SelectPalette()` aufrufen.

#### EINGABEN

`type`        Typ des Objekts, dessen Palette aktiv gemacht werden soll  
`id`         Identifikator des Objekts, dessen Palette aktiv gemacht werden soll

#### BEISPIEL

```
SelectPalette(#BRUSH, 1)
```

Der obige Code macht die Palette von Pinsel 1 zur aktiven Palette.

## 41.22 SetBorderPen

#### BEZEICHNUNG

SetBorderPen – setzt den Rahmenstift (V9.0)

#### ÜBERSICHT

```
SetBorderPen(pen)
```

#### BESCHREIBUNG

Dieser Befehl setzt den durch `pen` angegebenen Stift auf den Stift, der zum Zeichnen von Rahmen verwendet wird, wenn der Palettenmodus `#PALETTEMODE_PEN` ist und das aktuelle Ausgabeziel eine Palette ist. Der Palettenmodus kann mit `SetPaletteMode()` eingestellt werden.

Wenn der Palettenmodus auf `#PALETTEMODE_PEN` eingestellt wurde, verwenden alle Hollywood-Befehle, die einen Rahmen zeichnen, nicht die Rahmenfarbe, die mit den Befehlen `SetFormStyle()`, `SetFontStyle()` oder einem `BorderColor`-Tag gesetzt wurde, sondern sie verwenden den Stift, der mit diesem Befehl als Rahmenstift festgelegt wurde.

Siehe [Abschnitt 41.31 \[SetPaletteMode\]](#), [Seite 868](#), für Details.

#### EINGABEN

`pen`         Stift, der zum Zeichnen von Rahmen verwendet werden soll

## 41.23 SetBulletPen

#### BEZEICHNUNG

SetBulletPen – setzt den Stift für die Aufzählungszeichen (V9.0)

#### ÜBERSICHT

```
SetBulletPen(pen)
```

#### BESCHREIBUNG

Wenn sich das Hollywood-Display derzeit im Palettenmodus befindet, können Sie mit diesem Befehl den Stift festlegen, der zum Zeichnen von Aufzählungszeichen verwendet werden soll, wenn Sie `TextOut()` im Listenmodus verwenden. Standardmäßig werden Aufzählungszeichen mit dem aktuellen Zeichnungsstift gezeichnet, der mit `SetDrawPen()`

festgelegt wurde. Wenn Sie möchten, dass sie mit einem anderen Stift gezeichnet werden sollen, können Sie diesen Befehl verwenden.

Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für weitere Informationen zu Aufzählungslisten.

## EINGABEN

**pen**            Stift, mit dem Aufzählungszeichen gezeichnet werden (beginnend bei 0)

## 41.24 SetCycleTable

### BEZEICHNUNG

SetCycleTable – stellt die Farbwechseltabelle ein (V9.0)

### ÜBERSICHT

SetCycleTable(cycle[, t])

### BESCHREIBUNG

Dieser Befehl setzt die Farbwechseltabelle der aktuell aktiven Palette auf die in **cycle** angegebene. Sie müssen in **cycle** eine Tabelle mit Untertabellen übergeben, wobei jede Untertabelle eine Konfiguration eines Farbverlaufseffekts beschreibt. Jede Untertabelle unterstützt die folgenden Tags:

- Low:**            Der Stiftindex, der den Beginn des Farbbereichs markiert.
- High:**          Der Stiftindex, der das Ende des Farbbereichs markiert.
- Rate:**          Die gewünschte Geschwindigkeit des Farbwechseleffekts. Ein Wert von 16384 gibt 60 Bilder pro Sekunde an. Alle anderen Geschwindigkeiten skalieren linear von dieser Basis aus, z.B. ein Wert von 8192 gibt 30 Bilder pro Sekunde an.
- Reverse:**      Wenn dieser Tag auf **True** gesetzt ist, werden die Farben in umgekehrter Reihenfolge durchlaufen. Der Standardwert ist **False**.
- Active:**        Wenn dieser Tag auf **False** gesetzt ist, wird der Farbbereich als inaktiv markiert. Der Standardwert ist **True**.

Standardmäßig kopiert **SetCycleTable()** die Farbwechseltabelle in die Palette des aktuellen Displays, die die standardmäßig aktive Palette ist, aber natürlich nur, wenn das aktuelle Display ein Palettenmodus-Display ist. Siehe [Abschnitt 24.16 \[Palettenmodus-Displays\]](#), [Seite 393](#), für Details. Mit dem Befehl **SelectPalette()** können Sie eine andere Palette als die aktive auswählen.

Alternativ kann die angegebene Farbwechseltabelle auch auf ein anderes Palettenobjekt gesetzt werden. Dazu müssen Sie das optionale Tabellenargument an **SetCyclingTable()** übergeben und die Tags **Type** und **ID** angeben.

Die folgenden Tags werden durch das optionale Tabellenargument **t** unterstützt:

- Type:**            Setzen Sie dies auf den Typ des Objekts, dessen Farbwechseltabelle Sie einstellen möchten. Dies kann einer der folgenden Objekttypen sein:

**#BGPIC**

#BRUSH  
#PALETTE

**Type** ist standardmäßig der Typ der aktuell aktiven Palette, die mit dem Befehl `SelectPalette()` gewählt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.

**ID:** Setzen Sie diesen Tag auf den Identifikator des Objekts, dessen Farbtiefe Sie einstellen möchten. Die Standardeinstellung ist die ID der aktuell aktiven Palette, die mit `SelectPalette()` gesetzt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.

#### EINGABEN

**cycle**      Tabelle mit den Farbzyklusbereichen (siehe oben)  
**t**            optional: Tabelle mit weiteren Optionen (siehe oben)

## 41.25 SetDepth

### BEZEICHNUNG

SetDepth – stellt die Farbtiefe der Palette ein (V9.0)

### ÜBERSICHT

`SetDepth(depth[, t])`

### BESCHREIBUNG

Dieser Befehl setzt die Farbtiefe der aktuell aktiven Palette auf die in **depth** angegebene Farbtiefe. Die Farbtiefe in **depth** muss eine Bit-Tiefe zwischen 1 (= 2 Farben) und 8 (= 256 Farben) haben. Siehe [Abschnitt 41.1 \[Übersicht Paletten\]](#), [Seite 841](#), für Details. Beachten Sie, dass wenn die angegebene Farbtiefe geringer ist als die der an die Palette angehängten Pixeldaten, die Pixeldaten neu zugeordnet werden, um der neuen Farbtiefe zu entsprechen.

Standardmäßig ist die Palette des aktuellen Displays die aktive Palette, aber natürlich nur, wenn das aktuelle Display ein Palettenmodus-Display ist. Siehe [Abschnitt 24.16 \[Palettenmodus-Displays\]](#), [Seite 393](#), für Details. Eine Palette kann mit dem Befehl `SelectPalette()` zur aktiven Palette gemacht werden.

Alternativ kann die angegebene Farbtiefe auch auf ein anderes Palettenobjekt gesetzt werden. Dazu müssen Sie das optionale Tabellenargument an `SetDepth()` übergeben und die Tags **Type** und **ID** angeben. Siehe weiter unten für ein Beispiel.

Die folgenden Tags werden durch das optionale Tabellenargument **t** unterstützt:

**Type:** Setzen Sie dies auf den Typ des Objekts, dessen Farbtiefe Sie einstellen möchten. Dies kann einer der folgenden Objekttypen sein:

#ANIM  
#BGPIC  
#BRUSH  
#DISPLAY  
#LAYER  
#PALETTE

**#SPRITE**

Beachten Sie, dass Sie bei Verwendung der Typen **#ANIM** oder **#SPRITE** auch den Tag **Frame** (siehe unten) setzen müssen, um das Einzelbild anzugeben, dessen Pixeldaten verwendet werden sollen. Wenn Sie **#LAYER** verwenden und die angegebene Ebene eine Animationsebene ist, müssen Sie auch den Tag **Frame** setzen.

**Type** ist standardmäßig der Typ der aktuell aktiven Palette, die mit dem Befehl **SelectPalette()** gewählt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.

- ID:** Setzen Sie diesen Tag auf den Identifikator des Objekts, dessen Farbtiefe Sie einstellen möchten. Die Standardeinstellung ist die ID der aktuell aktiven Palette, die mit **SelectPalette()** ausgewählt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.
- Frame:** Wenn der Typ eine Animation, ein Sprite oder eine Animationsebene ist, müssen Sie diesen Tag setzen, um das Einzelbild anzugeben, dessen Farbtiefe eingestellt werden soll. Die Einzelbilder werden von 1 an gezählt. Bei Animationen und Sprites ist der Standardwert 1, bei Animationsebenen das aktuelle Einzelbild.
- Remap:** Wenn dieser Tag auf **False** gesetzt ist, werden Stifte außerhalb des Bereichs nicht den vorhandenen Stiften neu zugeordnet, sondern einfach auf den im Tag **ClipPen** (siehe unten) angegebenen Stift gesetzt, d.h. es findet keine Neuordnung statt. Beachten Sie, dass **Remap** nur beim Reduzieren von Farben wirksam ist. Wenn die neue Farbtiefe mehr Stifte hat als die alte, hat **Remap** keine Auswirkung. (V10.0)
- ClipPen:** Dies wird nur verwendet, wenn der Tag **Remap** auf **False** gesetzt ist (siehe oben). In diesem Fall werden Stifte außerhalb des Bereichs nicht den vorhandenen Stiften neu zugeordnet, sondern einfach auf den im Tag **ClipPen** angegebenen Stift gesetzt, d.h. es findet keine Neuordnung statt. Beachten Sie, dass **ClipPen** nur beim Reduzieren von Farben wirksam ist. Wenn die neue Farbtiefe mehr Stifte hat als die alte, führt **ClipPen** nichts aus. (V10.0)

**EINGABEN**

- depth** gewünschte neue Farbtiefe der Palette (im Bereich von 1 bis 8)
- t** optional: Tabelle mit weiteren Optionen (siehe oben)

**BEISPIEL**

```
SetDepth(4, {Type = #BRUSH, ID = 2})
```

Der obige Code setzt die Farbtiefe der Palette von Pinsel 2 auf 4 (= 16 Farben).

## 41.26 SetDitherMode

**BEZEICHNUNG**

**SetDitherMode** – stellt den Dithering-Modus ein (V9.0)

**ÜBERSICHT**

`SetDitherMode(mode)`

**BESCHREIBUNG**

Wenn der Palettenmodus mit dem Befehl `SetPaletteMode()` auf `#PALETTEMODE_REMAP` eingestellt ist, was auch die Standardeinstellung ist, können Sie diesen Befehl verwenden, um den zu verwendenden Dithering-Modus zu konfigurieren. Der gewünschte Dithering-Modus muss im Argument `mode` übergeben werden. Dithering kann die Qualität der neu zugeordneten Grafiken erhöhen, ist jedoch langsamer als das Neuordnen von Grafiken ohne Dithering.

Die folgenden Dithering-Modi sind derzeit verfügbar:

`#DITHERMODE_NONE:`

Kein Dithering. Dies ist der Standard-Dithering-Modus.

`#DITHERMODE_FLOYDSTEINBERG:`

Verwendet das Floyd-Steinberg-Dithering.

Siehe [Abschnitt 41.31 \[SetPaletteMode\]](#), [Seite 868](#), für Details.

**EINGABEN**

`mode`            gewünschter Dithering-Modus (siehe oben)

## 41.27 SetDrawPen

**BEZEICHNUNG**

`SetDrawPen` – setzt den Zeichnungsstift (V9.0)

**ÜBERSICHT**

`SetDrawPen(pen)`

**BESCHREIBUNG**

Dieser Befehl setzt den durch `pen` angegebenen Stift auf den Stift, der zum Zeichnen für die Palette des Ausgabeziels (Display, Pinsel usw.) verwendet wird, wenn der Palettenmodus `#PALETTEMODE_PEN` ist. Der Palettenmodus kann mit `SetPaletteMode()` eingestellt werden. Siehe [Abschnitt 41.31 \[SetPaletteMode\]](#), [Seite 868](#), für Details.

Wenn der Palettenmodus auf `#PALETTEMODE_PEN` gesetzt wurde, verwenden die folgenden Befehle den mit `SetDrawPen()` eingestellten Stift anstelle der Farbe, die an sie übergeben wird:

- `Arc()`
- `Box()`
- `Circle()`
- `Cls()`
- `CreateTextObject()`
- `DrawPath()`
- `Ellipse()`
- `Line()`



- `Plot()`
- `Polygon()`
- `Print()`
- `TextOut()`

Beachten Sie, dass im Palettenmodus `#PALETTEMODE_PEN` auch keine Schatten- oder Randeffekte in der Farbe gezeichnet werden, die mit `SetFormStyle()`, `SetFontStyle()` oder den Tags `ShadowColor` oder `BorderColor` festgelegt wurden. Stattdessen werden Schatten und Rahmen mit dem Stift gezeichnet, der über `SetShadowPen()` bzw. `SetBorderPen()` festgelegt wurde.

#### EINGABEN

`pen`            gewünschter Zeichnungsstift; Stifte beginnen bei 0

#### BEISPIEL

```
@DISPLAY {Palette = #PALETTE_DEFAULT}
SetFillStyle(#FILLCOLOR)
SetPaletteMode(#PALETTEMODE_PEN)
SetDrawPen(10)
Box(#CENTER, #CENTER, 320, 240)
```

Der obige Code erstellt ein Display im Palettenmodus und zeichnet dann mit dem Palettenstift 10 ein gefülltes Rechteck in die Mitte des Bildschirms.

## 41.28 SetGradientPalette

#### BEZEICHNUNG

`SetGradientPalette` – erzeugt einen Farbverlauf in der Palette (V9.0)

#### ÜBERSICHT

```
SetGradientPalette(id, startcolor, endcolor)
```

#### BESCHREIBUNG

Dieser Befehl erstellt einen Farbverlauf in der von `id` angegebenen Palette. Der erste Stift wird auf die in `startcolor` angegebene Farbe und der letzte Stift auf die in `endcolor` initialisiert. Alle Stifte zwischen dem ersten und dem letzten Stift werden mit Zwischenfarben gefüllt, so dass das Ergebnis ein gleichmäßiger Farbverlauf zwischen `startcolor` und `endcolor` ist. Je mehr Farben die Palette hat, desto glatter wird der resultierende Farbverlauf. Daher wird empfohlen, die Palettenfarbtiefe auf 8 (= 256 Farben) einzustellen, um das beste Ergebnis zu erzielen.

#### EINGABEN

`id`            Identifikator der zu verwendenden Palette

`startcolor`            Startfarbe des Farbverlaufs

`endcolor`       Endfarbe des Farbverlaufs

#### BEISPIEL

```
@DISPLAY {Palette = #PALETTE_DEFAULT, Height = 512}
```

```

CreatePalette(1)
SetGradientPalette(1, #BLACK, #BLUE)
SetPaletteMode(#PALETTE_MODE_PEN)
SetPalette(1)
SetFillStyle(#FILL_COLOR)
For Local y = 0 To 255
    SetDrawPen(y)
    Box(0, y * 2, 640, 2)
Next

```

Der obige Code erzeugt in Palette 1 einen Farbverlauf zwischen Schwarz und Blau und zeichnet ihn.

## 41.29 SetPalette

### BEZEICHNUNG

SetPalette – wechselt die Palette (V9.0)

### ÜBERSICHT

SetPalette(id[, t])

### BESCHREIBUNG

Dieser Befehl ersetzt alle Stifte in der aktuell aktiven Palette durch die Stifte aus der in *id* angegebenen Palette. Standardmäßig ist die Palette des aktuellen Displays die aktive Palette, aber natürlich nur, wenn das aktuelle Display ein Palettenmodus-Display ist. Siehe [Abschnitt 24.16 \[Palettenmodus-Displays\]](#), [Seite 393](#), für Details. Mit dem Befehl [SelectPalette\(\)](#) kann eine Palette aktiviert werden.

Alternativ kann die in *id* angegebene Palette auch auf andere Objekte kopiert werden. Dazu müssen Sie das optionale Tabellenargument an [SetPalette\(\)](#) übergeben und den Typ im Tag *Type* und den Identifikator im Tag *ID* festlegen. Gehen Sie beispielsweise wie folgt vor, um Pinsel 2 die Palette 1 zuzuweisen:

```
SetPalette(1, {Type = #BRUSH, ID = 2})
```

Die folgenden Tags werden durch das optionale Tabellenargument *t* unterstützt:

**Type:** Stellen Sie dies auf den Identifikator des Objekts ein, in das Sie die Palette kopieren möchten. Dies kann einer der folgenden Objekttypen sein:

```

#ANIM
#BGPIC
#BRUSH
#DISPLAY
#LAYER
#PALETTE
#SPRITE

```

Beachten Sie, dass Sie bei Verwendung der Typen *#ANIM* oder *#SPRITE* auch den Tag *Frame* (siehe unten) setzen müssen, um das Einzelbild anzugeben, dessen Pixeldaten verwendet werden sollen. Wenn Sie *#LAYER* verwenden und die angegebene Ebene eine Animationsebene ist, müssen Sie auch den Tag *Frame* setzen.

Type ist standardmäßig der Typ der aktuell aktiven Palette, die mit dem Befehl `SelectPalette()` gewählt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.

**ID:** Setzen Sie diesen Tag auf den Identifikator des Objekts, in das Sie die Palette kopieren möchten. Die Standardeinstellung ist die ID der aktuell aktiven Palette, die mit `SelectPalette()` ausgewählt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.

**Frame:** Wenn die Palette in eine Animation, ein Sprite oder eine Animationsebene kopiert werden soll, müssen Sie diesen Tag setzen, um das Einzelbild anzugeben, in das die Palette kopiert werden soll. Die Einzelbilder werden von 1 an gezählt. Bei der Verwendung mit Animationen und Sprites wird standardmäßig 1 und mit Animationsebenen das aktuelle Einzelbild verwendet.

**Remap:** Wenn dies auf `True` gesetzt ist, werden die Farben des Zielobjekts neu zugeordnet, um den Farben in der Quellpalette so genau wie möglich zu entsprechen. Standardmäßig erfolgt keine Neuordnung und die tatsächlichen Pixeldaten des Zielobjekts bleiben unberührt. Wenn Sie eine Neuordnung wünschen, setzen Sie diesen Tag auf `True`. Beachten Sie jedoch, dass die Neuordnung aller Pixel natürlich viel länger dauert als das Festlegen einer neuen Palette ohne Neuordnung. Der Standardwert ist `False`.

**Dither:** Wenn der Tag `Remap` (siehe oben) auf `True` gesetzt wurde, können Sie mit dem Tag `Dither` angeben, ob Dithering verwendet werden soll oder nicht. Der Standardwert ist `True`, was bedeutet, Dithering wird verwendet.

**CopyCycleTable:**

Paletten können eine Tabelle mit Informationen zum Farbwechsel enthalten. Wenn Sie diesen Tag auf `True` setzen, wird diese Wechseltabelle auch in das Zielobjekt kopiert. Der Standardwert ist `False`.

## EINGABEN

**id** id der Palette, welche gesetzt werden soll  
**t** optional: Tabelle mit weiteren Optionen (siehe oben)

## BEISPIEL

```
@DISPLAY {Palette = #PALETTE_MONOCHROME}
SetFillStyle(#FILLCOLOR)
SetPaletteMode(#PALETTEMODE_PEN)
SetDrawPen(1)
Box(#CENTER, #CENTER, 320, 240)
WaitLeftMouse
CreatePalette(1, {#WHITE, #BLACK}, {Depth = 1})
SetPalette(1)
```

Der obige Code erstellt eine monochrome Palettendarstellung mit einem schwarzen Hintergrund und einem weißen Rechteck in der Mitte. Nach einem Mausklick werden die Farben des Hintergrunds und des weißen Rechtecks vertauscht, indem eine neue Palette festgelegt wird, die in Stift 0 Weiß anstelle von Schwarz und in Stift 1 Schwarz anstelle von Weiß verwendet.

## 41.30 SetPaletteDepth

### BEZEICHNUNG

SetPaletteDepth – setzt die Farbtiefe der Palette (V9.0)

### ÜBERSICHT

SetPaletteDepth(id, depth)

### BESCHREIBUNG

Dieser Befehl setzt die Farbtiefe der in `id` angegebenen Palette auf die in `depth` angegebene. Die Farbtiefe muss eine Bittiefe zwischen 1 (= 2 Farben) und 8 (= 256 Farben) haben. Siehe [Abschnitt 41.1 \[Übersicht über die Paletten\]](#), [Seite 841](#), für Details.

### EINGABEN

`id`            Identifikator der zu ändernden Palette  
`depth`        gewünschte neue Farbtiefe der Paletten (im Bereich von 1 bis 8)

### BEISPIEL

SetPaletteDepth(1, 8)

Der obige Code ändert bei der Palette 1 die Farbtiefe auf 8 (= 256 Farben).

## 41.31 SetPaletteMode

### BEZEICHNUNG

SetPaletteMode – setzt den Palettenzeichnungsmodus (V9.0)

### ÜBERSICHT

SetPaletteMode(mode)

### BESCHREIBUNG

Dieser Befehl setzt den Palettenzeichnungsmodus auf den im Argument `mode` angegebenen. Dieser Modus wird immer dann verwendet, wenn die Ausgabe palettenbasiert ist, z.B. ein Palettenmodus-Display oder ein Palettenpinsel.

Die folgenden Modi werden derzeit unterstützt:

#### #PALETTEMODE\_REMAP:

Alle gezeichneten Grafiken werden der Palette des Ausgabeziels (z.B. Display, Pinsel usw.) neu zugeordnet. Dies ist der Standard-Palettenmodus. Aber seien Sie gewarnt, dass dies sehr langsam werden kann, da Hollywood für jedes einzelne gezeichnete Pixel die nächstgelegene Stiftübereinstimmung finden muss. Um die beste Zeichenleistung im Palettenmodus zu erzielen, sollten Sie stattdessen `#PALETTEMODE_PEN` verwenden (siehe unten). Die Art und Weise, wie Grafikdaten bei einer Palette des Ausgabeziels neu zugeordnet werden, kann durch den Befehl [SetDitherMode\(\)](#) konfiguriert werden. Auf diese Weise können Sie das Dithering aktivieren oder deaktivieren und den zu verwendenden Dithering-Algorithmus angeben. Beachten Sie, dass bei Verwendung von `#PALETTEMODE_REMAP` einfarbige Zeichenfunktionen wie [Box\(\)](#), [Circle\(\)](#) oder [TextOut\(\)](#) nicht mit dem Zeichnungsstift über [SetDrawPen\(\)](#) gezeichnet werden, sondern mit der RGB-Farbe, die an diese Befehle übergeben wird.

**#PALETTEMODE\_PEN:**

Bei Verwendung von **#PALETTEMODE\_PEN** werden alle Palettengrafiken ohne Pixel-Neuzuordnung auf das Ausgabeziel kopiert. Dies ist sehr schnell, erfordert jedoch, dass die Palette des zu zeichnenden Grafikobjekts und die Palette des Ausgabeziels identisch sind, da das Ergebnis sonst Farbverfälschungen aufweist. Wenn Sie also **#PALETTEMODE\_PEN** verwenden, sollten Sie sicherstellen, dass alle Ihre Grafikobjekte dieselbe Palette verwenden. Wenn **#PALETTEMODE\_PEN** aktiv ist, werden alle einfarbigen Zeichenfunktionen wie **Box()**, **Circle()** und **TextOut()** nicht in der RGB-Farbe zeichnen, die Sie an diese Befehle übergeben haben, sondern alle verwenden den Zeichnungsstift, der mit **SetDrawPen()** festgelegt wurde.

Gleiches gilt für die Schatten- und Rahmenfarbe: Wenn der Palettenmodus auf **#PALETTEMODE\_PEN** gesetzt ist, verwenden alle Grafikfunktionen, die Schatten und Rahmen unterstützen, nicht die in **SetFormStyle()**, **SetFontStyle()** oder im Standard-Zeichen-Tag wie **ShadowColor** angegebene Farbe. Sondern sie verwenden die Stifte, die mit Befehlen wie **SetShadowPen()**, **SetBorderPen()** oder Zeichen-Tags wie **ShadowPen** und **BorderPen** angegeben wurden.

Außerdem wird das Antialiasing von Text- und Grafikgrundelemente deaktiviert, wenn **#PALETTEMODE\_PEN** aktiv ist, da Paletten in den meisten Fällen nicht genügend Farben für ein zufriedenstellendes Antialiasing von Kanten haben.

Beachten Sie jedoch, dass RGB-Grafiken auch dann noch neu zugeordnet werden müssen, wenn **#PALETTEMODE\_PEN** aktiv ist, da es offensichtlich unmöglich ist, RGB-Grafiken auf ein Palettenausgabeziel zu zeichnen, ohne die RGB-Farben den Palettenstiften neu zuzuordnen. Daher sollte das Zeichnen von 32-Bit-Echtfarbgrafiken auf Palettenausgabeziele vermieden werden, da dies immer langsam ist, weil für diese Grafiken eine Neuordnung vorgenommen werden muss und es keine Möglichkeit gibt, dies zu umgehen.

Der Standardzeichnungsmodus ist **#PALETTEMODE\_REMAP**. Aus Leistungsgründen wird jedoch empfohlen, **#PALETTEMODE\_PEN** zu verwenden. Siehe [Abschnitt 24.16 \[Palettenmodus-Displays\]](#), [Seite 393](#), für Details.

**EINGABEN**

mode      gewünschter Palettenzeichnungsmodus (siehe oben)

**41.32 SetPalettePen****BEZEICHNUNG**

SetPalettePen – wechselt den Palettenstift (V9.0)

**ÜBERSICHT**

SetPalettePen(id, pen, color)

**BESCHREIBUNG**

Dieser Befehl setzt die Farbe des durch `pen` angegebenen Stifts auf die durch `color` gesetzte Farbe in der durch `id` angegebenen Palette.

**EINGABEN**

`id`            Identifikator der Palette

`pen`           Stift, den Sie ändern möchten (beginnend bei 0)

`color`        neue Farbe für den Stift, muss als **RGB-Farbe** angegeben werden.

**BEISPIEL**

```
SetPalettePen(1, 0, #RED)
```

Mit dem obigen Code wird Stift 0 in der Palette 1 auf Rot gesetzt.

### 41.33 SetPaletteTransparentPen

**BEZEICHNUNG**

SetPaletteTransparentPen – stellt den transparenten Stift der Palette ein (V9.0)

**ÜBERSICHT**

```
SetPaletteTransparentPen(id, pen)
```

**BESCHREIBUNG**

Dieser Befehl setzt den transparenten Stift der durch `id` angegebenen Palette auf den in `pen` angegebenen Stift. Stifte werden von 0 an gezählt.

**EINGABEN**

`id`            Identifikator der zu verwendenden Palette

`pen`           gewünschter transparenter Stift (beginnend bei 0)

**BEISPIEL**

```
SetPaletteTransparentPen(1, 4)
```

Der Code macht Stift 4 in Palette 1 transparent.

### 41.34 SetPen

**BEZEICHNUNG**

SetPen – ändert die Stiftfarbe (V9.0)

**ÜBERSICHT**

```
SetPen(pen, color[, t])
```

**BESCHREIBUNG**

Dieser Befehl setzt die Farbe des durch `pen` angegebenen Stifts auf die durch `color` angegebene Farbe. Die Änderung wird in der aktuell aktiven Palette vorgenommen. Standardmäßig ist die Palette des aktuellen Displays die aktive Palette, aber natürlich nur für den Fall, wenn das aktuelle Display im Palettenmodus ist. Siehe **Abschnitt 24.16**

[[Palettenmodus-Displays](#)], [Seite 393](#), für Details. Eine Palette kann zur aktiven Palette gemacht werden, indem der Befehl `SelectPalette()` verwendet wird.

Alternativ können Sie `SetPen()` auch dazu bringen, Stifte in einem anderen Palettenobjekt zu ändern. Dazu müssen Sie das optionale Tabellenargument `t` an `SetPen()` übergeben und die Tags `Type` und `ID` angeben. Ein Beispiel finden Sie weiter unten.

Die folgenden Tags werden durch das optionale Tabellenargument `t` unterstützt:

**Type:** Setzen Sie dies auf die Typkennung des Objekts, dessen Stift Sie ändern möchten. Dies kann einer der folgenden Objekttypen sein:

```
#ANIM
#BGPIC
#BRUSH
#DISPLAY
#LAYER
#PALETTE
#SPRITE
```

Beachten Sie, dass Sie bei Verwendung der Typen `#ANIM` oder `#SPRITE` auch den Tag `Frame` (siehe unten) setzen müssen, um das Einzelbild anzugeben, dessen Stift Sie ändern möchten. Wenn Sie `#LAYER` verwenden und die angegebene Ebene eine Animationsebene ist, müssen Sie auch den Tag `Frame` setzen.

`Type` ist standardmäßig der Typ der aktuell aktiven Palette, die mit dem Befehl `SelectPalette()` gewählt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.

**ID:** Setzen Sie diesen Tag auf den Identifikator des Objekts, dessen Stift Sie ändern möchten. Die Standardeinstellung ist die ID der aktuell aktiven Palette, die mit `SelectPalette()` ausgewählt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.

**Frame:** Wenn der Typ eine Animation, ein Sprite oder eine Animationsebene ist, müssen Sie diesen Tag setzen, um das Einzelbild anzugeben, dessen Stift Sie ändern möchten. Die Einzelbilder werden von 1 an gezählt. Bei Verwendung mit Animationen und Sprites wird standardmäßig 1 und bei Verwendung mit Animationsebenen das aktuelle Einzelbild verwendet.

## EINGABEN

<code>pen</code>	Stift, den Sie ändern möchten (beginnend bei 0)
<code>color</code>	neue Farbe für den Stift; muss als <b>RGB-Farbe</b> angegeben werden
<code>t</code>	optional: Tabelle mit weiteren Optionen (siehe oben)

## BEISPIEL

```
@DISPLAY {Palette = #PALETTE_MONOCHROME}
SetFillStyle(#FILLCOLOR)
SetPaletteMode(#PALETTEMODE_PEN)
SetDrawPen(1)
Box(#CENTER, #CENTER, 320, 240)
```

```
WaitLeftMouse
SetPen(0, #WHITE)
SetPen(1, #BLACK)
```

Der obige Code erstellt ein monochromes Palettenmodus-Display mit einem schwarzen Hintergrund und einem weißen Rechteck in der Mitte. Nach einem Mausklick werden die Farben des Hintergrunds und des weißen Rechtecks vertauscht, indem Stift 0 auf Weiß und Stift 1 auf Schwarz gesetzt wird.

```
SetPen(4, #RED, {Type = #BRUSH, ID = 2})
```

Der Code ändert den Stift 4 im Pinsel 2 auf rot.

## 41.35 SetShadowPen

### BEZEICHNUNG

SetShadowPen – stellt den Schattenstift ein (V9.0)

### ÜBERSICHT

```
SetShadowPen(pen)
```

### BESCHREIBUNG

Dieser Befehl setzt den durch **pen** angegebenen Stift auf den Stift, der zum Zeichnen von Schatten verwendet wird, wenn der Palettenmodus **#PALETTEMODE\_PEN** ist und das aktuelle Ausgabeziel ein Palettenmodus ist. Der Palettenmodus kann mit **SetPaletteMode()** eingestellt werden.

Wenn der Palettenmodus auf **#PALETTEMODE\_PEN** eingestellt ist, verwenden alle Hollywood-Befehle, die einen Schatten zeichnen, nicht die Schattenfarbe, die mit den Befehlen **SetFormStyle()**, **SetFontStyle()** oder dem Tag **ShadowColor** festgelegt wurde. Aber sie verwenden den Stift, der mit diesem Befehl als Schattenstift festgelegt wurde. Siehe [Abschnitt 41.31 \[SetPaletteMode\]](#), [Seite 868](#), für Details.

### EINGABEN

**pen**            Stift für das Zeichnen von Schatten

## 41.36 SetStandardPalette

### BEZEICHNUNG

SetStandardPalette – kopiert die Farben aus der Standardpalette (V9.0)

### ÜBERSICHT

```
SetStandardPalette(id, type)
```

### BESCHREIBUNG

Mit diesem Befehl können die Farben aus der durch **type** angegebenen Standardpalette in die durch **id** angegebene Palette kopiert werden. Beachten Sie, dass dies die Farbtiefe der von **id** angegebenen Palette ändern kann, da die Farbtiefe der Standardpalette auch in die von **id** angegebene Palette kopiert wird.



Die folgenden Standardpaletten sind derzeit verfügbar:

**#PALETTE\_MONOCHROME:**  
Zweifarbige Schwarz-Weiß-Palette.

**#PALETTE\_GRAY4:**  
4-Farben-Graustufen-Palette.

**#PALETTE\_GRAY8:**  
8-Farben-Graustufen-Palette.

**#PALETTE\_GRAY16:**  
16-Farben-Graustufen-Palette.

**#PALETTE\_GRAY32:**  
32-Farben-Graustufen-Palette.

**#PALETTE\_GRAY64:**  
64-Farben-Graustufen-Palette.

**#PALETTE\_GRAY128:**  
128-Farben-Graustufen-Palette.

**#PALETTE\_GRAY256:**  
256-Farben-Graustufen-Palette.

**#PALETTE\_CGA:**  
Standard-CGA-Palette (16 Farben).

**#PALETTE\_OCS:**  
Standard-OCS-Palette (32 Farben).

**#PALETTE\_EGA:**  
Standard-EGA-Palette (64 Farben).

**#PALETTE\_AGA:**  
Standard-AGA-Palette (256 Farben).

**#PALETTE\_WORKBENCH:**  
Standard-Palette der klassischen Amiga Workbench (256 Farben).

**#PALETTE\_MACINTOSH:**  
Standardmäßige klassische Macintosh-Palette (256 Farben).

**#PALETTE\_WINDOWS:**  
Standardmäßige klassische Windows-Palette (256 Farben).

**#PALETTE\_DEFAULT:**  
Gleich wie **#PALETTE\_AGA**.

## EINGABEN

**id**            Identifikator der zu verwendenden Palette

**type**          gewünschte Standardpalette zum Kopieren in die Zielpalette

## BEISPIEL

```
SetStandardPalette(1, #PALETTE_EGA)
```

Der obige Code kopiert die Standard-EGA-Palette in die Palette 1. Die neue Farbtiefe der Palette 1 wird nach der Operation 6 (= 64 Farben) sein.

## 41.37 SetTransparentPen

### BEZEICHNUNG

SetTransparentPen – setzt den transparenten Stift (V9.0)

### ÜBERSICHT

SetTransparentPen(pen[, t])

### BESCHREIBUNG

Dieser Befehl setzt den transparenten Stift in der aktuell aktiven Palette auf den in **pen** angegebenen. Stifte werden ab 0 gezählt. Standardmäßig ist die Palette des aktuellen Displays die aktive Palette, aber natürlich nur, wenn das aktuelle Display ein Palettenmodus-Display ist. Siehe [Abschnitt 24.16 \[Palettenmodus-Displays\]](#), [Seite 393](#), für Details. Mit dem Befehl [SelectPalette\(\)](#) kann eine Palette aktiviert werden.

Alternativ kann der angegebene transparente Stift auch auf ein anderes Palettenobjekt gesetzt werden. Dazu müssen Sie das optionale Tabellenargument an [SetTransparentPen\(\)](#) übergeben und die Tags **Type** und **ID** angeben. Ein Beispiel finden Sie weiter unten.

Die folgenden Tags werden durch das optionale Tabellenargument **t** unterstützt:

**Type:** Setzen Sie dies auf den Typ des Objekts, dessen transparenten Stift Sie einstellen möchten. Dies kann einer der folgenden Objekttypen sein:

```
#ANIM
#BGPIC
#BRUSH
#DISPLAY
#LAYER
#PALETTE
#SPRITE
```

Beachten Sie, dass Sie bei der Verwendung der Typen **#ANIM** oder **#SPRITE** auch den Tag **Frame** (siehe unten) setzen müssen, um das Einzelbild anzugeben, dessen transparenten Stift Sie ändern möchten. Wenn Sie **#LAYER** verwenden und die angegebene Ebene eine Animationsebene ist, müssen Sie auch den Tag **Frame** setzen.

**Type** ist standardmäßig der Typ der aktuell aktiven Palette, die mit [SelectPalette\(\)](#) ausgewählt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.

**ID:** Setzen Sie dieses Tag auf den Identifikator des Objekts, dessen transparenter Stift Sie festlegen möchten. Die Standardeinstellung ist die ID der aktuell aktiven Palette, die mit [SelectPalette\(\)](#) ausgewählt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.

**Frame:** Wenn der Typ eine Animation, ein Sprite oder eine Animationsebene ist, müssen Sie diesen Tag setzen, um das Einzelbild anzugeben, dessen trans-

parenter Stift festgelegt werden soll. Einzelbilder werden von 1 an gezählt. Bei Verwendung mit Animationen und Sprites wird standardmäßig 1 und bei Verwendung mit Animationsebenen das aktuelle Einzelbild verwendet.

**EINGABEN**

**pen**            gewünschter transparenter Stift (beginnend bei 0)  
**t**               optional: Tabelle mit weiteren Optionen (siehe oben)

**BEISPIEL**

```
SetTransparentPen(4, {Type = #BRUSH, ID = 2})
```

Der Code macht Stift 4 in Pinsel 2 transparent.

## 41.38 SetTransparentThreshold

**BEZEICHNUNG**

SetTransparentThreshold – stellt den Schwellenwert für Alpha-Mapping ein (V9.0)

**ÜBERSICHT**

```
SetTransparentThreshold(threshold)
```

**BESCHREIBUNG**

Mit diesem Befehl kann ein Schwellenwert zwischen 0 und 255 angegeben werden, der bei der Quantifizierung der Alpha-Transparenz in Monochrom-Transparenz verwendet werden soll. Wenn Sie beispielsweise einen RGB-Pinsel mit Alpha-Transparenz auf ein Palettenmodus-Display zeichnen, muss Hollywood entscheiden, welche Pixel neu zugeordnet und auf das Display gezeichnet und welche Pixel ignoriert werden sollen, da sie (teilweise) transparent sind.

Alle Pixel, deren Alphawert kleiner oder gleich dem in **threshold** angegebenen Schwellenwert ist, werden als transparent betrachtet. Der Standardwert ist 0, was bedeutet, dass nur vollständig unsichtbare Pixel als transparent betrachtet werden. Je nach den tatsächlichen Bilddaten, die Sie quantisieren möchten, kann es jedoch erforderlich sein, hier einen anderen Schwellenwert zu wählen.

Hier gibt es keinen besten Schwellenwert für eine "Einheitsgröße, passt für alle". Es hängt alles von den Quellbilddaten ab, die Sie quantisieren möchten. Manchmal möchten Sie möglicherweise teilweise transparente Pixel im Zielbild haben, manchmal nicht. Aus diesem Grund kann es erforderlich sein, diesen Befehl mit unterschiedlichen Schwellenwerten aufzurufen, je nachdem, welches Bild tatsächlich quantisiert werden muss.

**EINGABEN**

**threshold**  
                  gewünschter Schwellenwert für transparente Pixel (muss zwischen 0 und 255 liegen); der Standardwert ist 0

## 41.39 SolarizePalette

**BEZEICHNUNG**

SolarizePalette – wendet den Solarisationseffekt auf die Palette an (V9.0)

**ÜBERSICHT**

```
SolarizePalette(id, level)
```

**BESCHREIBUNG**

Mit diesem Befehl können Sie einen Solarisierungseffekt auf die angegebene Palette anwenden. Der Solarisationseffekt versucht, das Aussehen eines fotografischen Films zu simulieren, der Licht ausgesetzt wurde. Das zweite Argument steuert die Intensität des Solarisationseffekts und kann ein beliebiger Wert zwischen 0 und 255 oder eine prozentuale Angabe innerhalb einer Zeichenkette sein.

**EINGABEN**

<code>id</code>	Palette zum Solarisieren
<code>level</code>	gewünschter Solarisierungsgrad (0 bis 255, oder eine Zeichenkette mit einer Prozentangabe)

**41.40 TintPalette****BEZEICHNUNG**

TintPalette – tönt die Palette (V9.0)

**ÜBERSICHT**

```
TintPalette(id, color, level)
```

**BESCHREIBUNG**

Dieser Befehl färbt alle Farben in der von `id` angegebenen Palette mit dem in `color` angegebenen **RGB-Farbe** unter Verwendung der in `level` angegebenen Farbtonstufe. Das Argument `level` muss zwischen 0 (keine Tönung) und 255 (vollständige Tönung) liegen. Alternativ kann `level` auch eine Zeichenkette sein, die eine prozentuale Angabe enthält, z.B. "50%".

**EINGABEN**

<code>id</code>	Identifikator der zu färbenden Palette
<code>color</code>	eine <b>RGB-Farbe</b> , die für die Tönung verwendet wird
<code>level</code>	Farbtonstufe (0 bis 255 oder Prozentangabe)

**BEISPIEL**

```
TintPalette(1, #RED, 128)
```

Der obige Code fügt allen Farben in Palette 1 etwas Rot hinzu.

**41.41 WritePen****BEZEICHNUNG**

WritePen – schreibt den Stift in das Palettenobjekt (V9.0)

**ÜBERSICHT**

```
WritePen(x, y, pen[, t])
```

**BESCHREIBUNG**

Dieser Befehl schreibt den durch **pen** angegebenen Stift an die durch **x** und **y** angegebene Position im aktuell aktiven Palettenobjekt. Standardmäßig ist das aktuelle Display das aktive Palettenobjekt, aber natürlich nur, wenn das aktuelle Display ein Palettenmodus-Display ist. Siehe [Abschnitt 24.16 \[Palettenmodus-Displays\]](#), [Seite 393](#), für Details. Sie können das aktive Palettenobjekt mit dem Befehl [SelectPalette\(\)](#) festlegen.

Alternativ können Sie auch [WritePen\(\)](#) verwenden, um einen Stift in ein anderes Palettenobjekt zu schreiben. Dazu müssen Sie das optionale Tabellenargument **t** an [WritePen\(\)](#) übergeben und die Tags **Type** und **ID** angeben. Siehe unten für ein Beispiel.

Die folgenden Tags werden durch das optionale Tabellenargument **t** unterstützt:

**Type:** Setzen Sie dies auf den Typ des Objekts, in dessen Pixeldaten Sie schreiben möchten. Dies kann einer der folgenden Objekttypen sein:

```
#ANIM
#BGPIC
#BRUSH
#DISPLAY
#LAYER
#PALETTE
#SPRITE
```

Beachten Sie, dass Sie bei Verwendung der Typen **#ANIM** oder **#SPRITE** auch den Tag **Frame** (siehe unten) setzen müssen, um das Einzelbild anzugeben, dessen Pixeldaten verwendet werden sollen. Wenn Sie **#LAYER** verwenden und die angegebene Ebene eine Animationsebene ist, müssen Sie auch den Tag **Frame** setzen.

**Type** ist standardmäßig der Typ des aktuell aktiven Palettenobjekts, welches mit [SelectPalette\(\)](#) eingestellt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.

**ID:** Setzen Sie diesen Tag auf den Identifikator des Objekts, dessen Pixeldaten verwendet werden sollen. Die Standardeinstellung ist die ID des aktuell aktiven Palettenobjekts, welches mit [SelectPalette\(\)](#) festgelegt wurde. Siehe [Abschnitt 41.21 \[SelectPalette\]](#), [Seite 859](#), für Details.

**Frame:** Wenn der Typ eine Animation, ein Sprite oder eine Animationsebene ist, müssen Sie diesen Tag setzen, um das Einzelbild anzugeben, dessen Pixeldaten verwendet werden sollen. Einzelbilder werden von 1 an gezählt. Bei Verwendung mit Animationen und Sprites wird standardmäßig 1 und bei Verwendung mit Animationsebenen des aktuellen Einzelbildes verwendet.

**EINGABEN**

<b>x</b>	x-Position zum Schreiben
<b>y</b>	y-Position zum Schreiben
<b>pen</b>	der Stift zum Schreiben
<b>t</b>	optional: Tabelle mit weiteren Optionen (siehe oben)

**BEISPIEL**

```
WritePen(0, 0, 10, {Type = #BRUSH, ID = 2})
```

Der Code zeichnet ein Pixel, welches Stift 10 verwendet, in Pinsel 2 in der oberen linken Ecke.

## 42 Piktogramm-Bibliothek

### 42.1 AddIconImage

#### BEZEICHNUNG

`idx = AddIconImage` – fügt ein Bild zum Piktogramm hinzu (V8.0)

#### ÜBERSICHT

`AddIconImage(id, table)`

#### BESCHREIBUNG

Mit diesem Befehl können Sie dem mit `id` angegebenen Piktogramm ein neues Bild hinzufügen. Das Bild muss im Parameter `table` angegeben werden, der auf eine Tabelle gesetzt werden muss, die die folgenden Felder verwendet:

**Type:** Mit diesem Tag können Sie den Quelltyp des Bildes festlegen, das Sie zum Piktogramm hinzufügen möchten. Dies kann `#BRUSH` sein, wenn Sie einen Pinsel hinzufügen wollen oder `#FILE`, wenn Sie ein Bild aus einer externen Dateiquelle hinzufügen möchten. Der Standardwert ist `#BRUSH`. Beachten Sie, dass sich diese Standardeinstellung von der Standardeinstellung der Präprozessor-Anweisung `@ICON` unterscheidet. Da ist `#FILE` voreingestellt.

**Image:** Dieser Tag gibt die tatsächliche Bildquelle an und muss festgelegt werden. Wenn **Type** auf `#BRUSH` gesetzt wurde, müssen Sie diesen Tag auf den Identifikator eines Pinsels setzen, den Sie zum Piktogramm hinzufügen möchten. Andernfalls muss **Image** auf den Pfad einer Bilddatei gesetzt werden. Die Bilddatei kann in einem der von Hollywood unterstützten Bilddateiformate sein. Wenn die hier angegebene Bilddatei einen Alphakanal aufweist, werden die Alphakanaldaten automatisch geladen. Beachten Sie auch, dass in jedem Piktogramm jede Bildgröße nur einmal verwendet werden kann, d.h. es ist nicht möglich, zwei 48x48-Bilder zu einem einzelnen Piktogramm hinzuzufügen. Beachten Sie auch, dass es nicht möglich ist, einem Piktogramm ein Vektorgrafikbild hinzuzufügen, da Vektorpiktogramme nur ein einzelnes Bild enthalten dürfen. Siehe [Abschnitt 42.3 \[CreateIcon\]](#), [Seite 881](#), für Details.

**SelImage:**

Mit diesem Tag können Sie ein zusätzliches Bild einfügen, das als ausgewählte Version des in **Image** angegebenen Bildes verwendet werden soll. Dieser Tag ist optional. Wenn Sie ihn einstellen, muss das hier angegebene Bild genau dieselbe Größe wie das in **Image** angegebene haben. Darüber hinaus wird **SelImage** auf die gleiche Weise wie **Image** verwendet, d.h. es hängt von dem in **Type** eingestellten Bildtyp ab, was Sie hier übergeben müssen. Entweder ist dies ein Pinselidentifikator oder einen Pfad zu einer externen Bilddatei.

**Standard:**

Mit diesem Tag können Sie das Bild festlegen, das als Standardgröße für das Piktogramm hinzugefügt werden soll. Das Festlegen einer Standardgröße ist in manchen Kontexten wichtig, damit Hollywood weiß, welches Bild für

höhere Auflösungen ausgewählt werden soll, z.B. wenn Sie ein 64x64-Bild in einem Piktogramm als Standardgröße festlegen, wählt Hollywood das Bild mit 128x128 aus, falls die Auflösung des Monitors eine DPI-Einstellung verwendet, die doppelt so hoch ist wie die normale Einstellung. Natürlich kann in jedem Piktogramm nur ein Standardbild vorhanden sein. Wenn also bereits ein Standardbild im Piktogramm vorhanden ist, wird ein Fehler generiert.

`AddIconImage()` gibt den Index zurück, an dem das Bild innerhalb des Piktogramm hinzugefügt wurde. Beachten Sie, dass dies nicht unbedingt der letzte Index im Piktogramm ist, da die einzelnen Bilder in Piktogrammen nach ihrer Breite in aufsteigender Reihenfolge sortiert werden. Die von `AddIconImage()` zurückgegebenen Indizes beginnen bei 1.

Um ein Bild von einem Piktogramm zu entfernen, verwenden Sie den Befehl `RemoveIconImage()`.

#### EINGABEN

<code>id</code>	Identifikator des zu verwendenden Piktogramms
<code>table</code>	Tabelle zur Beschreibung des Bildes, das dem Piktogramm hinzugefügt werden soll (siehe oben)

#### RÜCKGABEWERTE

<code>idx</code>	Index, an dem das Bild hinzugefügt wurde (beginnend bei 1)
------------------	--

#### BEISPIEL

```
AddIconImage(1, {Image = "ic16x16.png"})
```

Der obige Code fügt die Bilddatei "ic16x16.png" zum Piktogramm 1 hinzu.

## 42.2 ChangeApplicationIcon

#### BEZEICHNUNG

`ChangeApplicationIcon` – ändert das Docky-Piktogramm (V6.0)

#### ÜBERSICHT

```
ChangeApplicationIcon(id1[, id2, type])
```

#### PLATTFORMEN

Nur AmigaOS 4

#### BESCHREIBUNG

Mit diesem Befehl können Sie das Programm-Piktogramm im AmiDock während der Laufzeit ändern. Sie müssen mindestens einen Pinsel an diesen Befehl übergeben. Wenn Sie einen zweiten Pinsel im optionalen Argument `id2` übergeben, wird dieser Pinsel als zweiter Zustand des Piktogrammes verwendet. Beide Pinsel müssen die gleichen Abmessungen haben. Für die beste visuelle Darstellung sollten Sie nur Pinsel mit Alphakanaltransparenz mit diesem Befehl verwenden.

Bitte beachten Sie, dass nur Standard-Dockies zwei Zustand-Piktogramme in AmiDock unterstützen. Wenn Ihre Anwendung durch eine App-Docky in AmiDock dargestellt



wird, können Sie hier nur ein Bild angeben. Siehe [Abschnitt 16.1 \[Informationen über AmiDock\]](#), [Seite 173](#), für mehr Infos über die Unterschiede der beiden Dockies.

Beachten Sie auch, dass das Ändern des Programm-Piktogramms von Standard-Dockies ein deutlich sichtbare Neugestaltung von AmiDock bewirkt und somit nicht geeignet ist, Animationen in AmiDock anzuzeigen. Wenn Sie das Piktogramm schnell ändern möchten, müssen Sie ein App-Docky verwenden. Siehe [Abschnitt 16.1 \[Informationen über AmiDock\]](#), [Seite 173](#), für Details.

Ab Hollywood 9.0 können Sie mit `ChangeApplicationIcon()` auch Hollywood-Piktogramme verwenden. Dazu müssen Sie `#ICON` im optionalen Argument `type` übergeben. In diesem Fall muss `id1` der Identifikator eines Piktogrammes sein, das verwendet werden soll. Wenn `type` auf `#ICON` gesetzt wurde, wird der Parameter `id2` ignoriert. Im Fall von `#ICON` ist der Parameter `id2` unnötig, da Hollywood-Piktogramme (im Gegensatz zu Pinseln) Bilder für mehrere Zustände enthalten können, so dass `ChangeApplicationIcon()` einfach das ausgewählte Bild verwenden kann, das im Piktogramm gespeichert ist.

Das anfängliche Piktogramm für Ihr Programm in AmiDock kann über die Präprozessor-Anweisung `@APPICON` oder den Tag `DockyBrush` der Anweisung `@OPTIONS` verwendet werden.

Bitte beachten Sie, dass dieser Befehl nur verwendet werden kann, wenn Sie den Tag `RegisterApplication` in der Anweisung `@OPTIONS` auf `True` setzen. Siehe [Abschnitt 50.25 \[OPTIONS\]](#), [Seite 1107](#), für Details.

## EINGABEN

<code>id1</code>	Pinsel oder Piktogramm, das den normalen Zustand des aktuellen Docky-Piktogrammes ersetzen soll
<code>id2</code>	optional: Pinsel, der den ausgewählten Status des aktuellen Docky-Piktogrammes ersetzen soll
<code>type</code>	optional: Typ des in <code>id1</code> übergebenen Objekts (muss entweder <code>#BRUSH</code> oder <code>#ICON</code> sein, voreingestellt ist <code>#BRUSH</code> ) (V9.0)

## 42.3 CreateIcon

### BEZEICHNUNG

`CreateIcon` – erstellt ein Piktogramm (V8.0)

### ÜBERSICHT

`[id] = CreateIcon(id, table)`

### BESCHREIBUNG

Mit diesem Befehl können Sie ein Piktogramm aus einer Sammlung einzelner Bilddateien oder Pinsel erstellen. Sie müssen den gewünschten Identifikator für das neue Piktogramm im Parameter `id` übergeben. Wenn Sie `Nil` als `id` angeben, wird automatisch eine ID für dieses Piktogramm ausgewählt und in `id` zurückgegeben. In Hollywood ist eine Piktogramm eine Sammlung desselben Bildes in verschiedenen Größen und Farbtiefen.

Durch die Verwendung individuell entworfener Bilder für jede Größe, anstatt nur ein und dasselbe Bild für jede Größe zu skalieren, wird eine bessere Qualität erzielt. Dies trifft

insbesondere bei kleineren Bildgrößen zu, die viel besser aussehen, wenn sie speziell für ihre Auflösung ausgelegt sind. Typische Größen für die einzelnen Bilder innerhalb eines Piktogrammes sind 16x16, 24x24, 32x32, 48x48, 64x64, 96x96, 128x128, 256x256 und 512x512. Sie können jedoch auch völlig willkürlich sein. Der Vorteil, dass ein Bild in verschiedenen Größen in einem Piktogramm angezeigt wird, besteht darin, dass Hollywood in Abhängigkeit von der Bildschirmauflösung eine geeignete Größe auswählen kann.

Außerdem können Bilder innerhalb von Hollywood-Piktogrammen auch speziell für unterschiedliche Farbtiefen gestaltet werden. Sie können beispielsweise 24x24-Bilder in verschiedenen Farbtiefen bereitstellen, z.B. in 256 Farben (8 Bit) und in Echtfarbe (**True Color**) mit Alphakanal (32 Bit). So ist es möglich, Bilder gleicher Größe innerhalb eines Piktogrammes zu haben, solange sie sich in ihrer Farbtiefe unterscheiden. Damit hat Hollywood einmal mehr den Vorteil, für eine bestimmte Bildschirmauflösung und Farbtiefe das beste Bild aus einem Piktogramm auszuwählen.

Schließlich kann jedes Bild innerhalb eines Piktogramm-Sets zwei verschiedene Zustände haben: Normal und Ausgewählt. Normalerweise braucht man immer nur den normalen Zustand, aber auf AmigaOS und kompatiblen Geräten wird manchmal auch der ausgewählte Zustand verwendet.

Neben dem Identifikator für das neue Piktogramm, das in `id` übergeben werden soll, müssen Sie auch eine Tabelle im Parameter `table` an `CreateIcon()` übergeben. Diese Tabelle muss eine Reihe von Untertabellen enthalten, eine für jede Bildgröße, die Sie dem Piktogramm hinzufügen möchten.

Die einzelnen Untertabellen können die folgenden Tags verwenden:

- Type:** Mit diesem Tag können Sie den Quelltyp des Bildes festlegen, das Sie zum Piktogramm hinzufügen möchten. Dies kann `#BRUSH` sein, wenn Sie einen Pinsel hinzufügen wollen oder `#FILE`, wenn Sie ein Bild aus einer externen Dateiquelle hinzufügen möchten. Der Standardwert ist `#BRUSH`. Beachten Sie, dass sich diese Standardeinstellung von der Standardeinstellung der Präprozessor-Anweisung `@ICON` unterscheidet. Da ist `#FILE` voreingestellt.
- Image:** Dieser Tag gibt die tatsächliche Bildquelle an und muss in jeder Untertabelle festgelegt werden. Wenn **Type** auf `#BRUSH` gesetzt wurde, müssen Sie diesen Tag auf den Identifikator eines Pinsels setzen, den Sie dem Piktogramm hinzufügen möchten. Der Pinsel kann ein RGB- oder Palettenpinsel sein. Andernfalls muss **Image** auf den Pfad einer Bilddatei gesetzt werden, die dem Piktogramm hinzugefügt werden soll. Die Bilddatei kann in jedem der von Hollywood unterstützten Bilddateiformate vorliegen. Beachten Sie, dass die Alphakanaldateien automatisch geladen werden, wenn die hier angegebene Bilddatei einen Alphakanal hat. Wenn es sich um ein Palettenbild handelt und der Tag `LoadPalette` auf `True` gesetzt ist (siehe unten), wird auch der transparente Stift des Bildes automatisch geladen. Beachten Sie auch, dass in jedem Piktogramm jede Bildgröße nur einmal für jede Farbtiefe verwendet werden darf, d.h. es ist nicht möglich, zwei 48x48 Bilder mit der gleichen Farbtiefe zu einem einzelnen Piktogramm hinzuzufügen. In jedem Piktogramm kann es nur ein Bild für jede Größe und Farbtiefe geben. Wenn das hier angegebene Bild in einem Vektorgrafikformat vorliegt, d.h. entweder ein Vektorpinsel oder eine Datei in einem Vektorbildformat, dürfen

Sie keine weiteren Bilder übergeben, da bei Vektorgrafiken nur ein Bild für alle Größen verwendet wird. Hollywood berechnet das Piktogramm automatisch in alle benötigten Größen. Wenn Sie also Vektor- statt Rastergrafiken verwenden, darf die Tabelle, die Sie an `CreateIcon()` übergeben, nur eine Untertabelle enthalten.

**SelImage:**

Mit diesem Tag können Sie ein zusätzliches Bild einfügen, das als ausgewählte Version des in `Image` angegebenen Bildes verwendet werden soll. Dieser Tag ist optional. Wenn Sie ihn einstellen, muss das hier angegebene Bild genau dieselbe Größe wie das in `Image` angegebene haben. Darüber hinaus wird `SelImage` auf die gleiche Weise wie `Image` verwendet, d.h. es hängt von dem in `Type` eingestellten Bildtyp ab, was Sie hier übergeben müssen. Entweder ist dies ein Pinselidentifikator oder einen Pfad zu einer externen Bilddatei.

**Standard:**

Mit diesem Tag können Sie das in dieser Untertabelle angegebene Bild als Standardgröße für das Piktogramm festlegen. Das Festlegen einer Standardgröße ist in manchen Kontexten wichtig, damit Hollywood weiß, welches Bild für höhere Auflösungen ausgewählt werden soll. Z.B. wenn Sie ein 64x64-Bild in einem Piktogramm als Standardgröße festlegen, wählt Hollywood das Bild mit 128x128 aus, falls die Auflösung des Monitors eine DPI-Einstellung verwendet, die doppelt so hoch ist wie die normale Einstellung. Natürlich kann sich in jedem Piktogramm nur ein Standardbild befinden. Wenn Sie diesen Tag zweimal auf `True` setzen, führt dies zu einem Fehler. Beachten Sie auch, dass Sie keine Standard-Piktogrammgröße angeben müssen. Für viele Anwendungsfälle wird jedoch empfohlen, dies zu tun.

**Loader:** Mit diesem Tag können Sie einen oder mehrere Formatlademodule angeben, die die in `Image` und `SelImage` angegebenen Bilddateien laden. Dies muss als eine Zeichenkette festgelegt werden, die den Namen eines oder mehrerer Formatlademodulen enthält. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. Dieser Tag wird natürlich nur verwendet, wenn `Type` auf `#FILE` gesetzt ist.

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, die die in `Image` und `SelImage` angegebenen Dateien öffnen sollen. Wenn angegeben, muss dies auf eine Zeichenkette festgelegt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. Dieser Tag wird natürlich nur verwendet, wenn `Type` auf `#FILE` gesetzt ist.

**LoadPalette:**

Wenn dieser Tag auf `True` und `Type` auf `#FILE` gesetzt ist, versucht Hollywood, die Palette des Bildes zu laden und das Bild als Palettenbild innerhalb des Piktogrammes zu speichern. Wenn das Bild einen transparenten Stift hat, wird dieser automatisch geladen. (V9.0)

Dieser Befehl ist auch als Präprozessor-Anweisung **@ICON** verfügbar.

Verwenden Sie zum Hinzufügen und Entfernen einzelner Bilder zu einem Piktogramm die Befehle **AddIconImage()** und **RemoveIconImage()**.

### EINGABEN

<b>id</b>	ID des Piktogrammes oder <b>Nil</b> für die automatische ID-Zuweisung
<b>table</b>	Tabelle mit den einzelnen Bildern, die dem Piktogramm hinzugefügt werden sollen (siehe oben)

### BEISPIEL

```
CreateIcon(1, {
    {Image = "ic16x16.png", Type = #FILE},
    {Image = "ic24x24.png", Type = #FILE},
    {Image = "ic32x32.png", Type = #FILE},
    {Image = "ic48x48.png", Type = #FILE},
    {Image = "ic64x64.png", Type = #FILE},
    {Image = "ic96x96.png", Type = #FILE},
    {Image = "ic128x128.png", Type = #FILE},
    {Image = "ic256x256.png", Type = #FILE},
    {Image = "ic512x512.png", Type = #FILE},
    {Image = "ic1024x1024.png", Type = #FILE}})
```

Der obige Code erstellt das Piktogramm 1 aus einem Satz externer Bilder in verschiedenen Größen im Bereich von 16x16 bis 1024x1024 Pixel.

```
CreateIcon(1, {{Image = "icon.svg", Type = #FILE}})
```

Der obige Code erstellt das Piktogramm 1 und verwendet nur ein einzelnes Bild, da das Bild in einem Vektorgrafikformat (SVG) vorliegt und in diesem Fall nur ein einzelnes Bild angegeben werden muss (siehe oben).

## 42.4 FreeIcon

### BEZEICHNUNG

**FreeIcon** – löscht ein Piktogramm aus dem Speicher (V8.0)

### ÜBERSICHT

```
FreeIcon(id)
```

### BESCHREIBUNG

Dieser Befehl löscht das in **id** angegebenen Piktogramm aus dem Speicher. Um Speicherplatz zu sparen, sollten Sie immer Piktogramme löschen, wenn Sie sie nicht mehr benötigen.

### EINGABEN

<b>id</b>	Identifikator des Piktogrammes, welches aus dem Speicher gelöscht werden soll
-----------	---

## 42.5 GetIconProperties

### BEZEICHNUNG

GetIconProperties – ruft die Eigenschaften von einem Piktogramm ab (V4.5)

### ÜBERSICHT

```
t = GetIconProperties(id)
type, tooltypes, deftool$ = GetIconProperties(file$)
```

### BESCHREIBUNG

Mit diesem Befehl können Sie die Eigenschaften eines Piktogrammes abrufen. Dies ist vor allem für Amiga-Piktogrammes nützlich, da sie neben den Bilddaten auch Metadaten enthalten, z.B. Piktogramm-Typ, Merkmale (ToolTypes), allgemeine Informationen und so weiter.

Es gibt zwei Möglichkeiten, diesen Befehl zu verwenden: Sie können entweder den Identifikator eines Piktogrammes im ersten Parameter oder den Dateinamen einer Piktogramm-Datei übergeben. Die Übergabe eines Dateinamens an `GetIconProperties()` wird nur unter AmigaOS und kompatiblen Systemen unterstützt. Die Übergabe einer Piktogramm-ID wird jedoch auf allen Plattformen unterstützt.

Wenn Sie die ID eines Piktogrammes im ersten Parameter übergeben, gibt `GetIconProperties()` eine Tabelle zurück, in der die folgenden Felder initialisiert sind:

**Type:** Dieser Tag wird auf den Amiga-Piktogramm-Typ des Piktogrammes gesetzt. Dies ist eine der folgenden Konstanten:

**#AMIGAICON\_DISK:**

Ein Piktogramm von einem Laufwerk (z.B. RAM, HD, CD-ROM, etc.)

**#AMIGAICON\_DRAWER:**

Ein Piktogramm von einer Schublade/Verzeichnis.

**#AMIGAICON\_TOOL:**

Ein Piktogramm von einem Programm.

**#AMIGAICON\_PROJECT:**

Ein Piktogramm eines Projekts. Ein Projekt ist eine Datendatei, die von einem anderen Programm geöffnet werden kann. Das Programm, welches zum Öffnen des Projekts verwendet werden soll, wird in `DefaultTool` übergeben, z.B. `SYS:Utilities/MultiView`.

**#AMIGAICON\_GARBAGE:**

Ein Papierkorb-Piktogramm.

**##AMIGAICON\_DEVICE:**

Ein Piktogramm eines Gerätes (Device).

**#AMIGAICON\_KICKSTART:**

Ein Kickstart-Piktogramm.

(V9.0)

- IconX:** Die x-Position des Piktogrammes relativ zur oberen linken Ecke der Schublade, in der es gespeichert ist. (V9.0)
- IconY:** Die y-Position des Piktogrammes relativ zur oberen linken Ecke der Schublade, in der es gespeichert ist. (V9.0)
- DrawerX:** Wenn **Type** auf einen Container-Typ wie **#AMIGAICON\_DRAWER** festgelegt ist, wird dieser Tag auf die x-Position des neuen Fensters gesetzt, das beim Doppelklicken auf das Piktogramm geöffnet wird. (V9.0)
- DrawerY:** Wenn **Type** auf einen Container-Typ wie **#AMIGAICON\_DRAWER** festgelegt ist, wird dieser Tag auf die y-Position des neuen Fensters gesetzt, das beim Doppelklicken auf das Piktogramm geöffnet wird. (V9.0)
- DrawerWidth:**  
Wenn **Type** auf einen Container-Typ wie **#AMIGAICON\_DRAWER** festgelegt ist, wird dieser Tag auf die Breite des neuen Fensters gesetzt, das beim Doppelklicken auf das Piktogramm geöffnet wird. (V9.0)
- DrawerHeight:**  
Wenn **Type** auf einen Container-Typ wie **#AMIGAICON\_DRAWER** festgelegt ist, wird dieser Tag auf die Höhe des neuen Fensters gesetzt, das beim Doppelklicken auf das Piktogramm geöffnet wird. (V9.0)
- ViewAll:** Wenn **Type** auf einen Container-Typ wie **#AMIGAICON\_DRAWER** festgelegt ist, wird dieser Tag auf **True** gesetzt, wenn alle Dateien des Verzeichnisses angezeigt werden sollen, anstatt nur diejenigen, die ein Piktogramm haben. (V9.0)
- StackSize:**  
Wenn **Type** auf **#AMIGAICON\_TOOL** oder **#AMIGAICON\_PROJECT** festgelegt ist, wird dieser Tag auf die Stackgröße für das zu startende Programm gesetzt. (V9.0)
- DefaultTool:**  
Wenn **Type** auf **#AMIGAICON\_PROJECT** festgelegt ist, wird dieser Tag auf eine Zeichenkette gesetzt, die den Namen (und optional den Pfad) des Programms enthält, mit dem die Datei geöffnet werden soll. (V9.0)
- ToolTypes:**  
Wenn das Piktogramm Merkmale enthält, werden diese in diesem Tag zurückgegeben. Der Tag **ToolTypes** enthält eine Tabelle mit einer Liste von Untertabellen, d.h. eine Untertabelle pro Tooltyp-Eintrag. Jede Untertabelle enthält die folgenden Tags:
- Key:** Dieser Tag enthält den Namen des Merkmals.
- Value:** Dieser Tag enthält den Wert des Merkmals.
- Enabled:** Dieser Tag ist ein Boolescher Wert, der angibt, ob der Merkmalstyp aktiviert ist oder nicht.
- Alternativ können Sie die Merkmale auch mit dem Tag **RawToolTypes** abrufen (siehe unten). (V9.0)

**RawToolTypes:**

Wenn Sie den Tag **ToolTypes** aus irgendeinem Grund nicht verwenden möchten, können Sie auch **RawToolTypes** verwenden. Im Gegensatz zu **ToolTypes** unterteilt der Tag **RawToolTypes** die Merkmale nicht in ihre einzelnen Bestandteile (Name, Wert, aktiviertes Flag). Stattdessen gibt der Tag **RawToolTypes** nur die Merkmale zurück, wie sie im Piktogramm gespeichert sind, d.h. ohne zusätzliche Verarbeitung. Auf diese Weise können auch benutzerdefinierte Daten in den Merkmalen abgerufen werden. **RawToolTypes** wird auf eine Tabelle gesetzt, die die Merkmale als Zeichenkette enthält. (V9.0)

**Images:** Dieser Tag wird auf eine Tabelle gesetzt, die Informationen zu den einzelnen Bildern innerhalb des Piktogrammes enthält. Die Tabelle enthält eine Untertabelle für jedes Bild des Piktogrammes. Jede Untertabelle enthält die folgenden Felder:

**Width:** Bildbreite in Pixel.

**Height:** Bildhöhe in Pixel.

**Standard:**

**True**, wenn das Bild das Standardbild ist, sonst **False**.

**Frames:** Anzahl der Einzelbilder. Dies kann entweder 1 oder 2 sein. Wenn es 1 ist, gibt es nur das normale Bild, wenn es 2 ist, gibt es auch ein ausgewähltes Zustandsbild.

(V9.0)

Um die Eigenschaften eines Piktogrammes einzustellen, verwenden Sie den Befehl **SetIconProperties()**.

**EINGABEN**

**id** Syntax 1: ID des Piktogrammes, welches untersucht werden soll

**file\$** Syntax 2: das Piktogramm, welches untersucht werden soll

**RÜCKGABEWERTE**

**t** Syntax 1: Tabelle mit Piktogramm-Eigenschaften

**type** Syntax 2: Typ des Piktogrammes; wird eine der oben geschriebenen Konstanten sein

**tooltypes**

Syntax 2: eine Tabelle mit einer Liste aller Merkmalstypen; für jeden Listeneintrag werden die Felder **Key**, **Value** und **Enabled** initialisiert

**deftool\$** Syntax 2: die Standard-Merkmalen für dieses Piktogramm; wird nur für Piktogrammes vom Typ **#AMIGAICON\_PROJECT** gesetzt

**BEISPIEL**

```
t = GetIconProperties(1)
For k = 0 To ListItems(t.ToolTypes) - 1
    DebugPrint("Item:", k, "Key:", t.ToolTypes[k].key,
```

```

        "Value:", t.ToolTypes[k].value,
        "Enabled:", t.ToolTypes[k].enabled)
Next

```

Next

Der obige Code ruft die Eigenschaften vom Piktogramm 1 ab und gibt dann alle Informationen aus, die in dessen Merkmalen gespeichert sind.

```

type, tt, deftool$ = GetIconProperties("MyIcon.info")
For k = 0 To ListItems(tt) - 1
    DebugPrint("Item:", k, "Key:", tt[k].key, "Value:", tt[k].value,
        "Enabled:", tt[k].enabled)
Next

```

Next

Der obige Code ruft die Eigenschaften des Piktogrammes "MyIcon.info" ab und gibt dann alle in seinen Merkmalen gespeicherten Informationen aus.

## 42.6 ICON

### BEZEICHNUNG

ICON – lädt ein Piktogramm für den späteren Gebrauch vor (V8.0)

### ÜBERSICHT

```

@ICON id, filename$[, table]
@ICON id, table

```

### BESCHREIBUNG

Mit dieser Präprozessor-Anweisung kann ein Piktogramm für die spätere Verwendung entweder aus Bilddatei- oder aus Pinselquellen vorgeladen werden. In Hollywood ist ein Piktogramm eine Sammlung desselben Bildes in verschiedenen Größen und Farbtiefen.

In Hollywood ist ein Piktogramm eine Sammlung desselben Bildes in verschiedenen Größen. Durch die Verwendung individuell entworfener Bilder für jede Größe wird eine bessere Qualität erzielt, anstatt nur ein und dasselbe Bild für jede Größe zu skalieren. Dies trifft insbesondere bei kleineren Bildgrößen zu, die viel besser aussehen, wenn sie speziell für ihre Auflösung ausgelegt sind. Typische Größen für die einzelnen Bilder innerhalb eines Piktogrammes sind 16x16, 24x24, 32x32, 48x48, 64x64, 96x96, 128x128, 256x256 und 512x512. Sie können jedoch auch völlig willkürlich sein. Der Vorteil, dass ein Bild in verschiedenen Größen in einem Piktogramm angezeigt wird, besteht darin, dass Hollywood in Abhängigkeit von der Bildschirmauflösung eine geeignete Größe auswählen kann.

Außerdem können Bilder innerhalb von Hollywood-Piktogramme auch speziell für unterschiedliche Farbtiefen gestaltet werden. Sie können beispielsweise 24x24-Bilder in verschiedenen Farbtiefen bereitstellen, z.B. in 256 Farben (8 Bit) und in Echtfarbe (True Color) mit Alphakanal (32 Bit). So ist es möglich, Bilder gleicher Größe innerhalb eines Piktogrammes zu haben, solange sie sich in ihrer Farbtiefe unterscheiden. Damit hat Hollywood einmal mehr den Vorteil, für eine bestimmte Bildschirmauflösung und Farbtiefe das beste Bild aus einem Piktogramm auszuwählen.

Schließlich kann jedes Bild innerhalb eines Piktogramm-Sets zwei verschiedene Zustände haben: Normal und Ausgewählt. Normalerweise braucht man immer nur den normalen



Zustand, aber auf AmigaOS und kompatiblen Geräten wird manchmal auch der ausgewählte Zustand verwendet.

Diese Präprozessor-Anweisung kann auf zwei verschiedene Arten verwendet werden: Sie können entweder eine einzelne Datei in `filename$` angeben, die als Hollywood-Piktogramm geladen werden soll oder Sie können eine Tabelle angeben, um ein Hollywood-Piktogramm aus einzelnen Bildern zu erstellen, die entweder als Pinsel oder als externe Dateien definiert werden.

Wenn Sie die erste Syntax verwenden, d.h. eine einzige Datei in `filename$` übergeben, erwartet `@ICON`, dass die Datei im benutzerdefinierten PNG-Piktogrammformat von Hollywood vorliegt. `@ICON` bewirkt in diesem Fall dasselbe wie der Befehl `LoadIcon()` mit der Ausnahme, dass das Piktogramm vorinstalliert ist und beim Kompilieren mit dem Applet oder der ausführbaren Datei verknüpft wird. Siehe [Abschnitt 42.7 \[LoadIcon\]](#), [Seite 892](#), für Details.

Die erste Syntax akzeptiert auch ein Tabellenargument, das auf den Parameter `filename$` folgt. Die folgenden Felder in der Tabelle können festgelegt werden:

- Link:** Setzen Sie dieses Feld auf `False`, wenn dieses Piktogramm beim Kompilieren des Skripts nicht mit Ihrer ausführbaren Datei/Ihrem Applet verknüpft werden soll. Dieses Feld ist standardmäßig auf `True` gesetzt, was bedeutet, dass das Piktogramm mit Ihrer ausführbaren Datei/Ihrem Applet verknüpft ist, wenn sich Hollywood im Kompiliermodus befindet.
- Loader:** Mit diesem Tag können Sie ein oder mehrere Formatlademodule angeben, die zum Laden dieses Piktogrammes benutzt werden sollen. Dies muss als Zeichenkette festgelegt werden, die den Namen eines oder mehrerer Formatlademodulen enthält. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details.
- Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, die zum Öffnen der angegebenen Datei benutzt werden sollen. Dies muss auf eine Zeichenkette gesetzt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details.
- UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Alternativ können Sie die zweite Syntax verwenden, mit der Sie ein Piktogramm aus einem Satz von Quellbildern erstellen können, die entweder externe Dateien oder Hollywood-Pinsel sind. In diesem Fall müssen Sie ein einzelnes Tabellenargument angeben, das eine Anzahl von Untertabellen enthalten muss, eine für jede Bildgröße, die Sie dem Piktogramm hinzufügen möchten. Dies ist ähnlich wie der Befehl `CreateIcon()`. Siehe [Abschnitt 42.3 \[CreateIcon\]](#), [Seite 881](#), für Details.

Die einzelnen Untertabellen können die folgenden Tags verwenden:

**Type:** Mit diesem Tag können Sie den Quelltyp des Bildes festlegen, das Sie zum Piktogramm hinzufügen möchten. Dies kann **#BRUSH** sein, wenn Sie einen Pinsel hinzufügen wollen oder **#FILE**, wenn Sie ein Bild aus einer externen Dateiquelle hinzufügen möchten. Der Standardwert ist **#FILE**. Beachten Sie, dass sich diese Standardeinstellung von der Standardeinstellung des Befehls **CreateIcon()** und **AddIconImage()** unterscheidet. Da ist **#BRUSH** voreingestellt.

**Image:** Dieser Tag gibt die tatsächliche Bildquelle an und muss in jeder Untertabelle festgelegt werden. Wenn **Type** auf **#BRUSH** gesetzt wurde, müssen Sie diesen Tag auf den Identifikator eines Pinsels setzen, den Sie dem Piktogramm hinzufügen möchten. Der Pinsel kann ein RGB- oder Palettenpinsel sein. Andernfalls muss **Image** auf den Pfad einer Bilddatei gesetzt werden, die dem Piktogramm hinzugefügt werden soll. Die Bilddatei kann in jedem der von Hollywood unterstützten Bilddateiformate vorliegen. Beachten Sie, dass die Alphakanaldateien automatisch geladen werden, wenn die hier angegebene Bilddatei einen Alphakanal hat. Wenn es sich um ein Palettenbild handelt und der Tag **LoadPalette** auf **True** gesetzt ist (siehe unten), wird auch der transparente Stift des Bildes automatisch geladen. Beachten Sie auch, dass in jedem Piktogramm jede Bildgröße nur einmal für jede Farbtiefe verwendet werden darf, d.h. es ist nicht möglich, zwei 48x48 Bilder mit der gleichen Farbtiefe zu einem einzelnen Piktogramm hinzuzufügen. In jedem Piktogramm kann es nur ein Bild für jede Größe und Farbtiefe geben. Wenn das hier angegebene Bild in einem Vektorgrafikformat vorliegt, d.h. entweder ein Vektorpinsel oder eine Datei in einem Vektorbildformat, dürfen Sie keine weiteren Bilder übergeben, da bei Vektorgrafiken nur ein Bild für alle Größen verwendet wird. Hollywood berechnet das Piktogramm automatisch in alle benötigten Größen. Wenn Sie also Vektor anstelle von Rastergrafiken verwenden, darf die Tabelle, die Sie an **@ICON** übergeben, nur eine Untertabelle enthalten.

**SelImage:** Mit diesem Tag können Sie ein zusätzliches Bild einfügen, das als ausgewählte Version des in **Image** angegebenen Bildes verwendet werden soll. Dieser Tag ist optional. Wenn Sie ihn einstellen, muss das hier angegebene Bild genau dieselbe Größe wie das in **Image** angegebene haben. Darüber hinaus wird **SelImage** auf die gleiche Weise wie **Image** verwendet, d.h. es hängt von dem in **Type** eingestellten Bildtyp ab, was Sie hier übergeben müssen. Entweder ist dies ein Pinselidentifikator oder einen Pfad zu einer externen Bilddatei.

**Standard:** Mit diesem Tag können Sie das in dieser Untertabelle angegebene Bild als Standardgröße für das Piktogramm festlegen. Das Festlegen einer Standardgröße ist in manchen Kontexten wichtig, damit Hollywood weiß, welches Bild für höhere Auflösungen ausgewählt werden soll. Z.B. wenn Sie ein 64x64-Bild in einem Piktogramm als Standardgröße festlegen, wählt Hollywood das Bild

mit 128x128 aus, falls die Auflösung des Monitors eine DPI-Einstellung verwendet, die doppelt so hoch ist wie die normale Einstellung. Natürlich kann sich in jedem Piktogramm nur ein Standardbild befinden. Wenn Sie diesen Tag zweimal auf **True** setzen, führt dies zu einem Fehler. Beachten Sie auch, dass Sie keine Standard-Piktogrammgröße angeben müssen. Für viele Anwendungsfälle wird jedoch empfohlen, dies zu tun.

**Link:** Setzen Sie dieses Feld auf **False**, wenn Sie nicht möchten, dass die in **Image** und **SelfImage** angegebenen Bilddateien beim Kompilieren des Skripts mit Ihrer ausführbaren Datei/Ihrem Applet verknüpft werden. Dieses Feld ist standardmäßig auf **True** gesetzt, womit die in **Image** und **SelfImage** angegebenen Bilddateien mit Ihrer ausführbaren Datei/Ihrem Applet verknüpft werden, wenn sich Hollywood im Kompiliermodus befindet. Natürlich wird dieser Tag nur verwendet, wenn **Type** auf **#FILE** gesetzt ist.

**Loader:** Mit diesem Tag können Sie einen oder mehrere Formatlademodule angeben, die zum Laden der in **Image** und **SelfImage** angegebenen Dateien benutzt werden sollen. Dies muss als eine Zeichenkette festgelegt werden, die den Namen eines oder mehrerer Formatlademodulen enthält. Standardmäßig wird der mit **SetDefaultLoader()** eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. Dieser Tag kann nur verwendet werden, wenn **Type** auf **#FILE** gesetzt ist.

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, die die zum Öffnen in den von **Image** und **SelfImage** angegebenen Dateien benutzt werden sollen. Dies muss auf eine Zeichenkette gesetzt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der mit **SetDefaultAdapter()** eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. Dieser Tag kann nur verwendet werden, wenn **Type** auf **#FILE** gesetzt ist.

**LoadPalette:** Wenn dieser Tag auf **True** und **Type** auf **#FILE** gesetzt ist, versucht Hollywood, die Palette des Bildes zu laden und das Bild als Palettenbild innerhalb des Piktogrammes zu speichern. Wenn das Bild einen transparenten Stift hat, wird dieser automatisch geladen. (V9.0)

**UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Um zur Laufzeit Piktogramme zu laden oder zu erstellen, sehen Sie sich die Befehle **LoadIcon()** und **CreateIcon()** an.

Verwenden Sie zum Hinzufügen und Entfernen einzelner Bilder zu einem Piktogramm die Befehle **AddIconImage()** und **RemoveIconImage()**.

## EINGABEN

**id** ein Wert, mit dem dieses Piktogramm später im Code identifiziert wird

`filename$` optional: die Piktogrammdatei, die Sie laden möchten (siehe oben)

`table` optional oder obligatorisch, abhängig von der verwendeten Syntax (siehe oben)

**BEISPIEL**

```
@ICON 1, "MyIcon.png"
```

Lädt "MyIcon.png" als Piktogramm 1.

```
@ICON 1, {
  {Image = "ic16x16.png"},
  {Image = "ic24x24.png"},
  {Image = "ic32x32.png"},
  {Image = "ic48x48.png"},
  {Image = "ic64x64.png"},
  {Image = "ic96x96.png"},
  {Image = "ic128x128.png"},
  {Image = "ic256x256.png"},
  {Image = "ic512x512.png"},
  {Image = "ic1024x1024.png"}}

```

Der obige Code erstellt das Piktogramm 1 aus einem Satz externer Bilder in verschiedenen Größen im Bereich von 16x16 bis 1024x1024 Pixel.

```
@ICON 1, {{Image = "icon.svg"}}
```

Der obige Code erstellt das Piktogramm 1 und verwendet nur ein einzelnes Bild, da das Bild in einem Vektorgrafikformat (SVG) vorliegt und in diesem Fall nur ein einzelnes Bild angegeben werden muss (siehe oben).

## 42.7 LoadIcon

**BEZEICHNUNG**

LoadIcon – lädt ein Piktogramm (V8.0)

**ÜBERSICHT**

```
[id] = LoadIcon(id, filename$[, table])
```

**BESCHREIBUNG**

Dieser Befehl lädt das durch `filename$` angegebene Piktogramm in den Speicher und weist ihm den Identifikator `id` zu. Wenn Sie `Nil` als `id` angeben, wird automatisch eine ID für dieses Piktogramm ausgewählt und in `id` zurückgegeben.

In Hollywood ist ein Piktogramm eine Sammlung desselben Bildes in verschiedenen Größen. Durch die Verwendung individuell entworfener Bilder für jede Größe, anstatt nur ein und dasselbe Bild für jede Größe zu skalieren, wird eine bessere Qualität erzielt. Dies trifft insbesondere bei kleineren Bildgrößen zu, die viel besser aussehen, wenn sie speziell für ihre Auflösung ausgelegt sind. Darüber hinaus kann jedes Bild innerhalb eines

Piktogrammsatzes zwei verschiedene Status haben: Normal und Ausgewählt. Normalerweise benötigen Sie immer nur den normalen Status, aber bei AmigaOS und kompatiblen Systemen wird manchmal auch der ausgewählte Status verwendet. Typische Größen für die einzelnen Bilder innerhalb eines Piktogrammes sind 16x16, 24x24, 32x32, 48x48, 64x64, 96x96, 128x128, 256x256 und 512x512. Sie können jedoch auch völlig willkürlich sein. Der Vorteil, dass ein Bild in verschiedenen Größen in einem Piktogramm angezeigt wird, besteht darin, dass Hollywood in Abhängigkeit von der Bildschirmauflösung eine geeignete Größe auswählen kann.

Das in `filename$` übergebene Piktogramm muss das benutzerdefinierten PNG-Piktogrammformat von Hollywood vorliegen. Sie können `SaveIcon()` verwenden, um solche Piktogramme zu erstellen. Obwohl Hollywood-Piktogramme normale PNG-Bilder sind, enthalten sie zusätzliche Metadaten. Aus diesem Grund dürfen Sie sie nicht mit Ihrem bevorzugten Bildbearbeitungswerkzeug bearbeiten, da dies zum Verlust der Metadaten führen kann. Hollywood-Piktogramme sollten immer nur mit dem Befehl `SaveIcon()` erstellt werden.

Wenn Sie das benutzerdefinierte Piktogrammformat von Hollywood nicht verwenden möchten, können Sie auch Piktogramme aus Pinseln oder normalen Bildern mit dem Befehl `CreateIcon()` oder dem Präprozessor-Anweisung `@ICON` erstellen. Diese Befehle haben auch den Vorteil, dass Sie Vektorpinsel verwenden können, die verlustfrei auf jede Größe skaliert werden können, was zu einem absolut gestochen scharfen Aussehen in allen möglichen unterschiedlichen Auflösungen führt.

Das dritte Argument ist optional. In dieser Tabelle können Sie weitere Optionen für den Ladevorgang festlegen. Die folgenden Felder der Tabelle können verwendet werden:

**Loader:** Mit diesem Tag können Sie ein oder mehrere Formatlademodule angeben, die zum Laden dieses Piktogrammes benutzt werden sollen. Dies muss als eine Zeichenkette festgelegt werden, die den Namen eines oder mehrerer Formatlademodulen enthält. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details.

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, die zum Öffnen der angegebenen Datei benutzt werden sollen. Dies muss auf eine Zeichenkette gesetzt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details.

**UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Dieser Befehl ist auch als Präprozessor-Anweisung verfügbar: Verwenden Sie `@ICON`, um die Piktogramme vorab zu laden.

**EINGABEN**

**id** ID des Piktogrammes oder **Nil** für die automatische ID-Zuweisung

**filename\$**  
zu ladende Datei

**table** optional: weitere Optionen (siehe oben)

**RÜCKGABEWERTE**

**id** optional: Identifikator des Piktogrammes; wird nur zurückgegeben werden, wenn Sie **Nil** als Argument 1 angegeben haben (siehe oben)

**BEISPIEL**

```
LoadIcon(1, "test.png")
```

Dieser Code lädt "test.png" als Piktogramm 1.

## 42.8 RemoveIconImage

**BEZEICHNUNG**

RemoveIconImage – entfernt ein Bild vom Piktogramm (V8.0)

**ÜBERSICHT**

```
RemoveIconImage(id, idx)
```

**BESCHREIBUNG**

Mit diesem Befehl können Sie ein einzelnes Bild von dem in **id** angegebenen Piktogramm entfernen. Das zu entfernende Bild muss anhand des Parameters **idx** durch seinen Index angegeben werden. Die Indizes beginnen bei 1 für das erste Bild und reichen bis zur Anzahl der Bilder im Piktogramm. Sie können die Anzahl der Bilder in einem Piktogramm abfragen, indem Sie das Attribut **#ATTRNUMENTRIES** mit dem Befehl **GetAttribute()** verwenden.

Beachten Sie, dass die einzelnen Bilder in Piktogrammen aufsteigend nach ihrer Breite sortiert werden. Dies bedeutet, dass die an **RemoveIconImage()** übergebenen Indizes nicht unbedingt der Reihenfolge von Bildern entsprechen, die an Befehle wie **CreateIcon()** oder der Präprozessor-Anweisung **@ICON** übergeben werden.

Da Piktogramme mindestens ein Bild enthalten müssen, darf auch das letzte Bild nicht aus einem Piktogramm entfernt werden.

Verwenden Sie zum Hinzufügen von Bildern zu Piktogrammen den Befehl **AddIconImage()**.

**EINGABEN**

**id** Identifikator des zu verwendenden Piktogrammes

**idx** Index des zu entfernenden Bildes (beginnend mit 1)

**BEISPIEL**

```
RemoveIconImage(1, 1)
```

Der obige Code entfernt das erste Bild vom Piktogramm 1.

## 42.9 SaveIcon

### BEZEICHNUNG

SaveIcon – speichert ein Piktogramm in einer Datei (V8.0)

### ÜBERSICHT

```
SaveIcon(id, f$[, fmt, t])
```

### BESCHREIBUNG

Dieser Befehl speichert das durch `id` angegebene Piktogramm in der durch `f$` angegebenen Datei. Standardmäßig wird das Piktogramm in Hollywoods benutzerdefiniertem Piktogrammformat gespeichert, das auf PNG basiert. Sie können dies ändern, indem Sie eine andere Piktogrammformatkonstante im Argument `fmt` übergeben. Das einzige von Hollywood intern unterstützte Piktogrammformat ist `#ICNFMT_HOLLYWOOD`, welches auf PNG basiert. Zusätzliche Piktogrammformate können durch Hollywood-Plugins verfügbar gemacht werden.

Beachten Sie, dass Hollywoods benutzerdefiniertes Piktogrammformat zwar Piktogramme als scheinbar normale PNG-Bilder speichert, aber zusätzliche Metadaten enthält, weshalb Sie sie nicht mit Ihrem bevorzugten Bildbearbeitungsprogramm bearbeiten sollten, da dies zum Verlust dieser Metadaten führen könnte. Hollywood-Piktogramme sollten immer nur mit dem Befehl `SaveIcon()` erstellt werden.

Beachten Sie auch, dass bei Verwendung des benutzerdefinierten Piktogrammformats von Hollywood das in `id` angegebene Piktogramm keine Vektorgrafiken enthalten darf. Hollywood-Piktogramme unterstützen nur Rastergrafiken, da sie auf PNG basieren. Wenn Sie Vektorgrafiken in einem Piktogramm verwenden möchten, können Sie solche Piktogramme mit dem Befehl `CreateIcon()` und der Präprozessor-Anweisung `@ICON` erstellen.

Schließlich dürfen Piktogramme in Hollywoods eigenem Piktogramm-Format auch keine Palettengrafiken enthalten. Aus diesem Grund schlägt `SaveIcon()` fehl, wenn das durch `id` angegebene Piktogramm Palettenbilder enthält.

Ab Hollywood 9.0 akzeptiert `SaveIcon()` ein optionales Tabellenargument, das die folgenden Optionen enthalten kann:

#### Compression:

Bei Piktogrammformaten, die Komprimierung unterstützen, können Sie diesen Tag auf `True` oder `False` setzen, um die Komprimierung zu aktivieren oder zu deaktivieren. Voreingestellt ist `True`. (V9.0)

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, die zum Speichern der angegebenen Datei benutzt werden sollen. Dies muss auf eine Zeichenkette gesetzt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V10.0)

#### UserTags:

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die



die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

## EINGABEN

<code>id</code>	Identifikator des zu speichernden Piktogrammes
<code>f\$</code>	Zieldatei
<code>fmt</code>	optional: gewünschtes Piktogrammformat (Voreingestellt ist <code>#ICNFMT_HOLLYWOOD</code> ) (V9.0)
<code>t</code>	optional: Tabelle mit weiteren Argumenten (V9.0)

## 42.10 SetIconProperties

### BEZEICHNUNG

`SetIconProperties` – ändert die Eigenschaften eines Amiga-Piktogrammes (V4.5)

### ÜBERSICHT

```
SetIconProperties(id, table)
SetIconProperties(file$, type[, tooltypes, deftool$]) (nur Amiga)
```

### BESCHREIBUNG

Mit diesem Befehl können Sie die Eigenschaften eines Piktogrammes ändern. Dies ist vor allem für Amiga-Piktogramme nützlich, da diese neben den Bilddaten auch Metadaten enthalten, z.B. Piktogramm-Typ, Merkmale, allgemeine Informationen und so weiter. Diese Metadaten können mit `SetIconProperties()` geändert werden.

Es gibt zwei Möglichkeiten, diesen Befehl zu verwenden: Sie können entweder den Identifikator eines Piktogrammes im ersten Parameter und eine Tabelle im zweiten Parameter oder den Dateinamen einer Piktogramm-Datei im ersten Parameter und weitere Argumente in den folgenden Parametern übergeben. Die Übergabe eines Dateinamens an `SetIconProperties()` wird nur unter AmigaOS und kompatiblen Systemen unterstützt. Die Übergabe einer Piktogramm-ID wird jedoch auf allen Plattformen unterstützt.

Wenn Sie eine Piktogramm-ID an `SetIconProperties()` übergeben, müssen Sie im zweiten Argument eine Tabelle übergeben, um die tatsächlich zu ändernden Eigenschaften anzugeben. Die folgenden Piktogramm-Eigenschaften können derzeit geändert werden:

**Type:** Mit diesem Tag können Sie den gewünschten Amiga-Piktogramm-Typ für das Piktogramm einstellen. Dies muss eine der folgenden Konstanten sein:

```
#AMIGAICON_DISK:
    Ein Piktogramm eines Laufwerks (z.B. RAM, HD, CD-ROM, usw.)

#AMIGAICON_DRAWER:
    Ein Piktogramm einer Schublade.

#AMIGAICON_TOOL:
    Ein Piktogramm eines Programms.
```



**#AMIGAICON\_PROJECT:**

Ein Piktogramm eines Projekts. Ein Projekt ist eine Datendatei, die von einem anderen Programm geöffnet werden kann. Das Programm, das verwendet werden sollte, um das Projekt zu öffnen, sollte im Tag `DefaultTool` übergeben werden (siehe unten).

**#AMIGAICON\_GARBAGE:**

Ein Papierkorb-Piktogramm.

**##AMIGAICON\_DEVICE:**

Ein Piktogramm eines Gerätes.

**#AMIGAICON\_KICKSTART:**

Ein Kickstart-Piktogramm.

(V9.0)

**IconX:** Die x-Position des Piktogramms in Bezug auf die linke obere Ecke der Schublade, in der es gespeichert ist. (V9.0)

**IconY:** Die y-Position des Piktogramms in Bezug auf die linke obere Ecke der Schublade, in der es gespeichert ist. (V9.0)

**DrawerX:** Wenn **Type** auf einen Container-Typ wie **#AMIGAICON\_DRAWER** festgelegt ist, kann dieser Tag verwendet werden, um die x-Position des neuen Fensters festzulegen, das beim Doppelklicken auf das Piktogramm geöffnet wird. (V9.0)

**DrawerY:** Wenn **Type** auf einen Container-Typ wie **#AMIGAICON\_DRAWER** festgelegt ist, kann dieser Tag verwendet werden, um die y-Position des neuen Fensters festzulegen, das beim Doppelklicken auf das Piktogramm geöffnet wird. (V9.0)

**DrawerWidth:**

Wenn **Type** auf einen Container-Typ wie **#AMIGAICON\_DRAWER** festgelegt ist, kann dieser Tag verwendet werden, um die Breite des neuen Fensters festzulegen, das beim Doppelklicken auf das Piktogramm geöffnet wird. (V9.0)

**DrawerHeight:**

Wenn **Type** auf einen Container-Typ wie **#AMIGAICON\_DRAWER** festgelegt ist, kann dieser Tag verwendet werden, um die Höhe des neuen Fensters festzulegen, das beim Doppelklicken auf das Piktogramm geöffnet wird. (V9.0)

**ViewAll:** Wenn **Type** auf einen Container-Typ wie **#AMIGAICON\_DRAWER** festgelegt ist, können Sie diesen Tag auf **True** setzen, um die Workbench anzuweisen, alle Dateien der Schublade anzuzeigen und nicht nur die, die ein Piktogramm haben. (V9.0)

**StackSize:**

Wenn **Type** auf **#AMIGAICON\_TOOL** oder **#AMIGAICON\_PROJECT** festgelegt ist, wird dieser Tag auf die Stackgröße für das zu startende Programm gesetzt. (V9.0)

**DefaultTool:**

Wenn **Type** auf **#AMIGAICON\_PROJECT** festgelegt ist, kann dies auf eine Zeichenkette gesetzt werden, die den Namen (und optional den Pfad) des Programms enthält, mit dem die Datei geöffnet werden soll. (V9.0)

**ToolTypes:**

Wenn Sie dem Piktogramm Merkmale hinzufügen möchten, müssen Sie im Tag **ToolTypes** eine Tabelle übergeben. Die Tabelle muss eine Liste von Untertabellen enthalten, und zwar eine Untertabelle pro Merkmaleintrag. Jede Untertabelle kann die folgenden Tags enthalten:

- Key:** Dieser Tag ist obligatorisch. Er gibt den Namen des ToolTypes an. Tooltype-Namen sollten nur Buchstaben des englischen Alphabets enthalten, sollten immer im Großbuchstabenformat sein und sie dürfen keine Leerzeichen verwenden. Wenn Sie ein Leerzeichen wünschen, verwenden Sie stattdessen einen Unterstrich ("\_"). Außerdem sollten Zahlen nicht als Anfangszeichen eines Tooltype-Namens verwendet werden.
- Value:** Dieser Tag ist optional. Mit diesem können Sie dem ToolType einen Wert zuordnen. Wenn Sie diesen Wert nicht setzen, wird der ToolType ein boolescher Wert sein.
- Enabled:** Auch dieser Tag ist optional. Die Voreinstellung ist **True**. Wenn Sie ToolTypes hinzufügen möchten, die anfänglich deaktiviert sind, können Sie diesen Tag auf **False** setzen. In diesem Fall wird der ToolType in Klammern eingeschlossen, was bedeutet, dass er deaktiviert ist.

Alternativ können Sie Merkmale auch mit dem Tag **RawToolTypes** festlegen (siehe unten).

(V9.0)

**RawToolTypes:**

Wenn Sie den Tag **ToolTypes** aus irgendeinem Grund nicht verwenden möchten, können Sie auch **RawToolTypes** verwenden. Im Gegensatz zu **ToolTypes** unterteilt der **RawToolTypes** die Merkmale nicht in ihre einzelnen Bestandteile (Name, Wert, aktiviertes Flag). Stattdessen kopiert der Tag **RawToolTypes** die Merkmale ohne zusätzliche Verarbeitung auf das Piktogramm. Dadurch ist es möglich, benutzerdefinierte Daten auch in den Merkmalen zu speichern. Wenn Sie dies tun möchten, setzen Sie einfach **RawToolTypes** auf eine Tabelle, die alle Merkmale enthält, die als einfache Zeichenkette festgelegt werden soll. (V9.0)

Um die Eigenschaften eines Piktogrammes zu lesen, benutzen Sie den Befehl **GetIconProperties()**.

**EINGABEN**

- |               |  |
|---------------|--|
| <b>id</b>     | Syntax 1: Identifikator des zu ändernden Piktogrammes                          |
| <b>table</b>  | Syntax 1: Tabelle mit den einzustellenden Piktogramm-Eigenschaften             |
| <b>file\$</b> | Syntax 2: das zu ändernde Piktogramm   |
| <b>type</b>   | Syntax 2: neuer Typ für das Piktogramm; muss eine der Konstanten von oben sein |

- tooltypes** Syntax 2, optional: eine Tabelle mit einer Liste aller ToolTypes, die Sie einstellen wollen; jeder Listeneintrag muss mindestens den Tag **Key** enthalten; Standard ist {} (eine leere Tabelle)
- deftool\$** Syntax 2, optional: das DefaultTool, das für dieses Piktogramm gesetzt wird; wird nur für Piktogramme vom Typ **#AMIGAICON\_PROJECT** gesetzt; Standardmäßig auf "".

**BEISPIEL**

```
SetIconProperties(1, {
  Type = #AMIGAICON_PROJECT,
  DefaultTool = "Hollywood:System/Hollywood",
  ToolTypes = {
    {Key = "BORDERLESS"},
    {Key = "BACKFILL", Value = "GRADIENT"},
    {Key = "STARTCOLOR", Value = "$000000"},
    {Key = "ENDCOLOR", Value = "$0000ff"},
    {Key = "FIXED", Enabled = False}
  }
})
```

Der obige Code setzt den Typ von Piktogramm 1 auf **#AMIGAICON\_PROJECT**, das Programm auf "Hollywood:System/Hollywood" und fügt einige Merkmale hinzu.

```
SetIconProperties("MyCoolScript.hws.info", #AMIGAICON_PROJECT, {
  {Key = "BORDERLESS"},
  {Key = "BACKFILL", Value = "GRADIENT"},
  {Key = "STARTCOLOR", Value = "$000000"},
  {Key = "ENDCOLOR", Value = "$0000ff"},
  {Key = "FIXED", Enabled = False} }, "Hollywood:System/Hollywood")
```

Der obige Code setzt Hollywood als DefaultTool für "MyCoolScript.hws". Darüber hinaus fügt es mehrere Tooltypes dem Piktogramm des Skripts hinzu, die Hollywood mitteilen, was als Hintergrund dem Skript hinzugefügt werden sollte (z.B. Farbverlauf des Hintergrunds).

## 42.11 SetStandardIconImage

**BEZEICHNUNG**

SetStandardIconImage – setzt das Standardbild des Piktogrammes (V8.0)

**ÜBERSICHT**

```
SetStandardIconImage(id, idx)
```

**BESCHREIBUNG**

Mit diesem Befehl kann das Standardbild in dem durch **id** angegebene Piktogramm festgelegt werden. Das Bild, das zum Standardbild gemacht werden soll, muss durch den Index mit dem Parameter **idx** angegeben werden. Die Indizes beginnen bei 1 für das

erste Bild und reichen bis zur Anzahl der Bilder im Piktogramm. Sie können die Anzahl der Bilder in einem Piktogramm abfragen, indem Sie das Attribut `#ATTRNUMENTRIES` mit `GetAttribute()` verwenden.

Beachten Sie, dass die einzelnen Bilder in Piktogrammen aufsteigend nach ihrer Breite sortiert werden. Dies bedeutet, dass die an `RemoveIconImage()` übergebenen Indizes nicht unbedingt der Reihenfolge von Bildern entsprechen, die an Befehle wie `CreateIcon()` oder der Präprozessor-Anweisung `@ICON` übergeben werden.

Das Festlegen eines Bildes innerhalb des Piktogrammes als Standardbild kann in manchen Kontexten wichtig sein, damit Hollywood weiß, welches Bild für höhere Auflösungen ausgewählt werden soll, z.B. wenn Sie ein 64x64-Bild in einem Piktogramm als Standardgröße festlegen, wählt Hollywood das Bild mit 128x128 aus, falls die Auflösung des Monitors eine DPI-Einstellung verwendet, die doppelt so hoch ist wie die normale Einstellung. Natürlich kann es in jedem Piktogramm nur ein Standardbild geben. Wenn Sie also ein Bild zum Standardbild machen, wird der Standardidentifikator für jedes Bild, das zuvor das Standardbild war, automatisch gelöscht. Um kein Bild zum Standardbild zu machen, übergeben Sie 0 in `idx`.

#### EINGABEN

<code>id</code>	Identifikator des zu verwendenden Piktogrammes
<code>idx</code>	Index des Bildes, welches zum Standardbild wird (beginnend mit 1)

#### BEISPIEL

```
SetStandardIconImage(1, 1)
```

Der obige Code macht das erste Bild 1 in Piktogramm 1 zum Standardbild.

## 42.12 SetTrayIcon

#### BEZEICHNUNG

SetTrayIcon – installiert ein Bild als Piktogramm in der Taskleiste (V5.2)

#### ÜBERSICHT

```
SetTrayIcon(id[, tooltip$, type])
```

#### PLATTFORMEN

Nur Microsoft Windows

#### BESCHREIBUNG

Mit diesem Befehl kann das in `id` angegebene Bild als Piktogramm in der Windows-Taskleiste installiert werden. Wenn der Benutzer auf dieses Piktogramm klickt, erhält das Skript ein Ereignis vom Typ `TrayIcon`, das Sie mit dem Befehl `InstallEventHandler()` bearbeiten können. Das optionale Argument `tooltip$` kann verwendet werden, um eine Zeichenfolge anzugeben, der als QuickInfo angezeigt werden soll, wenn die Maus über dem Taskleisten-Piktogramm schwebt.

Das Bild, das Sie diesem Befehl übergeben, sollte 16x16 Pixel groß sein und sollte einen Alphakanal für die Transparenz verwenden.

Wenn Sie beim Aufruf von diesem Befehl bereits ein Taskleisten-Piktogramm installiert haben, wird das Piktogramm in die Grafik des angegebenen Pinsels geändert. Wenn Sie

den Spezialwert `#NONE` als Pinsel-ID übergeben, wird das Piktogramm in der Taskleiste entfernt.

Ein weiterer besonderer Wert, den Sie an diesen Befehl übergeben können, ist `#DEFAULTICON`. Wenn Sie `#DEFAULTICON` in `id` übergeben, verwendet `SetTrayIcon()` das Piktogramm, das mit der Präprozessor-Anweisung `@APPICON` definiert wurde oder, falls keine `@APPICON-Definition` vorhanden ist, Hollywoods Standard-Piktogramm (die Filmklappe).

Ab Hollywood 8.0 gibt es ein optionales Argument `type`, mit dem Sie den Quellbildtyp für das Taskleisten-Piktogramm angeben können. Der Standardwert ist `#BRUSH`, was bedeutet, dass Sie den Identifikator eines Pinsels im Argument `id` übergeben müssen. Sie können das Argument `type` jedoch auch auf `#ICON` setzen. In diesem Fall müssen Sie den Identifikator eines Hollywood-Piktogrammes im Argument `id` übergeben. Dies hat den Vorteil, dass Hollywood je nach Auflösung des Monitors des Hostsystems unterschiedliche Bilder auswählen kann. Dies ist sehr nützlich für Systeme, die Monitore mit hohem DPI-Wert verwenden. Wenn Sie ein Piktogramm verwenden, das ein Bild in mehreren Auflösungen enthält, können Sie sicherstellen, dass das Taskleisten-Piktogramm auch auf Monitoren mit hoher DPI-Auflösung perfekt aussieht. Siehe [Abschnitt 42.3 \[CreateIcon\]](#), [Seite 881](#), für Details.

Wenn Sie ein Piktogramm in `id` übergeben, müssen Sie das 16x16-Bild als Standardbild innerhalb des Piktogrammes festlegen, da 16x16 Pixel die Standardgröße von Piktogrammen für die Windows-Taskleiste ist. Siehe [Abschnitt 42.11 \[SetStandardIconImage\]](#), [Seite 899](#), für Details.

## EINGABEN

<code>id</code>	ID des Bildes, das als Piktogramm in der Taskleiste verwendet werden soll oder <code>#NONE</code> oder <code>#DEFAULTICON</code>
<code>tooltip\$</code>	optional: Text, der als Kurzinfo des Piktogrammes angezeigt wird
<code>type</code>	optional: Typ des in <code>id</code> übergebenen Bildes; dies kann entweder <code>#BRUSH</code> oder <code>#ICON</code> sein (Voreingestellt ist <code>#BRUSH</code> ) (V8.0)

## BEISPIEL

```
InstallEventHandler({TrayIcon = ...})
SetTrayIcon(1, "My program")
```

Der obige Code aktiviert den `TrayIcon`-Ereignis-Handler und installiert dann die Pinselnummer 1 als Taskleisten-Piktogramm.

## 42.13 SetWBIcon

### BEZEICHNUNG

`SetWBIcon` – ändert Hollywoods ikonisiertes Piktogramm auf der Workbench (V4.5)

### ÜBERSICHT

```
SetWBIcon(icon$[, ...])
```

### PLATTFORMEN

Nur AmigaOS und kompatibel

## BESCHREIBUNG

Dieser Befehl kann verwendet werden, um Ihr benutzerdefiniertes Piktogramm festzulegen, welches Hollywood auf der Workbench anzeigen wird, wenn es sich im ikonifizierten Zustand befindet. Hier müssen Sie eine Piktogramm-Datei angeben, die sich im `*.info`-Format befindet. Alle Piktogramme, die die aktuell installierte Workbench lesen kann, werden unterstützt. Wenn Sie also auf MorphOS sind oder ein entsprechenden Patch verwenden, können Sie hier auch PNG-Piktogramme verwenden.

Für die beste Kompatibilität sollten Sie sich jedoch an die Standardpiktogramme Im GlowIcon-Format halten.

Folgende Sonderkonstanten können in `icon$` übergeben werden:

### #AMIGAICON\_NONE:

Übergeben Sie diese, wenn Sie nicht möchten, dass Hollywood ein App-Piktogramm der Workbench hinzufügt, wenn es ikonisiert ist. (V5.2)

### #AMIGAICON\_SHOW:

Zeigt das App-Piktogramm an. Dies ist nützlich, wenn Sie möchten, dass Ihr App-Piktogramm dauerhaft auf dem Workbench-Bildschirm angezeigt wird und nicht nur, wenn Ihr Programm ikonisiert ist. (V6.1)

### #AMIGAICON\_HIDE:

Blendet das App-Piktogramm aus. (V6.1)

### #AMIGAICON\_SETTITLE:

Setzt einen Text und zeigt ihn unterhalb des App-Piktogrammes an. Es ist der Titel, den Sie bei der Präprozessor-Anweisung `@APPTITLE` angegeben haben. Der zu zeigende Text muss als zweites Argument übergeben werden. (V6.1)

### #AMIGAICON\_SETPOSITION:

Ändert die Position des App-Piktogrammes. Sie müssen zwei zusätzliche Argumente übergeben: Die neue x- und y-Position des App-Piktogrammes. Wenn Sie die beiden zusätzlichen Argumente weglassen, wird die Position des App-Piktogrammes auf die Position zurückgesetzt, die in der `*.info`-Datei gespeichert ist. (V6.1)

Beachten Sie, dass Sie `SetWBIcon()` mehrere Male aufrufen müssen, um den gewünschten Effekt zu erzielen. Wenn Sie zum Beispiel das App-Piktogramm ändern und dauerhaft anzeigen lassen möchten, müssen Sie zuerst `SetWBIcon()` aufrufen, um die Datei `*.info` anzuzeigen und dann müssen Sie `SetWBIcon()` erneut aufrufen und an `#AMIGAICON_SHOW` übergeben, um das App-Piktogramm dauerhaft anzuzeigen.

## EINGABEN

<code>icon\$</code>	zu verwendende Piktogramm-Datei, wenn es ikonisiert ist oder eine spezielle Konstante (siehe oben)
<code>...</code>	zusätzliches Argument in Abhängigkeit von der bestehenden Konstante (siehe oben)

## BEISPIEL

```
SetWBIcon("MyCoolProg.info")
```

Dieser Code verwendet das Piktogramm des Programms als Standard-WB-Piktogramm.





## 43 Pinselbibliothek

### 43.1 Überblick

Der Pinsel ist die flexibelste Bildart in Hollywood. Sie können einen Pinsel erstellen, indem Sie entweder eine Bilddatei mit dem Befehl vom `LoadBrush()` vom Datenträger laden oder mit dem Befehl `CreateBrush()` Bilddaten im Speicher erstellen. Hollywoods Pinselbibliothek enthält eine Vielzahl von Befehlen, die es Ihnen ermöglichen, Pinsel zu verwandeln oder sie mit einer Vielzahl von Bildfiltern zu bearbeiten. Sie können auch in einem Pinsel zeichnen, indem Sie ihn als aktuelles Ausgabegerät auswählen. Dies geschieht mit dem Befehl `SelectBrush()`. Siehe [Abschnitt 43.64 \[SelectBrush\]](#), [Seite 968](#), für Details. Sie können auch mit den Befehlen `SelectMask()` oder `SelectAlphaChannel()` in die Pinselmaske oder den Alphakanal zeichnen.

Hier ist ein kurzer Code, der einen Pinsel aus einer Bilddatei lädt und ihn in die Mitte des Displays zieht:

```
LoadBrush(1, "test.jpg")
DisplayBrush(1, #CENTER, #CENTER)
```

Die meisten anderen Bildtypen in Hollywood können in Pinsel und umgekehrt umgewandelt werden. Aus diesem Grund sind Pinsel die flexibelste Bildart, welche Hollywood bietet. Für hardwarebeschleunigtes Zeichnen unterstützt Hollywood auch Hardware-Pinsel. Siehe [Abschnitt 43.37 \[Informationen über Hardware-Pinsel\]](#), [Seite 940](#), für Details.

Normalerweise enthalten Pinsel Rasterpixeldaten, aber Hollywood unterstützt auch spezielle Vektorpinsel, die stattdessen aus Vektorpfaddaten bestehen und somit frei transformiert werden können.

Normally, brushes contain raster pixel data, but Hollywood also supports special vector brushes which consist of vector path data instead and can thus be freely transformed. Siehe [Abschnitt 43.80 \[Informationen über Vektorpinsel\]](#), [Seite 982](#), für Details.

### 43.2 ArcDistortBrush

#### BEZEICHNUNG

`ArcDistortBrush` – stellt Pinsel als verzerrter Kreisringausschnitt dar (V5.0)

#### ÜBERSICHT

```
ArcDistortBrush(id, angle1[, angle2, rtop, rbottom, smooth])
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, ein verzerrter Kreisringausschnitt auf den Pinsel in `id` anzuwenden. Das Argument `angle1` gibt den Winkel des Kreisrings an, über den der Pinsel verzerrt werden soll. Die optionalen Argumente können die Bogenverzerrung steuern. `angle2` dreht den Pinsel dem Kreis entlang und mit `rtop` und `rbottom` werden die oberen und unteren Radien angepasst. Schließlich kann das optionale Argument `smooth` verwendet werden, um Antialiased Pixelinterpolation zu ermöglichen. Sieht zwar besser aus, die Berechnung des Aussehens dauert aber länger.

Weitere Pinseleffekte: `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`,

EmbossBrush(), FlipBrush() Befehl, GammaBrush(), InvertBrush() Befehl, MixBrush(), ModulateBrush(), OilPaintBrush(), PerspectiveDistortBrush(), PixelateBrush(), PolarDistortBrush(), RotateBrush(), SepiaToneBrush(), ScaleBrush(), SharpenBrush(), SolarizeBrush(), SwirlBrush(), TintBrush(), TransformBrush(), und WaterRippleBrush().

#### EINGABEN

<b>id</b>	Pinsel, der verzerrt werden soll
<b>angle1</b>	Winkel, in dem der Pinsel dargestellt wird
<b>angle2</b>	optional: Drehwinkel, rotiert den Pinsel dem Kreis entlang (voreingestellt ist 0)
<b>rtop</b>	optional: Oberer Kreisbogen des Pinsels wird auf diesen Radius eingestellt (standardmäßig ein automatisch berechneter Wert, der versucht, das Seitenverhältnis so gut wie möglich beizubehalten)
<b>rbottom</b>	optional: Unterer Kreisbogen des Pinsels wird auf diesen Radius eingestellt (standardmäßig ein automatisch berechneter Wert, der versucht, das Seitenverhältnis so gut wie möglich beizubehalten)
<b>smooth</b>	optional: mit <b>True</b> wird Antialiasing beim Verzerren verwendet (Standard ist <b>False</b> )

### 43.3 BarrelDistortBrush

#### BEZEICHNUNG

BarrelDistortBrush – stellt Pinsel als Kissen-/Tonnenverzerrung dar (V5.0)

#### ÜBERSICHT

```
BarrelDistortBrush(id, ...)
BarrelDistortBrush(id, A, B, C, D[, X, Y])
BarrelDistortBrush(id, Ax, Bx, Cx, Dx, Ay, By, Cy, Dy[, X, Y])
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, tonnen- und kissenförmige Verzerrungen (Verzeichnungen) auf den Pinsel in **id** anzuwenden. Sie können diesen Befehl auf zwei verschiedene Arten verwenden: Der erste Weg erfordert, dass Sie mindestens drei Koeffizienten (A, B, C) übergeben, die die Verzerrung definieren. Optional können Sie einen vierten Koeffizienten (D) und einen Mittelpunkt für die radiale Verzerrung (X und Y) angeben. Der Mittelpunkt wird in Pixel übergeben, während die Koeffizienten als Fließkommazahlen angegeben werden müssen. Wenn alle Koeffizienten addiert 1.0 ergeben, wird es keine Änderung in der Abbildung geben.

Der zweite Weg, um diesen Befehl zu verwenden, ist, getrennte Koeffizienten für die x und y-Achse zu liefern. In diesem Fall müssen Sie 8 Koeffizienten angeben (4 für jede Achse). Wie bei der ersten Variante können Sie optional einen Mittelpunkt angeben.

Schließlich kann das optionale Argument **smooth** verwendet werden, um Antialiasing Pixelinterpolation zu ermöglichen, die zu einem glatteren Aussehen führt, aber dessen Berechnung länger dauert.

Weitere Pinseleffekte: `ArcDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `InvertBrush()` Befehl, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `SepiaToneBrush()`, `ScaleBrush()`, `SharpenBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()`, `TransformBrush()`, und `WaterRippleBrush()`.

## EINGABEN

<code>id</code>	Pinsel, der verzerrt wird
<code>...</code>	Koeffizienten für die tonnenförmige Verzerrung (siehe oben)
<code>X</code>	optional: X-Koordinate vom Mittelpunkt (ist standardmäßig die Hälfte der Pinselbreite)
<code>Y</code>	optional: Y-Koordinate vom Mittelpunkt (ist standardmäßig die Hälfte der Pinselhöhe)
<code>smooth</code>	optional: mit <code>True</code> wird Antialiasing beim Verzerren verwendet (Standard ist <code>False</code> )

## 43.4 BGPicToBrush

### BEZEICHNUNG

BGPicToBrush – kopiert ein Hintergrundbild in einen Pinsel

### ÜBERSICHT

`BGPicToBrush(bgpicid, brushid)`

### BESCHREIBUNG

Dieser Befehl erstellt eine Kopie des durch `bgpicid` angegebenen Hintergrundbildes und konvertiert es in einen Pinsel, auf den danach durch die Nummer `brushid` zugegriffen werden kann. Alles wird geklont, deshalb ist der Pinsel unabhängig vom originalen Hintergrundbild (Sie könnten es zum Beispiel nach dieser Operation aus dem Speicher löschen und der Pinsel bleibt trotzdem benutzbar!).

Siehe auch `ConvertToBrush()` und `RasterizeBrush()`.

### EINGABEN

<code>bgpicid</code>	zu klonendes Hintergrundbild
<code>brushid</code>	ID für den neuen Pinsel

### BEISPIEL

```
BGPicToBrush(1,5)
DisplayBrush(5,#CENTER,#CENTER)
```

Obiger Code kopiert Hintergrundbild 1 in Pinsel 5 und stellt den Pinsel dann dar.

## 43.5 BlurBrush

### BEZEICHNUNG

BlurBrush – stellt Pinsel mit Gaußscher Weichzeichnung dar (V5.0)

### ÜBERSICHT

```
BlurBrush(id[, radius])
```

### BESCHREIBUNG

Dieser Befehl wendet die Gaußsche Weichzeichnung (Gaußsche Unschärfe) beim angegebenen Pinsel an. Das optionale Argument **radius** kann verwendet werden, um den Weichzeichnungsradius anzugeben. Je größer der Radius, desto länger braucht dieser Befehl, den Weichzeichnungseffekt anzuwenden. Wenn Sie das optionale Argument **radius** nicht angeben, wird **BlurBrush()** automatisch einen Weichzeichnerradius wählen.

Beachten Sie, dass dieser Befehl nicht mit Palettenpinseln verwendet werden kann.

Weitere Pinseleffekte: **ArcDistortBrush()**, **BarrelDistortBrush()**, **BrushToGray()**, **BrushToMonochrome()**, **CharcoalBrush()**, **ContrastBrush()**, **EdgeBrush()**, **EmbossBrush()**, **FlipBrush()** Befehl, **GammaBrush()**, **InvertBrush()** Befehl, **MixBrush()**, **ModulateBrush()**, **OilPaintBrush()**, **PerspectiveDistortBrush()**, **PixelateBrush()**, **PolarDistortBrush()**, **RotateBrush()**, **SepiaToneBrush()**, **ScaleBrush()**, **SharpenBrush()**, **SolarizeBrush()**, **SwirlBrush()**, **TintBrush()**, **TransformBrush()**, und **WaterRippleBrush()**.

### EINGABEN

<b>id</b>	Pinsel, auf den der Effekt angewandt wird
<b>radius</b>	optional: Weichzeichnungsradius (voreingestellt ist 0, womit Hollywood automatisch einen Radius wählt)

## 43.6 BRUSH

### BEZEICHNUNG

BRUSH – lädt einen Pinsel vor (V2.0)

### ÜBERSICHT

```
@BRUSH id, filename$[, table]
```

### BESCHREIBUNG

Diese Präprozessor-Anweisung lädt den in **filename\$** angegebenen Pinsel und er bekommt die Kennung **id**.

Bildformate, die auf allen Plattformen unterstützt werden, sind PNG, JPEG, BMP, IFF ILBM, GIF und Plugins für Bilder. Je nach Plattform, worauf Hollywood ausgeführt wird, können mehr Bildformate unterstützt werden. Zum Beispiel auf den Amigakompatiblen Systemen wird Hollywood in der Lage sein, alle Formate über Bilddatatypes zu öffnen. Unter Windows kann **@BRUSH** auch Bildformate laden, die von der Windows-Imaging-Komponente unterstützt werden.

Ab Hollywood 5.0 kann dieser Befehl auch Vektorformate wie SVG laden, wenn Sie ein entsprechendes Plugin installieren. Beachten Sie aber, dass wenn Sie Vektorbilder

mit diesen Befehl laden, wird der Pinsel ein spezieller Vektorpinsel sein, der nicht alle Merkmale der normalen Pinsel unterstützt. Sie können jedoch Vektorpinsel mit dem Befehl `RasterizeBrush()` in Rasterpinsel umwandeln. Siehe [Abschnitt 43.80 \[Vektorpinsel\]](#), [Seite 982](#), für weitere Informationen über Vektorpinsel.

Das dritte Argument `table` ist optional. Es ist eine Tabelle, um weitere Möglichkeiten für den Ladevorgang verwenden zu können. Die folgenden Tags stehen dafür zur Verfügung:

**Transparency:**

Dieser Tag kann verwendet werden, um eine Farbe in [RGB-Notation](#) anzugeben, die im Pinsel transparent erscheinen soll.

**LoadAlpha:**

Setzen Sie diesen Tag auf `True`, wenn der Alpha-Kanal des Bildes auch geladen werden soll. Bitte beachten Sie, dass nicht alle Bilder einen Alpha-Kanal haben und dass nicht alle Bildformate in der Lage sind, Alphakanal-daten zu speichern. Es wird vorgeschlagen, dass Sie das PNG-Format verwenden, wenn Sie Alphakanal-daten benötigen. Dieser Tag ist standardmäßig `False`. (V4.5)

**Link:**

Setzen Sie diesen Tag auf `False`, wenn Sie diesen Pinsel beim Kompilieren nicht in die ausführbare Datei/das Applet einbinden wollen. Dieser Tag ist standardmäßig auf `True` gesetzt, was bedeutet, dass der Pinsel mit der ausführbaren Datei/dem Applet beim Kompilieren verknüpft wird.

**X, Y, Width, Height:**

Diese Tags können verwendet werden, um nur einen Teil des Bildes in den Pinsel zu laden. Dies ist nützlich, wenn Sie ein großes Bild mit vielen verschiedenen kleinen Bildern haben und nun die kleinen Bilder in einzelne Pinsel laden wollen. Somit können Sie ein Rechteck innerhalb des Bildes angeben, aus dem Hollywood die Grafikdaten für den Pinsel nimmt.

**Hardware:**

Wenn Sie diesen Tag auf `True` setzen, wird Hollywood diesen Pinsel vollständig im Videospeicher für Hardwarezeichnen erstellen und in Verbindung mit einem Hardwaredoppelpuffer das Zeichnen beschleunigt. Hardware-Pinsel unterliegen verschiedenen Einschränkungen. Siehe [Abschnitt 43.37 \[Hardware-Pinsel\]](#), [Seite 940](#), für Details. (V5.0)

**ScaleWidth, ScaleHeight:**

Diese Tags können verwendet werden, um eine skalierte Version des Bildes zu laden. Wenn der Bildtreiber skaliertes Laden unterstützt, kann das die Geschwindigkeit erheblich steigern, wenn Sie zum Beispiel nur eine Version in Thumbnailgröße eines großen Bildes laden möchten. Wenn der Bildtreiber skaliertes Laden nicht unterstützt, wird zuerst das volle Bild geladen, bevor es skaliert wird. Dies ist nicht wesentlich schneller als das Bild nach dem Laden von Hand zu skalieren. Sie können die Größe entweder als direkter Wert übergeben oder Sie benutzen einen Prozentsatz als Zeichenfolge (z.B. `Scalewidth = "200%"`). (V5.3)

**SmoothScale:**

Wenn **ScaleWidth** oder **ScaleHeight** eingestellt ist, können Sie mit diesem Tag festlegen, ob Hollywood bei der Skalierung Antialiasing verwenden soll oder nicht. Der Standardwert ist **False** und bedeutet kein Antialiasing. Beachten Sie, dass Skalierung mit Antialiasing viel langsamer ist als die normale Skalierung. (V5.3)

**Display:** Wenn Sie hier die ID eines Displays angeben, wird Hollywood einen Displayabhängigen-Hardware-Pinsel für Sie erstellen. Displayabhängige Hardware-Pinsel können nur auf das Display zeichnen, dem sie zugewiesen wurden. Dieser Tag wird nur berücksichtigt, wenn der Tag **Hardware** auf **True** gesetzt wurde. Beachten Sie auch, dass Hollywoods eingebautes Display-Adaptermodul keine Displayabhängigen-Hardware-Pinsel unterstützt, aber Plugins können individuelle Display-Adaptermodule installieren, die displayabhängige Hardware-Pinsel unterstützen. Dieser Tag enthält standardmäßig die ID des aktuell aktiven Displays. Siehe [Abschnitt 43.37 \[Hardware-Pinsel\]](#), [Seite 940](#), für Details. (V6.0)

**Loader:** Mit diesem Tag können Sie ein oder mehrere Formatlademodule angeben, die diesen Pinsel laden sollen. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehrerer Lademodulen enthält. Standardmäßig wird der mit **SetDefaultLoader()** eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen sollen. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen enthält. Standardmäßig wird der mit **SetDefaultAdapter()** eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**LoadTransparency:**

Ist dieser Tag auf **True** gesetzt, wird die monochrome Transparenz des Bildes geladen. Bitte beachten Sie, dass dieser Tag speziell für monochrome Transparenzkanäle ist, das heißt der transparente Stift ist für ein Palettenbasiertes Bild ausgelegt. Wenn Sie den Alpha-Kanal eines Bildes laden möchten, stellen Sie den Tag **LoadAlpha** auf **True**. Dieser Tag ist auf **False** voreingestellt. (V6.0)

**LoadPalette:**

Wenn dieser Tag auf **True** gesetzt ist, lädt Hollywood den Pinsel als Palettenpinsel. Dies bedeutet, dass Sie die Palette des Pinsels abrufen und verändern können, was für bestimmte Effekte wie Farbwechsel nützlich ist. Sie können Stifte auch mit dem Tag **TransparentPen** (siehe unten) oder dem Tag **LoadTransparency** (siehe oben) transparent machen. Palettenpinsel haben auch den Vorteil, dass sie weniger Speicherplatz benötigen, da 1 Pixel nur 1 Byte Speicher anstelle von 4 Byte für 32-Bit-Bilder benötigt. Dieser Tag ist standardmäßig **False**. (V9.0)

**TransparentPen:**

Wenn der Tag `LoadPalette` auf `True` gesetzt wurde (siehe oben), kann mit dem Tag `TransparentPen` ein Stift definiert werden, der transparent gemacht werden soll. Stifte werden ab 0 gezählt. Alternativ können Sie auch den Tag `LoadTransparency` auf `True` setzen, um Hollywood zu zwingen, den transparenten Stift zu verwenden, der in der Bilddatei gespeichert ist (sofern das Bildformat die Speicherung von transparenten Stiften unterstützt). Dieser Tag ist standardmäßig `#NOPEN`. (V9.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Bitte beachten Sie, dass sich die Tags `Transparency`, `LoadTransparency` und `LoadAlpha` gegenseitig ausschließen. Ein Pinsel kann nur eine Transparenzeinstellung haben!

Wenn Sie Pinsel manuell laden möchten, verwenden Sie den Befehl `LoadBrush()`.

Siehe auch `CopyBrush()`, `FreeBrush()`, `SaveBrush()` und `MoveBrush()`.

**EINGABEN**

<code>id</code>	einen Wert, der verwendet wird, um diesen Pinsel später im Code zu identifizieren
<code>filename\$</code>	das Bild, welches Sie laden möchten
<code>table</code>	optionales Argument für weitere Optionen

**BEISPIEL**

```
@BRUSH 1, "MyBrush.png"
```

Lädt "MyBrush.png" als Pinsel 1 ohne Transparenz.

```
@BRUSH 1, "MyBrush.png", {Transparency = $FF0000}
```

Ist das Gleiche wie oben, aber der Pinsel ist jetzt transparent (Transparenzfarbe ist Rot = \$FF0000).

```
@BRUSH 1, "Sprites.png", {X = 64, Y = 32, Width = 32, Height = 32}
```

Lädt einen Ausschnitt von 32x32 Pixel des Bildes "Sprites.png", welcher bei X=64 und Y=32 beginnt.

## 43.7 BrushToGray

**BEZEICHNUNG**

`BrushToGray` – wandelt einen Pinsel in Graustufen um (1.5)

**ÜBERSICHT**

```
BrushToGray(id)
```

**BESCHREIBUNG**

Dieser Befehl wandelt den Pinsel, der durch `id` übergeben wurde, in Graustufen um. Wenn der Pinsel sehr groß ist, kann das ein wenig dauern.

Beachten Sie, dass wenn `id` einen Palettenpinsel angibt, `BrushToGray()` die Palettenfarben nur in Grau umwandelt, was diesen Befehl sehr schnell macht, wenn er mit Palettenpinseln verwendet wird.

Weitere Pinseleffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `InvertBrush()` Befehl, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `SepiaToneBrush()`, `ScaleBrush()`, `SharpenBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()`, `TransformBrush()`, und `WaterRippleBrush()`.

**EINGABEN**

`id` Kennung des zu wandelnden Pinsel

**BEISPIEL**

```
BrushToGray(1)
```

Wandelt Pinsel 1 in Graustufen um.

## 43.8 BrushToMonochrome

**BEZEICHNUNG**

`BrushToMonochrome` – wandelt einen Pinsel in Schwarz/Weiß um (V5.0)

**ÜBERSICHT**

```
BrushToMonochrome(id[, dither])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um einen Pinsel in Schwarz/Weiß abzubilden. Wenn das optionale Argument `dither` auf `True` gesetzt ist, wird Dithering angewendet werden. Dithering ist langsamer, aber erzeugt besser aussehende Bilder.

Beachten Sie, dass dieser Befehl nicht mit Palettenpinseln verwendet werden kann.

Weitere Pinseleffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `InvertBrush()` Befehl, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `SepiaToneBrush()`, `ScaleBrush()`, `SharpenBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()`, `TransformBrush()`, und `WaterRippleBrush()`.

**EINGABEN**

`id` Pinsel, der in Schwarz/Weiß konvertiert wird

`dither` optional: ob Dithering verwendet wird (`True`) oder nicht (`False`) (standardmäßig ist `True` voreingestellt)



## 43.9 BrushToPenArray

### BEZEICHNUNG

BrushToPenArray – wandelt einen Palettenpinsel in ein Stiftfeld um (V9.0)

### ÜBERSICHT

```
table = BrushToPenArray(id)
```

### BESCHREIBUNG

Dieser Befehl kopiert alle Stifte aus dem angegebenen Palettenpinsel in eine Tabelle und gibt diese Tabelle an Sie zurück. Die Tabelle kann als Matrix angesehen werden, die eine Anzahl von Zeilen enthält, die mit der Höhe des Pinsels identisch ist, wobei jede Zeile eine Anzahl von Elementen enthält, die mit der Breite des Pinsels identisch ist. Die Reihenfolge der Pixeldaten in dieser Tabelle ist wie folgt: Zeile für Zeile von oben nach unten, d.h. die Tabelle beginnt mit der ersten Pixelzeile.

Beachten Sie, dass die Zeilen nicht als Untertabellen gespeichert werden. Die von BrushToPenArray() zurückgegebene Tabelle ist eindimensional und enthält genau `height * width`-Elemente, die nacheinander Zeile für Zeile gespeichert werden.

Bitte beachten Sie auch, dass die Tabelle, die Sie von diesem Befehl erhalten, normalerweise viel Speicherplatz beansprucht. Daher sollten Sie diese Tabelle auf `Nil` setzen, sobald Sie sie nicht mehr benötigen. Andernfalls verschwenden Sie sehr viel Speicher und es kann sogar vorkommen, dass Ihrem Skript der Speicherplatz ausgeht. Denken Sie also daran, dass Sie Pixelfeld-Tabellen immer auf `Nil` setzen sollten, sobald Sie sie nicht mehr benötigen.

Um ein Stiftfeld wieder in einen Palettenpinsel zu konvertieren, können Sie den Befehl `PenArrayToBrush()` verwenden. Siehe [Abschnitt 43.48 \[PenArrayToBrush\]](#), Seite 950, für Details.

Wenn Sie RGB-Farben anstelle von Stiftwerten verwenden möchten, müssen Sie stattdessen den Befehl `BrushToRGBArray()` verwenden. Siehe [Abschnitt 43.10 \[BrushToRGBArray\]](#), Seite 913, für Details.

Siehe auch `BrushToPenArray()` und `RGBArrayToBrush()`.

### EINGABEN

<code>id</code>	Identifikator des Palettenpinsels, der in ein Stiftfeld umgewandelt werden soll
-----------------	---

### RÜCKGABEWERTE

<code>table</code>	eine Tabelle, die alle Stifte aus dem Quellpinsel enthält; vergessen Sie nicht, diese Tabelle auf <code>Nil</code> zu setzen, wenn Sie mit ihr fertig sind!
--------------------	---

## 43.10 BrushToRGBArray

### BEZEICHNUNG

BrushToRGBArray – wandelt den Pinsel in ein Pixelarray um (V5.0)

### ÜBERSICHT

```
table = BrushToRGBArray(id[, invalpha])
```

**BESCHREIBUNG**

Dieser Befehl kopiert alle Pixel aus dem angegebenen Pinsel in eine Tabelle und gibt Ihnen diese in `table` zurück. Die Tabelle kann als Matrix mit einer Anzahl von Zeilen angesehen werden, die der Höhe des Pinsels entspricht, wobei jede Zeile eine Anzahl von Elementen enthält, die der Breite des Pinsels entspricht. Die Reihenfolge der Pixeldaten in dieser Tabelle ist wie folgt: Zeile für Zeile von oben nach unten, d.h. die Tabelle beginnt mit der ersten Pixelzeile. Die einzelnen Pixel werden als ARGB-Werte gespeichert.

Beachten Sie, dass die Zeilen nicht als Untertabellen gespeichert werden. Die von `BrushToRGBArray()` zurückgegebene Tabelle ist eindimensional und enthält genau `height * width`-Elemente, die nacheinander, Zeile für Zeile gespeichert werden.

Das optionale Argument `invalpha` kann verwendet werden, damit `BrushToRGBArray()` alle Alphakanalwerte invertiert. Dies bedeutet, dass ein Wert von 0 hundertprozentig sichtbar und ein Wert von 255 völlig unsichtbar ist. Normalerweise ist es genau umgekehrt. Aus historischen Gründen verwendet die Darstellungsbibliothek von Hollywood **invertierte Alphawerte** und aus diesem Grund wird dies auch von `BrushToRGBArray()` unterstützt, obwohl es nicht der Standard ist.

Bitte beachten Sie, dass die Tabelle, die Sie von diesem Befehl erhalten, in der Regel viel Speicher in Anspruch nimmt. Daher sollten Sie diese Tabelle auf `Nil` setzen, sobald Sie sie nicht mehr benötigen. Andernfalls werden Sie riesige Mengen an Speicher verschwenden und es könnte sogar passieren, dass Ihrem Skript kein Speicher mehr zur Verfügung steht.

Um ein Pixelarray zurück zu einem Pinsel zu konvertieren, können Sie den Befehl `RGBArrayToBrush()` verwenden.

Beachten Sie, dass es für Palettenpinsel auch den Befehl `BrushToPenArray()` gibt, die die Stiftwerte des Pinsels anstelle der RGB-Farben zurückgibt. Siehe **Abschnitt 43.9 [BrushToPenArray]**, Seite 913, für Details.

Siehe auch `PenArrayToBrush()` und `RGBArrayToBrush()`.

**EINGABEN**

<code>id</code>	Identifikator des Pinsels
<code>invalpha</code>	optional: auf <code>True</code> gesetzt, werden Alphawerte invertiert (Standard ist <code>False</code> )

**RÜCKGABEWERTE**

<code>table</code>	eine Tabelle, die alle Pixel vom Pinsel enthält; vergessen Sie nicht, diese Tabelle auf <code>Nil</code> zu setzen, wenn Sie sie nicht mehr benötigen!
--------------------	--

## 43.11 ChangeBrushTransparency

**BEZEICHNUNG**

ChangeBrushTransparency – ändert den Transparenzmodus des Pinsels (V5.0)

**ÜBERSICHT**

`ChangeBrushTransparency(id, mode)`

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um den Transparenzmodus eines Pinsels zu ändern. Hollywood unterstützt derzeit drei verschiedene Transparenz-Modi:

- #NONE:** Keine Transparenz. Der gesamte Pinsel ist sichtbar.
- #MASK:** Monochrome Transparenz. Jedes Pixel kann entweder sichtbar oder unsichtbar sein.
- #ALPHACHANNEL:** Schrittweise Transparenz. Jedes Pixel kann 256 verschiedene Transparenzstufen haben. Ein Alphakanalwert von 0 bedeutet, volle Transparenz, während ein Alphakanalwert von 255 keine Transparenz darstellt.

`ChangeBrushTransparency()` ist besonders nützlich, sobald Sie zwischen den Betriebsarten **#MASK** und **#ALPHACHANNEL** wechseln. Wenn Sie zum Beispiel einen Pinsel mit `LoadBrush()` laden und Sie mit dem Tag **Transparency** eine Farbe transparent machen, werden Sie immer einen Pinsel mit dem Transparenzmodus **#MASK** erhalten. In einigen Fällen wollen Sie jedoch den Pinsel stattdessen mit **#ALPHACHANNEL** verwenden, damit Sie die Werte mit `SelectAlphaChannel()` ändern können. In diesem Fall kann `ChangeBrushTransparency()` sehr hilfreich sein.

Beachten Sie, dass dieser Befehl nicht mit Palettenpinseln verwendet werden kann.

Siehe auch `DeleteAlphaChannel()`, `DeleteMask()`, `InvertAlphaChannel()`, `InvertBrush()` Befehl, `InvertMask()`, `IsBrushEmpty()`, `ReduceAlphaChannel()`, `SetAlphaIntensity()`, `SetBrushTransparency()`, `SetBrushTransparentPen()` und `SetMaskMode()`.

#### EINGABEN

- id** Pinsel, dessen Transparentmodus Sie ändern möchten
- mode** neuer Transparentmodus; kann **#NONE**, **#MASK** oder **#ALPHACHANNEL** sein

#### BEISPIEL

```
LoadBrush(1, "test.iff", {Transparency = #RED})
ChangeBrushTransparency(1, #ALPHACHANNEL)
```

Der obige Code lädt das Bild "test.iff" als Pinsel 1, macht die Farbe Rot transparent und ändert dann den Transparenzmodus von **#MASK** zu **#ALPHACHANNEL**.

## 43.12 CharcoalBrush

#### BEZEICHNUNG

CharcoalBrush – stellt Pinsel als Kohlezeichnung dar (V5.0)

#### ÜBERSICHT

```
CharcoalBrush(id, radius)
```

#### BESCHREIBUNG

Dieser Befehl zeigt den Pinsel mit einem Holzkohle-Effekt. Das optionale Argument **radius** kann verwendet werden, um den Kohlenzeichnungsradius anzugeben. Je größer der Radius, desto länger braucht dieser Befehl, den Holzkohle-Effekt anzuwenden.

Beachten Sie, dass dieser Befehl nicht mit Palettenpinseln verwendet werden kann.

Weitere Pinseffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `InvertBrush()` Befehl,

MixBrush(), ModulateBrush(), OilPaintBrush(), PerspectiveDistortBrush(), PixelateBrush(), PolarDistortBrush(), RotateBrush(), SepiaToneBrush(), ScaleBrush(), SharpenBrush(), SolarizeBrush(), SwirlBrush(), TintBrush(), TransformBrush(), und WaterRippleBrush().

#### EINGABEN

**id**            Pinsel, der den Holzkohle-Effekt erhält  
**radius**       Radius des Holzkohle-Effektes

### 43.13 ContrastBrush

#### BEZEICHNUNG

ContrastBrush – verbessert oder verringert den Pinselkontrast (V5.0)

#### ÜBERSICHT

ContrastBrush(id, inc[, repeat])

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Farbkontrast des angegebenen Pinsels zu verbessern oder zu reduzieren. Wenn das Argument **inc** auf **True** gesetzt ist, wird der Kontrast erhöht. Wenn er **False** ist, wird der Kontrast verringert. Das optionale Argument **repeat** kann verwendet werden, um die Wirkung auf dem Pinsel mehrfach anzuwenden. Dies ist nützlich, wenn Sie schärfere Kontraste erstellen möchten.

Beachten Sie, dass wenn **id** einen Palettenpinsel angibt, **ContrastBrush()** nur den Kontrast auf die Palettenfarben anwendet, was dieser Befehl sehr schnell macht, wenn er mit Palettenpinseln verwendet wird.

Weitere Pinseffekte: ArcDistortBrush(), BarrelDistortBrush(), BlurBrush(), BrushToGray(), BrushToMonochrome(), CharcoalBrush(), EdgeBrush(), EmbossBrush(), FlipBrush() Befehl, GammaBrush(), InvertBrush() Befehl, MixBrush(), ModulateBrush(), OilPaintBrush(), PerspectiveDistortBrush(), PixelateBrush(), PolarDistortBrush(), RotateBrush(), SepiaToneBrush(), ScaleBrush(), SharpenBrush(), SolarizeBrush(), SwirlBrush(), TintBrush(), TransformBrush(), und WaterRippleBrush().

#### EINGABEN

**id**            Pinsel, der bearbeitet wird  
**inc**            **True**, um den Kontrast zu erhöhen, **False**, um den Kontrast zu verringern  
**repeat**       optional: gibt an, wie viele Male der Kontrast wiederholt wird (Standard: 1, der den Effekt nur einmal ausführt)

### 43.14 ConvertToBrush

#### BEZEICHNUNG

ConvertToBrush – konvertiert ein Objekt in einen Pinsel (V2.5)

**ÜBERSICHT**

```
[id] = ConvertToBrush(sourcetype, sourceid, dest[, t])
```

**BESCHREIBUNG**

Mit diesem Befehl können Sie einen neuen Pinsel aus einem vorhandenen Grafikobjekt erstellen. Dies ist zum Beispiel nützlich, um die Bilddaten von einzelnen Anim- oder Sprite-Einzelbilder in einen Pinsel zu kopieren. Sie könnten den Pinsel nun modifizieren, um ihn dann in eine Animation oder einen Sprite zurück zu wandeln. Sie können auch mit diesem Befehl auf die Grafiken von Ebenen und anderen Bildtypen zugreifen.

Grafiken als Pinsel zu handhaben, ist bequem, weil Pinsel der flexibelste Grafiktyp in Hollywood ist. Die meisten Manipulationen der Bildbefehle arbeiten nur mit einem Pinsel. Deshalb will man oft die Grafikdaten ins Pinselformat konvertieren.

Das Argument **sourcetype** gibt den Typ des Quellobjekts an, welches in einen Pinsel umgewandelt werden soll. Folgende Typen sind möglich:

- #ANIM**      Erstellt ein neuer Pinsel aus einem einzigen Einzelbild aus einem Animationsobjekt. Standardmäßig wird das erste Einzelbild in einen Pinsel umgewandelt. Sie können dies ändern, indem Sie den Tag **Frame** im optionalen Tabellenargument übergeben (siehe unten).
- #BGPIC**    Erstellt ein neuer Pinsel aus einem Hintergrundbild.
- #BRUSH**    Erstellt ein neuer Pinsel aus einem anderen Pinsel. Ist das selbe wie der Befehl **CopyBrush()**.
- #ICON**     Erstellt einen neuen Pinsel aus einem Bild in einem Piktogramm. Da ein Piktogramm mehrere Bilder enthalten können, können Sie den Tag **Frame** des optionalen Tabellenarguments verwenden, um den Index des Bildes anzugeben, das in einen Pinsel umgewandelt werden soll. Standardmäßig wird das erste Bild im Piktogramm in einen Pinsel umgewandelt. Sie können auch den Tag **Selected** des optionalen Tabellenarguments verwenden, um anzugeben, ob das ausgewählte Piktogrammbild in einen Pinsel umgewandelt werden soll oder nicht. Standardmäßig wird das normale Bild in einen Pinsel umgewandelt. (V8.0)
- #LAYER**    Erstellt ein neuer Pinsel aus einer Ebene (Ebenen müssen aktiviert sein!). Wenn die Ebene eine Animebene ist, können Sie den Tag **Frame** im optionalen Tabellenargument verwenden, um anzugeben, welches Einzelbild der Animationsebene in einen Pinsel umgewandelt werden soll (siehe unten). Standardmäßig wird das erste Einzelbild konvertiert.
- #SPRITE**   Erstellt ein neuer Pinsel aus einem einzigen Einzelbild eines Spriteobjekts. Standardmäßig wird das erste Sprite-Einzelbild in einen Pinsel umgewandelt. Sie können dies ändern, indem Sie den Tag **Frame** im optionalen Tabellenargument **t** übergeben (siehe unten).
- #TEXTOBJECT**  
               Erstellt ein neuer Pinsel aus einem Textobjekt.
- #VECTORPATH**  
               Erstellt einen Vektorpinsel aus einem oder mehreren Pfadobjekt(en). Wenn Sie diesen Typ verwenden, wird das Argument **sourceid** nicht verwendet.

Stattdessen müssen Sie ein Tabellenargument im Tag **Path** im optionalen Tabellenargument **t** übergeben. Diese Tabelle muss Informationen über die einzelnen Pfade enthalten, die in den neuen Vektorpinsel eingebettet werden sollen. Die Tabelle verwendet dasselbe Layout wie die Tabelle, die Sie an den Befehl **PathToBrush()** übergeben müssen. Siehe [Abschnitt 53.26 \[PathToBrush\]](#), [Seite 1199](#), für Details. (V7.0)

Mit dem optionalen Argument **t** können Sie die folgenden zusätzlichen Optionen übergeben:

**Frame:** Wenn der Quelltyp ein Grafikobjekt mit mehreren Einzelbilder angibt, können Sie diesen Tag verwenden, um das Einzelbild anzugeben, der in einen Pinsel umgewandelt werden soll. Einzelbilder werden von 1 bis zur Anzahl der Einzelbilder gezählt. Dieser Tag ist standardmäßig auf 1 eingestellt.

**Selected:** Wenn der Quelltyp **#ICON** ist, können Sie mit diesem Tag angeben, ob das ausgewählte oder normale Bild in einen Pinsel umgewandelt werden soll. Piktogrammbilder haben zwei Zustände: Normal und ausgewählt. Wenn Sie **Selected** auf **True** setzen, wird das ausgewählte Bild in einen Pinsel umgewandelt. Andernfalls konvertiert **ConvertToBrush()** das normale Bild in einen Pinsel. Der Standardwert ist **False** (V8.0).

**Path:** Wenn der Quelltyp **#VECTORPATH** ist, müssen Sie diesen Tag auf eine Tabelle setzen, die Informationen über die einzelnen Pfade enthält, die in den neuen Vektorpinsel eingebettet werden sollen. Die Tabelle verwendet dasselbe Layout wie die Tabelle, die Sie an den Befehl **PathToBrush()** übergeben. Siehe [Abschnitt 53.26 \[PathToBrush\]](#), [Seite 1199](#), für Details. (V7.0)

**Vector:** Standardmäßig wandelt **ConvertToBrush()** Vektorbilder in Rasterpinsel um. Wenn Sie sie in Vektorpinsel umwandeln möchten, setzen Sie diesen Tag auf **True**. Dadurch ist es möglich, Vektortextobjekte oder Vektoranimations-Einzelbilder in Vektorpinsel umzuwandeln, die ohne Qualitätsverlust skaliert und gedreht werden können. Standardwert ist **False** (V10.0).

Siehe auch **BGPicToBrush()** und **RasterizeBrush()**.

## EINGABEN

<b>sourcetype</b>	Typ des Quellobjekts (siehe Liste oben)
<b>sourceid</b>	Identifikator des Quellobjekts
<b>dest</b>	ID für den zu erstellenden Pinsel oder <b>Nil</b> für die <b>automatische ID-Zuweisung</b>
<b>t</b>	optional: Tabellenargument mit weiteren Optionen (siehe oben)

## RÜCKGABEWERTE

<b>id</b>	optional: ID des Pinsels; wird nur zurückgegeben, wenn Sie <b>Nil</b> in <b>dest</b> angegeben haben (siehe oben)
-----------	---

**BEISPIEL**

```
ConvertToBrush(#SPRITE, 1, 10, {Frame = 5})
```

Der obige Code erstellt einen neuen Pinsel mit der ID 10 von dem Einzelbild 5 der Spritenummer 1.

## 43.15 CopyBrush

**BEZEICHNUNG**

CopyBrush – kopiert einen Pinsel (V1.5)

**ÜBERSICHT**

```
[id] = CopyBrush(source, dest[, table])
```

**BESCHREIBUNG**

Dieser Befehl kloniert den durch **source** angegebenen Pinsel und erzeugt eine Kopie in Pinsel **dest**. Der neue Pinsel ist unabhängig vom alten Pinsel, deshalb könnten Sie den alten nach dem Kopieren löschen.

Wenn Sie beim Argument **dest** **Nil** angeben, wird **CopyBrush()** für den kopierten Pinsel automatisch eine ID auswählen und Ihnen übergeben.

Ab 5.0 kann diesem Befehl eine optionale Tabelle **table** mit folgenden Tags übergeben werden:

**Hardware:**

Wenn Sie diesen Tag auf **True** setzten, wird Hollywood diesen Pinsel vollständig im Videospeicher erstellen. Dieser Hardware-Pinsel beschleunigt in Verbindung mit einem Hardware-Doppelpuffer das Zeichnen dieses Pinsels. Hardware-Pinsel unterliegen verschiedenen Einschränkungen. Siehe [Abschnitt 43.37 \[Hardware-Pinsel\]](#), [Seite 940](#), für Details. (V5.0)

**Display:** Wenn Sie hier die ID eines Displays angeben, wird Hollywood einen Displayabhängigen-Hardware-Pinsel für Sie erstellen. Displayabhängige Hardware-Pinsel können nur auf dem Display, zu dem sie gehören, verwendet werden. Dieser Tag wird nur berücksichtigt, wenn der Tag **Hardware** auf **True** gesetzt wurde. Beachten Sie auch, dass Hollywoods eingebautes Display-Adaptermodul keine Displayabhängigen-Hardware-Pinsel unterstützt, aber Plugins können individuelle Display-Adaptermodule installieren, die displayabhängige Hardware-Pinsel unterstützen. Dieser Tag benutzt standardmäßig die ID des aktuell aktiven Displays. Siehe [Abschnitt 43.37 \[Hardware-Pinsel\]](#), [Seite 940](#), für Details. (V6.0)

**SmoothScale:**

Wenn Sie diesen Tag auf **True** setzen und der Tag **Hardware** ebenfalls auf **True** gesetzt wurde, verwendet Hollywood (oder das Display-Adaptermodul) bei der Transformation des neu erstellten Pinsels die bilineare Interpolation. Normalerweise wird beim Aufruf eines Pinseltransformationsbefehls wie [ScaleBrush\(\)](#) oder [RotateBrush\(\)](#) festgelegt, ob Interpolation verwendet werden soll oder nicht. Einige Display-Adaptermodule müssen diese Informationen jedoch bereits beim Erstellen des Hardware-Pinsels kennen. Aus diesem

Grund ist dieser Tag hier, obwohl er wahrscheinlich nicht sehr nützlich ist, weil er nur in speziellen Situationen mit Display-Adaptermodule wie RebelSDL oder Hardware-Pinseln unter Android benötigt wird. Normalerweise können Sie einfach angeben, ob die Interpolation direkt im Transformationsbefehl verwendet werden soll oder nicht. Beachten Sie, dass `SmoothScale` nur unterstützt wird, wenn `Hardware` auf `True` gesetzt ist. (V8.0)

Siehe auch `@BRUSH`, `LoadBrush()`, `FreeBrush()`, `SaveBrush()` und `MoveBrush()`.

#### EINGABEN

**source**      Identifikator des Quellpinsels

**dest**        ID des neuen Pinsel oder `Nil` für die automatische ID-Zuweisung

**table**       optional: Tabelle, um weitere Konfiguration vorzunehmen (V5.0)

#### RÜCKGABEWERTE

**id**            optional: Identifikator des Pinsels. Wird nur zurückgegeben werden, wenn Sie `Nil` als Argument 1 angegeben haben (siehe oben)

#### BEISPIEL

```
CopyBrush(1, 10)
FreeBrush(1)
```

Der obige Code erstellt einen neuen Pinsel 10, der die gleichen Grafikdaten wie Pinsel 1 enthält. Dann wird der Pinsel 1 aus dem Speicher gelöscht, da er nicht mehr benötigt wird.

## 43.16 CreateBorderBrush

#### BEZEICHNUNG

CreateBorderBrush – erstellt von einem Pinsel einen Randpinsel (V5.0)

#### ÜBERSICHT

```
[id] = CreateBorderBrush(id, src, color[, size])
```

#### BESCHREIBUNG

Dieser Befehl erstellt einen Rand von dem in `src` angegeben Pinsel und kopiert ihn als neuen Randpinsel, welcher in `id` angegeben ist. Wenn Sie beim Argument `id` `Nil` angeben, wird `CreatBorderBrush()` für den kopierten Pinsel automatisch eine ID auswählen und Ihnen übergeben. Im Argument `color` übergeben Sie die Randfarbe. Dies muss eine Farbe in `ARGB-Notation` sein, somit können Sie hier auch eine Transparenz verwenden. Schließlich kann das optionale Argument `size` verwendet werden, um die Randdicke anzugeben.

Beachten Sie, dass das Argument `size` (Größe) nicht die absolute Breite oder Höhe angibt, sondern ein relativer Faktor ist, mit dem der Rand beim Quellepinsel auf jeder Seite ergänzt wird. Das bedeutet, dass die Breite des Randpinsels die Quellpinselbreite plus zweimal `size` ist und das Gleiche gilt für die Höhe des Randes.

Siehe auch `CreateShadowBrush()` und `CreateTexturedBrush()`.



**EINGABEN**

<b>id</b>	ID des neuen Randpinsels oder <b>Nil</b> für die automatische ID-Zuweisung
<b>src</b>	ID des Pinsels, von dem ein Randpinsel erstellt wird
<b>color</b>	gewünschte Rahmenfarbe als <b>ARGB-Wert</b>
<b>size</b>	optional: Raddicke (voreingestellt ist 5)

**RÜCKGABEWERTE**

<b>id</b>	optional: ID des neuen BGPic oder <b>Nil</b> für die automatische ID-Zuweisung
-----------	--

**BEISPIEL**

```
bordersize = 20
CreateBrush(1, 320, 240, #RED)
CreateBorderBrush(2, 1, #BLACK, bordersize)
DisplayBrush(2, 0, 0)
DisplayBrush(1, bordersize, bordersize)
```

Der obige Code erzeugt einen Rand für ein 320x240 großes rotes Rechteck und zeigt es darunter an.

## 43.17 CreateBrush

**BEZEICHNUNG**

CreateBrush – erzeugt einen leeren Pinsel (V1.5)

**ÜBERSICHT**

```
[id] = CreateBrush(id, width, height[, color], table))
```

**BESCHREIBUNG**

Dieser Befehl erzeugt einen neuen Pinsel mit der angegebenen Höhe **height** und Breite **width** und füllt ihn mit der angegebenen Farbe **color**. Wird keine Farbe angegeben, wird ein schwarzer Pinsel erzeugt. Wenn Sie beim Argument **id** **Nil** angeben, wird CreateBrush() für den Pinsel automatisch eine ID auswählen und Ihnen übergeben.

Ab Hollywood 4.5 gibt es ein optionales Tabellenargument **table**, womit Sie Transparenz für diesen Pinsel verwenden können. Die folgenden Tags können Sie der optionalen Tabelle übergeben:

**Mask:** Setzen Sie diesen Tag auf **True**, wenn CreateBrush() eine Maske als Transparenz für den neuen Pinsel verwenden soll. Wenn dieser Tag **True** ist, muss AlphaChannel **False** sein. Der Standardwert ist **False**.

**AlphaChannel:** Setzen Sie diesen Tag auf **True**, wenn CreateBrush() einen Alphakanal als Transparenz für den neuen Pinsel verwenden soll. Wenn dieser Tag **True** ist, muss Mask **False** sein. Der Standardwert ist **False**.

**Clear:** Dieser Tag wird nur berücksichtigt, wenn entweder AlphaChannel oder Mask auf **True** gesetzt wurde. Ist dies der Fall, gibt Clear an, ob die Maske oder der Alphakanal vollständig transparent oder undurchsichtig sein soll. Der Standardwert ist **False**, was bedeutet, dass die neue Maske oder Alphakanal undurchsichtig ist.

**Hardware:**

Wenn Sie diesen Tag auf **True** setzten, wird Hollywood diesen Pinsel vollständig im Videospeicher erstellen. Dieser Hardware-Pinsel beschleunigt in Verbindung mit einem Hardware-Doppelpuffer das Zeichnen dieses Pinsels. Hardware-Pinsel unterliegen verschiedenen Einschränkungen. Siehe [Abschnitt 43.37 \[Hardware-Pinsel\]](#), Seite 940, für Details. (V5.0)

**Display:** Wenn Sie die ID eines Displays hier angeben, wird Hollywood einen Displayabhängigen-Hardware-Pinsel für Sie erstellen. Displayabhängige Hardware-Pinsel können nur auf dem Display, zu dem sie gehören, verwendet werden. Dieser Tag wird nur berücksichtigt, wenn der Tag **Hardware** auf **True** gesetzt wurde. Beachten Sie auch, dass Hollywoods eingebautes Display-Adaptermodul keine Displayabhängigen-Hardware-Pinsel unterstützt, aber Plugins können individuelle Display-Adaptermodule installieren, die displayabhängige Hardware-Pinsel unterstützen. Dieser Tag benutzt standardmäßig die ID des aktuell aktiven Displays. Siehe [Abschnitt 43.37 \[Hardware-Pinsel\]](#), Seite 940, für Details. (V6.0)

**SmoothScale:**

Wenn Sie diesen Tag auf **True** setzen und der Tag **Hardware** ebenfalls auf **True** gesetzt wurde, verwendet Hollywood (oder das Display-Adaptermodul) bei der Transformation des neu erstellten Pinsels die bilineare Interpolation. Normalerweise wird beim Aufruf eines Pinseltransformationsbefehls wie [ScaleBrush\(\)](#) oder [RotateBrush\(\)](#) festgelegt, ob Interpolation verwendet werden soll oder nicht. Einige Display-Adaptermodule müssen diese Informationen jedoch bereits beim Erstellen des Hardware-Pinsels kennen. Aus diesem Grund ist dieser Tag hier, obwohl er wahrscheinlich nicht sehr nützlich ist, weil er nur in speziellen Situationen mit Display-Adaptermodule wie RebelSDL oder Hardware-Pinseln unter Android benötigt wird. Normalerweise können Sie einfach angeben, ob die Interpolation direkt im Transformationsbefehl verwendet werden soll oder nicht. Beachten Sie, dass **SmoothScale** nur unterstützt wird, wenn **Hardware** auf **True** gesetzt ist. (V8.0)

**Palette:** Wenn dieser Tag auf den Identifikator einer Palette gesetzt ist, erstellt Hollywood einen Palettenpinsel für Sie. Paletten können mit Befehlen wie [CreatePalette\(\)](#) oder [LoadPalette\(\)](#) erstellt werden. Alternativ können Sie diesen Tag auch auf eine von Hollywoods integrierten Paletten setzen, z.B. **#PALETTE\_AGA**. Siehe [Abschnitt 41.36 \[SetStandardPalette\]](#), Seite 872, für eine Liste der integrierten Paletten. (V9.0)

**FillPen:** Wenn der Tag **Palette** gesetzt ist (siehe oben), können Sie mit diesem Tag den Stift einstellen, der zum Füllen des Pinselhintergrunds verwendet werden soll. Beachten Sie, dass der Parameter **color**, der an [CreateBrush\(\)](#) übergeben wird, ignoriert wird, wenn **Palette True** ist. Aus diesem Grund ist dieser Tag hier, damit Sie einen Stift angeben können, der beim Initialisieren der Pixel des Pinsels verwendet wird. Voreingestellt ist 0. (V9.0)

**TransparentPen:**

Wenn **Palette** auf **True** gesetzt ist, kann mit diesem Tag ein Stift angegeben werden, der im neuen Pinsel transparent werden soll. Voreingestellt ist

`#NOPEN`, was bedeutet, dass kein transparenter Stift vorhanden sein sollte. (V9.0)

**Depth:** Mit diesem Tag können Sie die gewünschte Farbtiefe des Pinsels festlegen. Wenn dieser Wert kleiner oder gleich 8 ist, erstellt `CreateBrush()` einen Palettenpinsel. Sie können den Tag `Palette` auch zusammen mit dem Tag `Depth` angeben. Wenn die angegebene Palette mehr Farben als die angegebene Tiefe hat, werden diese Farben verworfen. Wenn er weniger Farben enthält, werden die nicht verwendeten Stifte auf Schwarz gesetzt. Standardmäßig erstellt `CreateBrush()` 24-Bit- oder 32-Bit-Pinsel, je nachdem, ob der Tag `AlphaChannel` auf `True` oder `False` gesetzt ist. (V10.0)

**Callback:** Wenn Sie diesen Tag auf eine Callback-Funktion setzen, erstellt `CreateBrush()` einen benutzerdefinierten Pinsel für Sie. Im Vergleich zu normalen Pinseln werden benutzerdefinierte Pinsel von einer Callback-Funktion unterstützt, die Hollywood immer dann aufruft, wenn sich die Abmessungen des Pinsels ändern oder eine Transformation angewendet wird. Auf diese Weise können Sie Pinsel erstellen, die sich dynamisch an neue Auflösungen und Transformationen anpassen. Dies ist dem, was Vektorpinsel tun, sehr ähnlich, außer dass Sie mit benutzerdefinierten Pinseln die volle Kontrolle über den Prozess übernehmen können, da Ihre Callback-Funktion das gesamte Zeichnen übernimmt. Ihre Callback-Funktion wird auch sofort von `CreateBrush()` aufgerufen, um das anfängliche Zeichnen des Pinsels durchzuführen.

Benutzerdefinierte Pinsel können sehr nützlich sein, um Ihren eigenen Vektorpinseltyp zu implementieren. Da benutzerdefinierte Pinsel die Möglichkeit haben, sich selbst neu zu zeichnen, wenn sich ihre Auflösung oder Transformation ändert, können Sie sie verwenden, um Pinsel zu erstellen, die ohne Qualitätsverlust skaliert oder transformiert werden können. Der Grund ist, da Ihre Callback-Funktion die Grafiken immer dann neu zeichnet, wenn sich die Pinselabmessungen ändern, anstatt nur die Pixel mit herkömmlicher verlustbehafteter Größenveränderung zu skalieren. Dies ist besonders nützlich, wenn Sie die **Ebenenskalisierung** mit benutzerdefinierten Grafiken verwenden. Wenn Sie benutzerdefinierte Pinsel für eine Ebene verwenden, können Sie sicher sein, dass diese Ebene verlustfrei auf alle Auflösungen skaliert wird.

Die Callback-Funktion, die Sie in `Callback` angeben, erhält als Parameter 1 eine Nachricht mit den folgenden initialisierten Feldern:

**Action:** Initialisiert auf `Draw`.

**ID:** Identifikator des Pinsels, mit dem gezeichnet werden soll. Beachten Sie, dass dies nicht dieselbe ID ist wie der Pinsel, den Sie mit `CreateBrush()` erstellt haben. Ihre Callback-Funktion muss `SelectBrush()` auf dem in ID angegebenen Pinsel aufrufen und die gewünschten Grafiken unter Berücksichtigung der aktuellen Transformation (siehe unten) auf den Pinsel zeichnen. Sie können auch `SelectMask()` und `SelectAlphaChannel()` für den

Pinsel aufrufen, falls Sie die Transparenzeinstellungen anpassen müssen.

**Width:** Aktuelle Breite des Pinsels.

**Height:** Aktuelle Höhe des Pinsels.

**SX:** Gibt den Skalierungsbetrag auf der x-Achse an. Wenn er negativ ist, wird das Bild auf der y-Achse gespiegelt. Dieser Wert wird niemals 0 sein.

**RX:** Gibt den Grad der Drehung auf der x-Achse an. Dies kann 0 sein.

**RY:** Gibt den Grad der Drehung auf der y-Achse an. Dies kann 0 sein.

**SY:** Gibt den Skalierungsbetrag auf der y-Achse an. Wenn er negativ ist, wird das Bild auf der x-Achse gespiegelt. Dieser Wert wird niemals 0 sein.

**UserData:** Dies wird auf die Benutzerdaten gesetzt, die Sie im Tag **UserData** des optionalen Tabellenarguments übergeben haben, das von **CreateBrush()** unterstützt wird (siehe unten). Wenn Sie an **CreateBrush()** keine Benutzerdaten übergeben haben, wird dieser Tag nicht gesetzt.

(V10.0)

**UserData:**

Wenn Sie den Tag **Callback** gesetzt haben (siehe oben), können Sie diesen Tag verwenden, um einige Benutzerdaten zu speichern, die an Ihre Callback-Funktion weitergegeben werden, wenn Hollywood sie aufruft. Die Benutzerdaten können von beliebigem Typ sein. (V10.0)

Siehe auch **CreateGradientBrush()**.

## EINGABEN

<b>id</b>	ID des neuen Randpinsels oder <b>Nil</b> für die automatische ID-Zuweisung
<b>width</b>	Breite des Pinsels
<b>height</b>	Höhe des Pinsels
<b>color</b>	optional: <b>RGB-Farbe</b> für den Pinsel (voreingestellt ist <b>#BLACK</b> )
<b>table</b>	optional: Tabelle für weitere Optionen (siehe oben) (V4.5)

## RÜCKGABEWERTE

<b>id</b>	optional: Identifikator des Pinsels; Wird nur zurückgegeben werden, wenn Sie <b>Nil</b> als Argument 1 angegeben haben (siehe oben)
-----------	---

## BEISPIEL

```
CreateBrush(2, 320, 256, #BLUE)
```

Der obige Code erstellt einen neuen blauen Pinsel mit der ID 2 und den Dimension von 320x256.

```
CreateBrush(2, 320, 256, #BLUE, {AlphaChannel = True, Clear = True})
```

Der obige Code erstellt einen neuen blauen Pinsel mit ID 2 in einer Größe von 320x256. Der neue Pinsel erhält auch einen Alphakanal, dessen Transparenz auf 100% gesetzt wird. Wenn Sie also den neuen Pinsel anzuzeigen, werden Sie nichts sehen, weil der Pinsel zur Zeit vollkommen transparent ist.

```
CreateBrush(1, 320, 240, 0, {AlphaChannel = True, Clear = True, Callback =
    Function(msg)
        SelectBrush(msg.id, #SELMODE_COMBO)
        SetFormStyle(#ANTIALIAS)
        SetFillStyle(#FILLCOLOR)
        Ellipse(0, 0, msg.width / 2, msg.height / 2, #RED)
        EndSelect
    EndFunction
})
ScaleBrush(1, 640, 480)
DisplayBrush(1, 0, 0)
```

Der obige Code zeigt, wie Sie einen benutzerdefinierten Pinsel erstellen. Er erstellt einen Pinsel, der in seiner Callback-Funktion eine Ellipse mit Anti-Aliasing zeichnet. Da die Callback-Funktion immer dann aufgerufen wird, wenn sich die Abmessungen des Pinsels ändern, skaliert die Ellipse wie ein echtes Vektorbild und ist in allen Auflösungen perfekt scharf.

## 43.18 CreateGradientBrush

### BEZEICHNUNG

CreateGradientBrush – erstellt einen Pinsel mit Farbverlauf (V5.0)

### ÜBERSICHT

```
[id] = CreateGradientBrush(id, width, height, type, startcolor,
                           endcolor[, angle, table])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen neuen Pinsel mit einem Farbverlauf zu erstellen. Wenn Sie beim Argument `id` `Nil` angeben, wird `CreatBrush()` für den Pinsel automatisch eine Zahl auswählen und Ihnen übergeben. Die gewünschte Abmessung geben Sie mit `width` (Breite) und `height` (Höhe) an. Mit `type` können Sie die Art des Farbverlaufs angeben. Folgende Farbverläufe stehen zur Zeit zur Verfügung: `#LINEAR`, `#RADIAL` und `#CONICAL`. Den Drehwinkel (in Grad) des Verlaufs geben Sie in `angle` an, welcher nur durch die beiden Arten `#LINEAR` und `#CONICAL` unterstützt werden. Der Farbverlauf `#RADIAL` kann nicht gedreht werden.

Das optionale Argument `table` kann verwendet werden, um erweiterte Optionen anzugeben:

**CenterX, CenterY:**

Diese beiden Tags können verwendet werden, um den Mittelpunkt des Farbverlaufs anzugeben. Lineare Verläufe haben keinen Mittelpunkt, darum werden diese beiden Variablen nur dann berücksichtigt, wenn Sie Farbverläufe vom Typ **#RADIAL** oder **#CONICAL** verwenden. Der Mittelpunkt muss als Fließkommawert angegeben werden, der zwischen 0.0 (linke/obere Ecke) und 1.0 (rechts/unten Ecke) ist. Wenn nichts angegeben wurde, wird als Standard 0.5 genommen, so dass der Mittelpunkt des Verlaufes in der Mitte des Bildes ist. (V5.0)

**Border:** Dieser Tag kann verwendet werden, um die Rahmengröße für den Verlauf des Typs **#RADIAL** einzustellen. Für die anderen Verlaufstypen wird dieser Tag ignoriert. Die Rahmengröße des radialen Verlaufes muss ein Fließkommawert zwischen 0.0 und 1.0 sein. Voreingestellt ist 0.0, welches keinen Rahmen bedeutet.

**Balance:** Dieser Tag kann verwendet werden, um den Gleichgewichtspunkt für den Verlauf des Typs **#CONICAL** einzustellen. Für die anderen Verlaufstypen wird dieser Tag ignoriert. Der Gleichgewichtspunkt des konischen Verlaufes muss ein Fließkommawert zwischen 0.0 und 1.0 sein. Der Standardwert ist 0.5. Beachten Sie, dass dies nur bei der Erstellung eines zweifarbigen Farbverlaufs verwendet wird. Bei der Erstellung eines mehrfarbigen Farbverlaufs mit Hilfe der Tabelle **Colors** wird **Balance** ignoriert, da Sie mit der Tabelle **Colors** die Farben im Farbverlauf über Farbstopps individuell ausgleichen können.

**Colors:** Mit diesem Tag können Sie Verläufe erstellen, die mehrere Farben enthalten. Dieser Tag ist eine Tabelle, die eine Folge von abwechselnden Farben und Stoppwerte enthält. Die Farben müssen im **RGB-Format** angegeben werden. Der Stoppwert ist ein Fließkommawert zwischen 0.0 und 1.0 und definiert die Position, wo die entsprechende Farben in dem Verlauf verschmolzen werden. Eine Position von 0.0 bedeutet die Startposition des Verlaufes und eine Position von 1.0 ist die Endposition. Bitte beachten Sie, dass die Stoppositionen in aufsteigender Reihenfolge sortiert werden müssen, das heißt von 0.0 bis 1.0. Wenn Sie diese Felder angeben, werden die Farben in den Argumenten **startcolor** und **endcolor** ignoriert und Hollywood wird nur die in diesem Tag angegebenen Farben verwenden. (V5.0)

Siehe auch **CreateBrush()**.

**EINGABEN**

<b>id</b>	ID des neuen Farbverlaufspinsel oder <b>Nil</b> für die <b>automatische ID-Zuweisung</b>
<b>width</b>	Breite des Pinsels
<b>height</b>	Höhe des Pinsels
<b>type</b>	Typ des Farbverlaufs; siehe oben für die verschiedenen Typen
<b>startcolor</b>	<b>RGB-Wert</b> für die Startfarbe
<b>endcolor</b>	<b>RGB-Wert</b> als Endfarbe

<b>angle</b>	Drehwinkel für den Farbverlauf (Voreingestellt ist 0)
<b>table</b>	Tabelle, welche weitere Optionen ermöglicht; siehe oben für eine Beschreibung der verfügbaren Optionen

**RÜCKGABEWERTE**

<b>id</b>	optional: Identifikator des Pinsels; Wird nur zurückgegeben werden, wenn Sie <b>Nil</b> als Argument 1 angegeben haben (siehe oben)
-----------	---

**BEISPIEL**

Siehe [Abschnitt 30.6 \[CreateGradientBGPic\]](#), Seite 624.

## 43.19 CreateShadowBrush

**BEZEICHNUNG**

CreateShadowBrush – erstellt aus einem Pinsel einen Schattenpinsel (V5.0)

**ÜBERSICHT**

```
[id] = CreateShadowBrush(id, src, color[, size])
```

**BESCHREIBUNG**

Dieser Befehl erstellt einen Schlagschatten von dem in **src** angegeben Pinsel und kopiert ihn als neuen Schattenpinsel, welcher in **id** angegeben ist. Wenn Sie beim Argument **id** **Nil** angeben, wird CreateShadowBrush() für den kopierten Pinsel automatisch eine Zahl auswählen und Ihnen übergeben. Im Argument **color** übergeben Sie die Schattenfarbe. In den meisten Fällen wird es **#BLACK** kombiniert mit einem Transparenzwert sein, da undurchsichtige Schatten nicht sehr gut aussehen. Somit geben Sie die Farbe und die Transparenz in der **ARGB-Notation** an. Schließlich kann das optionale Argument **size** verwendet werden, um die Größe des Schlagschattens anzugeben.

Beachten Sie, dass das Argument **size** (Größe) nicht die absolute Breite oder Höhe angibt, sondern einen relativen Faktor, mit dem der Schatten beim Quellpinsel auf jeder Seite ergänzt wird. Das bedeutet, dass die Breite des Schattenpinsels die Quellpinselbreite plus zweimal **size** ist und das Gleiche gilt für die Höhe des Schattens.

Siehe auch [CreateBorderBrush\(\)](#) und [CreateTexturedBrush\(\)](#).

**EINGABEN**

<b>id</b>	ID des neuen Schattenpinsels oder <b>Nil</b> für die <b>automatische ID-Zuweisung</b>
<b>src</b>	Quellpinsel, von dem ein Schatten erstellt wird
<b>color</b>	gewünschte Schattenfarbe als <b>ARGB-Wert</b>
<b>size</b>	optional: gewünschte Schattengröße (voreingestellt ist 5)

**RÜCKGABEWERTE**

<b>id</b>	optional: Identifikator des Pinsels; Wird nur zurückgegeben werden, wenn Sie <b>Nil</b> als Argument 1 angegeben haben (siehe oben)
-----------	---

**BEISPIEL**

```
shadowsize = 20
CreateBrush(1, 320, 240, #RED)
```

```
CreateShadowBrush(2, 1, ARGB(40, #BLACK), shadowsize)
DisplayBrush(2, 0, 0)
DisplayBrush(1, shadowsize, shadowsize)
```

Der obige Code erzeugt einen Schlagschatten für ein 320x240 großes rotes Rechteck und zeigt es an.

## 43.20 CreateTexturedBrush

### BEZEICHNUNG

CreateTexturedBrush – erstellt einen Texturpinsel (V5.0)

### ÜBERSICHT

```
[id] = CreateTexturedBrush(id, brushid, width, height[, x, y])
```

### BESCHREIBUNG

Mit diesem Befehl wird ein neuer Pinsel mit der Textur vom Pinsel `brushid` erstellt. Wenn Sie beim Argument `id` `Nil` angeben, wird `CreatTexturedBrush()` für den kopierten Pinsel automatisch eine Zahl auswählen und Ihnen übergeben. Die gewünschte Abmessung geben Sie mit `width` (Breite) und `height` (Höhe) an. Mit den optionalen Parametern `x` und `y` können Sie einen Versatz in dem Texturpinsel angeben. Die Texturierung wird dann von diesem Punkt versetzt im Pinsel beginnen. Der Standardwert für diese Argumente ist 0/0 und bedeutet, dass die Textur in der linken oberen Ecke vom Texturpinsel beginnt.

Siehe auch `CreateBorderBrush()` und `CreateShadowBrush()`.

### EINGABEN

<code>id</code>	ID des neuen Texturpinsels oder <code>Nil</code> für die automatische ID-Zuweisung
<code>brushid</code>	ID des Pinsels, den Sie als Textur verwenden
<code>width</code>	Breite des neuen Pinsels
<code>height</code>	Höhe des neuen Pinsels
<code>x</code>	optional: Startpunkt x im Texturpinsel
<code>y</code>	optional: Startpunkt y im Texturpinsel

### RÜCKGABEWERTE

<code>id</code>	optional: Identifikator des Pinsels; Wird nur zurückgegeben werden, wenn Sie <code>Nil</code> als Argument 1 angegeben haben (siehe oben)
-----------------	---

## 43.21 CropBrush

### BEZEICHNUNG

CropBrush – stutzt einen Pinsel (V2.0)

### ÜBERSICHT

```
CropBrush(id, x, y, width, height)
```



**BESCHREIBUNG**

Dieser Befehl schneidet/stutzt den in `id` angegebenen Pinsel von der Position `x` und `y` an auf die neue Dimension von `width` (Breite) und `height` (Höhe). Wenn der Pinsel eine Maske oder einen Alphakanal hat, werden diese auch gestutzt werden.

Siehe auch `TrimBrush()`.

**EINGABEN**

<code>id</code>	Pinsel, der gestutzt/zugeschnitten wird
<code>x</code>	Position x, wo das Stutzen/Zuschneiden startet
<code>y</code>	Position y, wo das Stutzen/Zuschneiden beginnt
<code>width</code>	Breite des Zuschnittes
<code>height</code>	Höhe des Zuschnittes

**BEISPIEL**

```
CreateBrush(1, 200, 200)
SelectBrush(1)
Circle(50, 50, 50, #RED)
EndSelect
```

```
CropBrush(1, 50, 50, 101, 101)
DisplayBrush(1, #CENTER, #CENTER)
```

Erstellt einen neuen Pinsel und zeichnet in die Mitte einen roten Kreis. Danach wird die umgebende leere Fläche des Kreises abgeschnitten.

## 43.22 DeleteAlphaChannel

**BEZEICHNUNG**

DeleteAlphaChannel – entfernt den Alphakanal des Pinsels (V2.0)

**ÜBERSICHT**

```
DeleteAlphaChannel(id)
```

**BESCHREIBUNG**

Dieser Befehl entfernt den Alphakanal des in `id` angegebenen Pinsels. Siehe [Abschnitt 43.63 \[SelectAlphaChannel\]](#), [Seite 966](#), für weitere allgemeine Informationen über Alphakanäle.

Siehe auch `ChangeBrushTransparency()`, `DeleteMask()`, `InvertAlphaChannel()`, `InvertBrush()` Befehl, `InvertMask()`, `IsBrushEmpty()`, `ReduceAlphaChannel()`, `SetAlphaIntensity()`, `SetBrushTransparency()`, `SetBrushTransparentPen()` und `SetMaskMode()`.

**EINGABEN**

<code>id</code>	Pinsel, dessen Alphakanal entfernt werden soll
-----------------	--

## 43.23 DeleteMask

### BEZEICHNUNG

DeleteMask – entfernt die Maske des Pinsels (V2.0)

### ÜBERSICHT

DeleteMask(id)

### BESCHREIBUNG

Dieser Befehl entfernt die Maske des in `id` angegebenen Pinsels, das heißt der Pinsel erscheint dann undurchsichtig.

Siehe auch `ChangeBrushTransparency()`, `DeleteAlphaChannel()`, `InvertAlphaChannel()`, `InvertBrush()` Befehl, `InvertMask()`, `IsBrushEmpty()`, `ReduceAlphaChannel()`, `SetAlphaIntensity()`, `SetBrushTransparency()`, `SetBrushTransparentPen()` und `SetMaskMode()`.

### EINGABEN

`id`            Pinsel, dessen Maske entfernt werden soll.

## 43.24 DisplayBrush

### BEZEICHNUNG

DisplayBrush – stellt einen Pinsel dar

### ÜBERSICHT

DisplayBrush(id, x, y[, table])

### BESCHREIBUNG

Dieser Befehl stellt den durch `id` angegebenen Pinsel an den durch `x` und `y` angegebenen Koordinaten dar.

Wenn Ebenen aktiviert sind, wird dieser Befehl eine neue Ebene des Typs `#BRUSH` dem Ebenenstapel hinzuzufügen.

Ab Hollywood 4.0 kann mit der optionale Tabelle `table` auch eine oder mehrere der [Standard-Tags für alle Zeichnungsbefehle](#) benutzt werden. Siehe [Abschnitt 29.17 \[Standard-Tags zum Zeichnen\]](#), Seite 613, für weitere Informationen über die Standard-Tags, die fast alle Hollywood Zeichnungsbefehle unterstützen.

Siehe auch `DisplayBrushFX()`, `DisplayBrushPart()` und `MoveBrush()`.

### EINGABEN

`id`            ID des Pinsels, der dargestellt werden soll  
`x`            x-Koordinaten des Pinsels  
`y`            y-Koordinaten des Pinsels  
`table`       optional: Tabelle für weitere Optionen (siehe oben) (V4.0)

### BEISPIEL

DisplayBrush(1, #CENTER, #CENTER)

Zeigt Pinsel 1 zentriert auf dem Bildschirm an.

```
DisplayBrush(1, 0, 0, {Width = 640, Height = 480})
```

Zeigt Pinsel 1 auf 640x480 skaliert an.

## 43.25 DisplayBrushFX

### BEZEICHNUNG

DisplayBrushFX – zeigt einen Pinsel mit einem Übergangseffekt an

### ÜBERSICHT

```
[handle] = DisplayBrushFX(id, x, y[, table])
```

### BESCHREIBUNG

Dieser Befehl ist eine erweiterte Version des [DisplayBrush\(\)](#) Befehls. Sie zeigt den durch `id` angegebenen Pinsel an der durch `x/y` angegebenen Position an und benutzt dazu einen der vielen Übergangseffekte, die Hollywood besitzt (angegeben in `Type`). Sie müssen ebenfalls die Übergangsgeschwindigkeit angeben.

Wenn Ebenen aktiviert sind, wird dieser Befehl eine neue Ebene des Typs `#BRUSH` dem Ebenenstapel hinzuzufügen.

Ab Hollywood 4.0 verwendet dieser Befehl eine neue Syntax. Die alte Syntax wird aus Kompatibilitätsgründen weiterhin unterstützt. Das optionale Tabellenargument kann den Übergangseffekt konfigurieren. Folgende Optionen sind möglich:

- Type:** Hier geben Sie den gewünschten Übergangseffekt an. Siehe [Abschnitt 30.11 \[DisplayTransitionFX\]](#), [Seite 630](#), für eine Liste aller unterstützten Übergangseffekte. (Standardeinstellung ist `#RANDOMPARAMETER`)
- Speed:** Legt die gewünschte Geschwindigkeit für den Übergang fest. Je höher der Wert, den Sie hier angeben, desto schneller wird der Effekt angezeigt werden. (Standardeinstellung ist `#NORMALSPEED`)
- Parameter:** Einige Übergangseffekte akzeptieren einen zusätzlichen Parameter, der hier angegeben werden kann. (Standardeinstellung ist `#RANDOMPARAMETER`)
- Async:** Sie können diesen Tag verwenden, um ein asynchrones Zeichnungsobjekt für diesen Übergang zu erstellen. Wenn Sie hier `True` angeben, wird `DisplayBGPicPartFX()` sofort verlassen und es wird ein Handler für das asynchrone Zeichnungsobjekt kreiert, den Sie dann mit dem Befehl `AsyncDrawFrame()` verwenden können. Ein Beispielskript finden Sie unter dem Befehl `AsyncDrawFrame()`. Siehe [Abschnitt 19.1 \[AsyncDrawFrame\]](#), [Seite 231](#), für weitere Informationen über asynchrone Zeichnungsobjekte.

Siehe auch [DisplayBrush\(\)](#), [DisplayBrushPart\(\)](#) und [MoveBrush\(\)](#).

### EINGABEN

<code>id</code>	ID des Pinsels
<code>x</code>	x-Koordinaten des Pinsels
<code>y</code>	y-Koordinaten des Pinsels

`table` optional: Konfigurationen für den Übergangseffekt

## RÜCKGABEWERTE

`handle` optional: Handler auf ein asynchrones Zeichnungsobjekt; wird nur zurückgegeben, wenn `Async` auf `True` gesetzt wurde (siehe oben)

## BEISPIEL

```
DisplayBrushFX(1, 0, 0, #VLINES, 10) ; alte Syntax
```

OR

```
DisplayBrushFX(1, 0, 0, {Type = #VLINES, Speed = 10}) ; neue Syntax
```

Der obige Code zeigt den Pinsel 1 bei 0:0 mit einem `#VLINES` Übergang mit der Geschwindigkeit 10 an.

## 43.26 DisplayBrushPart

### BEZEICHNUNG

`DisplayBrushPart` – stellt einen Ausschnitt des Pinsels dar

### ÜBERSICHT

```
DisplayBrushPart(id, srcx, srcy, destx, desty, width, height[, table])
```

### BESCHREIBUNG

Dieser Befehl stellt einen Ausschnitt des durch `id` angegebenen Pinsels auf dem Bildschirm dar. Der Ausschnitt wird durch `srcx`, `srcy` und seiner Breite `width` sowie Höhe `height` definiert. Er wird im Display an den Koordinaten `destx` und `desty` erscheinen.

Wenn Ebenen aktiviert sind, wird dieser Befehl eine neue Ebene des Typs `#BRUSHPART` dem Ebenenstapel hinzuzufügen.

Ab Hollywood 4.0 kann mit der optionale Tabelle `table` auch eine oder mehrere der [Standard-Tags für alle Zeichnungsbefehle](#) benutzt werden. Siehe [Abschnitt 29.17 \[Standard-Tags zum Zeichnen\]](#), Seite 613, für weitere Informationen über die Standard-Tags, die fast alle Hollywood Zeichnungsbefehle unterstützen.

Siehe auch [DisplayBrush\(\)](#), [DisplayBrushFX\(\)](#) und [MoveBrush\(\)](#).

### EINGABEN

<code>id</code>	ID des Pinsels, der als Quelle benutzt werden soll
<code>srcx</code>	linke Kante im Pinsel
<code>srcy</code>	obere Kante im Pinsel
<code>destx</code>	gewünschte X-Position des Pinsels auf dem Bildschirm
<code>desty</code>	gewünschte Y-Position des Pinsels auf dem Bildschirm
<code>width</code>	Breite des Ausschnitts
<code>height</code>	Höhe des Ausschnitts
<code>table</code>	optional: Tabelle für weitere Optionen (siehe oben) (V4.0)

**BEISPIEL**

```
DisplayBrushPart(1,0,0,50,50,100,100)
```

Zeigt die ersten 100 Pixel und Zeilen vom Pinsel 2 auf dem Bildschirm an der Position 50,50 an.

**43.27 EdgeBrush****BEZEICHNUNG**

EdgeBrush – betont die Kanten innerhalb eines Pinsels (V5.0)

**ÜBERSICHT**

```
EdgeBrush(id[, radius])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die Kanten eines Pinsels zu betonen. Das optionale Argument **radius** kann dazu verwendet werden, um den Radius anzugeben. Je größer der Radius, desto länger braucht dieser Befehl, den Kantenbetonungseffekt anzuwenden. Wenn Sie bei **radius** nichts angeben, wird **EdgeBrush()** automatisch einen entsprechenden Radius auswählen.

Beachten Sie, dass dieser Befehl nicht mit Palettenpinseln verwendet werden kann.

Weitere Pinseffekte: **ArcDistortBrush()**, **BarrelDistortBrush()**, **BlurBrush()**, **BrushToGray()**, **BrushToMonochrome()**, **CharcoalBrush()**, **ContrastBrush()**, **EmbossBrush()**, **FlipBrush()** Befehl, **GammaBrush()**, **InvertBrush()** Befehl, **MixBrush()**, **ModulateBrush()**, **OilPaintBrush()**, **PerspectiveDistortBrush()**, **PixelateBrush()**, **PolarDistortBrush()**, **RotateBrush()**, **SepiaToneBrush()**, **ScaleBrush()**, **SharpenBrush()**, **SolarizeBrush()**, **SwirlBrush()**, **TintBrush()**, **TransformBrush()**, und **WaterRippleBrush()**.

**EINGABEN**

<b>id</b>	ID des Pinsels
<b>radius</b>	optional: Radius des Kantenbetonungseffekts (voreingestellt ist 0, das bedeutet, dass der Radius automatisch ausgewählt wird)

**43.28 EmbossBrush****BEZEICHNUNG**

EmbossBrush – stellt Pinsel als Relief dar (V5.0)

**ÜBERSICHT**

```
EmbossBrush(id[, radius])
```

**BESCHREIBUNG**

Dieser Befehl legt einen Reliefeffekt auf den angegebenen Pinsel. Das optionale Argument **radius** kann dazu verwendet werden, den Radius anzugeben. Je größer der Radius, desto länger braucht dieser Befehl, den Reliefeffekt anzuwenden. Wenn Sie bei **radius** nichts angeben, wird **EmbossBrush()** automatisch einen entsprechenden Radius auswählen.

Beachten Sie, dass dieser Befehl nicht mit Palettenpinseln verwendet werden kann.

Weitere Pinseleffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `InvertBrush()` Befehl, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `SepiaToneBrush()`, `ScaleBrush()`, `SharpenBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()`, `TransformBrush()`, und `WaterRippleBrush()`.

#### EINGABEN

<code>id</code>	ID des Pinsels
<code>radius</code>	optional: Radius des Reliefeffekts (voreingestellt ist 0, das bedeutet, dass der Radius automatisch ausgewählt wird)

## 43.29 EndSelect

#### BEZEICHNUNG

`EndSelect` – selektiert das Display als Ausgabeziel (V1.5)

#### ÜBERSICHT

`EndSelect()`

#### BESCHREIBUNG

Dieser Befehl beendet Grafikausgaben abseits des Bildschirms, indem sie wieder das Display als Ausgabeziel einstellt. Die Ausgaben aller Befehle, die Grafik ausgeben, werden nun wieder direkt auf dem Display erscheinen.

Beachten Sie, dass `EndSelect()` nicht mit `SelectDisplay()` benutzt werden kann.

Bitte beachten Sie außerdem, dass `EndSelect()` immer das Display als Ausgabeziel wählen wird, auch wenn es nicht das Ausgabeziel war, als `SelectXXX()` aufgerufen wurde.

```
SelectBrush(1)
Box(0, 0, 100, 100, #RED)    ; zeichnet ein Rechteck in Pinsel #1
SelectBrush(2)
Box(0, 0, 100, 100, #GREEN) ; zeichnet ein Rechteck in Pinsel #2
EndSelect                   ; wählt Display (nicht Pinsel #1 !!!)
EndSelect                   ; Fehler! Display ist bereits gewählt!
```

Siehe auch `SelectAlphaChannel()`, `SelectBrush()`, `SelectMask()`, `SelectLayer()`, `SelectBGPic()` und `SelectAnim()`.

#### EINGABEN

keine

#### BEISPIEL

Siehe Abschnitt 30.15 [`SelectBGPic`], Seite 639.

Siehe Abschnitt 43.64 [`SelectBrush`], Seite 968.

Siehe Abschnitt 17.22 [`SelectAnim`], Seite 212.

Siehe Abschnitt 25.39 [`SelectLayer`], Seite 436.

### 43.30 ExtendBrush

#### BEZEICHNUNG

ExtendBrush – vergrößert einen Pinsel (V10.0)

#### ÜBERSICHT

ExtendBrush(id, left, top, right, bottom[, t])

#### BESCHREIBUNG

Dieser Befehl vergrößert (die Fläche) den durch `id` angegebenen Pinsel. Sie können die gewünschte Ausdehnung für alle Seiten des Pinsels festlegen, indem Sie einen Pixelwert in den Parametern `left`, `top`, `right` und `bottom` übergeben. Ein Wert von 0 bedeutet keine Ausdehnung auf dieser Seite. Beachten Sie, dass die Ausdehnungswerte nicht negativ sein dürfen. Wenn Sie einen Pinsel zuschneiden möchten, verwenden Sie stattdessen den Befehl `CropBrush()`.

Das optionale Argument `t` kann verwendet werden, um die folgenden zusätzlichen Optionen anzugeben:

- Clear:** Wenn der Pinsel Transparenz verwendet, können Sie diesen Tag auf `True` setzen, wenn Sie möchten, dass die neu zugewiesenen Bereiche des Pinsels ebenfalls transparent sind. Wenn Sie diesen Tag auf `False` setzen, sind die neuen Bereiche undurchsichtig. Der Standardwert ist `False`.
- Color:** Mit diesem Tag kann die Füllfarbe für die neu zugewiesenen Bereiche des Pinsels festgelegt werden. Dies wird nur für RGB-Pinsel verwendet. Benutzen Sie für Palettenpinsel stattdessen den Tag `Pen` (siehe unten). Die Voreinstellung ist `#BLACK`.
- Pen:** Mit diesem Tag kann der Füllstift für die neu zugewiesenen Bereiche des Pinsels angegeben werden. Dies wird nur für Palettenpinsel verwendet. Benutzen Sie für RGB-Pinsel stattdessen den Tag `Color` (siehe oben). Voreingestellt ist 0.

#### EINGABEN

<code>id</code>	Pinsel zum Vergrößern
<code>left</code>	Anzahl der Pixel, die auf der linken Seite hinzugefügt werden sollen
<code>top</code>	Anzahl der Pixel, die auf der oberen Seite hinzugefügt werden sollen
<code>right</code>	Anzahl der Pixel, die auf der rechten Seite hinzugefügt werden sollen
<code>bottom</code>	Anzahl der Pixel, die auf der unteren Seite hinzugefügt werden sollen
<code>t</code>	optional: Tabelle zur Angabe weiterer Optionen (siehe oben)

#### BEISPIEL

```
CreateBrush(1, 200, 200)
ExtendBrush(1, 50, 50, 50, 50, {Color = #RED})
```

Der obige Code erstellt einen Pinsel mit einer Größe von 200x200 Pixeln und fügt dann einen roten Rand von 50 Pixeln um ihn herum hinzu.

### 43.31 FlipBrush

#### BEZEICHNUNG

FlipBrush – spiegelt einen Pinsel (V1.5)

#### ÜBERSICHT

FlipBrush(id, xflip)

#### BESCHREIBUNG

Dieser Befehl spiegelt den Pinsel, der durch `id` angegeben wurde. Wenn `xflip` auf `True` gesetzt ist, wird der Pinsel in x-Richtung gespiegelt, ansonsten in y-Richtung.

Weitere Pinseleffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `GammaBrush()`, `InvertBrush()` Befehl, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `SepiaToneBrush()`, `ScaleBrush()`, `SharpenBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()`, `TransformBrush()`, und `WaterRippleBrush()`.

#### EINGABEN

`id` zu spiegelnder Pinsel

`xflip` `True` für eine horizontale Spiegelung, `False` für eine vertikale Spiegelung

#### BEISPIEL

FlipBrush(1, TRUE)

Der obige Code spiegelt den Pinsel 1 horizontal.

### 43.32 FloodFill

#### BEZEICHNUNG

FloodFill – füllt einen Pinselbereich mit einer Farbe (V2.0)

#### ÜBERSICHT

FloodFill(id, x, y, bordercolor, color[, t])

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen begrenzten Bereich in dem in `id` angegebenen Pinsel mit der Farbe `color` zu füllen. Sie müssen eine Startposition in den Argumenten `x` und `y` angeben. `FloodFill()` startet dann in alle Richtungen und ersetzt dann die Pixel mit der angegebenen Farbe, bis sie die Grenzfarbe `bordercolor` erreicht. Die Ausgangslage ist in der Regel ein beliebiger Punkt innerhalb des begrenzten Bereichs. Wenn der Pinsel ein Palettenpinsel ist, arbeitet `FloodFill()` im Stiftmodus und nicht im Farbmodus. Das bedeutet, dass sowohl das Argument `bordercolor` als auch `color` auf einen Stiftindex anstelle einer RGB-Farbe gesetzt werden müssen. Im Stiftmodus verhält sich `FloodFill()` genauso wie im RGB-Modus, außer dass Stifte anstelle von RGB-Farben verwendet werden.

Ab Hollywood 9.0 können Sie auch `#NOCOLOR` im Argument `bordercolor` übergeben. In diesem Fall wird eine randlose Füllung verwendet, was bedeutet, dass alle benachbarten Pixel, die der Farbe (oder dem Stift) des Startpixels entsprechen, gefüllt werden.



Darüber hinaus führt Hollywood 9.0 ein optionales Tabellenargument ein, mit dem Sie die folgenden Optionen angeben können:

**AlphaChannel:**

Wenn Sie diesen Tag auf **True** setzen, arbeitet **FloodFill()** auf dem Alphakanal des Pinsels statt auf seinen Farbkanälen. Dies bedeutet, dass Sie Werte im Bereich von 0 bis 255 anstelle von RGB-Farben an **FloodFill()** übergeben müssen.

**ColorSource:**

Wenn dies und der Tag **AlphaChannel** auf **True** gesetzt sind, wird der zu füllende Bereich durch die Farbkanäle bestimmt, während die gesamte Ausgabe in den Alphakanal geschrieben wird. Das bedeutet, dass die an **FloodFill()** übergebene Rahmenfarbe eine RGB-Farbe (oder **#NOCOLOR**) und die Füllfarbe ein Alphawert zwischen 0 und 255 sein muss. (V9.1)

Siehe auch **QuantizeBrush()**, **ReadBrushPixel()**, **ReplaceColors()** und **WriteBrushPixel()**.

## EINGABEN

<b>id</b>	Pinsel, den Sie füllen möchten
<b>x</b>	Startposition x für den Füllvorgang
<b>y</b>	Startposition y für den Füllvorgang
<b>bordercolor</b>	Farbe (oder Stift) des Rahmens oder <b>#NOCOLOR</b> eine randlose Füllung
<b>color</b>	Farbe (oder Stift) für die Füllung
<b>t</b>	optional: Tabellenargument mit weiteren Optionen (V9.0)

## BEISPIEL

```
CreateBrush(1, 241, 201)
SelectBrush(1)
SetFillStyle(#FILLNONE)
Ellipse(0, 0, 120, 100, #RED)
EndSelect
FloodFill(1, 120, 100, #RED, #WHITE)
DisplayBrush(1, 0, 0)
```

Es wird eine rote Ellipse mit einer Kontur erzeugt, die dann mit der Farbe Weiß gefüllt wird. **FloodFill()** startet vom Zentrum der Ellipse aus in alle Richtungen.

## 43.33 FreeBrush

### BEZEICHNUNG

**FreeBrush** – gibt den Speicher eines Pinsels frei

### ÜBERSICHT

**FreeBrush(id)**

**BESCHREIBUNG**

Dieser Befehl löscht den Speicher, der von dem durch `id` angegebenen Pinsel belegt wird. Um den Speicherverbrauch zu reduzieren, sollten Sie Pinsel aus dem Speicher löschen, wenn Sie sie nicht mehr benötigen.

Siehe auch `@BRUSH`, `LoadBrush()`, `CopyBrush()`, `SaveBrush()` und `MoveBrush()`.

**EINGABEN**

`id`            Identifikator des Pinsels

## 43.34 GammaBrush

**BEZEICHNUNG**

GammaBrush – korrigiert die Gammawerte des Pinsels (V5.0)

**ÜBERSICHT**

`GammaBrush(id, red, green, blue)`

**BESCHREIBUNG**

Dieser Befehl kann die Gammafarbkanäle des angegebenen Pinsels ändern. Für jeden Farbkanal müssen Sie einen Fließkommawert übergeben, der die gewünschte Gamma-korrektur angibt. Ein Wert von 1.0 bedeutet keine Veränderung, ein Wert kleiner als 1.0 dunkelt den Kanal ab und ein Wert von mehr als 1.0 wird den Kanal aufhellen.

Beachten Sie, dass wenn `id` einen Palettenpinsel angibt, `InvertBrush()` nur die Gamma-korrektur auf die Palettenfarben anwendet, was diesen Befehl bei Verwendung mit Palettenpinseln sehr schnell macht.

Weitere Pinseffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `InvertBrush()` Befehl, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `SepiaToneBrush()`, `ScaleBrush()`, `SharpenBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()`, `TransformBrush()`, und `WaterRippleBrush()`.

**EINGABEN**

`id`            Pinsel für die Gammakorrektur

`red`           Gammakorrektur für den roten Kanal

`green`        Gammakorrektur für den grünen Kanal

`blue`         Gammakorrektur für den blauen Kanal

**BEISPIEL**

`GammaBrush(1, 1.5, 1.0, 0.5)`

Der obige Code hellt den roten Kanal auf, der blaue wird abgedunkelt, während der grüne Farbkanal unangetastet bleibt.

## 43.35 GetBrushLink

### BEZEICHNUNG

GetBrushLink – erzeugt einen Link auf einen Pinsel (V1.5)

### ÜBERSICHT

GetBrushLink(id, sourcetype, sourceid[, par])

### BESCHREIBUNG

**Hinweis ab Hollywood 2.0:** Pinsellinks werden nicht mehr unterstützt. Sie können immer noch diesen Befehl verwenden, aber es werden keine Links mehr erstellt. Es wird einfach eine vollständige Kopie der Bilddaten generiert; mit anderen Worten: `GetBrushLink()` ruft nur den Befehl `ConvertToBrush()` auf.

Dieser Befehl erzeugt Ihnen einen neuen Pinsel, der auf die Grafikdaten eines anderen Objekts zurückgreift. Daher können Sie den neuen Pinsel nur lesen. Das bedeutet, dass Sie ihn z.B. darstellen und bewegen können, aber nicht seine Daten ändern (z.B. durch Aufruf von `ScaleBrush()` oder `SelectBrush()`).

Ihr Pinsel ist voll abhängig vom Quellobjekt. Wenn Sie das Quellobjekt löschen, wird der Pinsel ebenfalls freigegeben und ist nicht länger verfügbar. Links auf Pinsel benötigen nur sehr wenig Speicher, da die Grafikdaten alle vom Quellobjekt geholt werden.

Es ist hilfreich Links auf Pinsel zu verwenden, wenn Sie viele Objekte mit denselben Grafikdaten haben, auf die Sie über mehrere getrennte Pinsel zugreifen wollen (z.B. weil es so einfacher ist). Ein weiterer guter Grund ist, dass Sie mit Pinseln weit mehr Dinge anstellen können als mit anderen Objekten. Zum Beispiel könnten Sie einen Link auf das erste Bild einer Animation erfragen, es dann mit `DisplayBrushFX()` darstellen und erst dann mit `PlayAnim()` die Animation starten. Das würde dann die Animation mit einem Übergangseffekt darstellen.

Sourcetype kann einer der folgenden Typen sein:

#ANIM	erzeugt einen Link auf ein einzelnes Animations-Einzelbild; dieser Typ benötigt das optionale Argument <code>par</code> , welches das Einzelbild angibt, welches Sie verlinkt haben möchten
#BGPIC	erzeugt einen Link auf ein Hintergrundbild
#BRUSH	erzeugt einen Link auf einen anderen Pinsel
#LAYER	erzeugt einen Link auf eine Ebene (Ebenen müssen eingeschaltet sein!)
#TEXTOBJECT	erzeugt einen Link auf ein Textobjekt

### EINGABEN

id	ID für den zu erzeugenden verlinkten Pinsel
sourcetype	Typ des Quellobjekts (siehe Liste oben)
sourceid	Kennung des Quellobjekts
par	optional: momentan nur für Typ #ANIM benötigt

**BEISPIEL**

```
LoadAnim(1, "MyAnim.gif")
GetBrushLink(1, #ANIM, 1, 1)
DisplayBrushFX(1, #CENTER, #CENTER, #CROSSFADE)
PlayAnim(1, #CENTER, #CENTER)
```

Obenstehender Code lädt die Animation "MyAnim.anm" von "dh0:Animations", erzeugt einen Link auf das erste Bild, überblendet dieses Bild auf dem Display und beginnt dann, die Animation abzuspielen. So könnten Sie eine Animation mit einem Übergangseffekt darstellen.

**43.36 GetBrushPen****BEZEICHNUNG**

GetBrushPen – gibt die Stiftfarbe aus der Palette des Pinsels zurück (V9.0)

**ÜBERSICHT**

```
color = GetBrushPen(id, pen)
```

**BESCHREIBUNG**

Dieser Befehl ruft die Farbe des durch **pen** angegebenen Stifts aus der Palette des in **id** angegebenen Pinsels ab. Die Farbe wird als **RGB-Farbe** zurückgegeben.

Siehe auch **RemapBrush()**, **RemoveBrushPalette()**, **SetBrushDepth()**, **SetBrushPalette()**, **SetBrushPen()** und **SetBrushTransparentPen()**.

**EINGABEN**

**id**            Identifikator des zu verwendenden Pinsels  
**pen**           Stift, den Sie erhalten möchten (beginnend bei 0)

**RÜCKGABEWERTE**

**color**        Farbe des Stifts, angegeben als **RGB-Farbe**

**BEISPIEL**

```
color = GetBrushPen(1, 0)
```

Der Code gibt die Farbe des ersten Stifts von Pinsel 1 zurück.

**43.37 Hardware-Pinsel**

Hardware-Pinsel werden für beschleunigte Hardwarezeichnungen in Verbindung mit einem Hardware-Doppelpuffer verwendet. Um einen Hardware-Pinsel zu erstellen, setzen Sie einfach bei den Befehlen **LoadBrush()** oder **@BRUSH** den Tag **Hardware** auf **True**. Alternativ können Sie auch von einem normalen Pinsel einen Hardware-Pinsel erstellen, indem Sie **CopyBrush()** benutzen und bei diesem Befehl den Tag **Hardware** auf **True** setzen. Um herauszufinden, ob ein Hardware-Pinsel erfolgreich erstellt wurde, können Sie den Befehl **GetAttribute()** mit dem Attribut **#ATTRHARDWARE** aufrufen. Beachten Sie, dass dieses Attribut **False** zurückgeben kann, auch wenn der Tag **Hardware** auf **True** gesetzt wurde. Nicht alle Systeme unterstützen Hardware-Pinsel. Wenn Hollywood auf einem System ohne Hardware-Pinsel Unterstützung läuft, wird stattdessen ein Softwarepinsel erstellt.

Der Vorteil der Hardware-Pinsel ist, dass sie vollständig in dem Videospeicher gesichert sind und können somit sehr schnell dargestellt werden. Allerdings können Hardware-Pinsel nichts anderes, als nur in den hardwarebeschleunigte Doppelpuffer gezeichnet zu werden. Deshalb arbeiten fast alle Befehle der Pinsel-Bibliothek nicht mit Hardware-Pinseln. Wenn Sie einen Hardware-Pinsel ändern möchten, müssen Sie aus ihm zunächst einen Softwarepinsel erstellen. Diesen Softwarepinsel ändern Sie und wandeln ihn dann wieder in einen Hardware-Pinsel um. Sie können einen hardwarebeschleunigten Doppelpuffer erstellen, indem Sie beim Befehl `BeginDoubleBuffer()` das erste Argument auf `True` setzen. Siehe [Abschnitt 28.3 \[BeginDoubleBuffer\]](#), [Seite 568](#), für Details.

Denken Sie daran, dass Sie nur unter Verwendung von Hardware-Pinsel auf hardwarebeschleunigte Doppelpuffer zeichnen sollten. Alle anderen Zeichnungsbefehle sind viel langsamer! Nur durch verwenden von Hardware-Pinsel können Sie eine vollständige hardwarebeschleunigte Zeichnung erhalten. Benutzen Sie normale Zeichnungsbefehle mit einem Hardware-Doppelpuffer, kann er sogar langsamer sein, als wenn Sie einen Software-Doppelpuffer gebrauchen. Dies ist insbesondere bei Grafiken der Fall, die einen Alphakanal verwenden, wie z.B. Antialias-Text oder Vektorformen, weil das Lesen der Daten aus dem Videospeicher sehr langsam ist. Daher sollten Sie versuchen, Hardware-Pinsel wo auch immer möglich zu verwenden, wenn Sie mit einem Hardware-Doppelpuffer arbeiten.

Auf einigen Systemen (z.B. AmigaOS 4.1) bieten die Befehle `ScaleBrush()`, `RotateBrush()` und `TransformBrush()` sowie die [Standard-Tags zum Zeichnen](#) für on-the-fly Bildbearbeitung (z.B. `ScaleX` und `ScaleY`) beschleunigte Hardwareunterstützung für Hardware-Pinsel. In diesem Fall ist das Skalieren und Transformieren von Pinseln extrem schneller als im Softwaremodus (insbesondere für antialiased Transformationen).

Standardmäßig werden die Hardware-Pinsel nur auf AmigaOS und Android unterstützt. Ab Hollywood 6.0 jedoch können Plugins benutzt werden, die ein Display-Adaptermodul für Grafikkarten installieren können und so auch Hardware-Pinsel unterstützen. In diesem Fall können Sie auch Hardware-Pinsel auf anderen Systemen als AmigaOS und Android verwenden. Zum Beispiel können die Plugins GL Galore und RebelSDL Hardware-Pinsel und hardwarebeschleunigte Doppelpuffer unter Windows, macOS und Linux verwenden.

Hardware-Pinsel können auch Display abhängig sein. Dies bedeutet, dass sie nur auf dem Display dargestellt werden, das verwendet wurde. Dies ist häufig der Fall, wenn Display-Adaptermodule benutzerdefinierte Displays von Plugins zur Verfügung stellen. Hollywoods eingebaute Hardware-Pinsel für AmigaOS und Android sind jedoch nicht Display abhängig und können auf jedem Display gezeichnet werden, die derzeit geöffnet sind. Um einen displayabhängigen Hardware-Pinsel zuzuweisen, müssen Sie die ID des Displays übergeben, die die Hardware-Pinsel über den Tag `Display` der Befehle `LoadBrush()`, `@BRUSH` oder `CopyBrush()` erhalten. Beachten Sie, dass alle displayabhängigen Hardware-Pinsel automatisch von Hollywood freigegeben werden, wenn das zugehörige Display geschlossen wird.

### 43.38 InvertAlphaChannel

#### BEZEICHNUNG

`InvertAlphaChannel` – invertiert den Alphakanal eines Pinsels (V2.0)

#### ÜBERSICHT

`InvertAlphaChannel(id)`

**BESCHREIBUNG**

Dieser Befehl invertiert den Alphakanal den durch `id` angegebenen Pinsel. Dies bedeutet, dass die Transparenz für jedes Pixel umgedreht wird. Wenn z.B. ein Pixel vorher 80% transparent war, wird es nur noch 20% transparent sein. Andererseits werden nach einer Inversion der Pixel, die 20% transparent waren, nachher 80% transparent haben.

Siehe auch `ChangeBrushTransparency()`, `DeleteAlphaChannel()`, `DeleteMask()`, `InvertBrush()` Befehl, `InvertMask()`, `IsBrushEmpty()`, `ReduceAlphaChannel()`, `SetAlphaIntensity()`, `SetBrushTransparency()`, `SetBrushTransparentPen()` und `SetMaskMode()`.

**EINGABEN**

`id` Pinsel, dessen Alphakanal invertiert wird

**43.39 InvertBrush****BEZEICHNUNG**

`InvertBrush` – invertiert die Farben eines Pinsels (V1.5)

**ÜBERSICHT**

`InvertBrush(id)`

**BESCHREIBUNG**

Dieser Befehl invertiert alle Farben des Pinsels, der durch `id` angegeben wurde. Durch die Invertierung werden alle Farben mit ihrer Komplementärfarbe ersetzt (Weiß wird Schwarz, Blau wird Gelb usw.)

Beachten Sie, dass wenn `id` einen Palettenpinsel angibt, `InvertBrush()` nur die Palettenfarben invertiert, was diesen Befehl bei Verwendung mit Palettenpinseln sehr schnell macht.

Weitere Pinseleffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `SepiaToneBrush()`, `ScaleBrush()`, `SharpenBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()`, `TransformBrush()`, `WaterRippleBrush()`, `InvertMask()` und `InvertAlphaChannel()`.

**EINGABEN**

`id` Pinsel, der invertiert werden soll

**BEISPIEL**

`InvertBrush(1)`

Der obige Code invertiert die Farben von Pinsel 1.

**43.40 InvertMask****BEZEICHNUNG**

`InvertMask` – invertiert die Maske eines Pinsels (V2.0)

**ÜBERSICHT**

`InvertMask(id)`

**BESCHREIBUNG**

Dieser Befehl kehrt die Maske des in `id` angegebenen Pinsels um. Das bedeutet, dass alle Bereiche, die vorher transparent waren, werden undurchsichtig und Bereiche, die zuvor undurchsichtig waren, werden transparent.

Siehe auch `ChangeBrushTransparency()`, `DeleteAlphaChannel()`, `DeleteMask()`, `InvertAlphaChannel()`, `InvertBrush()` Befehl, `IsBrushEmpty()`, `ReduceAlphaChannel()`, `SetAlphaIntensity()`, `SetBrushTransparency()`, `SetBrushTransparentPen()` und `SetMaskMode()`.

**EINGABEN**

`id` Pinsel, dessen Maske invertiert wird

**43.41 IsBrushEmpty****BEZEICHNUNG**

`IsBrushEmpty` – überprüft, ob ein Pinsel nur unsichtbare Pixel hat (V8.0)

**ÜBERSICHT**

`r = IsBrushEmpty(id)`

**BESCHREIBUNG**

Mit diesem Befehl kann geprüft werden, ob der in `id` angegebene Pinsel nur unsichtbare Pixel aufweist. In diesem Fall kann er als "leer" betrachtet werden. Wenn der Pinsel nur unsichtbare Pixel enthält, wird **True** zurückgegeben, andernfalls **False**.

Offensichtlich kann ein Pinsel nur "leer" sein, wenn er eine Transparenz verwendet, entweder eine Maske oder einen Alphakanal. Wenn Sie diesen Befehl mit einem Pinsel aufrufen, an den weder eine Maske noch ein Alphakanal angehängt ist, ist der Rückgabewert immer **False**.

Siehe auch `ChangeBrushTransparency()`, `DeleteAlphaChannel()`, `DeleteMask()`, `InvertAlphaChannel()`, `InvertBrush()` Befehl, `InvertMask()`, `ReduceAlphaChannel()`, `SetAlphaIntensity()`, `SetBrushTransparency()`, `SetBrushTransparentPen()` und `SetMaskMode()`.

**EINGABEN**

`id` Pinsel, der überprüft werden soll

**RÜCKGABEWERTE**

`r` **True**, wenn der Pinsel nur unsichtbare Pixel enthält, andernfalls **False**

**BEISPIEL**

```
CreateBrush(1, 100, 100, #RED, {Mask = True, Clear = True})
Print(IsBrushEmpty(1))
```

Der obige Code gibt den Wert 1 aus, da der Pinsel zwar mit roten Pixeln gefüllt ist, jedoch keines davon sichtbar ist, da die Maske alle Pixel auf unsichtbar gesetzt hat.

## 43.42 LoadBrush

### BEZEICHNUNG

LoadBrush – lädt einen Pinsel

### ÜBERSICHT

```
[id] = LoadBrush(id, filename$[, table])
```

### BESCHREIBUNG

Dieser Befehl lädt den durch `filename$` angegebenen Pinsel in den Speicher und weist ihm die `id` zu. Wenn Sie beim Argument `id` **Nil** angeben, wird `LoadBrush()` für den kopierten Pinsel automatisch eine Zahl auswählen und Ihnen übergeben.

Bildformate, die auf allen Plattformen unterstützt werden, sind PNG, JPEG, BMP, IFF ILBM, GIF und Plugins für Bilder. Je nach Plattform, worauf Hollywood ausgeführt wird, können mehr Bildformate unterstützt werden. Zum Beispiel auf den Amigakompatiblen Systemen wird Hollywood in der Lage sein, alle Formate über `Bilddatatypes` zu öffnen. Unter Windows kann `LoadBrush()` auch Bildformate laden, die von der Windows-Imaging-Komponente unterstützt werden.

Ab Hollywood 5.0 kann dieser Befehl auch Vektorformate wie SVG laden, wenn Sie ein entsprechendes Plugin installieren. Beachten Sie aber, dass wenn Sie Vektorbilder mit diesen Befehl laden, der Pinsel ein spezieller Vektorpinsel sein wird, der nicht alle Merkmale der normalen Pinsel unterstützt. Sie können jedoch Vektorpinsel mit dem Befehl `RasterizeBrush()` in Rasterpinsel umwandeln. Siehe [Abschnitt 43.80 \[Vektorpinsel\]](#), [Seite 982](#), für weitere Informationen über Vektorpinsel.

Das dritte Argument `table` ist optional. Es ist eine Tabelle, die weitere Möglichkeiten für den Ladevorgang zur Verfügung stellt. Die folgenden Tags können verwendet werden:

#### Transparency:

Dieser Tag kann benutzt werden, um eine Farbe in **RGB-Notation** anzugeben, die im Pinsel transparent erscheinen soll.

#### LoadAlpha:

Setzen Sie diesen Tag auf **True**, wenn der Alpha-Kanal des Bildes auch geladen werden soll. Bitte beachten Sie, dass nicht alle Bilder einen Alpha-Kanal haben und dass nicht alle Bildformate in der Lage sind, Alphakanaldaten zu speichern. Es wird vorgeschlagen, dass Sie das PNG-Format verwenden, wenn Sie Alphakanaldaten benötigen. Dieser Tag ist standardmäßig **False**.  
(4.5)

#### X, Y, Width, Height:

Diese Tags können verwendet werden, um nur einen Teil des Bildes in den Pinsel zu laden. Dies ist nützlich, wenn Sie ein großes Bild mit vielen verschiedenen kleinen Bildern haben und nun die kleinen Bilder in einzelne Pinsel laden wollen. Somit können Sie ein Rechteck innerhalb des Bildes angeben, aus dem Hollywood die Grafikdaten für den Pinsel nimmt.

#### Hardware:

Wenn Sie diesen Tag auf **True** setzen, wird Hollywood diesen Pinsel vollständig im Videospeicher für Hardwarezeichnen erstellen und in Verbindung mit einem Hardwaredoppelpuffer das Zeichnen beschleunigt.



Hardware-Pinsel unterliegen verschiedenen Einschränkungen. Siehe [Abschnitt 43.37 \[Hardware-Pinsel\]](#), Seite 940, für Details. (V5.0)

**ScaleWidth, ScaleHeight:**

Diese Tags können verwendet werden, um eine skalierte Version des Bildes zu laden. Wenn der Bildtreiber skaliertes Laden unterstützt, kann das die Geschwindigkeit erheblich steigern, wenn Sie zum Beispiel nur eine Version in Thumbnailgröße eines großen Bildes laden möchten. Wenn der Bildtreiber skaliertes Laden nicht unterstützt, wird zuerst das volle Bild geladen, bevor es skaliert wird. Dies ist nicht wesentlich schneller als das Bild nach dem Laden von Hand zu skalieren. Sie können die Größe entweder als direkter Wert übergeben oder Sie benutzen einen Prozentsatz als Zeichenfolge (z.B. `Scalewidth = "200%"`). (V5.3)

**SmoothScale:**

Wenn `ScaleWidth` oder `ScaleHeight` eingestellt ist, können Sie mit diesem Tag festlegen, ob Hollywood bei der Skalierung Antialiasing verwenden soll oder nicht. Der Standardwert ist `False` und bedeutet kein Antialiasing. Beachten Sie, dass Skalierung mit Antialiasing viel langsamer ist als die normale Skalierung. (V5.3)

**Display:** Wenn Sie hier die ID eines Displays angeben, wird Hollywood einen Displayabhängigen-Hardware-Pinsel für Sie erstellen. Displayabhängige Hardware-Pinsel können nur auf das Display zeichnen, dem sie zugewiesen wurden. Dieser Tag wird nur berücksichtigt, wenn der Tag `Hardware` auf `True` gesetzt wurde. Beachten Sie auch, dass Hollywoods eingebautes Display-Adaptermodul keine Displayabhängigen-Hardware-Pinsel unterstützt, aber Plugins können individuelle Display-Adaptermodule installieren, die displayabhängige Hardware-Pinsel unterstützen. Dieser Tag enthält standardmäßig die ID des aktuell aktiven Displays. Siehe [Abschnitt 43.37 \[Hardware-Pinsel\]](#), Seite 940, für Details. (V6.0)

**Loader:** Mit diesem Tag können Sie einen oder mehrere Formatlademodule angeben, die diesen Pinsel laden sollen. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehreren Lademodulen enthält. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), Seite 96, für Details. (V6.0)

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen sollen. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehreren Adaptermodulen enthält. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), Seite 96, für Details. (V6.0)

**LoadTransparency:**

Ist dieser Tag auf `True` gesetzt, wird die monochrome Transparenz des Bildes geladen. Bitte beachten Sie, dass dieser Tag speziell für monochrome Transparenzkanäle ist, das heißt der transparente Stift ist für ein Palettenbasiertes

Bild ausgelegt. Wenn Sie den Alpha-Kanal eines Bildes laden möchten, stellen Sie den Tag `LoadAlpha` auf `True`. Dieser Tag ist auf `False` voreingestellt. (V6.0)

#### **LoadPalette:**

Wenn dieser Tag auf `True` gesetzt ist, lädt Hollywood den Pinsel als Palettenpinsel. Dies bedeutet, dass Sie die Palette des Pinsels abrufen und verändern können, was für bestimmte Effekte wie Farbwechsel nützlich ist. Sie können Stifte auch mit dem Tag `TransparentPen` (siehe unten) oder dem Tag `LoadTransparency` (siehe oben) transparent machen. Palettenpinsel haben auch den Vorteil, dass sie weniger Speicherplatz benötigen, da 1 Pixel nur 1 Byte Speicher anstelle von 4 Byte für 32-Bit-Bilder benötigt. Dieser Tag ist standardmäßig auf `False` gesetzt. (V9.0)

#### **TransparentPen:**

Wenn der Tag `LoadPalette` auf `True` gesetzt wurde (siehe oben), kann mit dem Tag `TransparentPen` ein Stift definiert werden, der transparent gemacht werden soll. Stifte werden ab 0 gezählt. Alternativ können Sie auch den Tag `LoadTransparency` auf `True` setzen, um Hollywood zu zwingen, den transparenten Stift zu verwenden, der in der Bilddatei gespeichert ist (sofern das Bildformat die Speicherung von transparenten Stiften unterstützt). Dieser Tag ist standardmäßig auf `#NOPEN` gesetzt. (V9.0)

#### **UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Bitte beachten Sie, dass sich die Tags `Transparency`, `LoadTransparency` und `LoadAlpha` gegenseitig ausschließen. Ein Pinsel kann nur eine Transparenzeinstellung haben!

Dieser Befehl ist auch als Präprozessor vorhanden: Verwenden Sie `@BRUSH` um Pinsel vorzuladen!

Siehe auch `CopyBrush()`, `FreeBrush()`, `IsPicture()`, `SaveBrush()`, `LoadBGPic()` und `MoveBrush()`.

### **EINGABEN**

`id` ID des neuen Pinsels oder `Nil` für die automatische ID-Zuweisung

`filename$` das Bild, welches Sie laden möchten

`table` optional: optionales Argument für weiterer Optionen (siehe oben) (V2.0)

### **RÜCKGABEWERTE**

`id` optional: Identifikator des Pinsels; Wird nur zurückgegeben werden, wenn Sie `Nil` als Argument 1 angegeben haben (siehe oben)

### **BEISPIEL**

```
LoadBrush(2, "MyBrush.png", {Transparency = #RED})
```

Dies lädt "MyBrush.png" als Pinsel 2 und die Farbe Rot ist transparent.

### 43.43 Masken und Alphakanal

Hollywood unterstützt zwei Arten von Transparenzen für seine Grafikobjekte: Masken- und Alpha-Transparenz. In diesem Abschnitt wird der Unterschied zwischen den beiden erklärt.

Masken-Transparenz kennt nur zwei Einstellungen pro Pixel: Sichtbar und unsichtbar. Alpha-Transparenz unterstützt jedoch 256 verschiedene Transparenzstufen für jedes Pixel. Ein Alpha-Niveau von 0 bedeutet, dass das Pixel nicht sichtbar ist, und ein Wert von 255, dass das Pixel vollständig sichtbar ist. Ein Niveau von 128 bedeutet also, dass ein Pixel nur 50% sichtbar ist. Alpha-Transparenz ist sehr nützlich, wenn Sie Bilder einbetten, die sich nahtlos an Ihren Hintergrund anzupassen. Zum Beispiel kann ein Pinsel mit einem Schatten erstellt werden, wo der Hintergrund hindurch scheint, oder ein Pinsel mit antialiased Kanten. Für solche Zwecke reicht eine Maske nicht.

Bitte beachten Sie auch, dass Maske und Alphakanal sich gegenseitig ausschließen. Das heißt, ein Pinsel kann entweder eine Maske oder einen Alphakanal haben, aber nicht beides.

### 43.44 MixBrush

#### BEZEICHNUNG

MixBrush – mischt zwei Pinsel (V1.5)

#### ÜBERSICHT

MixBrush(*brush1*, *brush2*, *level*)

#### BESCHREIBUNG

Dieser Befehl mischt den Pinsel, der in *brush1* angegeben wurde in den Pinsel, der in *brush2* angegeben wurde mit einer bestimmten Intensität, die in *level* übergeben wird. Diese darf von 0 bis 255 reichen.

Ab Version 2.0 kann bei *level* eine prozentige Angabe in Form einer Zeichenkette angegeben werden, z.B. "50%".

Weitere Pinseffekte: [ArcDistortBrush\(\)](#), [BarrelDistortBrush\(\)](#), [BlurBrush\(\)](#), [BrushToGray\(\)](#), [BrushToMonochrome\(\)](#), [CharcoalBrush\(\)](#), [ContrastBrush\(\)](#), [EdgeBrush\(\)](#), [EmbossBrush\(\)](#), [FlipBrush\(\)](#) Befehl, [GammaBrush\(\)](#), [InvertBrush\(\)](#) Befehl, [ModulateBrush\(\)](#), [OilPaintBrush\(\)](#), [PerspectiveDistortBrush\(\)](#), [PixelateBrush\(\)](#), [PolarDistortBrush\(\)](#), [RotateBrush\(\)](#), [SepiaToneBrush\(\)](#), [ScaleBrush\(\)](#), [SharpenBrush\(\)](#), [SolarizeBrush\(\)](#), [SwirlBrush\(\)](#), [TintBrush\(\)](#), [TransformBrush\(\)](#), und [WaterRippleBrush\(\)](#).

#### EINGABEN

*brush1*      Pinsel, in den gemischt werden soll  
*brush2*      Pinsel, der in *brush1* gemischt werden soll  
*level*        Mischstufe

#### BEISPIEL

MixBrush(1, 2, 128)

Der obige Code mischt Pinsel 2 in Pinsel 1 mit einer Intensität von 50% (= 128).

## 43.45 ModulateBrush

### BEZEICHNUNG

ModulateBrush – ändert die Helligkeit, Sättigung und den Farbton des Pinsels (V5.0)

### ÜBERSICHT

`ModulateBrush(id, brightness, saturation, hue)`

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die Einstellungen der Helligkeit, Sättigung und des Farbtons eines Pinsels zu ändern. Für jede Einstellung übergeben Sie einen Fließkommawert, der die gewünschte Änderung beschreibt. Ein Wert von 1.0 bedeutet keine Veränderung, ein Wert kleiner als 1.0 verringert die Helligkeit/Sättigung/den Farbton, während ein Wert von mehr als 1.0 sie verbessert.

Wenn `id` einen Palettenpinsel angibt, moduliert `ModulateBrush()` nur die Palettenfarben, was diesen Befehl bei Verwendung von Palettenpinseln sehr schnell macht.

Weitere Pinseleffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `InvertBrush()` Befehl, `MixBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `SepiaToneBrush()`, `ScaleBrush()`, `SharpenBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()`, `TransformBrush()`, und `WaterRippleBrush()`.

### EINGABEN

<code>id</code>	ID des Pinsels zum Modulieren
<code>brightness</code>	gewünschte Korrektur der Helligkeit
<code>saturation</code>	gewünschte Korrektur der Sättigung
<code>hue</code>	gewünschte Korrektur des Farbtons

### BEISPIEL

`ModulateBrush(1, 1.0, 2.0, 1.0)`

Der obige Code erhöht die Sättigung, während die Helligkeit und der Farbton unberührt bleiben. Das Ergebnis ist ein Bild mit betonten Farben wie in einem Cartoon.

## 43.46 MoveBrush

### BEZEICHNUNG

MoveBrush – bewegt einen Pinsel von a nach b

### ÜBERSICHT

`[handle] = MoveBrush(id, xa, ya, xb, yb[, table])`

**BESCHREIBUNG**

Dieser Befehl bewegt (scrollt) den durch in `id` angegebenen Pinsel sanft von der durch `xa/ya` angegebenen Stelle zu der Stelle `xb/yb`.

Weitere Parameter können in dem optionalen Argument `table` angegeben werden:

**Speed:** Der Parameter `Speed` definiert die Anzahl Pixel, die der Pinsel pro Bild bewegt wird. Daher bedeutet eine höhere Zahl eine höhere Geschwindigkeit. Sie können als Geschwindigkeit auch eine der Geschwindigkeitskonstanten angeben (`#SLOWSPEED`, `#NORMALSPEED` oder `#FASTSPEED`).

**FX:** Diesen Parameter können Sie benutzen können, um der Bewegung einen Effekt hinzuzufügen. Die folgende Effekte können momentan benutzt werden:

**#BOUNCE:** prallt das Objekt am Ende der Bewegung

**#DAMPED:** schwingt das Objekt am Ende aus

**#SMOOTHOUT:**  
bremst die Bewegung gegen Ende ab

**#SINE:** führt die Bewegung auf einer Sinuswelle aus (\*)

**#BIGSINE:**  
benutzt eine größere Sinuswelle für die Bewegung (\*)

**#LOWERCURVE:**  
bewegt das Objekt auf einer Kurve unter der Normalzeile (\*)

**#UPPERCURVE:**  
bewegt das Objekt auf einer Kurve über der Normalzeile (\*)

Effekte, die mit einem Sternchen gekennzeichnet sind, können nur mit horizontalen Bewegungen benutzt werden, d.h. `ya` und `yb` müssen gleich sein.

**Async:** Sie können diesen Tag verwenden, um ein asynchrones Zeichnungsobjekt für diesen Übergang zu erstellen. Wenn Sie hier `True` angeben, wird `MoveBrush()` sofort verlassen und es wird ein Handler für das asynchrone Zeichnungsobjekt kreiert, den Sie dann mit dem Befehl `AsyncDrawFrame()` verwenden können. Ein Beispieldskript finden Sie unter dem Befehl `AsyncDrawFrame()`. Siehe [Abschnitt 19.1 \[AsyncDrawFrame\]](#), [Seite 231](#), für weitere Informationen über asynchrone Zeichnungsobjekte.

Siehe auch [@BRUSH](#), [LoadBrush\(\)](#), [CopyBrush\(\)](#), [FreeBrush\(\)](#) und [SaveBrush\(\)](#).

**EINGABEN**

<code>id</code>	ID des Pinsels, der bewegt wird
<code>xa</code>	Startposition x
<code>ya</code>	Startposition y
<code>xb</code>	Zielposition x
<code>yb</code>	Zielposition y
<code>table</code>	optional: Argument für weiterer Optionen (siehe oben)

**RÜCKGABEWERTE**

**handle** optional: Handler auf ein asynchrones Zeichnungsobjekt; wird nur zurückgegeben, wenn **Async** auf **True** gesetzt wurde (siehe oben)

**BEISPIEL**

```
MoveBrush(1, 100, 50, 0, 50, {Speed = 5})
```

Bewegt den Pinsel 1 mit Geschwindigkeit 5 von 100:50 nach 0:50.

```
MoveBrush(1, #RIGHTOUT, #BOTTOM, #LEFTOUT, #BOTTOM, {Speed = #NORMALSPEED})
```

Bewegt den Pinsel von der äusseren rechten Position zur äusseren linken Position mit normaler Geschwindigkeit.

**43.47 OilPaintBrush****BEZEICHNUNG**

OilPaintBrush – stellt Pinsel mit Ölfarbeneffekt dar (V5.0)

**ÜBERSICHT**

```
OilPaintBrush(id, radius)
```

**BESCHREIBUNG**

Dieser Befehl wendet einen Ölfarbeneffekt beim angegebenen Pinsel an. Das optionale Argument **radius** kann verwendet werden, um den Effektradius anzugeben. Je größer der Radius, desto länger braucht dieser Befehl, den Ölfarbeneffekt anzuwenden.

Beachten Sie, dass dieser Befehl nicht mit Palettenpinseln verwendet werden kann.

Weitere Pinseleffekte: **ArcDistortBrush()**, **BarrelDistortBrush()**, **BlurBrush()**, **BrushToGray()**, **BrushToMonochrome()**, **CharcoalBrush()**, **ContrastBrush()**, **EdgeBrush()**, **EmbossBrush()**, **FlipBrush()** Befehl, **GammaBrush()**, **InvertBrush()** Befehl, **MixBrush()**, **ModulateBrush()**, **PerspectiveDistortBrush()**, **PixelateBrush()**, **PolarDistortBrush()**, **RotateBrush()**, **SepiaToneBrush()**, **ScaleBrush()**, **SharpenBrush()**, **SolarizeBrush()**, **SwirlBrush()**, **TintBrush()**, **TransformBrush()**, und **WaterRippleBrush()**.

**EINGABEN**

**id** ID des Pinsels

**radius** Radius des Effektes

**43.48 PenArrayToBrush****BEZEICHNUNG**

PenArrayToBrush – wandelt ein Stiftfeld in einen Palettenpinsel um (V9.0)

**ÜBERSICHT**

```
[id] = PenArrayToBrush(id, table, width, height[, t])
```

## BESCHREIBUNG

Dieser Befehl erstellt einen neuen Palettenpinsel aus dem in **table** angegebenen Feld von Stiften. Die Tabelle kann als eine Matrix angesehen werden, die in **height** die Anzahl der Zeilen enthält und in **width** die Anzahl der Elemente, die nacheinander gespeichert werden. Die Reihenfolge der Stiftdaten in dieser Tabelle muss wie folgt sein: Zeile für Zeile von oben nach unten, d.h. die Tabelle beginnt mit der ersten Reihe von Stiften. Jede Zeile muss genau die Anzahl der Stifte in **width** und die Anzahl der Zeilen in **height** enthalten. Die Tabelle muss eindimensional sein, d.h. sie darf keine Untertabellen für die einzelnen Zeilen verwenden, sondern nur die Stiftwerte nacheinander speichern.

Die Palette, die die Stifte verwenden sollen, kann im optionalen Tabellenargument **t** festgelegt werden. Die folgenden Tabellenelemente werden derzeit erkannt:

**Palette:** Setzen Sie dies auf die ID einer Palette, die mit den Befehlen **CreatePalette()**, **LoadPalette()** oder der Präprozessor-Anweisung **@PALETTE** erstellt wurde. Wenn Sie diesen Tag nicht festlegen, verwendet Hollywood eine Standardpalette, in der alle Farben auf Schwarz initialisiert sind.

**TransparentPen:**

Mit diesem Tag kann ein Stift angegeben werden, der transparent erscheinen soll. Wenn kein Stift transparent gemacht werden soll, setzen Sie diesen Tag auf **#NOPEN**, was ebenfalls die Standardeinstellung ist.

Bitte beachten Sie, dass die Tabelle, die Sie an diesen Befehl übergeben, normalerweise viel Speicherplatz beansprucht. Daher sollten Sie diese Tabelle auf **Nil** setzen, sobald Sie sie nicht mehr benötigen. Andernfalls verschwenden Sie sehr viel Speicher und es kann sogar vorkommen, dass Ihrem Skript der Speicherplatz ausgeht. Denken Sie also daran, dass Sie Pixelfeld-Tabellen immer auf **Nil** setzen sollten, sobald Sie sie nicht mehr benötigen.

Um einen Palettenpinsel in ein Stiftfeld umzuwandeln, können Sie den Befehl **BrushToPenArray()** verwenden. Siehe **Abschnitt 43.9 [BrushToPenArray]**, **Seite 913**, für Details.

Siehe auch **BrushToRGBArray()** und **RGBArrayToBrush()**.

## EINGABEN

<b>id</b>	ID der neuen Pinselpalette oder <b>Nil</b> für die <b>automatische ID-Zuweisung</b>
<b>table</b>	Tabelle mit einem Feld von Stiften, die den Inhalt des neuen Pinsels beschreiben
<b>width</b>	Anzahl der Elemente in jeder Zeile
<b>height</b>	Anzahl der Zeilen in der Tabelle
<b>t</b>	optional: Tabelle mit weiteren Optionen (siehe oben)

## RÜCKGABEWERTE

<b>id</b>	optional: Identifikator der neuen Pinselpalette; wird nur zurückgegeben, wenn Sie <b>Nil</b> als Argument 1 angegeben haben (siehe oben)
-----------	--

## BEISPIEL

```
pixels = {}
```

```

For Local y = 0 To 479
  For Local x = 0 To 639 Do pixels[y * 640 + x] = y \ 2
Next

```

```

PenArrayToBrush(1, pixels, 640, 480, {Palette = #PALETTE_AGA})
pixels = Nil      ; WICHTIG: Speicher freigeben!
DisplayBrush(1, 0, 0)

```

Der obige Code erstellt einen Palettenpinsel, der die ersten 240 Stifte aus der Standardpalette `#PALETTE_AGA` enthält. Jeder Palettenstift verwendet zwei Zeilen. Wichtig: Vergessen Sie nicht, das Pixelfeld auf `Nil` zu setzen, wenn Sie es nicht mehr benötigen, da es sonst im Speicher bleibt und Pixelfelder sehr viel Speicher verbrauchen!

## 43.49 PerspectiveDistortBrush

### BEZEICHNUNG

PerspectiveDistortBrush – stellt Pinsel mit perspektivischer Verzerrung dar (V5.0)

### ÜBERSICHT

`PerspectiveDistortBrush(id,cx1,cy1,cx2,cy2,cx3,cy3,cx4,cy4[,smooth])`

### BESCHREIBUNG

Dieser Befehl verzerrt perspektivisch den in `id` angegebenen Pinsel. Sie müssen vier Kontrollpunkte übergeben, die ein Viereck beschreiben, in das der Pinsel abgebildet werden soll. Das optionale Argument `smooth` kann verwendet werden, um Antialiasing Pixelinterpolation zu ermöglichen, die zu einem glatteren Aussehen führt, aber dessen Berechnung länger dauert.

Das Mapping der Kontrollpunkte ist wie folgt: Die linke obere Ecke des Pinsels ist Nummer 1, die rechte obere Ecke Nummer 2, die untere rechte Ecke Punkt 3 und die untere linke Ecke ist Punkt 4.

Weitere Pinseffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `InvertBrush()` Befehl, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `SepiaToneBrush()`, `ScaleBrush()`, `SharpenBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()`, `TransformBrush()`, und `WaterRippleBrush()`.

### EINGABEN

<code>id</code>	Pinsel, der verzerrt werden soll
<code>cx1</code>	Koordinate x vom Punkt 1
<code>cy1</code>	Koordinate y vom Punkt 1
<code>cx2</code>	Koordinate x vom Punkt 2
<code>cy2</code>	Koordinate y vom Punkt 2
<code>cx3</code>	Koordinate x vom Punkt 3



<code>cy3</code>	Koordinate y vom Punkt 3
<code>cx4</code>	Koordinate x vom Punkt 4
<code>cy4</code>	Koordinate y vom Punkt 4
<code>smooth</code>	optional: mit <code>True</code> wird Antialiasing beim Verzerren verwendet (Standard ist <code>False</code> )

**BEISPIEL**

```
PerspectiveDistortBrush(1, 100, 0, 400, 0, 500, 300, 0, 300)
```

Der obige Code verzerrt den Pinsel 1 trapezförmig.

## 43.50 PixelateBrush

**BEZEICHNUNG**

PixelateBrush – vergrößert die Pixel eines Pinsels (V5.0)

**ÜBERSICHT**

```
PixelateBrush(id, cellsize)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die Pixel des angegebenen Pinsels zu vergrößern/vergrößern. Jedes Pixel im Pinsel wird auf im Argument `cellsize` angegebene Größe gezoomt werden. Die Vergrößerung startet in der oberen linken Ecke des Pinsels.

Weitere Pinseffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `InvertBrush()` Befehl, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `SepiaToneBrush()`, `ScaleBrush()`, `SharpenBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()`, `TransformBrush()`, und `WaterRippleBrush()`.

**EINGABEN**

`id` ID des Pinsels

`cellsize` Stufe der Vergrößerung; muss größer als 1 sein

## 43.51 PolarDistortBrush

**BEZEICHNUNG**

PolarDistortBrush – stellt Pinsel mit polarer Verzerrung dar (V5.0)

**ÜBERSICHT**

```
PolarDistortBrush(id[, rmax, rmin, cx, cy, start, end, smooth])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die polare Verzerrung auf den in `id` angegebenen Pinsel anzuwenden. Mit den optionalen Argumenten können Sie die Parameter

für die polare Verzerrung anzugeben. Die Argumente **rmin** und **rmax** geben die minimalen und maximalen Radien an. **cx** und **cy** werden verwendet, um den Mittelpunkt der Verzerrung anzugeben. Sowohl die Radiuswerte und der Mittelpunkt müssen in Pixel angegeben werden. In **start** und **end** geben Sie den Start- und Endwinkel für die polare Verzerrung an. Schließlich kann das optionale Argument **smooth** verwendet werden, um Antialiasing Pixelinterpolation zu ermöglichen, die zu einem glatteren Aussehen führt, aber dessen Berechnung länger dauert.

Weitere Pinseleffekte: **ArcDistortBrush()**, **BarrelDistortBrush()**, **BlurBrush()**, **BrushToGray()**, **BrushToMonochrome()**, **CharcoalBrush()**, **ContrastBrush()**, **EdgeBrush()**, **EmbossBrush()**, **FlipBrush()** Befehl, **GammaBrush()**, **InvertBrush()** Befehl, **MixBrush()**, **ModulateBrush()**, **OilPaintBrush()**, **PerspectiveDistortBrush()**, **PixelateBrush()**, **RotateBrush()**, **SepiaToneBrush()**, **ScaleBrush()**, **SharpenBrush()**, **SolarizeBrush()**, **SwirlBrush()**, **TintBrush()**, **TransformBrush()**, und **WaterRippleBrush()**. ■

## EINGABEN

<b>id</b>	Identifikator des Pinsels
<b>rmax</b>	optional: maximum Radius (voreingestellt ist die Diagonale des Pinsels dividiert durch 2)
<b>rmin</b>	optional: minimum Radius (voreingestellt ist 0)
<b>cx</b>	optional: Koordinate x des Mittelpunktes (voreingestellt ist die Hälfte der Pinselbreite)
<b>cy</b>	optional: Koordinate y des Mittelpunktes (voreingestellt ist die Hälfte der Pinselhöhe)
<b>start</b>	optional: Startwinkel (voreingestellt ist -180)
<b>end</b>	optional: Endwinkel (voreingestellt ist 180)
<b>smooth</b>	optional: mit <b>True</b> wird Antialiasing beim Verzerren verwendet (Standard ist <b>False</b> )

## 43.52 QuantizeBrush

### BEZEICHNUNG

QuantizeBrush – reduziert die Anzahl Farben im Pinsel (V6.0)

### ÜBERSICHT

```
QuantizeBrush(id[, t])
```

### FRÜHERE SYNTAX

```
QuantizeBrush(id[, colors, dither])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die Farben eines Pinsels zu reduzieren. Dies ist nützlich, um ein retropalettenbasiertes Aussehen für Ihren Pinsel zu erstellen.

Ab Hollywood 9.0 verwendet dieser Befehl eine neue Syntax mit einem optionalen Tabellenargument, um zusätzliche Optionen anzugeben. Die folgenden Tabellen-Tags werden derzeit erkannt:

- Colors:** Mit diesem Tag können Sie die gewünschte Anzahl von Farben für den Pinsel angeben. Dies muss ein Wert zwischen 1 und 256 sein. Der Standardwert ist 256. Alternativ können Sie auch den Tag **Depth** setzen, um die gewünschte Anzahl von Farben für den Pinsel festzulegen (siehe unten).
- Dither:** Mit diesem Tag können Sie steuern, ob Dithering verwendet werden soll oder nicht. Setzen Sie dies auf **True**, um das Dithering zu aktivieren oder auf **False**, um es zu deaktivieren. Der Voreinstellung ist **True**.
- Depth:** Mit diesem Tag können Sie die gewünschte Anzahl von Farben für den Pinsel angeben. Dies muss ein Wert zwischen 1 (= 2 Farben) und 8 (= 256 Farben) sein. Der Standardwert ist 8. Dieser Tag ist eine Alternative zum Tag **Colors** (siehe oben). (V9.0)
- Palette:** Setzen Sie diesen Tag auf **True**, wenn Sie möchten, dass **QuantizeBrush()** Ihren Pinsel in einen Palettenpinsel umwandelt. Beachten Sie, dass **QuantizeBrush()** standardmäßig keine Palettenpinsel erstellt, obwohl es Farben effektiv immer auf eine Anzahl reduziert, die in eine Palette passen würde. Aus Kompatibilitätsgründen ist dies jedoch nicht der Fall, da Palettenpinsel vor Hollywood 9.0 nicht unterstützt wurden. Wenn Sie also möchten, dass **QuantizeBrush()** einen Palettenpinsel für Sie erstellt, müssen Sie diesen Tag auf **True** setzen. Der Standardwert ist **False**. (V9.0)

**TransparentPen:**

Wenn **Palette** auf **True** gesetzt wurde (siehe oben) und der zu quantisierende Pinsel eine Maske hat, werden alle unsichtbaren Pixel auf die Nummer des hier angegebenen Stiftes gesetzt, so dass dieser Stift zum transparenten Stift wird. Der Standardwert ist 0, was bedeutet, dass der erste Stift standardmäßig transparent gemacht werden soll. (V9.0)

**TransparentColor:**

Wenn **Palette** auf **True** gesetzt wurde und der zu quantisierende Pinsel eine Maske hat, wird der transparente Stift des Pinsels auf die hier angegebene Farbe gesetzt. Diese Farbe muss als **RGB-Farbe** angegeben werden. Wenn dieser Tag nicht gesetzt ist, wird der transparente Stift nicht auf eine bestimmte Farbe gesetzt. (V9.0)

Siehe auch **FloodFill()**, **ReadBrushPixel()**, **ReplaceColors()** und **WriteBrushPixel()**.

**EINGABEN**

- id** ID des Pinsels
- t** optional: Tabelle mit weiteren Argumenten (siehe oben) (V9.0)

**BEISPIEL**

```
QuantizeBrush(1, 32)
```

Konvertiert den Pinsel 1 in einen Pinsel mit 32 Farben und das Dithering ist aktiviert.

### 43.53 RasterizeBrush

#### BEZEICHNUNG

RasterizeBrush – wandelt Vektorpinsel in einen Rasterpinsel (V5.0)

#### ÜBERSICHT

```
RasterizeBrush(id)
```

#### BESCHREIBUNG

Mit diesem Befehl wird der in `id` angegebener **Vektorpinsel** in einen Rasterpinsel konvertiert. Rasterpinsel ist in Hollywood der normale Pinseltyp und wird von allen Befehlen der Pinselbibliothek unterstützt. Der Nachteil ist jedoch, dass die Skalierung, Drehung und Transformation nur mit Qualitätseinbußen durchgeführt werden kann.

Sie können mit dem Befehl `GetAttribute()` und dem Attribut `#ATTRTYPE` den Typ eines Pinsels herausfinden.

Siehe auch `BGPicToBrush()` und `ConvertToBrush()`.

#### EINGABEN

`id`            Vektorpinsel, der konvertiert wird

### 43.54 ReadBrushPixel

#### BEZEICHNUNG

ReadBrushPixel – liest ein einzelnes Pixel aus dem Pinsel (V5.0)

#### ÜBERSICHT

```
color, trans = ReadBrushPixel(id, x, y)
```

#### BESCHREIBUNG

Dieser Befehl liest die Farbe und Transparenzzustände des Pixels der in `id` angegebenen Pinsels. Die Farbe wird im **RGB-Format**, während das Format in `trans` abhängig von der Art der Transparenz ist. Wenn der Pinsel eine Maske hat, wird `trans` entweder 0 (unsichtbar) oder 1 (sichtbar) enthalten. Wenn der Pinsel hingegen einen Alphakanal besitzt, dann wird `trans` im Bereich von 0 (unsichtbar) bis 255 (sichtbar) sein. Falls der Pinsel keinen Transparenzkanal hat, wird in `trans` -1 zurückgegeben.

Sie können auch Pixel von Pinseln lesen, indem der Pinsel mit `SelectBrush()` als Ausgabeziel ausgewählt wurde und dann den Befehl `ReadPixel()` aufrufen. Die Verwendung von `ReadBrushPixel()` ist für die meisten Fälle jedoch schneller, weil es gleichzeitig auf Farbe und Transparenzkanäle zugreifen kann und Sie vermeiden evtl. einen Overhead, der durch den Aufruf von `SelectBrush()` und `EndSelect()` generiert werden kann.

Beachten Sie, dass wenn Sie `ReadBrushPixel()` mit einem Palettenpinsel verwenden, dieser Befehl nicht die **RGB-Farbe** zurückgibt, sondern den Stift an der angegebenen Position.

Siehe auch `FloodFill()`, `QuantizeBrush()`, `ReplaceColors()` und `WriteBrushPixel()`.

#### EINGABEN

`id`            ID des Pinsels  
`x`            Koordinate x des Pixels

y            Koordinate y des Pixels

## RÜCKGABEWERTE

color        RGB-Farbe oder Stift des Pixels

trans        Transparenzwert des Pixels

## BEISPIEL

```
color, trans = ReadBrushPixel(1, 100, 100)
```

Liest den Pixelzustand des Pinsels 1 an der Position 100:100.

## 43.55 ReduceAlphaChannel

### BEZEICHNUNG

ReduceAlphaChannel – reduziert die Intensität des Alphakanals (V6.0)

### ÜBERSICHT

```
ReduceAlphaChannel(id, ratio)
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die Intensität des Alphakanals des Pinsels zu reduzieren. Jedes Alphapixel wird in dem Verhältnis vervielfacht, das Sie in Argument **ratio** übergeben. Der Wert muss zwischen 0 und 255 liegen. **ratio** von 255 bedeutet 1.0 oder 100%, während 0 die Bedeutung 0.0 bzw. 0% hat. Wenn Sie also die Alpha-transparenz aller Pixel um 50% reduzieren wollen, würden Sie 128 im Argument **ratio** übergeben.

**ratio** kann auch eine Zeichenfolge mit einer prozentualen Angabe enthalten, z.B. "50%". Siehe auch [ChangeBrushTransparency\(\)](#), [DeleteAlphaChannel\(\)](#), [DeleteMask\(\)](#), [InvertAlphaChannel\(\)](#), [InvertBrush\(\)](#) Befehl, [InvertMask\(\)](#), [IsBrushEmpty\(\)](#), [SetAlphaIntensity\(\)](#), [SetBrushTransparency\(\)](#), [SetBrushTransparentPen\(\)](#) und [SetMaskMode\(\)](#).

### EINGABEN

id            ID des Pinsels, dessen Alphakanal Sie ändern möchten

ratio        Wert zwischen 0 und 255 oder eine prozentuale Angabe als Zeichenkette wie z.B. "50%"

## 43.56 RemapBrush

### BEZEICHNUNG

RemapBrush – ordnet die Pinselfarben neu zu (V9.0)

### ÜBERSICHT

```
RemapBrush(id, palid[, t])
```

### BESCHREIBUNG

Mit diesem Befehl können die Farben des von **id** angegebenen Pinsels den Farben der in **palid** angegebenen Palette neu zugeordnet werden. Der Quellpinsel kann entweder ein

normaler Pinsel oder ein Palettenpinsel sein. Handelt es sich um einen normalen Pinsel, wandelt `RemapBrush()` ihn beim erneuten Zuordnen auch in einen Palettenpinsel um, sodass der resultierende Pinsel immer ein Palettenpinsel ist.

Mit dem optionalen Tabellenargument `t` können zusätzliche Optionen angegeben werden. Die folgenden Tabellen-Tags werden derzeit erkannt:

**Dither:** Mit diesem Tag können Sie steuern, ob Dithering verwendet werden soll oder nicht. Setzen Sie dies auf `True`, um das Dithering zu aktivieren, oder auf `False`, um es zu deaktivieren. Der Standardwert ist `True`.

Beachten Sie, dass Sie, wenn der Pinsel Transparenz verwendet, zuerst `SetTransparentPen()` für die Palette verwenden müssen, um einen Stift zu definieren, der transparent gemacht werden soll.

Siehe auch `GetBrushPen()`, `RemoveBrushPalette()`, `SetBrushDepth()`, `SetBrushPalette()`, `SetBrushPen()` und `SetBrushTransparentPen()`.

## EINGABEN

<code>id</code>	Identifikator des neu zuzuordnenden Pinsels
<code>palid</code>	Identifikator der Palette, auf deren Farben der Pinsel abgebildet werden soll
<code>t</code>	optional: Tabelle mit weiteren Argumenten

## BEISPIEL

```
CreatePalette(1, {#BLACK, #WHITE}, {Depth = 1})
RemapBrush(1, 1, {Dither = True})
```

Konvertiert Pinsel 1 in einen schwarz-weißen Palettenpinsel. Die Neuordnung erfolgt bei aktiviertem Dithering.

## 43.57 RemoveBrushPalette

### BEZEICHNUNG

`RemoveBrushPalette` – wandelt Paletten- in RGB-Pinsel um (V9.0)

### ÜBERSICHT

```
RemoveBrushPalette(id[, trans])
```

### BESCHREIBUNG

Mit diesem Befehl kann der von `id` angegebene Palettenpinsel in einen RGB-Pinsel konvertiert werden. Das bedeutet, dass alle Pinselpixel in 32-Bit-RGB konvertiert werden und die Palette wird aus dem Pinsel entfernt. Mit dem optionalen Argument `trans` können Sie angeben, wie die Transparenz des Pinsels konvertiert werden soll. Dies kann entweder `#MASK` oder `#ALPHACHANNEL` sein. Wenn Sie es auf `#MASK` setzen, was auch die Voreinstellung ist, wird der transparente Stift des Pinsels einer Maske zugeordnet. Wenn `#ALPHACHANNEL` eingestellt ist, wird der transparente Stift des Pinsels einem Alphakanal zugeordnet.

Siehe auch `GetBrushPen()`, `RemapBrush()`, `SetBrushDepth()`, `SetBrushPalette()`, `SetBrushPen()` und `SetBrushTransparentPen()`.

**EINGABEN**

<b>id</b>	Identifikator des zu konvertierenden Pinsels
<b>trans</b>	optional: gewünschter Typ der Pinseltransparenz; muss entweder <b>#MASK</b> oder <b>#ALPHACHANNEL</b> sein; Voreinstellung ist <b>#MASK</b>

## 43.58 ReplaceColors

**BEZEICHNUNG**

ReplaceColors – ersetzt eine Farbe in einem Pinsel (V1.5)

**ÜBERSICHT**

ReplaceColors(id, colors)

**BESCHREIBUNG**

Dieser Befehl ersetzt die Farben, die in der Tabelle **colors** definiert sind, durch die angegebenen Ersatzfarben im Pinsel **id**. Sie übergeben ein Feld, welches wie folgt organisiert sein muss:

Suchfarbe 1, Ersatzfarbe 1, Suchfarbe 2, Ersatzfarbe 2, .....

Beachten Sie, dass Sie bei der Übergabe eines Palettenpinsels in **id** im Argument **colors** Stifte anstelle von Farben übergeben müssen.

Siehe auch [FloodFill\(\)](#), [QuantizeBrush\(\)](#), [ReadBrushPixel\(\)](#) und [WriteBrushPixel\(\)](#).

**EINGABEN**

<b>id</b>	Kennung des zu benutzenden Pinsels
<b>colors</b>	Farbfeld mit Informationen über Such- und Ersatzfarben (oder Stifte)

**BEISPIEL**

```
ReplaceColors(1, {#BLACK, #WHITE, #RED, #GREEN})
```

Der obige Code ersetzt alle schwarzen Bildpunkte in Pinsel 1 durch weiße und alle roten durch grüne.

## 43.59 RGBArrayToBrush

**BEZEICHNUNG**

RGBArrayToBrush – wandelt ein Pixelarray in einen Pinsel um (V5.0)

**ÜBERSICHT**

```
[id] = RGBArrayToBrush(id, table, width, height[, transtype, invalpha])
```

**BESCHREIBUNG**

Dieser Befehl erstellt einen neuen Pinsel mit dem Array von RGB-Pixel, welche in **table** angegeben wurde. Die Tabelle kann als Matrix angesehen werden: Die Pinselhöhe ist die Anzahl der Zeilen, die Pinselbreite die Anzahl Elemente. Die Reihenfolge der Pixeldaten in dieser Tabelle geht von oben nach unten, das heißt die Tabelle mit der ersten Reihe von Pixeln beginnt. Jede Zeile muss genau die Breite **width** des Pinsels haben und und es muss mindestens die Anzahl Zeilen **height** für die Pixelhöhe vorhanden sein.

Die einzelnen Pixel müssen im RGB-Format mit einem optionalen Alphawert übergeben werden. Das Argument **transtype** ermöglicht es Ihnen, den Transparenztyp anzugeben, welcher der neue Pinsel verwenden soll. Dies kann entweder **#NONE** (ohne Transparenz), **#MASK** (monochrome Transparenz) oder **#ALPHACHANNEL** (Alphakanal Transparenz) sein. Das optionale Argument **invalpha** kann verwendet werden, damit **RGBArrayToBrush()** alle Alphakanalwerte invertiert. Dies bedeutet, dass ein Wert von 0 100% sichtbar und ein Wert von 255 völlig unsichtbar ist. Normalerweise ist es genau umgekehrt. Aus historischen Gründen verwendet die Darstellungsbibliothek von Hollywood **invertierte Alphawerte** und aus diesem Grund wird dies auch von **RGBArrayToBrush()** unterstützt, obwohl es nicht der Standard ist.

Wenn **transtype** auf **#NONE** gesetzt ist, werden die Alphawerte ignoriert und **invalpha** hat keine Auswirkungen.

Bitte beachten Sie, dass die Tabelle, die Sie diesem Befehl übergeben, in der Regel viel Speicher in Anspruch nimmt. Daher sollten Sie diese Tabelle auf **Nil** setzen, sobald Sie sie nicht mehr benötigen. Andernfalls werden Sie riesige Mengen an Speicher verschwenden und es könnte sogar passieren, dass Ihrem Skript kein Speicher mehr zur Verfügung steht. Um einen Pinsel in ein Pixelarray zu konvertieren, können Sie den Befehl **BrushToRGBArray()** verwenden.

Siehe auch **BrushToPenArray()** und **PenArrayToBrush()**.

## EINGABEN

<b>id</b>	ID des neuen Pinsels oder <b>Nil</b> für die <b>automatische ID-Zuweisung</b>
<b>table</b>	Tabelle mit Array aus RGB-Pixel, die den Inhalt des neuen Pinsels beschreibt; wenn <b>transtype</b> gesetzt ist ( <b>#MASK</b> oder <b>#ALPHACHANNEL</b> ), müssen Sie auch Alphawerte in den höchsten 8 Bits angeben
<b>width</b>	Anzahl der Elemente in jeder Reihe
<b>height</b>	Anzahl der Spalten in der Tabelle
<b>transtype</b>	optional: gewünschte Transparenz für den Pinsel (voreingestellt ist <b>#NONE</b> )
<b>invalpha</b>	optional: ob Alphawerte invertiert werden sollen (voreingestellt ist <b>False</b> , somit werden Alphawerte nicht invertiert)

## RÜCKGABEWERTE

<b>id</b>	optional: Identifikator des Pinsels; Wird nur zurückgegeben werden, wenn Sie <b>Nil</b> als Argument 1 angegeben haben (siehe oben)
-----------	---

## BEISPIEL

```

pixels = {}
col = #BLUE
stp = 256 / 480
For Local y = 0 To 479
    For Local x = 0 To 639 Do pixels[y * 640 + x] = col
    col = col - stp
Next

```



```

RGBArrayToBrush(1, pixels, 640, 480)
pixels = Nil      ; WICHTIG: Speicher freigeben!
DisplayBrush(1, 0, 0)

```

Der obige Code erzeugt einen Farbverlauf von Rot nach Schwarz (**#BLACK**) und wandelt ihn mit **RGBArrayToBrush()** in einen Pinsel um. Wichtig: Vergessen Sie nicht, den Pixelarray auf **Nil** zu setzen, wenn Sie ihn nicht mehr benötigen, weil Pixelarrays eine riesige Mengen an Speicher beanspruchen!

## 43.60 RotateBrush

### BEZEICHNUNG

RotateBrush – dreht einen Pinsel (V1.5)

### ÜBERSICHT

```
RotateBrush(id, angle[, factorx, factory, smooth])
```

### BESCHREIBUNG

Dieser Befehl rotiert den Pinsel, der in **id** übergeben wurde, in einem bestimmten Winkel (in Grad). Ein positiver Winkel rotiert gegen den Uhrzeigersinn, eine negativer rotiert im Uhrzeigersinn.

Ab Hollywood 2.5 kann dieser Befehl auch den Pinsel skalieren, während er sich dreht. Dies wird in einem Durchgang durchgeführt, so dass die Qualität der resultierenden Bilddaten viel besser ist, als wenn Sie zuerst die Befehle **ScaleBrush()** und dann **RotateBrush()** aufrufen. Wenn Sie den Pinsel mit der Rotation skaliert haben wollen, übergeben Sie einfach zwei Skalierungsfaktoren in **factorx** und **factory**. Diese beiden Faktoren sind Fließkommazahlen und repräsentieren den Zoomfaktor (1 entspricht 100%, 0.5 50%, 1.5 150% etc.)

Schließlich kann das optionale Argument **smooth** verwendet werden, um Antialiasing Pixelinterpolation beim rotieren und/oder skalieren zu ermöglichen, die zu einem glatteren Aussehen führt, aber dessen Berechnung länger dauert.

Bitte beachten Sie:

- Wenn Sie einen Pinsel zum Beispiel durch einen Winkel von 45 Grad drehen und den Pinsel nicht über eine Maske verfügt, wird Hollywood automatisch eine Maske für diesen Pinsel erstellen, da der Drehvorgang in der Regel einige ungenutzte Flächen im Pinsel führt. Wenn der Pinsel eine Maske hat, dann wird Hollywood diese Maske auch drehen.
- Sie sollten auch nicht einen schon rotierten Pinsel nochmals drehen, da dies Datenverlust zur Folge hat. Benutzen Sie immer den Originalpinsel, wenn Sie Ihn drehen wollen; z.B. wenn Sie einen Pinsel erst um 45 Grad drehen und dann wieder um -45 Grad zurück, dann wird der Pinsel nicht mehr dieselbe Qualität haben wie der Originalpinsel.
- Beachten Sie, dass **RotateBrush()** bei Vektorpinseln immer mit dem nicht transformierten Pinsel arbeitet. Das bedeutet, dass alle vorherigen Transformationen, die mit **RotateBrush()**, **ScaleBrush()** oder **TransformBrush()** auf den Pinsel angewendet wurden, beim Aufruf von **RotateBrush()** rückgängig gemacht werden.

Weitere Pinseleffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `InvertBrush()` Befehl, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `SepiaToneBrush()`, `ScaleBrush()`, `SharpenBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()`, `TransformBrush()`, und `WaterRippleBrush()`.

## EINGABEN

<code>id</code>	Pinself, der gedreht wird
<code>angle</code>	Rotationswinkel in Grad
<code>factorx</code>	optional: Skalierfaktor der x-Achse (voreingestellt ist 1 und bedeutet keine Skalierung dieser Achse) (V2.5)
<code>factory</code>	optional: Skalierfaktor der y-Achse (voreingestellt ist 1 und bedeutet keine Skalierung dieser Achse) (V2.5)
<code>smooth</code>	optional: mit <code>True</code> wird Antialiasing beim Drehen und/oder Skalieren verwendet (Standard ist <code>False</code> ) (V2.5)

## 43.61 SaveBrush

### BEZEICHNUNG

`SaveBrush` – speichert den Pinsel als Datei ab (V2.0)

### ÜBERSICHT

`SaveBrush(id, f$[, t])`

### FRÜHERE SYNTAX

`SaveBrush(id, f$[, transcolor, fmt, table])`

### BESCHREIBUNG

Dieser Befehl speichert den in `id` angegebenen Pinsel in die von Ihnen in `f$` angegebenen Datei ab. Standardmäßig wird der Pinsel als Windows-Bitmap-Datei (BMP) gespeichert. Dies kann geändert werden, wenn Sie beim Tag `Format` ein anderes Format auswählen (siehe unten für Details).

`SaveBrush()` unterstützt mehrere optionale Argumente. Vor Hollywood 9.0 mussten diese als optionale Parameter übergeben werden (siehe oben). Seit Hollywood 9.0 wird jedoch empfohlen, die neue Syntax zu verwenden, die ein einzelnes optionales Tabellenargument hat, mit dem ein oder mehrere optionale Argumente an `SaveBrush()` übergeben werden können.

Die folgenden Tabellenfelder werden von diesem Befehl erkannt:

**Format:** Setzen Sie diesen Tag auf das Bildformat, das verwendet werden soll. Dies kann entweder eine der folgenden Konstanten oder ein von einem Plugin bereitgestelltes Bildformat sein:

**#IMGFMT\_BMP:**

Windows-Bitmap. Hollywood unterstützt beim BMP-Format RGB- und Palettenbilder. **#IMGFMT\_BMP** ist das von **SaveBrush()** verwendete Standardformat.

**#IMGFMT\_PNG:**

PNG-Format. Hollywood unterstützt beim PNG-Format RGB- und Palettenbilder. RGB-Bilder können auch einen Alphakanal und Palettenbilder können einen transparenten Stift haben. (V2.5)

**#IMGFMT\_JPEG:**

JPEG-Format. Beachten Sie, dass das JPEG-Format keine Alphakanäle oder palettenbasierte Grafiken unterstützt. Im Feld **Qualität** (siehe unten) können Sie die Qualitätsstufe für das JPEG-Bild festlegen (gültige Werte sind 0 bis 100, wobei 100 die beste Qualität ist). (V4.0)

**#IMGFMT\_GIF:**

GIF-Format. Da GIF-Bilder immer palettenbasiert sind, müssen RGB-Grafiken quantisiert werden, bevor sie als GIF exportiert werden können. Sie können die Tags **Colors** und **Dither** (siehe unten) verwenden, um entweder die Farbtiefe oder die Anzahl Farben in der Palette festzulegen, die dem Bild zugewiesen werden sollen und ob Dithering angewendet werden soll oder nicht. Wenn **#IMGFMT\_GIF** mit einem Palettenpinsel verwendet wird, erfolgt keine Quantisierung. **#IMGFMT\_GIF** unterstützt auch Palettenbilder mit einem transparenten Stift. (V4.5)

**#IMGFMT\_ILBM:**

IFF-ILBM-Format. Hollywood unterstützt beim IFF-ILBM-Format RGB- und Palettenbilder. Palettenbilder können auch einen transparenten Stift haben, allerdings werden Alphakanäle für dieses Ausgabeformat nicht unterstützt. (V4.5)

Voreingestellt ist **#IMGFMT\_BMP**.

- Dither:** Wenn Sie diesen Tag auf **True** setzen, wird Dithering angewendet. Dieser Tag wird nur verarbeitet, wenn das Zielformat Paletten unterstützt und die Quelldaten sind im RGB-Format. Der Standardwert ist **False**, was kein Dithering bedeutet.
- Depth:** Gibt die gewünschte Bildfarbtiefe an. Dies wird nur berücksichtigt, wenn das Format palettenbasiert ist und die Quelldaten im RGB-Format vorliegen. Gültige Werte liegen zwischen 1 (= 2 Farben) und 8 (= 256 Farben). Die Voreinstellung ist 8. (V9.0)
- Colors:** Dies ist eine Alternative zum Tag **Depth**. Anstelle einer Bittiefe können Sie hier angeben, wie viele Farben das Bild verwenden soll. Auch dies wird nur berücksichtigt, wenn das Format palettenbasiert ist und die Quelldaten im RGB-Format vorliegen. Gültige Werte liegen zwischen 1 und 256. Die Voreinstellung ist 256.

**Quality:** Hier können Sie einen Wert zwischen 0 und 100 für die Komprimierungsqualität bei verlustbehafteten Kompressionsformate angeben. Ein Wert von 100 bedeutet beste Qualität, 0 hingegen schlechteste Qualität. Dies wird nur bei Bildformaten berücksichtigt, die verlustbehaftete Kompression unterstützen. Der Standardwert ist 90, welcher eine ziemlich gute Qualität bedeutet.

**FillColor:**

Beim Speichern eines RGB-Bildes mit transparenten Pixeln können Sie hier eine RGB-Farbe angeben, die in alle transparenten Pixel geschrieben werden soll. Dies ist wahrscheinlich von wenig praktischem Nutzen. Der Standardwert ist `#NOCOLOR`, was bedeutet, dass transparente Pixel so belassen werden, wie sie sind. (V9.0)

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, die überprüfen werden, ob sie die angegebene Datei speichern können. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Zeichenkette setzen, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V10.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an den Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Hier ist eine Übersicht der Felder über die unterstützten Formate:

	<b>BMP</b>	<b>PNG</b>	<b>JPEG</b>	<b>GIF</b>	<b>ILBM</b>
Dither	Nein	Nein	Nein	Ja	Nein
Colors	Nein	Nein	Nein	Ja	Nein
Quality	Nein	Nein	Ja	Nein	Nein

Siehe auch [@BRUSH](#), [LoadBrush\(\)](#), [CopyBrush\(\)](#), [FreeBrush\(\)](#) und [MoveBrush\(\)](#).

## EINGABEN

**id** ID des Pinsels, der gespeichert werden soll

**f\$** Dateiname

**t** optional: Tabelle mit weiteren Optionen (siehe oben) (V9.0)

## BEISPIEL

```
SaveBrush(1, "test.jpg", {Format = #IMGFMT_JPEG, Quality = 80})
```

Der obige Code speichert Pinsel 1 als "test.jpg" mit einer Qualität von 80%.

## 43.62 ScaleBrush

### BEZEICHNUNG

ScaleBrush – skaliert einen Pinsel

### ÜBERSICHT

ScaleBrush(id, width, height[, smooth])

### BESCHREIBUNG

Dieser Befehl skaliert den in `id` angegebenen Pinsel auf die Breite `width` und der Höhe `height`. Optional können Sie das Argument `smooth` auf `True` setzen, um Antialiasing beim skalieren zu benutzen, die zu einem glatteren Aussehen führt, aber dessen Berechnung länger dauert.

Bitte beachten Sie: Sie sollten beim Skalieren immer eine Kopie des Originalpinsel benutzen. Zum Beispiel, wenn Sie Pinsel 1 auf 12x8 skalieren und später wieder auf 640x480 skalieren, erhalten Sie ein unbrauchbares Bild. Deshalb sollten Sie immer vom Originalpinsel eine Kopie erstellen und erst dann skalieren.

Beachten Sie, dass `ScaleBrush()` bei Vektorpinseln immer mit dem nicht transformierten Pinsel arbeitet. Das bedeutet, dass alle vorherigen Transformationen, die mit `ScaleBrush()`, `TransformBrush()` oder `RotateBrush()` auf den Pinsel angewendet wurden, beim Aufruf von `ScaleBrush()` rückgängig gemacht werden.

Neu in V2.0: Sie können `#KEEPASPRAT` entweder bei der Breite oder Höhe angeben. Hollywood wird dann die Größe automatisch berechnen, indem das Seitenverhältnis des Pinsels berücksichtigt wird.

Ab Hollywood 2.0 können Breite und Höhe auch eine Zeichenfolge als eine prozentuale Angabe enthalten, z.B. "50%".

Weitere Pinseffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `InvertBrush()` Befehl, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `SepiaToneBrush()`, `SharpenBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()`, `TransformBrush()`, und `WaterRippleBrush()`.

### EINGABEN

<code>id</code>	ID des Pinsels, der skaliert wird
<code>width</code>	gewünschte neue Breite des Pinsels
<code>height</code>	gewünschte neue Höhe des Pinsels
<code>smooth</code>	optional: mit <code>True</code> wird Antialiasing beim Skalieren verwendet (Standard ist <code>False</code> ) (V2.5)

### BEISPIEL

```
ScaleBrush(1,640,480)
```

Skaliert den Pinsel 1 auf die Größe 640x480.

## 43.63 SelectAlphaChannel

### BEZEICHNUNG

SelectAlphaChannel – wählt einen Alphakanal als Ausgabeziel (V2.0)

### ÜBERSICHT

SelectAlphaChannel(id[, type, frame])

### BESCHREIBUNG

Dieser Befehl wählt den Alphakanal des in `id` angegebenen Grafikobjekts als aktuelles Ausgabeziel. Dies bedeutet, dass alle Grafikdaten in diesen Alphakanal gezeichnet werden. Alphakanäle sind in Hollywood keine eigenständige Objekte; sie gehören immer zu einem Bildobjekt, einem Pinsel oder einer Animation.

In der Standardeinstellung arbeitet `SelectAlphaChannel()` immer mit den Alphakanälen von Pinseln. Ab Hollywood 4.5 können Sie jedoch auch in die Alphakanäle von Animationen und BGPics zeichnen. Sie müssen dann beim optionalen Argument `type` `#ANIM` oder `#BGPIC` angeben. Wenn Sie `#ANIM` verwenden, müssen Sie auch das Einzelbild angeben. Siehe `SelectAnim()` für weitere Informationen. Wenn Sie `#BGPIC` in `type` angeben, beachten Sie, dass Sie nur den Alphakanal eines BGPics ändern können, das zur Zeit nicht einem Display zugewiesen ist. Ab 4.7 können Sie auch als `type` `#LAYER` angeben, um den Alphakanal einer Ebene zu ändern. Beachten Sie, dass wenn die Ebene eine Animeebene ist, müssen Sie auch die Nummer des Einzelbildes angeben.

Alphakanäle können verwendet werden, um jedem Pixel seine eigene Transparenzeinstellung zu geben. Es stehen 256 verschiedene Transparenzstufen für jedes Pixel zur Verfügung. Ein Alphakanalwert von 0 bedeutet, dass das Pixel vollständig transparent ist und bei einem Wert von 255 das Pixel undurchsichtig ist. Alle Grafikbefehle von Hollywood wenden eine statische Alphakanalintensität auf den Alphakanal des Grafikobjekts an. Sie können diese Intensität mit dem Befehl `SetAlphaIntensity()` konfigurieren. Alternativ können Sie den Alphadarstellungsmodus in den Vanilla-Kopiermodus einstellen. Dies geschieht mit dem Befehl `SetAlphaIntensity()` und mit `#VANILLACOPY` als Argument. Dann werden alle Grafikbefehle von Hollywood, die Alphakanalpixel ausgeben, die exakten Alphakanaldaten auf den Alphakanal des Pinsels kopieren. Dieser Vanilla-Kopiermodus ist eine neue Funktionalität von Hollywood 2.5.

Das Argument `color`, welches mehrere Hollywood-Befehle (wie `Box()` oder `Circle()`) erwarten, ist überflüssig, wenn sie Alphakanäle bearbeiten. Für die Einstellung der Intensität benutzen Sie den Befehl `SetAlphaIntensity()`, wenn Sie mit Alphakanälen arbeiten.

Alphakanäle werden in der Regel für die schönen Glanzeffekte im Hintergrund oder für Antialiasing-Kanten verwendet. Wenn Sie anstelle von unterschiedlichen Transparenzstufen nur zwei Möglichkeiten brauchen, nämlich transparente und undurchsichtige Pixel, sollten Sie `SelectMask()` verwenden, weil Masken schneller gezeichnet werden können. Bitte beachten Sie, dass Grafikobjekte nicht eine Maske und einen Alphakanal haben können, womit nur eine Transparenzeinstellung möglich ist. Wenn Sie also auf ein Grafikobjekt mit diesem Befehl zugreifen, welches eine Maske hat, wird diese Maske zuerst gelöscht werden.

Wenn die in `id` angegebene Grafik noch keinen Alphakanal hat, wird automatisch einer erstellt, wenn Sie einen Befehl aufrufen, der auf den Alphakanal zeichnen will. Der Alphakanal wird undurchsichtig erstellt, das heißt, jedes Pixel wird eine Alphaintensität von

255 haben (0% transparent). Um den Alphakanal-Darstellungsmodus zu beenden und wieder zurück zur Displayausgabe zu gelangen, rufen Sie einfach den Befehl `EndSelect()` auf.

Wenn Sie einen Alphakanal nicht mehr benötigen, sollten Sie ihn mit dem Befehl `DeleteAlphaChannel()` vom Pinsel entfernen.

Sie können keine Pinsellinks mit diesem Befehl erstellen, da die Grafikdaten zurzeit geändert werden. Es ist auch verboten Befehle aufzurufen, die die Abmessungen des Pinsels/der Animation ändern, die derzeit als Ausgabeziel verwendet werden. Dies betrifft z.B. die Befehle `ScaleBrush()` oder `ScaleAnim()`. Darüber hinaus ist es für Animationen auch nicht erlaubt, `SelectAlphaChannel()` zu verwenden, wenn sie von der Festplatte geladen wird. Animationen müssen sich immer vollständig im Speicher befinden, wenn Sie mit `SelectAlphaChannel()` zeichnen möchten.

Nur Befehle, die statische Grafiken ausgeben, können verwendet werden, wenn `SelectAlphaChannel()` aktiv ist. Sie können keine animierte Befehle wie `MoveBrush()` oder `DisplayBrushFX()` benutzen, wenn `SelectAlphaChannel()` aktiviert wurde.

Wenn Sie bei `type #LAYER` verwenden und die angegebene Ebene eine Vektorebene ist, wird `SelectAlphaChannel()` die Ebene zuerst in eine Pinselebene rastern. Siehe [Abschnitt 25.39 \[SelectLayer\]](#), [Seite 436](#), für Details.

Siehe auch `EndSelect()`, `SelectBrush()`, `SelectMask()`, `SelectLayer()`, `SelectBGPic()` und `SelectAnim()`.

## EINGABEN

<code>id</code>	Grafikobjekt, dessen Alphakanal als Ausgabeziel verwendet wird
<code>type</code>	optional: Typ des in <code>id</code> angegebenen Grafikobjekt; kann <code>#BRUSH</code> , <code>#ANIM</code> , <code>#BGPIC</code> oder <code>#LAYER</code> sein (voreingestellt ist <code>#BRUSH</code> ) (V4.5)
<code>frame</code>	optional: Einzelbild der Animation; nur erforderlich, wenn als Typ <code>#ANIM</code> oder <code>#LAYER</code> (falls die angegebene Ebene eine Animationsebene ist) angegeben wurde (V4.5)

## BEISPIEL

```
CreateBrush(1, 256, 50, #RED)      ; erstellt 256x50 großen Pinsel
SelectAlphaChannel(1)              ; Ausgabeziel ist der Alphakanal
For k = 255 To 0 Step -1
    SetAlphaIntensity(k)           ; Setzt Intensität auf k
    Line(255 - k, 0, 255 - k, 49) ; zeichnet Linien mit verschiedenen ...
Next                               ; ... Intensitäten
EndSelect                          ; Display ist wieder das Ausgabeziel
```

```
DisplayBrush(1, #CENTER, #CENTER)
```

Dieser Code erstellt einen Pinsel mit einer Breite von 256 Pixel und 25 Linien. Anschließend werden die 256 verschiedenen Transparenzstufen in den Pinsel gezeichnet. Das Ergebnis wird ein rotes Rechteck sein, welches glatt in das Hintergrundbild übergeht. Bitte beachten Sie, dass Sie einen Bildschirm mit 24 Bit für den vollen Augenschmaus benötigen. Bildschirme mit 15 und 16 Bit haben nicht genügend Farben, um alle unterschiedlichen Ebenen anzuzeigen.

## 43.64 SelectBrush

### BEZEICHNUNG

SelectBrush – wählt einen Pinsel als Ausgabeziel (V1.5)

### ÜBERSICHT

```
SelectBrush(id[, mode, combomode])
```

### BESCHREIBUNG

Dieser Befehl selektiert den in `id` angegebenen Pinsel als aktuelles Ziel für Ausgaben. Das bedeutet, dass alle Grafikdaten, die Hollywood ausgibt, in diesem Pinsel landen.

Das optionale Argument `mode` ist standardmäßig auf `#SELMODE_NORMAL` gesetzt, was bedeutet, dass nur die Farbkanäle des Pinsels geändert werden. Der Transparenzkanal des Pinsels (kann entweder eine Maske oder ein Alphakanal sein) wird nie verändert werden. Wenn Sie hingegen beim optionalen Argument `mode` `#SELMODE_COMBO` angeben, werden alle Grafikbefehle von Hollywood, welche nach `SelectBrush()` aufgerufen werden, in den Farb- und Transparenzkanal des Pinsels zeichnen. Wenn der Pinsel keinen Transparenzkanal hat, verhält sich `#SELMODE_COMBO` gleich wie `#SELMODE_NORMAL`.

Ab Hollywood 5.0 können Sie das optionale Argument `combomode` verwenden, um festzulegen, wie sich `#SELMODE_COMBO` verhalten soll. Wenn `combomode` auf 0 gesetzt ist, werden die Farb- und Transparenzinformation aller Pixel in dem Quellenbild in jedem Fall zu dem Zielbild kopiert, selbst wenn die Pixel unsichtbar sind. Dies ist auch voreingestellt. Wenn `combomode` hingegen auf 1 gesetzt ist, werden nur die sichtbaren Pixel in das Zielbild kopiert. Dies bedeutet, dass, wenn der Alphawert eines Pixels in dem Quellenbild 0 ist (unsichtbar), wird es nicht in das Zielbild kopiert werden.

Ab Hollywood 6.0 steht der neue `combomode` 2 zur Verfügung. Wenn Sie 2 in `combomode` übergeben, wird Hollywood die Farbkanäle und Alphakanal des Quellbildes mit der Farbe und Alpha der Zielbildkanäle mischen. Wenn Sie später das Zielbild zeichnen, wird es so aussehen, als ob ein Bild hintereinander auf das jeweils andere gezeichnet wurde. Bitte beachten Sie, dass das Argument `combomode` nur zusammen mit `#SELMODE_COMBO` unterstützt wird. Es hat keine Wirkung, wenn sie es mit den anderen Modi verwenden.

Ein alternativer Weg in die Transparenzkanäle eines Pinsels zu zeichnen, ist dies separat mit den Befehlen `SelectMask()` oder `SelectAlphaChannel()` zu erledigen. Diese beiden Befehle werden jedoch die Schreibdaten nur auf den Transparenzkanal anwenden. Sie werden den Farbkanal nicht ändern. Also, wenn Sie beide Kanäle (Farbe und Transparenz) ändern wollen, müssen Sie `SelectBrush()` mit `mode` `#SELMODE_COMBO` verwenden.

Wenn Sie mit dem ändern des Pinsels fertig sind und möchten, dass Ihr Display wieder das Ausgabeziel wird, rufen Sie einfach den Befehl `EndSelect()` auf.

Während `SelectBrush()` aktiv ist, ist es verboten, Befehle aufzurufen, welche die Abmessungen des Pinsels ändern, der derzeit als Ausgabepinsel verwendet wird. Z.B. können Sie Den Befehl `ScaleBrush()` nicht verwenden, um diesen Pinsel, zu skalieren.

Nur Befehle von Hollywood können verwendet werden, welche Grafiken direkt zeichnen, wenn `SelectBrush()` aktiv ist. Sie können keine animierten Befehle wie `MoveBrush()` oder `DisplayBrushFX()` aufrufen, während `SelectBrush()` aktiv ist.

Siehe auch `EndSelect()`, `SelectAlphaChannel()`, `SelectMask()`, `SelectLayer()`, `SelectBGPic()` und `SelectAnim()`.



**EINGABEN**

<b>id</b>	Pinsel, der als Ausgabeziel verwendet werden soll
<b>mode</b>	optional: Darstellungsmodus (siehe oben); dies kann entweder <b>#SELMODE_NORMAL</b> oder <b>#SELMODE_COMBO</b> sein; voreingestellt ist <b>#SELMODE_NORMAL</b> (V4.5)
<b>combomode</b>	optional: Wenn <b>#SELMODE_COMBO</b> aktiv ist, kann 0,1 oder 2 eingesetzt werden (siehe oben); voreingestellt ist 0 (V5.0)

**BEISPIEL**

```
CreateBrush(1, 320, 256)
SelectBrush(1)
SetFillStyle(#FILLCOLOR)
Box(0, 0, 320, 256, #RED)
EndSelect()
MoveBrush(1, #CENTER, #BOTTOMOUT, #CENTER, #TOPOUT, 10)
```

Der obige Code erzeugt einen 320x256 großen Pinsel, zieht ein rotes Rechteck hinein und rollt dann das Rechteck auf dem Bildschirm. Dies ist ein sehr abstraktes Beispiel. Sie können natürlich viel mehr mit diesem Befehl bewirken. Werfen Sie einen Blick auf die Beispiele, welche mit der Hollywood-Distribution mitgeliefert werden. Sie nutzen `SelectBrush()` in fortgeschritteneren Kontexten.

## 43.65 SelectMask

**BEZEICHNUNG**

SelectMask – wählt eine Maske als Ausgabeziel (V2.0)

**ÜBERSICHT**

```
SelectMask(id[, type, frame])
```

**BESCHREIBUNG**

Dieser Befehl wählt die Maske des in **id** angegebenen grafischen Objekts als das aktuelle Ausgabeziel. Dies bedeutet, dass alle Grafikdaten von Hollywood in diese Maske gezeichnet werden. Masken sind immer mit einem Bildobjekt verbunden, beispielsweise einem Pinsel oder einer Animation.

In der Standardeinstellung arbeitet `SelectMask()` immer mit den Masken von Pinseln. Um die Masken von Animationen und BGPics zu bearbeiten, können Sie ab Hollywood 4.5 im optionalen Argument **type** **#ANIM** oder **#BGPIC** übergeben. Wenn Sie **#ANIM** in **type** angeben, müssen Sie auch das Einzelbild der Animation festlegen. Siehe `SelectAnim()` für weitere Informationen. Wenn Sie **#BGPIC** in **type** verwenden, beachten Sie, dass Sie nur die Masken von BGPics ändern können, die derzeit nicht einem Display zugewiesen sind. Ab Hollywood 4.7 können Sie als **type** auch **#LAYER** verwenden, um die Maske einer Ebene zu verändern. Falls die Ebene eine Animebene ist, müssen Sie auch die Nummer des Einzelbildes angeben.

Masken werden verwendet, um die Transparenz eines graphischen Objekts zu steuern und tragen keine Farbinformationen in sich. Jedes Pixel in einer Maske kann nur zwei

verschiedene Zustände annehmen: 1, was bedeutet, dass dieses Pixel sichtbar, und 0, dass es nicht sichtbar ist. Daher müssen Sie Hollywood informieren, ob die Zeichnungsbefehle sichtbare (1) oder unsichtbare Pixel (0) in die Maske zeichnen soll. Dies wird durch die Verwendung des Befehls `SetMaskMode()` festgelegt. Das `color` Argument, welches mehrere Hollywood-Befehle (wie `Box()` oder `Circle()`) erwarten, ist bei Masken überflüssig. Sie müssen nur den Befehl `SetMaskMode()` verwenden.

Wenn die in `id` angegebene Grafik noch keine Maske hat, wird sie automatisch erstellt, wenn Sie einen Befehl aufrufen, der in die Maske zeichnen will. Wenn eine Maske durch `SelectMask()` erstellt wird, wird sie zunächst vollständig undurchsichtig sein.

Wenn Sie den Maskendarstellungsmodus verlassen möchten, und wieder in den Displaymodus zurück zu gehen, rufen Sie einfach den Befehl `EndSelect()` auf.

Wenn Sie eine Maske nicht mehr benötigen, können Sie sie aus dem Pinsel entfernen, indem Sie den Befehl `SetBrushTransparency()` mit dem Argument `#NOTRANSARENCY` aufrufen oder einfach `DeleteMask()` verwenden.

Sie können keine Pinsellinks mit diesem Befehl erstellen, da die Grafikdaten des Pinsels geändert werden. Es ist auch verboten Befehle aufzurufen, die die Abmessungen des Pinsels oder der Anim ändern, die derzeit als Ausgabeziel verwendet werden. Z.B. können Sie die Befehle `ScaleBrush()` oder `ScaleAnim()` nicht aufrufen, um den Pinsel/die Anim zu skalieren, die im Moment als Ausgabeziel definiert sind. Darüber hinaus darf für Animationen während dieser Zeit auch der Befehl `SelectMask()` nicht benutzt werden, wenn die Animation von der Festplatte geladen wird. Animationen müssen sich immer vollständig im Speicher befinden, wenn Sie mit `SelectMask()` in Einzelbilder zeichnen möchten.

Nur Befehle, die Grafikdaten direkt ausgeben, können nach `SelectMask()` verwendet werden. Auch animierte Befehle wie `MoveBrush()` oder `DisplayBrushFX()` dürfen nicht aufgerufen werden, während `SelectMask()` aktiv ist.

Bitte beachten Sie, dass Grafikobjekte nicht eine Maske und einen Alphakanal haben können, womit nur eine Transparenzeinstellung möglich ist. Wenn Sie also auf ein Grafikobjekt mit diesem Befehl zugreifen, das einen Alphakanal hat, wird dieser Alphakanal gelöscht werden.

Wenn Sie bei `type #LAYER` verwenden und die angegebene Ebene ist eine Vektorebene, wird `SelectMask()` die Ebene zuerst auf eine Pinselebene rastern. Siehe [Abschnitt 25.39 \[SelectLayer\]](#), Seite 436, für Details.

Siehe auch `EndSelect()`, `SelectAlphaChannel()`, `SelectBrush()`, `SelectLayer()`, `SelectBGPic()` und `SelectAnim()`.

## EINGABEN

<code>id</code>	Grafikobjekt, dessen Maske als Ausgabeziel verwendet wird
<code>type</code>	optional: Typ des in <code>id</code> angegebenen Grafikobjekt; kann <code>#BRUSH</code> , <code>#ANIM</code> , <code>#BGPIC</code> oder <code>#LAYER</code> sein (voreingestellt ist <code>#BRUSH</code> ) (V4.5)
<code>frame</code>	optional: Einzelbild der Animation; nur erforderlich, wenn als Typ <code>#ANIM</code> oder <code>#LAYER</code> (falls die angegebene Ebene eine Animationsebene ist) angegeben wurde (V4.5)

## BEISPIEL

```
w = GetAttribute(#BRUSH, 1, #ATTRWIDTH)
```

```

h = GetAttribute(#BRUSH, 1, #ATTRHEIGHT)

SetFillStyle(#FILLCOLOR)
SelectMask(1)           ; wählt die Maske als Ausgabeziel
SetMaskMode(#MASKINVISIBLE) ; mit unsichtbaren Pixel zeichnen
Cls                     ; löscht alle Pixel
SetMaskMode(#MASKVISIBLE)  ; mit sichtbaren Pixel zeichnen
Box(0, 0, w, h, 0, 20)   ; zeichnet Rechteck mit runden Kanten
EndSelect               ; wählt Display als Ausgabeziel

```

Der obige Code erstellt ein Rechteck mit abgerundeten Kanten auf die Maske des Pinsels 1. Wenn Pinsel 1 jetzt angezeigt wird, erscheint es mit abgerundeten Kanten.

## 43.66 SepiaToneBrush

### BEZEICHNUNG

SepiaToneBrush – wendet den Sepiatoneffekt auf den Pinsel an (V5.0)

### ÜBERSICHT

SepiaToneBrush(id, level)

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen Sepiatoneffekt auf den in `id` angegebenen Pinsel aufzutragen. Der Sepiatoneffekt versucht, das Aussehen von alten Fotografien zu simulieren. Das zweite Argument `level` steuert die Intensität der Sepiatönung und kann sich von 0 bis 255 bewegen. Sie können auch einen Prozentsatz als Zeichenkette angeben ("50%"). Üblicherweise wird ein Wert von etwa 204 verwendet (= 80%), der am besten aussieht.

Beachten Sie, dass dieser Befehl nicht mit Palettenpinseln verwendet werden kann.

Weitere Pinseleffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `InvertBrush()` Befehl, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `ScaleBrush()`, `SharpenBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()`, `TransformBrush()`, und `WaterRippleBrush()`. ■

### EINGABEN

<code>id</code>	Identifikator des Pinsels
<code>level</code>	gewünschte Stufe der Sepiatönung (0 bis 255 oder eine Zeichenfolge, die einen Prozentsatz enthält)

### BEISPIEL

```
SepiaToneBrush(1, "80%")
```

Der obige Code wendet den Sepiatoneffekt auf den Pinsel 1 mit einer Intensität von 80% an.

## 43.67 SetAlphaIntensity

### BEZEICHNUNG

SetAlphaIntensity – definiert die Intensität der Alphadarstellung (V2.0)

### ÜBERSICHT

SetAlphaIntensity(level)

### BESCHREIBUNG

Mit diesem Befehl können Sie den Grad der Transparenz aller Grafikbefehlen einstellen, wenn sie in den Alphakanal eines Pinsels zeichnen (wenn `SelectAlphaChannel()` aktiv ist). Der Transparenzgrad muss sich im Bereich von 0 bis 255 befinden, wobei 0 100% Transparenz und 255 keine Transparenz bedeutet. Die hier angegebene Intensität wird von allen Grafikbefehlen von Hollywood anstelle einer Farbe verwendet. Die Voreinstellung der Alpha-Intensität ist 128.

level kann auch eine Zeichenfolge mit prozentualer Angabe sein, z.B. "50%".

Bitte beachten Sie, dass dies genau die Umgekehrten Angaben von `SetLayerTransparency()` sind: Dort gilt 0 für keine und 255 für volle Transparenz.

Siehe [Abschnitt 43.63 \[SelectAlphaChannel\]](#), Seite 966, für mehr Informationen über den Alphakanal.

Neu in V2.5: Sie können nun die spezielle Konstante `#VANILLACOPY` im Argument `level` angeben. Wenn Sie dies tun, wird Hollywood den neuen Vanilla-Kopiermodus aktivieren. Dies bedeutet, dass alle Grafikbefehle, die Alphakanalpixel erstellen, diese Pixel direkt auf Ihrem Alphakanal des Pinsels kopieren. Z.B. wenn Sie einen Alphakanal eines Pinsels auswählen und dann mit dem Befehl `TextOut()` einen Antialiasing-Text zeichnen, wird Hollywood die genauen Alphakanaldaten des Antialiasing-Text in Ihren Alphakanal des Pinsels zeichnen, wo Sie sie weiter verarbeiten können. Wenn `#VANILLACOPY` aktiv ist und Sie Grafiken auf den Alphakanal zeichnen, der keine Alphadaten hat, wird Hollywood eine Alphaintensität von 255 (das heißt vollständig sichtbar) in den Alphakanal schreiben.

Siehe auch `ChangeBrushTransparency()`, `DeleteAlphaChannel()`, `DeleteMask()`, `InvertAlphaChannel()`, `InvertBrush()` Befehl, `InvertMask()`, `IsBrushEmpty()`, `ReduceAlphaChannel()`, `SetBrushTransparency()`, `SetBrushTransparentPen()` und `SetMaskMode()`.

### EINGABEN

level gewünschte Transparenzstufe (0 bis 255 oder Prozentangabe) oder: spezielle Konstante `#VANILLACOPY` für den speziellen Vanilla-Kopiermodus (V2.5)

### BEISPIEL

Siehe [Abschnitt 43.63 \[SelectAlphaChannel\]](#), Seite 966.

## 43.68 SetBrushDepth

### BEZEICHNUNG

SetBrushDepth – stellt die Farbtiefe der Pinselpalette ein (V9.0)

**ÜBERSICHT**

```
SetBrushDepth(id, depth[, t])
```

**BESCHREIBUNG**

Dieser Befehl setzt die Farbtiefe der von `id` angegebenen Pinselpalette auf die in `depth` angegebene. `depth` muss eine Bittiefe im Bereich von 1 (= 2 Farben) bis 8 (= 256 Farben) haben. Siehe [Abschnitt 41.1 \[Übersicht über die Paletten\], Seite 841](#), für Details. Beachten Sie, dass die Pixeldaten neu zugeordnet werden, wenn die angegebene Farbtiefe geringer ist als die der an die Palette angehängten Pixeldaten.

Ab Hollywood 10.0 akzeptiert `SetBrushDepth()` das optionale Tabellenargument `t`, welches die folgenden Tags enthalten kann:

**Remap:** Wenn dieser Tag auf `False` gesetzt ist, werden außerhalb des Bereichs befindliche Stifte nicht auf vorhandene Stifte neu zugeordnet, sondern einfach auf den im Tag `ClipPen` (siehe unten) angegebenen Stift gesetzt, d.h. es findet keine Neuordnung statt. Beachten Sie, dass **Remap** nur beim Reduzieren von Farben wirksam ist. Wenn die neue Farbtiefe mehr Stifte hat als die alte -tiefe, wird **Remap** nichts bewirken. (V10.0)

**ClipPen:** Dies wird nur verwendet, wenn der Tag **Remap** auf `False` gesetzt ist (siehe oben). In diesem Fall werden Stifte außerhalb des Bereichs nicht den bestehenden Stiften neu zugeordnet, sondern einfach auf den im Tag `ClipPen` angegebenen Stift gesetzt, d.h. es findet keine Neuordnung statt. Beachten Sie, dass **ClipPen** nur beim Reduzieren von Farben wirksam ist. Wenn die neue Farbtiefe mehr Stifte hat als die alte -tiefe, wird **ClipPen** nichts tun. (V10.0)

Siehe auch [GetBrushPen\(\)](#), [RemapBrush\(\)](#), [RemoveBrushPalette\(\)](#), [SetBrushPalette\(\)](#), [SetBrushPen\(\)](#) und [SetBrushTransparentPen\(\)](#).

**EINGABEN**

<code>id</code>	Identifikator des zu ändernden Pinsels
<code>depth</code>	gewünschte neue Palettenfarbtiefe (von 1 bis 8)
<code>t</code>	optional: Tabellenargument mit weiteren Optionen (siehe oben) (V10.0)

**BEISPIEL**

```
SetBrushDepth(1, 8)
```

Der obige Code ändert die Farbtiefe der Palette von Pinsel 1 auf 8 (= 256 Farben).

## 43.69 SetBrushPalette

**BEZEICHNUNG**

`SetBrushPalette` – wechselt die Pinselpalette (V9.0)

**ÜBERSICHT**

```
SetBrushPalette(id, palid[, t])
```

**BESCHREIBUNG**

Dieser Befehl ersetzt die durch **id** angegebene Palette des Pinsels durch die in **palid** angegebene. Mit dem optionalen Tabellenargument **t** können Sie einige weitere Optionen angeben. Die folgenden Tags werden derzeit vom optionalen Tabellenargument **t** unterstützt:

**Remap:** Wenn dies auf **True** gesetzt ist, werden die Pixel des Pinsels neu zugeordnet, um den Farben der neuen Palette so genau wie möglich zu entsprechen. Standardmäßig erfolgt keine Neuordnung und die tatsächlichen Pixeldaten des Pinsels bleiben unberührt. Wenn Sie eine Neuordnung wünschen, setzen Sie diesen Tag auf **True**. Beachten Sie jedoch, dass die Neuordnung aller Pixel natürlich viel länger dauert als das Festlegen einer neuen Palette ohne Neuordnung. Die Voreinstellung ist **False**.

**Dither:** Wenn Sie den Tag **Remap** (siehe oben) auf **True** gesetzt haben, können Sie mit dem Tag **Dither** angeben, ob Dithering verwendet werden soll oder nicht. Voreingestellt ist **True**, was bedeutet, dass Dithering verwendet wird.

**CopyCycleTable:**  
Paletten können eine Tabelle mit Farbwechselinformationen enthalten. Wenn Sie diesen Tag auf **True** setzen, wird diese Wechseltabelle ebenfalls in den Pinsel kopiert. Der Standardwert ist **False**.

Siehe auch **GetBrushPen()**, **RemapBrush()**, **RemoveBrushPalette()**, **SetBrushDepth()**, **SetBrushPen()** und **SetBrushTransparentPen()**.

**EINGABEN**

<b>id</b>	Identifikator des Pinsels
<b>palid</b>	Identifikator der Palette, die in den Pinsel kopiert werden soll
<b>t</b>	optional: Tabelle zur Angabe weiterer Optionen (siehe oben)

**43.70 SetBrushPen****BEZEICHNUNG**

**SetBrushPen** – wechselt den Stift der Pinselpalette (V9.0)

**ÜBERSICHT**

**SetBrushPen(id, pen, color)**

**BESCHREIBUNG**

Dieser Befehl setzt die Farbe des in **pen** angegebenen Stifts auf die durch **color** angegebene Farbe in der Palette des durch **id** angegebenen Pinsels.

Siehe auch **GetBrushPen()**, **RemapBrush()**, **RemoveBrushPalette()**, **SetBrushDepth()**, **SetBrushPalette()** und **SetBrushTransparentPen()**.

**EINGABEN**

<b>id</b>	Identifikator des Pinsels
<b>pen</b>	Stift, den Sie ändern möchten (ab 0)

`color`      neue Farbe für den Stift; muss als **RGB-Farbe** angegeben werden

### BEISPIEL

```
SetBrushPen(1, 0, #RED)
```

Mit dem obigen Code wird Stift 0 in der Palette von Pinsel 1 auf Rot gesetzt.

## 43.71 SetBrushTransparency

### BEZEICHNUNG

SetBrushTransparency – definiert die transparente Farbe eines Pinsels (V1.5)

### ÜBERSICHT

```
SetBrushTransparency(id, col)
```

### BESCHREIBUNG

Dieser Befehl lässt den Pinsel mit dem Identifikator `id` die durch in `col` angegebene Farbe transparent erscheinen. Dies wird durch die Erstellung einer Maske für den Pinsel durchgeführt. `SetBrushTransparency()` untersucht alle Pixel des Pinsels und maskiert alle Pixel, welche die in `col` angegebene Farbe haben. Die von diesem Befehl erstellte Maske wird nicht automatisch aktualisiert, wenn Sie `SelectBrush()` aufrufen. Daher ist es notwendig, den Befehl `SetBrushTransparency()` erneut nach einem Aufruf von `SelectBrush()` auszuführen, so dass die Maske aktualisiert wird.

Sie können mit diesem Befehl auch eine Maske aus einem Pinsel entfernen. Geben Sie einfach in `col` `#NOTTRANSPARENCY` als Farbe an.

Beachten Sie, dass dieser Befehl nicht mit Palettenpinseln verwendet werden kann. Sie können `SetTransparentPen()` verwenden, um den transparenten Stift in einem Palettenpinsel zu ändern. Siehe [Abschnitt 41.37 \[SetTransparentPen\]](#), [Seite 874](#), für Details.

Siehe auch `ChangeBrushTransparency()`, `DeleteAlphaChannel()`, `DeleteMask()`, `InvertAlphaChannel()`, `InvertBrush()` Befehl, `InvertMask()`, `IsBrushEmpty()`, `ReduceAlphaChannel()`, `SetAlphaIntensity()`, `SetBrushTransparentPen()` und `SetMaskMode()`.

### EINGABEN

`id`            Identifikator des Pinsels

`col`           Farbe, die transparent angezeigt wird oder `#NOTTRANSPARENCY`, um die Maske des Pinsels zu löschen

### BEISPIEL

```
CreateBrush(1, 320, 256)
SelectBrush(1)
SetFillStyle(#FILLCOLOR)
Circle(0, 0, 100, #RED)
EndSelect()
SetBrushTransparency(1, #BLACK)
MoveBrush(1, #CENTER, #BOTTOMOUT, #CENTER, #TOPOUT, 10)
```

Obenstehender Code erzeugt einen Pinsel, zeichnet einen gefüllten Kreis hinein und macht dann die schwarze Hintergrundfarbe transparent. Danach wird der Pinsel über

den Bildschirm bewegt. Es ist wichtig, dass Sie `SetBrushTransparency()` erst aufrufen, wenn die Ausgaben beendet sind, d.h. nachdem Sie `EndSelect()` aufgerufen haben.

## 43.72 SetBrushTransparentPen

### BEZEICHNUNG

`SetBrushTransparentPen` – stellt den transparenten Stift der Pinselpalette ein (V9.0)

### ÜBERSICHT

`SetBrushTransparentPen(id, pen)`

### BESCHREIBUNG

Dieser Befehl setzt den transparenten Stift der in `id` angegebenen Pinselpalette auf den in `pen` angegebenen Stift. Stifte werden ab 0 gezählt.

Siehe auch `ChangeBrushTransparency()`, `DeleteAlphaChannel()`, `DeleteMask()`, `InvertAlphaChannel()`, `InvertBrush()` Befehl, `InvertMask()`, `IsBrushEmpty()`, `ReduceAlphaChannel()`, `SetAlphaIntensity()`, `SetBrushTransparency()` und `SetMaskMode()`.

### EINGABEN

`id`            Identifikator des Pinsels

`pen`           gewünschter transparenter Stift (ab 0)

### BEISPIEL

`SetBrushTransparentPen(1, 4)`

Stift 4 in der Palette von Pinsel 1 wird transparent.

## 43.73 SetMaskMode

### BEZEICHNUNG

`SetMaskMode` – definiert den Darstellungsmodus für Masken (V2.0)

### ÜBERSICHT

`SetMaskMode(mode)`

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Darstellungsmodus zu definieren, wenn `SelectMask()` aktiv ist. Das Argument `mode` kann entweder `#MASKVISIBLE` oder `#MASKINVISIBLE` sein. Voreingestellt ist `#MASKVISIBLE`. Wenn Sie den sichtbaren Modus auswählen, werden alle Grafikbefehle sichtbare Pixel in die Maske, andernfalls unsichtbare zeichnen. Offensichtlich ist, dass eine Maske keine Farbinformationen, sondern nur diese beiden Möglichkeiten pro Pixel haben kann (sichtbar oder unsichtbar).

Seit Hollywood 4.0 werden folgende Maskenmodi unterstützt:

**#MASKVANILLACOPY:**

Die Maskendaten des Quellbildes werden genau auf das Ziel kopiert werden.



**#MASKAND:**

Maskendaten des Quellbildes werden an die Zielmaske mit einer logischen UND-Operation an jedem Pixel kopiert werden.

**#MASKOR:**

Maskendaten des Quellbildes wird an die Zielmaske mit einer logischen ODER-Operation an jedem Pixel kopiert werden.

**#MASKXOR:**

Maskendaten des Quellbildes wird an die Zielmaske mit einer logischen XOR-Operation an jedem Pixel kopiert werden.

Nachfolgend finden Sie eine Tabelle, die die verschiedenen Maskenmodi zusammenfasst. Bitte beachten Sie, dass die Maskenmodi **#MASKVISIBLE**, **#MASKINVISIBLE** und **#MASKVANILLACOPY** unabhängig von den Zielmaskendaten sind. Zielmaskendaten werden nur von den Modi **#MASKAND**, **#MASKOR** und **#MASKXOR** berücksichtigt.

Quelle	Ziel	VISIBLE	INVISIBLE	VANILLA	AND	OR	XOR
0	0	0	0	0	0	0	0
0	1	0	0	0	0	1	1
1	0	1	0	1	0	1	1
1	1	1	0	1	1	1	0

Siehe [Abschnitt 43.65 \[SelectMask\]](#), [Seite 969](#), für mehr Informationen über die Maske.

Siehe auch [ChangeBrushTransparency\(\)](#), [DeleteAlphaChannel\(\)](#), [DeleteMask\(\)](#), [InvertAlphaChannel\(\)](#), [InvertBrush\(\)](#) Befehl, [InvertMask\(\)](#), [IsBrushEmpty\(\)](#), [ReduceAlphaChannel\(\)](#), [SetAlphaIntensity\(\)](#), [SetBrushTransparency\(\)](#) und [SetBrushTransparentPen\(\)](#).

**EINGABEN**

**mode**          Darstellungsmodus der Maske (siehe oben)

**BEISPIEL**

Siehe [Abschnitt 43.65 \[SelectMask\]](#), [Seite 969](#).

## 43.74 SharpenBrush

**BEZEICHNUNG**

SharpenBrush – wendet den Scharfzeichnungsfilter auf den Pinsel an (V5.0)

**ÜBERSICHT**

SharpenBrush(id[, radius])

**BESCHREIBUNG**

Dieser Befehl schärft das Aussehen des angegebenen Pinsels. Das optionale Argument **radius** kann dazu verwendet werden, um den Schärfungsradius anzugeben. Je größer der Radius, desto länger braucht dieser Befehl, den Schärfungseffekt anzuwenden. Wenn Sie das optionale Argument nicht angeben, wird dieser Befehl automatisch einen entsprechenden Radius wählen.

Beachten Sie, dass dieser Befehl nicht mit Palettenpinseln verwendet werden kann.

Weitere Pinseleffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `InvertBrush()` Befehl, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `SepiaToneBrush()`, `ScaleBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()`, `TransformBrush()`, und `WaterRippleBrush()`.

#### EINGABEN

<code>id</code>	Pinself, der geschärft wird
<code>radius</code>	optional: Schärferradius (voreingestellt ist 0, was bedeutet, dass der Radius automatisch ausgewählt wird)

### 43.75 SolarizeBrush

#### BEZEICHNUNG

`SolarizeBrush` – wendet den Solarisationseffekt auf den Pinsel an (V5.0)

#### ÜBERSICHT

`SolarizeBrush(id, level)`

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen Solarisationseffekt auf den angegebenen Pinsel anzuwenden. Der Solarisationseffekt versucht, das Erscheinungsbild eines Films zu simulieren, der kurz dem Licht ausgesetzt wurde. Das zweite Argument `level` steuert die Intensität des Solarisationseffekts und kann einen beliebigen Wert zwischen 0 und 255 sein oder Sie benutzen einen Prozentsatz als Zeichenfolge (z.B. "50%").

Beachten Sie, dass wenn `id` einen Palettenpinsel angibt, `SolarizeBrush()` nur die Palettenfarben solarisiert, was diesen Befehl sehr schnell macht, wenn er mit Palettenpinseln verwendet wird.

Weitere Pinseleffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `InvertBrush()` Befehl, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `SepiaToneBrush()`, `ScaleBrush()`, `SharpenBrush()`, `SwirlBrush()`, `TintBrush()`, `TransformBrush()`, und `WaterRippleBrush()`.

#### EINGABEN

<code>id</code>	Pinself, auf den der Effekt angewendet wird
<code>level</code>	gewünschte Stufe der Solarisation (0 bis 255 oder eine Zeichenfolge, die einen Prozentsatz enthält)

## 43.76 SwirlBrush

### BEZEICHNUNG

SwirlBrush – wendet den Wirbeleffekt auf den Pinsel an (V5.0)

### ÜBERSICHT

SwirlBrush(id, degrees)

### BESCHREIBUNG

Dieser Befehl legt eine Wirbelwirkung auf den angegebenen Pinsel. Er wird um seinen Mittelpunkt um den in **degrees** angegebenen Betrag (in Grad) verwirbelt werden. Dies kann von 0 für keine Verwirbelung bis zu 360 für den dramatischsten Wirbeleffekt reichen.

Weitere Pinseleffekte: ArcDistortBrush(), BarrelDistortBrush(), BlurBrush(), BrushToGray(), BrushToMonochrome(), CharcoalBrush(), ContrastBrush(), EdgeBrush(), EmbossBrush(), FlipBrush() Befehl, GammaBrush(), InvertBrush() Befehl, MixBrush(), ModulateBrush(), OilPaintBrush(), PerspectiveDistortBrush(), PixelateBrush(), PolarDistortBrush(), RotateBrush(), SepiaToneBrush(), ScaleBrush(), SharpenBrush(), SolarizeBrush(), TintBrush(), TransformBrush(), und WaterRippleBrush().

### EINGABEN

id            Pinsel, der gewirbelt wird

degrees      gewünschte Wirbelstärke in Grad (von 0 bis 360)

## 43.77 TintBrush

### BEZEICHNUNG

TintBrush – färbt einen Pinsel (V1.5)

### ÜBERSICHT

TintBrush(id, color, level)

### BESCHREIBUNG

Dieser Befehl färbt den Pinsel, der durch **id** angegeben wurde mit der **RGB-Farbe**, die durch **color** angegeben wurde. Die Färbungsintensität wird durch **level** angegeben und darf von 0 bis 255 reichen.

Ab Version 2.0 kann bei **level** eine prozentige Angabe in Form einer Zeichenkette angegeben werden, z.B. "50%".

Beachten Sie, dass wenn **id** einen Palettenpinsel angibt, **TintBrush()** nur die Palettenfarben färbt, was diesen Befehl sehr schnell macht, wenn er mit Palettenpinsel verwendet wird.

Weitere Pinseleffekte: ArcDistortBrush(), BarrelDistortBrush(), BlurBrush(), BrushToGray(), BrushToMonochrome(), CharcoalBrush(), ContrastBrush(), EdgeBrush(), EmbossBrush(), FlipBrush() Befehl, GammaBrush(), InvertBrush() Befehl, MixBrush(), ModulateBrush(), OilPaintBrush(), PerspectiveDistortBrush(), PixelateBrush(), PolarDistortBrush(), RotateBrush(), SepiaToneBrush(), ScaleBrush(), SharpenBrush(), SolarizeBrush(), SwirlBrush(), TransformBrush() und WaterRippleBrush().

**EINGABEN**

<code>id</code>	Identifikator des zu färbenden Pinsels
<code>color</code>	eine <b>RGB-Farbe</b> mit der gefärbt werden soll
<code>level</code>	Färbungsstufe (0 bis 255)

**BEISPIEL**

```
TintBrush(1, #RED, 128)
```

Der obige Code gibt dem Pinsel Nummer 1 einen roten Look.

## 43.78 TransformBrush

**BEZEICHNUNG**

TransformBrush – wendet eine affine Transformation auf einen Pinsel an (V4.5)

**ÜBERSICHT**

```
TransformBrush(id, sx, rx, ry, sy[, smooth])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um eine affine Transformation auf einen Pinsel anzuwenden. Sie müssen eine 2x2 Transformationsmatrix diesem Befehl übergeben, die definiert, wie jedes Pixel in dem Pinsel umgewandelt wird. Dieser Befehl ist nützlich, wenn Sie Rotation und Skalierung gleichzeitig anwenden möchten. Natürlich könnte man dies mit Aufrufen der Befehle **ScaleBrush()** und **RotateBrush()** erledigen, aber dies würde zu Qualitätsverlusten führen. Wenn Sie die Transformation stattdessen mit **TransformBrush()** durchführen, wird alles in einem einzigen Durchlauf durchgeführt.

Die 2x2-Transformationsmatrix besteht aus vier Fließkommazahlenfaktoren:

<b>sx:</b>	Gibt den Faktor der Skalierung für die x-Achse an und darf nicht Null sein. Wenn er negativ ist, wird das Bild zusätzlich auf der y-Achse gekippt.
<b>rx:</b>	Gibt den Faktor der Drehung auf der x-Achse an und gegenüber <b>sx</b> und <b>sy</b> kann dies auch 0 sein.
<b>ry:</b>	Gibt den Faktor der Drehung auf der y-Achse wie rx an und kann wie <b>rx</b> auch 0 sein.
<b>sy:</b>	Gibt den Faktor der Skalierung für die y-Achse an und darf wie <b>sx</b> nicht Null sein. Wenn er negativ ist, wird das Bild zusätzlich auf der x-Achse gekippt.

Die Identitätsmatrix ist definiert als

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Wenn Sie diese Matrix übergeben, dann wird keine Transformation durchgeführt werden, da keine Rotation und keine Skalierung festgelegt wurde. Wenn Hollywood diese Matrix auf jeden Pixel in Ihrem Pinsel anwenden würde, wäre das Ergebnis nur eine Kopie des Pinsels. Wenn aber **TransformBrush()** eine Einheitsmatrix erkennt, wird dieser Befehl nichts tun.

Optional können Sie das Argument `smooth` auf `True` setzen, um Antialiasing beim Transformieren zu benutzen, welches zu einem glatteren Aussehen führt, aber dessen Berechnung länger dauert.

Bitte beachten Sie: Sie sollten beim Transformieren immer eine Kopie des Originalpinsel benutzen. Zum Beispiel, wenn Sie Pinsel 1 auf 12x8 skalieren und später wieder auf 640x480 transformieren, erhalten Sie ein völlig unbrauchbares Bild. Deshalb sollten Sie immer vom Originalpinsel eine Kopie erstellen und erst dann transformieren.

Beachten Sie, dass `TransformBrush()` bei Vektorpinseln immer mit dem nicht transformierten Pinsel angewendet wird. Das bedeutet, dass alle vorherigen Transformationen, die mit `TransformBrush()`, `ScaleBrush()` oder `RotateBrush()` auf den Pinsel angewendet wurden, beim Aufruf von `TransformBrush()` rückgängig gemacht werden.

Weitere Pinseleffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `InvertBrush()` Befehl, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `SepiaToneBrush()`, `ScaleBrush()`, `SharpenBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()` und `WaterRippleBrush()`.

## EINGABEN

<code>id</code>	Identifikator des Pinsels, der transformiert wird
<code>sx</code>	Skalierungsfaktor x; darf nie 0 sein
<code>rx</code>	Rotierungsfaktor x
<code>ry</code>	Rotierungsfaktor y
<code>sy</code>	Skalierungsfaktor y; darf nie 0 sein
<code>smooth</code>	optional: Mit <code>True</code> wird Antialiasing bei der affine Transformation verwendet (Standard ist <code>False</code> )

## BEISPIEL

```
angle = Rad(45)      ; konvertiert Grad in Radiant
TransformBrush(1, Cos(angle), Sin(angle), -Sin(angle), Cos(angle))
```

Der obige Code dreht den Pinsel 1 um 45 Grad.

## 43.79 TrimBrush

### BEZEICHNUNG

`TrimBrush` – entfernt transparente Pixel vom Rand des Pinsels (V4.6)

### ÜBERSICHT

```
left, top = TrimBrush(id)
```

### BESCHREIBUNG

Mit diesem Befehl werden alle transparenten Pixel vom äusseren Rand des Pinsels entfernt. Wenn der angegebene Pinsel keine Maske oder einen Alpha-Kanal besitzt, bewirkt dieser Befehl nichts. `TrimBrush()` gibt die Anzahl der gelöschten Spalten von der linken

Seite sowie die Anzahl der gelöschten Zeilen von der Oberseite des Pinsels zurück. Die Trimmwerte von rechts und unten können Sie dann selber berechnen.

Siehe auch `CropBrush()`.

#### EINGABEN

`id`            Identifikator des Pinsels

#### RÜCKGABEWERTE

`left`            Anzahl entfernter Spalten der linken Seite

`top`            Anzahl entfernter Zeilen von oben

#### BEISPIEL

```
CreateBrush(1, 640, 480, #BLACK, {Mask = True, Clear = True})
SelectBrush(1, #SELMODE_COMBO)
SetFillStyle(#FILLCOLOR)
Box(#CENTER, #CENTER, 100, 100, #RED)
EndSelect
TrimBrush(1)
```

Der obige Code wird eine 640x480 Pinsel erstellen, zeichnet ein 100x100 rotes Rechteck hinein und trimmt es dann auf 100x100 da alle Pixel am Rand leer sind.

## 43.80 Vektorpinsel

Wenn Sie ein Vektorbild mit `LoadBrush()` oder `@BRUSH` laden, werden Sie eine spezielle Art von Pinsel erhalten: einen Vektorpinsel. Wenn Sie normale Bilder wie PNG, JPEG etc. laden, werden Sie immer einen Rasterpinsel bekommen. Sie können mit dem Befehl `GetAttribute()` und dem Attribut `#ATTRTYPE` den Typ eines Pinsels herausfinden.

Der Vorteil eines Vektorpinsels ist, Sie können ohne Qualitätsverlust die Größe ändern und/oder den Pinsel transformieren. Zum Beispiel können die Befehle `ScaleBrush()`, `RotateBrush()` und `TransformBrush()` qualitativ hochwertige Bilder erzeugen, wenn sie mit Vektorpinsel verwendet werden. Auch wenn `Ebenen` und `Ebenenskalierung` aktiviert sind, werden Vektorpinsel ohne Qualitätsverlust skaliert. Also, wenn Sie nur Vektorpinsel und TrueType-Text in Ihrem Skript verwenden, können die Objekte auf jede Auflösung skaliert und werden immer noch perfekt aussehen.

Der Nachteil der Vektorpinsel besteht darin, dass sie nicht von allen Bildmanipulationsbefehlen unterstützt werden. Natürlich arbeiten alle wichtigen Befehlen wie `DisplayBrush()`, `DisplayBrushPart()`, `ScaleBrush()`, `RotateBrush()` usw. mit Vektorpinseln, aber bestimmte Befehle wie `TintBrush()`, `GammaBrush()`, `SelectBrush()` usw. können nur mit Rasterpinsel zurechtkommen. Möchten Sie eine dieser Befehle mit einem Vektorpinsel verwenden, müssen Sie zuerst den Vektor- mit dem Befehl `RasterizeBrush()` in einen Rasterpinsel konvertieren. Beachten Sie aber, dass dann beim skalieren und zu transformieren zu Qualitätseinbußen kommt.

## 43.81 WaterRippleBrush

#### BEZEICHNUNG

WaterRippleBrush – wendet den Wasserwelleneffekt auf den Pinsel an (V5.0)

**ÜBERSICHT**

`WaterRippleBrush(id[, wavelength, amplitude, phase, cx, cy])`

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um einen kreisförmigen Wasserwelleneffekt auf den angegebenen Pinsel anzuwenden. Mit den fünf optionalen Argumenten können Sie die Parameter des Wasserwelleneffekts steuern. Mit **wavelength** (Wellenlänge), **amplitude** (Wellenweite) und **phase** steuern Sie das Aussehen der Wellen, während die **cx** und **cy** Argumente verwendet werden, um den Mittelpunkt des Effektes festzulegen. Dieser Punkt muss als Fließkommawert im Bereich von 0.0 (links/oben) bis 1.0 (rechts/unten) angegeben werden. Das Zentrum des Pinsels ist somit an der Position 0.5/0.5.

Weitere Pinseleffekte: `ArcDistortBrush()`, `BarrelDistortBrush()`, `BlurBrush()`, `BrushToGray()`, `BrushToMonochrome()`, `CharcoalBrush()`, `ContrastBrush()`, `EdgeBrush()`, `EmbossBrush()`, `FlipBrush()` Befehl, `GammaBrush()`, `InvertBrush()` Befehl, `MixBrush()`, `ModulateBrush()`, `OilPaintBrush()`, `PerspectiveDistortBrush()`, `PixelateBrush()`, `PolarDistortBrush()`, `RotateBrush()`, `SepiaToneBrush()`, `ScaleBrush()`, `SharpenBrush()`, `SolarizeBrush()`, `SwirlBrush()`, `TintBrush()` und `TransformBrush()`.

**EINGABEN**

<b>id</b>	Identifikator des Pinsels
<b>wavelength</b>	optional: gewünschte Wellenlänge (voreingestellt ist 32)
<b>amplitude</b>	optional: gewünschte Wellenweite (voreingestellt ist 1)
<b>phase</b>	optional: gewünschte Phase (voreingestellt ist 0)
<b>cx</b>	optional: Koordinate x des Effektmittelpunktes (voreingestellt ist 0.5)
<b>cy</b>	optional: Koordinate y des Effektmittelpunktes (voreingestellt ist 0.5)

**43.82 WriteBrushPixel****BEZEICHNUNG**

`WriteBrushPixel` – schreibt ein einzelnes Pixel in den Pinsel (V5.0)

**ÜBERSICHT**

`WriteBrushPixel(id, x, y, color[, trans])`

**BESCHREIBUNG**

Dieser Befehl schreibt die angegebene **RGB-Farbe** bei den Koordinaten **x** und **y** in den in **id** angegebenen Pinsel. Wenn das optionale Argument **trans** angegeben wird und der Pinsel eine Maske oder einen Alphakanal besitzt, dann wird der angegebene Wert in den Transparenzkanal des Pinsels geschrieben. Wenn der Pinsel eine Maske hat, kann **trans** auf 0 oder 1 gesetzt werden. Besitzt der Pinsel hingegen einen Alphakanal, dann muss **trans** im Bereich von 0 bis 255 sein.

Sie können auch Pixel in Pinsel schreiben, indem Sie den Pinsel mit `SelectBrush()` als Ausgabeziel wählen und dann den Befehl `Plot()` aufrufen. `WriteBrushPixel()` wird jedoch in den meisten Fällen schneller sein, weil er es Ihnen ermöglicht, die Farbe und

die Transparenzkanäle zur gleichen Zeit zu ändern. Sie können so auch den Aufwand vermeiden, der durch den Aufruf von `SelectBrush()` und `EndSelect()` erzeugt wird.

Beachten Sie, dass wenn `id` einen Palettenpinsel angibt, `color` keine **RGB-Farbe**, sondern ein Stift-Wert sein muss.

Siehe auch `FloodFill()`, `QuantizeBrush()`, `ReadBrushPixel()` und `ReplaceColors()`.

#### EINGABEN

<code>id</code>	Identifikator des Pinsels
<code>x</code>	Koordinate x
<code>y</code>	Koordinate y
<code>color</code>	<b>RGB-Farbe</b> oder Stift des zu schreibenden Pixels
<code>trans</code>	optional: Wert, der in den Transparenzkanal kopiert wird (siehe oben)

#### BEISPIEL

```
WriteBrushPixel(1, 100, 100, #RED)
```

Schreibt ein rotes Pixel an die Position 100:100 in den Pinsel 1.



## 44 Pluginbibliothek

### 44.1 DisablePlugin

#### BEZEICHNUNG

DisablePlugin – deaktiviert ein Plugin (V6.0)

#### ÜBERSICHT

DisablePlugin(name\$)

#### BESCHREIBUNG

Mit diesem Befehl kann das in **name\$** angegebene Plugin deaktiviert werden. Bitte beachten Sie, dass nicht alle Plugins deaktiviert werden können. Das Deaktivieren von Plugins wird nur für Plugins unterstützt, die Laden und Speichern für zusätzliche Formate bereitstellen. Es wird nicht bei Plugins unterstützt, die komplette Kernkomponente in Hollywood ersetzen, z.B. indem ein benutzerdefiniertes Display-Adaptermodul bereitgestellt wird. Diese Plugins können nicht deaktiviert werden.

Um ein Plugin später wieder zu aktivieren, rufen Sie den Befehl **EnablePlugin()** auf. Siehe [Abschnitt 44.2 \[EnablePlugin\]](#), Seite 985, für Details.

#### EINGABEN

**name\$**      Name des Plugins das deaktiviert werden soll

### 44.2 EnablePlugin

#### BEZEICHNUNG

EnablePlugin – aktiviert ein Plugin (V6.0)

#### ÜBERSICHT

EnablePlugin(name\$)

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um das angegebene Plugin zu aktivieren. Dies ist nur erforderlich, wenn Sie es zuvor mit **DisablePlugin()** deaktiviert haben.

#### EINGABEN

**name\$**      Name des Plugins das aktiviert werden soll

### 44.3 GetPlugins

#### BEZEICHNUNG

GetPlugins – ruft Informationen über die verfügbaren Plugins ab (V5.3)

#### ÜBERSICHT

**t** = GetPlugins()

**BESCHREIBUNG**

Dieser Befehl gibt eine Tabelle zurück, die Informationen über alle verfügbaren Plugins enthält. Sie erhalten eine Untertabelle für jedes Plugin, das geladen wurde. Um herauszufinden, wie viele Plugin-Untertabellen in der Tabelle sind, verwenden Sie den Befehl `ListItems()`. Jede Untertabelle enthält die folgenden Felder:

**Name:** Name des Plugins.

**Version:** Versionsnummer des Plugins.

**Revision:**  
Revisionsnummer des Plugins.

**Capabilities:**  
Bitmask, welche die Hollywood-Fähigkeiten beschreibt, die dieses Plugin erweitert. Dies wird eine Kombination der folgenden Fähigkeiten sein:

```
#PLUGINCAPS_CONVERT
#PLUGINCAPS_LIBRARY
#PLUGINCAPS_IMAGE
#PLUGINCAPS_ANIM
#PLUGINCAPS_SOUND
#PLUGINCAPS_VECTOR
#PLUGINCAPS_VIDEO
#PLUGINCAPS_SAVEIMAGE
#PLUGINCAPS_SAVEANIM
#PLUGINCAPS_SAVESAMPLE
#PLUGINCAPS_REQUIRE
#PLUGINCAPS_DISPLAYADAPTER
#PLUGINCAPS_TIMERADAPTER
#PLUGINCAPS_REQUESTERADAPTER
#PLUGINCAPS_FILEADAPTER
#PLUGINCAPS_DIRADAPTER
#PLUGINCAPS_AUDIOADAPTER
#PLUGINCAPS_EXTENSION
#PLUGINCAPS_NETWORKADAPTER
#PLUGINCAPS_SERIALIZE
#PLUGINCAPS_ICON
#PLUGINCAPS_SAVEICON
#PLUGINCAPS_IPCADAPTER
#PLUGINCAPS_FONT
```

**Author:** Autor des Plugins.

**Description:**  
Beschreibung des Plugins.

**Copyright:**  
Urheberrecht-Zeichenfolge des Plugins.

**URL:** URL der Plugin-Homepage (wo Aktualisierungen zu bekommen sind usw.)

**Date:** Zusammenstellungsdatum des Plugins.

**Settings:**

Vollständig qualifizierter Pfad zum Einstellungs-Tool des Plugins.

**HelpFile:**

Vollständig qualifizierter Pfad zur Hilfedatei des Plugins.

**Path:**

Vollständig qualifizierter Pfad zum Binärcode des Plugins.

**FileTypes:**

Dieses Element enthält eine Untertabelle, die Tabellen enthält, die alle Dateiformate beschreiben, die dieses Plugin zur Verfügung stellt. Dies ist zum Beispiel nützlich, um Ihre Dateivorlagen an zusätzliche Erweiterungen anzupassen, die von Plugins unterstützt werden. Für jeden neuen Dateityp gibt es eine Tabelle mit folgenden Initialisierungsfeldern:

**Type:** Wird auf den Typ des Dateiformats gesetzt. Dies wird eines der folgenden Konstanten sein:

```
#FILETYPE_IMAGE
#FILETYPE_ANIM
#FILETYPE_SOUND
#FILETYPE_VIDEO
#FILETYPE_ICON
#FILETYPE_FONT
```

**Name:** Wird auf den Namen des Dateiformats als Zeichenkette gesetzt, z.B. "TIFF".

**Extensions:**

Setzt dieses Feld auf eine Zeichenfolge, die alle Erweiterungen von diesem Dateiformat enthalten. Die Erweiterungen werden durch das Zeichen ("|") getrennt und enthalten keinen Punkt, z.B. "tif|tiff".

**MIMEType:**

Setzt dieses Feld auf eine Zeichenfolge, welche die MIME des Dateiformats beschreibt. Dieses Feld kann auch leer sein.

**Flags:**

Setzt dieses Feld auf eine Bitmaskenkombination, die die Fähigkeiten dieses Dateityps beschreibt.

**#FILETYPEFLAGS\_SAVE:**

Wenn dieses Flag gesetzt ist, beschreibt der Eintrag einen Dateityp, den dieses Plugin speichern kann. Der `FormatID`-Tag enthält die Konstante, die verwendet wird, um sich in diesem Fall auf dieses Dateityp-Speichermodul des Plugin zu beziehen.

**#FILETYPEFLAGS\_ALPHA:**

Zeigt an, dass dieser Dateityp geladene oder gespeicherte Alphakanäle unterstützt (abhängig davon, ob `#FILETYPEFLAGS_SAVE` gesetzt ist).

**#FILETYPEFLAGS\_QUALITY:**

Wird nur für #FILETYPE\_IMAGE oder #FILETYPE\_ANIM mit aktiviertem Speichermodus verwendet. In diesem Fall zeigt dieses Flag an, dass das Bild-/Animspeichermodul unterschiedliche Qualitätsstufen (von 0 bis 100) unterstützt.

**#FILETYPEFLAGS\_FPS:**

Wird nur für #FILETYPE\_ANIM mit aktiviertem Speichermodus verwendet. In diesem Fall zeigt dieses Flag an, dass das Animspeichermodul verschiedene Einzelbilder pro Sekunde unterstützt.

**FormatID:**

Wenn in **Flags** #FILETYPEFLAGS\_SAVE gesetzt ist, enthält dieser Tag die Konstantenkennung, die an den jeweiligen Speicherbefehl übergeben werden muss, um dieses Dateitypen-Speichermodul zu verwenden. Zum Beispiel für Dateien vom Typ #FILETYPE\_IMAGE enthält **FormatID** die Kennung, welche an **SaveBrush()** übergeben werden muss.

**ModuleName:**

Enthält den Modulnamen des Plugins. Dies entspricht dem Dateinamen des Plugins abzüglich der Dateierweiterung. Der Modulname des Plugins ist unter allen geladenen Plugins eindeutig. Hollywood wird niemals zwei Plugins mit demselben Modulnamen laden. (V6.0)

**Disabled:**

Zeigt an, ob dieses Plugin aufgrund eines Aufrufs von **DisablePlugin()** zurzeit deaktiviert ist oder nicht. (V6.0)

**EINGABEN**

keine

**RÜCKGABEWERTE**

**t**           Tabelle, welche Informationen über die geladenen Plugins enthält

## 44.4 HavePlugin

**BEZEICHNUNG**

**HavePlugin** – prüft, ob ein Plugin verfügbar ist (V6.0)

**ÜBERSICHT**

**ok, loaded** = **HavePlugin(name\$, [version, revision])**

**BESCHREIBUNG**

Mit diesem Befehl können Sie überprüfen, ob das in **name\$** angegebene Plugin zurzeit verfügbar ist. In diesem Fall liefert **HavePlugin()** im ersten Rückgabewert **ok True** zurück. Der zweite Rückgabewert **loaded** gibt an, ob das Plugin beim Start automatisch geladen wurde oder nicht. Standardmäßig wird Hollywood beim Start automatisch

alle Plugins laden. Dieses Verhalten kann entweder durch das Präfixieren von Plugin-Dateinamen mit einem Unterstrich ('\_') oder durch die Verwendung des Konsolenargument `-skipplugins` geändert werden. Wenn das Plugin nicht geladen wurde, können Sie es mit dem Befehl `LoadPlugin()` manuell laden.

`HavePlugin()` akzeptiert zwei optionale Argumente, mit denen überprüft werden kann, ob eine bestimmte Version des Plugins verfügbar ist. Beachten Sie, dass dies nur überprüft werden kann, wenn das Plugin bereits geladen wurde. Wenn es noch nicht geladen ist, wird `HavePlugin()` nicht in der Lage sein diese Version zu überprüfen.

#### EINGABEN

<code>name\$</code>	Plugin, welches überprüft werden soll
<code>version</code>	optional: überprüft zusätzlich die Versionsnummer (standardmäßig 0, das heißt, jede Version ist akzeptabel)
<code>revision</code>	optional: überprüft zusätzlich auch die Revisionsnummer (standardmäßig 0, das heißt, jede Revision ist akzeptabel)

#### RÜCKGABEWERTE

<code>ok</code>	<code>True</code> wenn das Plugin in der angegebenen Version verfügbar ist, andernfalls <code>False</code>
<code>loaded</code>	<code>True</code> wenn das Plugin bereits geladen wurde, andernfalls <code>False</code>

## 44.5 LoadPlugin

#### BEZEICHNUNG

`LoadPlugin` – lädt ein Plugin zur Laufzeit (V6.0)

#### ÜBERSICHT

`LoadPlugin(name$[, table])`

#### BESCHREIBUNG

Dieser Befehl kann während der Laufzeit zum Laden und Initialisieren des in `name$` angegebenen Plugin verwendet werden. `LoadPlugin()` macht im Grunde das gleiche wie `@REQUIRE`, kann aber aufgerufen werden, während Ihr Skript bereits ausgeführt wird. `@REQUIRE` hingegen wird durch den Präprozessor ausgeführt. Da `LoadPlugin()` ein Laufzeitbefehl ist, können bestimmte Plugins nicht durch diesen Befehl geladen werden, die eine Initialisierung mit niedriger Ebene erfordern. Somit ist es nicht möglich während der Laufzeit ein Display-Adaptermodul als Plugin zu installieren, da Hollywood nach den Präprozessor-Anweisungen bereits seinen internen Displaytreiber eingerichtet hat. Plugins, die nur das Laden oder Speichern von zusätzlichen Dateiformaten ermöglichen, können während der Laufzeit installiert werden.

`LoadPlugin()` akzeptiert das optionale Tabellenargument `table`, das folgende Tags enthalten kann:

**Version:** Minimale Plugin-Version erforderlich. Hollywood scheitert, wenn das installierte Plugin nicht mindestens diese Versionsnummer hat. Die Voreinstellung ist 0, was bedeutet, dass jede Version funktioniert.

**Revision:**

Minimale Plugin-Revision erforderlich. Hollywood scheitert, wenn das installierte Plugin nicht mindestens diese Revisionsnummer hat. Die Voreinstellung ist 0, was bedeutet, dass jede Revision funktioniert.

**SkipRequire:**

Setzen Sie diesen Tag auf **True**, wenn Sie möchten, dass Hollywood den Aufruf des Plugins im Initialisierungscode überspringen soll. Dies ist nur für einige erweiterte Debugging-Zwecke nützlich und sollte normalerweise nicht ausgeführt werden. Standardwert ist **False**.

Darüber hinaus kann das optionale Tabellenargument eine unbegrenzte Anzahl zusätzlicher Tags enthalten, die direkt an die Initialisierungsroutine des Plugins auf dieselbe Weise weitergegeben wird, wie durch die Präprozessor-Anweisung **@REQUIRE**. Die zusätzlichen Argumente hängen vom jeweiligen Plugin ab. Bitte benutzen Sie die Dokumentation des Plugins, um herauszufinden, ob es zusätzliche Parameter akzeptiert. Siehe [Abschnitt 44.6 \[REQUIRE\]](#), [Seite 990](#), für Details.

**EINGABEN**

**name\$**      Name des Plugin, welches geladen werden soll

**table**      optional: Tabelle mit weiteren Parametern (siehe oben)

## 44.6 REQUIRE

**BEZEICHNUNG**

REQUIRE – installiert ein Plugin vor (V5.0)

**ÜBERSICHT**

**@REQUIRE** plugin\$[, table]

**BESCHREIBUNG**

Mit dieser Präprozessor-Anweisung wird Hollywood sicherstellen, dass das angegebene Plugin vor dem Ausführen des Skripts installiert wird. Optional können Sie zusätzliche Parameter an das Plugin übergeben, womit Sie bestimmen können, wie das Plugin initialisiert wird.

Bitte beachten Sie, obwohl Hollywood alle Plugins beim Start automatisch geladen werden, viele Plugins den Aufruf von **@REQUIRE** benötigen, bevor sie verwendet werden können. Diese Plugins benötigen einen benutzerdefinierten Initialisierungscode, der nur ausgeführt wird, wenn Sie explizit **@REQUIRE** aufrufen. Zum Beispiel, Plugins die ein Display-Adaptermodul installieren, nicht aktiviert werden, es sei denn, Sie rufen **@REQUIRE** auf. Plugins, die nur für zusätzliche Dateiformate laden oder speichern, werden jedoch automatisch aktiviert, auch wenn Sie **@REQUIRE** nicht für diese aufrufen.

Dieser Präprozessor-Anweisung kann auch zum Laden von Plugins verwendet werden, die vom automatischen Laden beim Start ausgeschlossen wurden. Plugins können vom automatischen Laden ausgeschlossen werden, indem Sie dem Dateiname einen Unterstrich ('\_') voranstellt oder Sie das Konsolenargument **-skipplugins** verwenden. Wenn Sie ein Plugin laden möchten, das vom automatischen Laden übersprungen wurde, rufen Sie

einfach `@REQUIRE` in Ihrem Skript auf und es wird vom Präprozessor geladen. Alternativ können Sie diese Plugins auch mit dem Befehl `LoadPlugin()` laden.

Ab Hollywood 6.0 akzeptiert diese Präprozessor-Anweisung ein optionales Tabellenargument `table`, mit dem Sie zusätzliche Parameter an die Initialisierungsroutine des Plugins übergeben können. Die hier akzeptierten Parameter variieren von Plugin zu Plugin. Bitte benutzen Sie die Dokumentation des Plugins, um herauszufinden, ob es zusätzliche Parameter akzeptiert, die an `@REQUIRE` übergeben werden können. Die folgenden beiden Parameter werden für jedes Plugin unterstützt:

**Version:** Minimale Plugin-Version erforderlich. Hollywood scheitert, wenn das installierte Plugin nicht mindestens diese Versionsnummer hat. Die Voreinstellung ist 0, was bedeutet, dass jede Version funktioniert (V6.0)

**Revision:** Minimale Plugin-Revision erforderlich. Hollywood scheitert, wenn das installierte Plugin nicht mindestens diese Revisionsnummer hat. Die Voreinstellung ist 0, was bedeutet, dass jede Revision funktioniert. (V6.0)

**Link:** Wenn dieses Feld auf `True` gesetzt ist, wird das angegebene Plugin bei der Kompilierung Ihres Skripts in Ihre ausführbare Datei eingebunden. Dies funktioniert nur, wenn Sie die Plugin-Linker-Infrastruktur korrekt eingerichtet haben. Siehe [Abschnitt 4.5 \[Einbinden von Plugins\]](#), [Seite 62](#), für Details. Achten Sie darauf, die Lizenzen aller Plugins sorgfältig zu lesen, die Sie in Ihre ausführbaren Datei einbinden, weil Lizenzen wie LGPL Ihr Projekt beeinflussen, wenn Sie sich statisch mit LGPL-Software verknüpfen. Dieses Feld ist standardmäßig auf `False` gesetzt, was bedeutet, dass das Plugin nicht eingebunden wird. (V7.0)

Bitte beachten Sie, dass Sie keinen absoluten Pfad im `plugin$` angeben müssen. Geben Sie einfach den Namen des Plugins an.

Siehe [Abschnitt 5.1 \[Plugins\]](#), [Seite 67](#), für mehr Informationen über Plugins.

## EINGABEN

<code>plugin\$</code>	Name des benötigten Plugins
<code>table</code>	optional: Tabelle, die weitere Optionen enthält, die an das Plugin übergeben werden sollen (V6.0)

## BEISPIEL

```
@REQUIRE "xml"
```

Dieser Code installiert das Plugin "xml.hwp". Jede Version wird akzeptiert.

```
@REQUIRE "myplugin", {Version = 2, Revision = 1, User = "John"}
```

Der obige Code überprüft das Plugin auf Version 2.1 von "myplugin.hwp" und übergibt auch das zusätzliche Argument (User="John") an den Initialisierungscode des Plugins.





## 45 Serialisierungsbibliothek

### 45.1 DeserializeTable

#### BEZEICHNUNG

DeserializeTable – deserialisiert eine Zeichenkette in eine Tabelle (V9.0)

#### ÜBERSICHT

```
t = DeserializeTable(s$[, adapter])
```

#### BESCHREIBUNG

Dieser Befehl deserialisiert die durch `s$` angegebene Zeichenkette in eine Tabelle und gibt sie zurück.

Die Tabelle wird mit dem Deserialisierer deserialisiert, der im optionalen Argument `adapter` angegeben wird. Dies kann der Name eines externen Deserialisierungs-Plugins sein (z.B. `xmlparser`) oder einer der folgenden integrierten Deserialisierer:

**Default:** Verwendet Hollywoods Standard-Deserialisierer. Dadurch werden Daten aus dem JSON-Format in eine Hollywood-Tabelle deserialisiert. Dies ist auch die Standardeinstellung, wenn das Argument `adapter` nicht angegeben wird.

**Inbuilt:** Verwendet Hollywoods alten integrierten Deserialisierer. Die Verwendung dieses Deserialisierers wird nicht mehr empfohlen, da die Daten in einem proprietären, nicht für den Menschen lesbaren Format vorliegen. Die Verwendung von JSON ist eine viel bessere Wahl.

Tabellen können mit dem Befehl `SerializeTable()` in Zeichenketten serialisiert werden. Siehe [Abschnitt 45.4 \[SerializeTable\]](#), Seite 995, für Details.

#### EINGABEN

`s$` zu deserialisierende Zeichenkette

`adapter` optional: zu verwendender Deserialisierer (siehe oben)

#### BEISPIEL

Siehe [Abschnitt 45.4 \[SerializeTable\]](#), Seite 995.

### 45.2 GetSerializeMode

#### BEZEICHNUNG

GetSerializeMode – ruft den Serialisierungsmodus ab (V10.0)

#### ÜBERSICHT

```
mode = GetSerializeMode()
```

#### BESCHREIBUNG

Dieser Befehl gibt den aktuellen Serialisierungsmodus zurück, der mit `SetSerializeMode()` eingestellt wurde. Siehe [Abschnitt 45.5 \[SetSerializeMode\]](#), Seite 997, für Details.

#### EINGABEN

keine

**RÜCKGABEWERTE**

`mode`            aktuellen Serialisierungsmodus

**45.3 ReadTable****BEZEICHNUNG**

`ReadTable` – liest eine Tabelle aus einer Datei (V4.0)

**ÜBERSICHT**

```
table = ReadTable(id[, t])
```

**BESCHREIBUNG**

Dieser Befehl liest eine Hollywood-Tabelle aus der durch `id` angegebenen Datei und gibt sie in `table` zurück. Das Lesen beginnt an der aktuellen Position des Dateicursors, die Sie mit dem Befehl `Seek()` ändern können.

Ab Hollywood 9.0 werden die Daten mit dem Deserializer bearbeitet, der im Tag **Adapter** im optionalen Tabellenargument angegeben werden kann. Vor Version 9.0 verwendete `ReadTable()` immer den alten Deserializer von Hollywood, der ein proprietäres, nicht von Menschen lesbares Format verwendet.

Die folgenden Tags werden derzeit im optionalen Tabellenargument erkannt:

**Adapter:** Dieser Tabellen-Tag kann benutzt werden, um den Deserialisierer anzugeben, der zum Importieren der Daten in eine Hollywood-Tabelle verwendet werden soll. Dies kann der Name eines externen Deserialisierungs-Plugins (z.B. `xmlparser`) oder einer der folgenden integrierten Deserializer sein:

**Default:** Verwenden Sie den standardmäßigen Deserializer von Hollywood. Dadurch werden Daten aus dem JSON-Format in eine Hollywood-Tabelle deserialisiert. Beachten Sie, dass der Name dieses Deserializers zwar behauptet, der Standardname zu sein, dies jedoch nicht der Fall ist. Aus Kompatibilitätsgründen verwendet `ReadTable()` standardmäßig den **Inbuilt** Deserializer (siehe unten). Wenn Sie möchten, dass `ReadTable()` den JSON-Deserializer verwendet, müssen Sie ihn explizit anfordern, indem Sie **Adapter** auf **Default** setzen.

**Inbuilt:** Verwendet Hollywoods integrierten alten Deserialisierer. Die einzigen Daten, die dieser Deserialisierer akzeptiert, sind Daten, die von Hollywoods altem integrierten Serialisierer während des `WriteTable()`-Aufrufs geschrieben wurden. Beachten Sie, dass dies aus Kompatibilitätsgründen immer noch der Standard-Deserializer ist. Es wird jedoch nicht mehr empfohlen, da die Daten in einem proprietären, nicht von Menschen lesbaren Format vorliegen. Die Verwendung von JSON ist eine viel bessere Wahl.

Wenn der Tag **Adapter** nicht angegeben ist, wird er standardmäßig auf den mit dem Befehl `SetDefaultAdapter()` gesetzt. Beachten Sie, dass diese Vor-

einstellung aus Kompatibilitätsgründen nicht `Default`, sondern `Inbuilt` ist. Siehe oben für eine Erklärung.

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Serializer-Plugins übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

**Mode:**

Dieser Tag kann verwendet werden, um den für die Operation zu verwendenden Serialisierungsmodus festzulegen. Er verwendet standardmäßig den Serialisierungsmodus, der mit `SetSerializeMode()` festgelegt wurde. Siehe [Abschnitt 45.5 \[SetSerializeMode\]](#), [Seite 997](#), für Details. (V10.0)

**Options:**

Dieser Tag kann verwendet werden, um die für den Vorgang zu verwendenden Serialisierungsoptionen festzulegen. Standardmäßig werden die Serialisierungsoptionen verwendet, die mit `SetSerializeOptions()` festgelegt wurden. Siehe [Abschnitt 45.6 \[SetSerializeOptions\]](#), [Seite 1000](#), für Details. (V10.0)

**SrcEncoding:**

Dieser Tag kann verwendet werden, um die Zeichencodierung der Quelle anzugeben. Dies ist standardmäßig die Standard-Zeichencodierung der Zeichenkettenbibliothek, wie sie von `SetDefaultEncoding()` festgelegt wird. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details. (V10.0)

**DstEncoding:**

Dieser Tag kann verwendet werden, um die Zielzeichencodierung anzugeben. Dies ist standardmäßig die Standard-Zeichencodierung der Zeichenkettenbibliothek, wie sie von `SetDefaultEncoding()` festgelegt wird. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details. (V10.0)

## EINGABEN

`id`            Datei, aus der gelesen werden soll

`t`            optional: Tabelle mit weiteren Optionen (V9.0)

## RÜCKGABEWERTE

`table`        die aus der Datei gelesene Tabelle

## BEISPIEL

```
see @link{WriteTable, WriteTable()}
```

## 45.4 SerializeTable

### BEZEICHNUNG

`SerializeTable` – serialisiert die Tabelle in Zeichenketten (V9.0)

### ÜBERSICHT

```
s$ = SerializeTable(table[, t])
```

**FRÜHERE SYNTAX**

```
s$ = SerializeTable(t[, adapter])
```

**BESCHREIBUNG**

Dieser Befehl serialisiert die durch **table** angegebene Tabelle in eine Zeichenkette und gibt sie zurück.

Das optionale Tabellenargument **t** kann verwendet werden, um zusätzliche Optionen anzugeben. Die folgenden Tags werden derzeit im optionalen Tabellenargument erkannt:

**Adapter:** Die Tabelle wird mit dem Serializer serialisiert, der im Tag **Adapter** angegeben ist. Dies kann der Name eines externen Serializer-Plugins (z.B. **xml**) oder einer der folgenden integrierten Serializer sein:

**Default:** Verwendet den Standard-Serializer von Hollywood. Dadurch wird die Tabelle in das JSON-Format serialisiert. Dies ist auch die Voreinstellung, wenn keine andere Voreinstellung mit **SetDefaultAdapter()** gesetzt wurde.

**Inbuilt:** Verwendet den Legacy-Serializer von Hollywood. Dadurch wird die Tabelle in ein benutzerdefiniertes, proprietäres Format serialisiert.

Wenn der Tag **Adapter** nicht angegeben ist, wird er standardmäßig auf den mit dem Befehl **SetDefaultAdapter()** gesetzt.

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Serializer-Plugins übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

**Mode:** Dieser Tag kann verwendet werden, um den für die Operation zu verwendenden Serialisierungsmodus festzulegen. Er verwendet standardmäßig den Serialisierungsmodus, der mit **SetSerializeMode()** festgelegt wurde. Siehe [Abschnitt 45.5 \[SetSerializeMode\]](#), [Seite 997](#), für Details. (V10.0)

**Options:** Dieser Tag kann verwendet werden, um die für den Vorgang zu verwendenden Serialisierungsoptionen festzulegen. Standardmäßig werden die Serialisierungsoptionen verwendet, die mit **SetSerializeOptions()** festgelegt wurden. Siehe [Abschnitt 45.6 \[SetSerializeOptions\]](#), [Seite 1000](#), für Details. (V10.0)

**SrcEncoding:**

Dieser Tag kann verwendet werden, um die Zeichencodierung der Quelle anzugeben. Dies ist standardmäßig die Standard-Zeichencodierung der Zeichenkettenbibliothek, wie sie von **SetDefaultEncoding()** festgelegt wird. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details. (V10.0)

**DstEncoding:**

Dieser Tag kann verwendet werden, um die Zielzeichencodierung anzugeben. Dies ist standardmäßig die Standard-Zeichencodierung der Zeichenkettenbibliothek, wie sie von **SetDefaultEncoding()** festgelegt wird. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details. (V10.0)

Die Zeichenkette kann später mithilfe des Befehls `DeserializeTable()` wieder in eine Tabelle deserialisiert werden. Siehe [Abschnitt 45.1 \[DeserializeTable\]](#), [Seite 993](#), für Details.

### EINGABEN

`table`      zu serialisierende Tabelle

`t`            optional: Tabelle mit weiteren Optionen (siehe oben)

### BEISPIEL

```
mytable = {1, 2, 3, 4, 5,
  "Hello World",
  x = 100, y = 150,
  subtable = {10, 9, 8, 7}
}
```

```
s$ = SerializeTable(mytable)
mytable2 = DeserializeTable(s$)
```

Der obige Code serialisiert `mytable` in eine Zeichenkette im JSON-Format und deserialisiert diese Zeichenkette dann zurück in eine Tabelle. Am Ende sind `mytable` und `mytable2` zwei unabhängige Tabellen, aber mit identischem Inhalt.

## 45.5 SetSerializeMode

### BEZEICHNUNG

`SetSerializeMode` – stellt den Serialisierungsmodus ein (V10.0)

### ÜBERSICHT

`SetSerializeMode(mode)`

### BESCHREIBUNG

Dieser Befehl setzt den Serialisierungsmodus auf den durch `mode` angegebenen. Derzeit sind die folgenden Serialisierungsmodi verfügbar: `#SERIALIZEMODE_HOLLYWOOD`, `#SERIALIZEMODE_LIST` und `#SERIALIZEMODE_NAMED`. Wie die einzelnen Serialisierungsmodi interpretiert werden, hängt vom Serialisierer ab.

Der ältere Serialisierer von Hollywood, der in ein proprietäres Format serialisiert, unterstützt nur `#SERIALIZEMODE_HOLLYWOOD`. Der JSON-Serialisierer von Hollywood unterstützt alle drei Serialisierungsmodi, die er folgendermaßen interpretiert:

**#SERIALIZEMODE\_HOLLYWOOD:**

Dies ist der standardmäßige Serialisierungsmodus. Alle JSON-Elemente werden in Tabellenfelder serialisiert. Darüber hinaus kann der Serialisierer von Hollywood auch binäre Daten serialisieren und sogar Hollywood-Funktionen vervollständigen. Hollywood-Funktionen und Binärdaten werden als Base64-Daten serialisiert. Der Hollywood-Serialisierer unterstützt auch Sparse-Arrays, also Tabellen, deren Indizes nicht streng sequentiell sind, sondern Lücken zwischen den einzelnen Indizes aufweisen. Ein Nachteil des Hollywood-Serialisierers besteht darin, dass er manchmal einige spezielle Markierungen in der JSON-Datei verwendet, um Hollywood

über die in einem JSON-Element gespeicherten Daten zu informieren, z.B. ob die Daten als Zeichenkette, als Binärdaten oder als Hollywood-Funktion interpretiert werden sollen. Folglich können Sie möglicherweise keine beliebige JSON-Datei mit dem Hollywood-Serialisierungsprogramm deserialisieren, da einige Dinge im JSON möglicherweise falsch als einer der speziellen Marker von Hollywood interpretiert werden. Solange Sie nur vom Hollywood-Serialisierer geschriebene Daten deserialisieren, werden Sie natürlich nie auf Probleme stoßen. Betrachten Sie beispielsweise die folgende Tabelle:

```
t = {foo = "bar", seqarray = {1,2,3,4,5}, sparsearray =
    {1, [2]=2, [4]=3, [6]=4, [8]=5}}
```

Wenn Sie dies mit dem Hollywood-Serialisierungsprogramm in JSON serialisieren, sieht das Ergebnis so aus:

```
{
  "seqarray": [1,2,3,4,5],
  "sparsearray": {"0": 1, "2": 2, "4": 3, "6": 4, "8": 5},
  "foo": "bar"
}
```

Sie können sehen, dass Sparse-Arrays mithilfe von benannten JSON-Indizes serialisiert werden. Das wiederum bedeutet, dass bei der Deserialisierung von JSON-Dateien mit dem Hollywood-Serialisierer benannte Elemente, die nur aus Zahlen bestehen, als Sparse-Array-Felder interpretiert werden, weshalb der Hollywood-Serialisierer nicht zur Deserialisierung beliebiger JSON-Dateien verwendet werden kann, sondern sollte nur mit JSONs verwendet werden, die vom Hollywood-Serialisierungsprogramm erstellt wurden.

Ein weiterer Nachteil des Hollywood-Serialisierers besteht darin, dass die Position von Elementen in der JSON-Datei völlig zufällig sein kann, da sie aus Hollywood-Tabellenfeldern serialisiert werden, die keine bestimmte Reihenfolge haben. Wenn Sie möchten, dass die JSON-Elemente eine feste Reihenfolge beibehalten, müssen Sie stattdessen `#SERIALIZEMODE_LIST` verwenden.

Standardmäßig konvertiert `#SERIALIZEMODE_HOLLYWOOD` alle JSON-Schlüsselnamen in Kleinbuchstaben. Wenn Sie das nicht möchten, können Sie dies ändern, indem Sie in `SetSerializeOptions()` den Tag `NoLowerCase` auf `True` setzen. Siehe [Abschnitt 45.6 \[SetSerializeOptions\]](#), [Seite 1000](#), für Details.

#### `#SERIALIZEMODE_NAMED:`

Dieser ist wie `#SERIALIZEMODE_HOLLYWOOD`, außer dass er keine Hollywood-Erweiterungen unterstützt. Das bedeutet, dass Sie nur Zahlen, Zeichenketten und Tabellen serialisieren können, aber keine Binärdaten oder Hollywood-Funktionen. Außerdem legt dieser Serialisierer Tabellen einige Einschränkungen auf, nämlich dass sie entweder Zeichenkettenindizes oder numerische Indizes verwenden müssen, aber nicht beides. Wenn numerische Indizes verwendet werden, müssen diese Indizes auch strikt sequentiell sein, d.h. Tabellenindizes müssen innerhalb eines bestimmten Bereichs `[0..n]` sequentiell sein. Es dürfen keine Lücken wie im obigen Beispiel vorhanden

sein, daher ist es nicht möglich, Sparse-Arrays mit dem genannten Serialisierer zu serialisieren. Der Vorteil von `#SERIALIZEMODE_NAMED` gegenüber `#SERIALIZEMODE_HOLLYWOOD` besteht darin, dass Sie damit jede beliebige JSON-Datei ohne Probleme deserialisieren können, da der benannte Serialisierer keine Hollywood-Erweiterungen unterstützt. Dies liegt daran, dass die benannten Serialisierer keine Markierungen verwenden, sodass keine Gefahr von Konflikten zwischen JSON-Daten und Hollywood-Markierungen besteht. Genau wie bei `#SERIALIZEMODE_HOLLYWOOD` kann die Position von Elementen in der JSON-Datei jedoch völlig zufällig sein, da sie aus Hollywood-Tabellenfeldern serialisiert werden, die keine bestimmte Reihenfolge haben. Wenn Sie möchten, dass die JSON-Elemente eine feste Reihenfolge beibehalten, müssen Sie stattdessen `#SERIALIZEMODE_LIST` verwenden.

Standardmäßig konvertiert `#SERIALIZEMODE_NAMED` alle JSON-Schlüsselnamen in Kleinbuchstaben. Wenn Sie das nicht möchten, können Sie dies ändern, indem Sie in `SetSerializeOptions()` den Tag `NoLowerCase` auf `True` setzen. Siehe [Abschnitt 45.6 \[SetSerializeOptions\]](#), [Seite 1000](#), für Details.

#### `#SERIALIZEMODE_LIST`:

Dieser Modus serialisiert JSON-Elemente mithilfe von Listen anstelle von benannten Tabellenfeldern. Das hat den Vorteil, dass die Reihenfolge aller JSON-Elemente erhalten bleibt. Außerdem bleibt die Schreibweise der einzelnen JSON-Schlüssel erhalten und Sie könnten sogar denselben Schlüssel mehrmals verwenden. Ein Nachteil ist, dass der Zugriff auf die JSON-Daten etwas schwieriger ist, da sie in Schlüssel-Wert-Paar-Tabellen gespeichert sind. Betrachten Sie beispielsweise die folgenden JSON-Daten:

```
{
  "firstName": "John",
  "lastName": "Smith",
  "isAlive": true,
  "age": 27,
  "address": {
    "streetAddress": "21 2nd Street",
    "city": "New York",
    "state": "NY",
    "postalCode": "10021-3100"
  }
}
```

Der Listenserialisierer speichert jeden Schlüsselwert in seiner eigenen Tabelle unter Verwendung von `key` und `value` als benannte Tabellenindizes. Um also auf Daten aus dem obigen JSON zuzugreifen, müssten Sie diesen Code verwenden:

```
Print(t[3].key, t[3].value) ; "age 27"
Print(t[4].value[2].key, t[4].value[2].value) ; "state NY"
```

Mit dem Namen- oder Hollywood-Serialisierer können Sie auf die JSON-Daten zugreifen, indem Sie einfach den Schlüsselnamen verwenden:

```
Print(t.age)                ; prints "27"
Print(t.address.state)      ; prints "NY"
```

Wie Sie sehen können, führt die Verwendung des Namen- oder Hollywood-Serialisierers zu besser lesbarem Code, aber der Nachteil ist, dass die Elementreihenfolge im JSON nicht beibehalten wird, sodass das JSON möglicherweise schwieriger zu lesen ist.

Beachten Sie, dass bei Verwendung eines externen Serialisierers (z.B. eines Plugins) die Interpretation der verschiedenen Serialisierungsmodi völlig unterschiedlich sein kann. Die obige Dokumentation gilt nur für den integrierten JSON-Serialisierer von Hollywood.

Beachten Sie auch, dass dieser Befehl den Serialisierungsmodus global ändert. Sie können den Serialisierungsmodus auch lokal ändern, indem Sie den Tag **Mode** in den optionalen Tabellenargumenten von Befehlen wie `SerializeTable()` setzen. Siehe [Abschnitt 45.4 \[SerializeTable\]](#), [Seite 995](#), für Details.

#### EINGABEN

**mode**            gewünschter Serialisierungsmodus

## 45.6 SetSerializeOptions

#### BEZEICHNUNG

SetSerializeOptions – legt die Serialisierungsoptionen fest (V10.0)

#### ÜBERSICHT

SetSerializeOptions(**t**)

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einige Optionen für den Serialisierer festzulegen. Sie müssen eine Tabelle in **t** festlegen. Die Tabelle kann die folgenden Tags enthalten:

##### NoLowerCase:

Setzen Sie dies auf **True**, wenn der Serialisierer nicht automatisch alle Schlüsselnamen in Kleinbuchstaben umwandeln soll. Der integrierte JSON-Serialisierer von Hollywood konvertiert derzeit alle Schlüsselnamen in Kleinbuchstaben, wenn die Serialisierungsmodi **#SERIALIZEMODE\_HOLLYWOOD** und **#SERIALIZEMODE\_NAMED** verwendet werden. Externe Serialisierer wie Plugins interpretieren diese Option möglicherweise anders. Voreingestellt ist **False**.

Beachten Sie, dass dieser Befehl die Serialisierungsoptionen global ändert. Sie können die Serialisierungsoptionen auch lokal ändern, indem Sie den Tag **Mode** in den optionalen Tabellenargumenten von Befehlen wie `SerializeTable()` setzen. Siehe [Abschnitt 45.4 \[SerializeTable\]](#), [Seite 995](#), für Details.

#### EINGABEN

**t**            Tabelle mit Serialisierungsoptionen (siehe oben)



## 45.7 WriteTable

### BEZEICHNUNG

WriteTable – schreibt die Tabelle in eine Datei (V4.0)

### ÜBERSICHT

```
WriteTable(id, table[, t])
```

### FRÜHERE SYNTAX

```
WriteTable(id, table[, txtmode, nobrk])
```

### BESCHREIBUNG

Dieser Befehl schreibt die durch **table** angegebene Hollywood-Tabelle in die durch **id** angegebene Datei. Die Tabelle wird mit dem Serialisierer serialisiert, der in den optionalen Argumenten angegeben werden kann. Es wird an der aktuellen Cursorposition in die Datei geschrieben, die Sie mit dem Befehl **Seek()** ändern können. In Dateien geschriebene Tabellen können später mit dem Befehl **ReadTable()** wieder in Hollywood-Tabellen geladen werden.

Dieser Befehl ist vollständig rekursiv. Ihre Tabelle kann beliebig viele Untertabellen enthalten. Zusätzlich kann die Tabelle sogar Hollywood-Funktionen enthalten. Siehe unten für ein Beispiel.

**WriteTable()** unterstützt mehrere optionale Argumente. Vor Hollywood 9.0 mussten diese als optionale Parameter übergeben werden (siehe oben). Seit Hollywood 9.0 wird jedoch empfohlen, die neue Syntax zu verwenden, die ein einzelnes optionales Tabellenargument hat, das verwendet werden kann, um ein oder mehrere optionale Argumente an **WriteTable()** zu übergeben.

Die folgenden Tabellenfelder werden von diesem Befehl erkannt:

**Adapter:** Dieser Tabellen-Tag kann verwendet werden, um den Serialisierer anzugeben, der zum Exportieren der Hollywood-Tabelle verwendet werden soll. Dies kann der Name eines externen Serialisierer-Plugins (z.B. **xml**) oder einer der folgenden integrierten Serialisierer sein:

**Default:** Verwendet den Standard-Serialisierer von Hollywood. Dadurch werden die Tabellendaten in das JSON-Format serialisiert. Beachten Sie, dass der Name dieses Serialisierers zwar behauptet, der Standardname zu sein, dies jedoch nicht der Fall ist. Aus Kompatibilitätsgründen verwendet **WriteTable()** standardmäßig den **Inbuilt** Serialisierer (siehe unten). Wenn Sie möchten, dass **WriteTable()** den JSON-Serialisierer verwendet, müssen Sie ihn explizit anfordern, indem Sie **Adapter** auf **Default** setzen.

**Inbuilt:** Verwendet den (alten) Legacy-Serialisierer von Hollywood. Dadurch wird die Tabelle in ein benutzerdefiniertes, proprietäres Format serialisiert. Dies ist das Format, das **WriteTable()** seit Hollywood 4.0 verwendet und aus Kompatibilitätsgründen immer noch der Standard-Serialisierer ist. Er wird jedoch nicht mehr empfohlen, da dieser Serialisierer Daten ausgibt, die nicht in einem für Menschen lesbaren Format vorliegen. Die Verwendung des JSON-Serialisierers ist eine viel bessere Wahl.

Wenn **Adapter** nicht angegeben ist, wird standardmäßig der Adapter verwendet, welcher mit dem Befehl **SetDefaultAdapter()** gesetzt wurde. Beachten Sie, dass dieser Standardadapter aus Kompatibilitätsgründen nicht **Default**, sondern **Inbuilt** ist. Siehe oben für eine Erklärung.

**TextMode:**

Bei der Verwendung von Hollywoods älterem Serialisierer, der immer noch der Standard ist, kann dieses Argument auf **True** gesetzt werden, um **WriteTable()** anzuweisen, binäre Daten als Text zu exportieren. Beachten Sie, dass der Text auch dann nicht in einem für Menschen lesbaren Format vorliegt, wenn Sie diesen Tag auf **True** setzen. Wenn Sie die Tabelle in menschenlesbaren Text serialisieren möchten, verwenden Sie den JSON-Serialisierer (siehe oben). Voreingestellt ist **False**.

**NoLineBreak:**

Wenn der Tag **TextMode** auf **True** gesetzt wurde, fügt **WriteTable()** zur besseren Lesbarkeit automatisch nach jeweils 72 Zeichen Zeilenumbrüche ein. Wenn Sie das nicht möchten, setzen Sie **NoLineBreak** auf **True**. In diesem Fall werden keine Zeilenumbrüche eingefügt. Beachten Sie, dass dieser Tag nur den (alten) Legacy-Serialisierer von Hollywood betrifft. Es hat keine Auswirkungen auf andere Serialisierer. Voreingestellt ist **False**. (V6.1)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Serialisierer-Plugins übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

**Mode:**

Dieser Tag kann verwendet werden, um den für die Operation zu verwendenden Serialisierungsmodus festzulegen. Er verwendet standardmäßig den Serialisierungsmodus, der mit **SetSerializeMode()** festgelegt wurde. Siehe [Abschnitt 45.5 \[SetSerializeMode\]](#), [Seite 997](#), für Details. (V10.0)

**Options:**

Dieser Tag kann verwendet werden, um die für den Vorgang zu verwendenden Serialisierungsoptionen festzulegen. Standardmäßig werden die Serialisierungsoptionen verwendet, die mit **SetSerializeOptions()** festgelegt wurden. Siehe [Abschnitt 45.6 \[SetSerializeOptions\]](#), [Seite 1000](#), für Details. (V10.0)

**SrcEncoding:**

Dieser Tag kann verwendet werden, um die Zeichencodierung der Quelle anzugeben. Dies ist standardmäßig die Standard-Zeichencodierung der Zeichenkettenbibliothek, wie sie von **SetDefaultEncoding()** festgelegt wird. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details. (V10.0)

**DstEncoding:**

Dieser Tag kann verwendet werden, um die Zielzeichencodierung anzugeben. Dies ist standardmäßig die Standard-Zeichencodierung der Zeichenkettenbibliothek, wie sie von **SetDefaultEncoding()** festgelegt wird. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details. (V10.0)

**EINGABEN**

<b>id</b>	Datei, in die geschrieben werden soll
<b>table</b>	Tabelle, welche in die Datei geschrieben wird
<b>t</b>	optional: Tabelle mit weiteren Argumenten (siehe oben) (V9.0)

**BEISPIEL**

```
mytable = {1, 2, 3, 4, 5,  
  "Hello World",  
  x = 100, y = 150,  
  subtable = {10, 9, 8, 7},  
  mulfunc = Function(a, b) Return(a*b) EndFunction  
}
```

```
OpenFile(1, "table.json", #MODE_WRITE)  
WriteTable(1, mytable, {Adapter = "default"})  
CloseFile(1)
```

```
OpenFile(1, "table.json", #MODE_READ)  
newtable = ReadTable(1, {Adapter = "default"})  
CloseFile(1)
```

```
Print(newtable[0], newtable[5], newtable.x, newtable.y,  
  newtable.subtable[0], newtable.mulfunc(9, 9))
```

Der obige Code schreibt die Tabelle `mytable` in die Datei `"table.json"`. Danach öffnet er die Datei `"table.json"` erneut und liest die Tabelle wieder in Hollywood ein. Die importierte Tabelle wird in der Variablen `newtable` gespeichert. Schließlich greifen wir auf die neu importierte Tabelle zu und geben einige ihrer Daten auf dem Bildschirm aus. Die Ausgabe des obigen Codes lautet `"1 Hello World 100 150 10 81"`.



## 46 Serielle Schnittstellenbibliothek

### 46.1 ClearSerialQueue

#### BEZEICHNUNG

ClearSerialQueue – löscht den Lesepuffer der seriellen Schnittstelle (V8.0)

#### ÜBERSICHT

ClearSerialQueue(id)

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Lesepuffer der in `id` angegebenen seriellen Schnittstellenverbindung zu löschen. Diese serielle Schnittstellenverbindung muss zuvor mit `OpenSerialPort()` geöffnet worden sein.

Verwenden Sie den Befehl `PollSerialQueue()`, um die Anzahl der im Lesepuffer befindlichen Bytes abzufragen. Siehe [Abschnitt 46.12 \[PollSerialQueue\]](#), [Seite 1013](#), für Details.

Beachten Sie, dass dieser Befehl derzeit unter Android nicht unterstützt wird.

#### EINGABEN

`id`            Identifikator der seriellen Schnittstelle, deren Lesepuffer gelöscht werden soll

### 46.2 CloseSerialPort

#### BEZEICHNUNG

CloseSerialPort – schließt die serielle Schnittstellenverbindung (V8.0)

#### ÜBERSICHT

CloseSerialPort(id)

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die durch `id` angegebene serielle Schnittstellenverbindung zu schließen. Diese serielle Schnittstellenverbindung muss zuvor mit `OpenSerialPort()` geöffnet worden sein. Sie sollten immer die seriellen Verbindungen schließen, sobald Sie damit fertig sind.

#### EINGABEN

`id`            Identifikator der seriellen Schnittstellenverbindung, die geschlossen werden soll

### 46.3 FlushSerialPort

#### BEZEICHNUNG

FlushSerialPort – leert die serielle Schnittstellenverbindung (V8.0)

#### ÜBERSICHT

FlushSerialPort(id)

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die durch `id` angegebene serielle Schnittstellenverbindung zu leeren. Diese serielle Schnittstellenverbindung muss zuvor mit `OpenSerialPort()` geöffnet worden sein.

**EINGABEN**

`id`            Identifikator der seriellen Schnittstellenverbindung

## 46.4 GetBaudRate

**BEZEICHNUNG**

`GetBaudRate` – gibt die Baudrate für die serielle Schnittstellenverbindung zurück (V8.0)

**ÜBERSICHT**

```
baud = GetBaudRate(id)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die in `id` angegebene Baudrate für die serielle Schnittstellenverbindung abzurufen. Diese serielle Schnittstellenverbindung muss zuvor mit `OpenSerialPort()` geöffnet worden sein.

Der Rückgabewert ist eine der folgenden speziellen Konstanten:

```
#BAUD_300:
    300 Bits pro Sekunde.

#BAUD_600:
    600 Bits pro Sekunde. (V9.0)

#BAUD_1200:
    1200 Bits pro Sekunde. (V9.0)

#BAUD_2400:
    2400 Bits pro Sekunde.

#BAUD_4800:
    4800 Bits pro Sekunde.

#BAUD_9600:
    9600 Bits pro Sekunde.

#BAUD_19200:
    19200 Bits pro Sekunde.

#BAUD_38400:
    38400 Bits pro Sekunde.

#BAUD_57600:
    57600 Bits pro Sekunde.

#BAUD_115200:
    115200 Bits pro Sekunde.

#BAUD_460800:
    460800 Bits pro Sekunde.
```

**EINGABEN**

`id`            Identifikator der seriellen Schnittstellenverbindung

**RÜCKGABEWERTE**

`baud`            aktuelle Baudrate als spezielle Konstante (siehe oben)

## 46.5 GetDataBits

**BEZEICHNUNG**

GetDataBits – gibt die Datenbits für die serielle Schnittstellenverbindung zurück (V8.0)

**ÜBERSICHT**

```
bits = GetDataBits(id)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die Anzahl der Datenbits für die in `id` angegebene serielle Schnittstellenverbindung abzurufen. Diese serielle Schnittstellenverbindung muss zuvor mit `OpenSerialPort()` geöffnet worden sein.

Der Rückgabewert ist eine der folgenden speziellen Konstanten:

`#DATA_5:`    Verwendet 5 Datenbits.

`#DATA_6:`    Verwendet 6 Datenbits.

`#DATA_7:`    Verwendet 7 Datenbits.

`#DATE_8:`    Verwendet 8 Datenbits.

**EINGABEN**

`id`            Identifikator der seriellen Schnittstellenverbindung

**RÜCKGABEWERTE**

`bits`            aktuelle Datenbits als spezielle Konstante (siehe oben)

## 46.6 GetDTR

**BEZEICHNUNG**

GetDTR – gibt den DTR-Pin-Status für die Verbindung der seriellen Schnittstelle zurück (V8.0)

**ÜBERSICHT**

```
state = GetDTR(id)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um den DTR-Pin-Status für die in `id` angegebene Verbindung zur seriellen Schnittstelle abzurufen. Diese serielle Schnittstellenverbindung muss zuvor mit `OpenSerialPort()` geöffnet worden sein.

Der Rückgabewert ist eine der folgenden speziellen Konstanten:

`#DTR_ON:`    DTR-Pin ist gesetzt.

**#DTR\_OFF:**  
DTR-Pin ist gelöscht.

#### EINGABEN

**id**            Identifikator der zu verwendenden seriellen Schnittstellenverbindung

#### RÜCKGABEWERTE

**state**        aktueller Zustand des DTR-Pins als spezielle Konstante (siehe oben)

## 46.7 GetFlowControl

#### BEZEICHNUNG

GetFlowControl – gibt die Datenflusssteuerung/das Protokoll zurück (V8.0)

#### ÜBERSICHT

```
flow = GetFlowControl(id)
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die Datenflusssteuerung (Protokoll) für die in **id** angegebene serielle Schnittstellenverbindung abzurufen. Diese serielle Schnittstellenverbindung muss zuvor mit **OpenSerialPort()** geöffnet worden sein.

Der Rückgabewert ist eine der folgenden speziellen Konstanten:

**#FLOW\_OFF:**  
Keine Datenflusssteuerung.

**#FLOW\_HARDWARE:**  
Verwendet die Hardware-Datenflusssteuerung CTS/RTS.

**#FLOW\_XON\_XOFF:**  
Verwendet die Software-Datenflusssteuerung XON/XOFF-Handshaking.

#### EINGABEN

**id**            Identifikator der zu verwendenden seriellen Schnittstellenverbindung

#### RÜCKGABEWERTE

**flow**        Datenflusssteuerung/Protokoll als spezielle Konstante (siehe oben)

## 46.8 GetParity

#### BEZEICHNUNG

GetParity – gibt den Paritätsmodus für serielle Schnittstellenverbindung zurück (V8.0)

#### ÜBERSICHT

```
parity = GetParity(id)
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Paritätsmodus für die in **id** angegebene Verbindung zur seriellen Schnittstelle abzurufen. Diese serielle Schnittstellenverbindung muss zuvor mit **OpenSerialPort()** geöffnet worden sein.



Der Rückgabewert ist eine der folgenden speziellen Konstanten:

**#PARITY\_NONE:**

Verwendet keine Paritätsbit.

**#PARITY\_EVEN:**

Verwendet 1 Bit gerade Parität.

**#PARITY\_ODD:**

Verwendet die 1 Bit ungerade Parität.

#### EINGABEN

**id**            Identifikator der zu verwendenden seriellen Schnittstellenverbindung

#### RÜCKGABEWERTE

**parity**        Paritätsbit als spezielle Konstante (siehe oben)

## 46.9 GetRTS

#### BEZEICHNUNG

GetRTS – gibt den RTS-Pin-Status für die Verbindung der seriellen Schnittstelle zurück (V8.0)

#### ÜBERSICHT

`state = GetRTS(id)`

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den RTS-Pin-Status für die in `id` angegebene serielle Schnittstellenverbindung abzurufen. Diese serielle Schnittstellenverbindung muss zuvor mit `OpenSerialPort()` geöffnet worden sein.

Der Rückgabewert ist eine der folgenden speziellen Konstanten:

**#RTS\_ON:**    RTS-Pin ist gesetzt.

**#RTS\_OFF:**

RTS-Pin ist gelöscht.

#### EINGABEN

**id**            Identifikator der zu verwendenden seriellen Schnittstellenverbindung

#### RÜCKGABEWERTE

**state**        Zustand des RTS-Pins als spezielle Konstante (siehe oben)

## 46.10 GetStopBits

#### BEZEICHNUNG

GetStopBits – gibt die Stoppbits für die serielle Schnittstellenverbindung zurück (V8.0)

#### ÜBERSICHT

`bits = GetStopBits(id)`

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die Anzahl der Stoppbits für die in `id` angegebene serielle Schnittstellenverbindung zu erhalten. Diese serielle Schnittstellenverbindung muss zuvor mit `OpenSerialPort()` geöffnet worden sein.

Der Rückgabewert ist eine der folgenden speziellen Konstanten:

`#STOP_1`: Verwendet 1 Stoppbit.

`#STOP_2`: Verwendet 2 Stoppbits.

**EINGABEN**

`id` Identifikator der zu verwendenden seriellen Schnittstellenverbindung

**RÜCKGABEWERTE**

`bits` Aktuelle Stoppbits als spezielle Konstante (siehe oben)

## 46.11 OpenSerialPort

**BEZEICHNUNG**

`OpenSerialPort` – öffnet die serielle Schnittstellenverbindung (V8.0)

**ÜBERSICHT**

```
[id] = OpenSerialPort(id, portname$, [table])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um eine Verbindung zu dem in `portname$` angegebene serielle Schnittstelle zu öffnen und der Verbindung den Identifikator `id` zuzuweisen. Wenn Sie `Nil` in `id` übergeben, wählt `OpenSerialPort()` automatisch eine ID aus und gibt diese zurück.

Der Name, den Sie in `portname$` übergeben, hängt von der Plattform ab, auf der Ihr Skript ausgeführt wird. Unter Windows könnte es `COM1` sein, unter Linux und macOS `/dev/ttyS0` oder `/dev/ttyUSB0`, falls Sie einen USB-Adapter verwenden. Unter AmigaOS müssen Sie die Einheit des `serial.device` übergeben, die Sie in `portname$` öffnen möchten. Bei Android wird davon ausgegangen, dass es nur einen Port gibt, sodass `portname$` ignoriert wird.

Ab Hollywood 9.0 kann `portname$` jetzt auf AmigaOS und kompatiblen Systemen auch eine Zeichenkette im Format "`<Gerätename>:<Port>`" sein. Dies ist nützlich, wenn Sie möchten, dass `OpenSerialPort()` ein alternatives seriellles Gerät anstelle des AmigaOS-Standards `serial.device` öffnet. Wenn Sie beispielsweise "`serialpl2303.device:0`" in `portname$` übergeben, wird versucht, `serialpl2303.device` auf Port 0 zu öffnen.

Darüber hinaus können Sie ein optionales Tabellenargument übergeben, mit dem Sie die Parameter für die serielle Schnittstellenverbindung festlegen können. Folgende Felder werden derzeit erkannt:

**BaudRate:**

Die gewünschte Baudrate für die Verbindung. Dies kann eine der folgenden speziellen Konstanten sein:

`#BAUD_300`:

300 Bits pro Sekunde.

**#BAUD\_600:**  
600 Bits pro Sekunde. (V9.0)

**#BAUD\_1200:**  
1200 Bits pro Sekunde. (V9.0)

**#BAUD\_2400:**  
2400 Bits pro Sekunde.

**#BAUD\_4800:**  
4800 Bits pro Sekunde.

**#BAUD\_9600:**  
9600 Bits pro Sekunde. Dies ist die Voreinstellung.

**#BAUD\_19200:**  
19200 Bits pro Sekunde.

**#BAUD\_38400:**  
38400 Bits pro Sekunde.

**#BAUD\_57600:**  
57600 Bits pro Sekunde.

**#BAUD\_115200:**  
115200 Bits pro Sekunde.

**#BAUD\_460800:**  
460800 Bits pro Sekunde.

**DataBits:**

Die gewünschten Datenbits für die Verbindung. Dies kann auf eine der folgenden speziellen Konstanten gesetzt werden:

**#DATA\_5:** Verwendet 5 Datenbits.

**#DATA\_6:** Verwendet 6 Datenbits.

**#DATA\_7:** Verwendet 7 Datenbits.

**#DATE\_8:** Verwendet 8 Datenbits. Dies ist die Voreinstellung.

**StopBits:**

Die gewünschten Stoppbits für die Verbindung. Dies kann auf eine der folgenden speziellen Konstanten gesetzt werden:

**#STOP\_1:** Verwendet 1 Stoppbit. Dies ist die Voreinstellung.

**#STOP\_2:** Verwendet 2 Stoppbits.

**Parity:** Der gewünschte Paritätsmodus. Dies kann auf eine der folgenden speziellen Konstanten gesetzt werden:

**#PARITY\_NONE:**  
Verwendet keine Paritätsbit. Dies ist die Voreinstellung.

**#PARITY\_EVEN:**  
Verwendet 1 Bit gerade Parität.

**#PARITY\_ODD:**

Verwendet die 1 Bit ungerade Parität.

**FlowControl:**

Die gewünschte Art der Datenflusssteuerung (Protokoll), die verwendet werden soll. Dies kann auf eine der folgenden speziellen Konstanten gesetzt werden:

**#FLOW\_OFF:**

Verwendet keine Datenflusssteuerung. Dies ist die Voreinstellung.

**#FLOW\_HARDWARE:**

Verwendet die Hardware-Datenflusssteuerung CTS/RTS.

**#FLOW\_XON\_XOFF:**

Verwendet die Software-Datenflusssteuerung XON/XOFF-Handshaking.

**RTS:** Der gewünschte Zustand des RTS-Pins. Beachten Sie, dass das manuelle Setzen des RTS-Pins nicht auf jeder Plattform unterstützt wird. Sofern unterstützt, kann eine der folgenden speziellen Konstanten festgelegt werden:

**#RTS\_ON:** Setzt das RTS-Pin.

**#RTS\_OFF:**  
Löscht das RTS-Pin.

**DTR:** Der gewünschte Zustand des DTR-Pins. Beachten Sie, dass das manuelle Setzen des DTR-Pins nicht auf jeder Plattform unterstützt wird. Sofern unterstützt, kann eine der folgenden speziellen Konstanten festgelegt werden:

**#DTR\_ON:** Setzt das DTR-Pin.

**#DTR\_OFF:**  
Löscht das DTR-Pin.

Wie Sie oben sehen können, ist die von `OpenSerialPort()` verwendete Standardkonfiguration 9600/8-N-1, d.h. 9600 bps, 8 Datenbits, kein Paritätsbit, 1 Stoppbit. Dies ist die am häufigsten verwendete Konfiguration und sollte auf jeder Plattform funktionieren.

**EINGABEN**

**id** Identifikator für die neue serielle Verbindung oder **Nil** für die automatische ID-Zuweisung

**portname\$**  
die zu öffnende serielle Schnittstelle

**table** optional: weitere Optionen (siehe oben)

**RÜCKGABEWERTE**

**id** optional: Identifikator der seriellen Verbindung; Wird nur zurückgegeben werden, wenn Sie **Nil** als Argument 1 angegeben haben (siehe oben)

**BEISPIEL**

```
OpenSerialPort(1, "COM1")
WriteSerialData(1, "Hello World!")
CloseSerialPort(1)
```

Der obige Code öffnet die serielle Schnittstelle COM1 unter Windows und sendet die Zeichenfolge "Hello World!" an den Empfänger und schließt die serielle Schnittstellenverbindung. Beachten Sie, dass es keine Garantie gibt, dass alle 12 Bytes an die serielle Schnittstelle gesendet werden konnten. In Ihrem Code müssten Sie den Rückgabewert von `WriteSerialData()` überprüfen und ggf. erneut aufrufen, um die restlichen Bytes zu senden.

## 46.12 PollSerialQueue

**BEZEICHNUNG**

PollSerialQueue – gibt die Anzahl der Bytes im Lesebuffer zurück (V8.0)

**ÜBERSICHT**

```
n = PollSerialQueue(id)
```

**BESCHREIBUNG**

Mit diesem Befehl kann die Anzahl der Bytes abgefragt werden, die sich aktuell im Lesebuffer der durch `id` angegebenen seriellen Schnittstellenverbindung befindet. Diese serielle Schnittstellenverbindung muss zuvor mit `OpenSerialPort()` geöffnet worden sein. Verwenden Sie den Befehl `ClearSerialQueue()`, um den Lesebuffer einer seriellen Schnittstellenverbindung zu löschen. Siehe [Abschnitt 46.1 \[ClearSerialQueue\]](#), [Seite 1005](#), für Details.

Beachten Sie, dass dieser Befehl derzeit unter Android nicht unterstützt wird.

**EINGABEN**

<code>id</code>	Identifikator der seriellen Schnittstelle, deren Lesebuffer Sie abfragen möchten
-----------------	--

**RÜCKGABEWERTE**

<code>n</code>	Anzahl Bytes im Lesebuffer
----------------	----------------------------

## 46.13 ReadSerialData

**BEZEICHNUNG**

ReadSerialData – liest Daten von der serieller Schnittstelle (V8.0)

**ÜBERSICHT**

```
data$, count = ReadSerialData(id, len[, timeout])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um Datenbytes von der in `id` angegebenen seriellen Schnittstellenverbindung zu lesen. Die Verbindung der seriellen Schnittstelle muss zuvor mit `OpenSerialPort()` geöffnet worden sein. Darüber hinaus können Sie im Argument

`timeout` eine Dauer in Millisekunden übergeben, um ein Zeitlimit für die Leseoperation festzulegen. Wenn der Parameter `timeout` angegeben ist, wird `ReadSerialData()` niemals länger als die angegebene Dauer blockieren. Andernfalls wird es ewig auf die Daten warten.

`ReadSerialData()` gibt die von der seriellen Schnittstelle gelesenen Daten in `data$` und die Länge der Daten in Byte in `count` zurück. Beachten Sie, dass diese Länge unter der in `len` angegebenen Länge liegen kann. Wenn `ReadSerialData()` weniger Bytes zurückgibt, als Sie in `len` angefordert haben, müssen Sie `ReadSerialData()` immer wieder aufrufen, bis Sie alle benötigten Daten erhalten haben.

Beachten Sie, dass der in `count` zurückgegebene Wert immer der `ByteLen()` für `data$` entspricht. Der einzige Grund für den Rückgabewert von `count` ist eine Leistungssteigerung, da Sie auf diese Weise nicht `ByteLen()` aufrufen müssen, um die Länge von `data$` zu berechnen.

Verwenden Sie den Befehl `PollSerialQueue()`, um die Anzahl der im Lesebuffer befindlichen Bytes abzufragen. Siehe [Abschnitt 46.12 \[PollSerialQueue\]](#), [Seite 1013](#), für Details.

## EINGABEN

<code>id</code>	Identifikator der seriellen Schnittstellenverbindung
<code>len</code>	Anzahl der Bytes, die von der seriellen Schnittstelle gelesen werden sollen
<code>timeout</code>	optional: Anzahl der Millisekunden, nach denen die Operation abgebrochen werden soll (Standardeinstellung ist 0, d.h. es wird solange blockiert, bis Daten ankommen)

## RÜCKGABEWERTE

<code>data\$</code>	die Daten, welche von der seriellen Schnittstelle gelesen werden
<code>count</code>	Anzahl der von der seriellen Schnittstelle gelesenen Bytes

## BEISPIEL

```
OpenSerialPort(1, "COM1")
Print(ReadSerialData(1, 256))
```

Der obige Code wartet immer auf die Daten vom seriellen Port. Sobald Daten ankommen, werden sie ausgegeben. Dies kann weniger als 256 Byte sein. Das einzige, was garantiert ist, ist, dass es niemals mehr als 256 Bytes sein wird.

# 46.14 SetBaudRate

## BEZEICHNUNG

`SetBaudRate` – setzt die Baudrate für die serielle Schnittstellenverbindung (V8.0)

## ÜBERSICHT

```
SetBaudRate(id, baud)
```

## BESCHREIBUNG

Mit diesem Befehl kann die Baudrate für die in `id` angegebene serielle Schnittstellenverbindung eingestellt werden. Diese serielle Schnittstellenverbindung muss zuvor mit

`OpenSerialPort()` geöffnet worden sein. Sie müssen die gewünschte Baudrate im Parameter `baud` übergeben. Dies muss eine der folgenden speziellen Konstanten sein:

`#BAUD_300:`  
300 Bits pro Sekunde.

`#BAUD_600:`  
600 Bits pro Sekunde. (V9.0)

`#BAUD_1200:`  
1200 Bits pro Sekunde. (V9.0)

`#BAUD_2400:`  
2400 Bits pro Sekunde.

`#BAUD_4800:`  
4800 Bits pro Sekunde.

`#BAUD_9600:`  
9600 Bits pro Sekunde.

`#BAUD_19200:`  
19200 Bits pro Sekunde.

`#BAUD_38400:`  
38400 Bits pro Sekunde.

`#BAUD_57600:`  
57600 Bits pro Sekunde.

`#BAUD_115200:`  
115200 Bits pro Sekunde.

`#BAUD_460800:`  
460800 Bits pro Sekunde.

## EINGABEN

<code>id</code>	Identifikator der seriellen Schnittstellenverbindung
<code>baud</code>	gewünschte Baudrate

## 46.15 SetDataBits

### BEZEICHNUNG

`SetDataBits` – setzt die Datenbits für die serielle Schnittstellenverbindung (V8.0)

### ÜBERSICHT

`SetDataBits(id, bits)`

### BESCHREIBUNG

Mit diesem Befehl kann die Anzahl der Datenbits für die in `id` angegebene serielle Schnittstellenverbindung festgelegt werden. Diese serielle Schnittstellenverbindung muss zuvor mit `OpenSerialPort()` geöffnet worden sein. Sie müssen die gewünschten Datenbits im Parameter `bits` übergeben. Dies muss eine der folgenden speziellen Konstanten sein:

`#DATA_5:` Verwendet 5 Datenbits.

#DATA\_6: Verwendet 6 Datenbits.  
 #DATA\_7: Verwendet 7 Datenbits.  
 #DATE\_8: Verwendet 8 Datenbits.

**EINGABEN**

`id`            Identifikator der seriellen Schnittstellenverbindung  
`baud`          gewünschte Datenbits

## 46.16 SetDTR

**BEZEICHNUNG**

SetDTR – setzt den DTR-Pin-Status für die Verbindung der seriellen Schnittstelle (V8.0)

**ÜBERSICHT**

SetDTR(`id`, `state`)

**BESCHREIBUNG**

Mit diesem Befehl kann der DTR-Pin-Status für die in `id` angegebene serielle Schnittstellenverbindung festgelegt werden. Diese serielle Schnittstellenverbindung muss zuvor mit `OpenSerialPort()` geöffnet worden sein. Beachten Sie, dass das manuelle Setzen des DTR-Pins nicht auf jeder Plattform unterstützt wird. Sie müssen den gewünschten Zustand im Parameter `state` übergeben. Dies muss eine der folgenden speziellen Konstanten sein:

#DTR\_ON: Setzt den DTR-Pin.  
 #DTR\_OFF:            Löscht den DTR-Pin.

**EINGABEN**

`id`            Identifikator der seriellen Schnittstellenverbindung  
`baud`          gewünschter DTR-Pin-Status

## 46.17 SetFlowControl

**BEZEICHNUNG**

SetFlowControl – setzt die Datenflusssteuerung/das Protokoll (V8.0)

**ÜBERSICHT**

SetFlowControl(`id`, `flow`)

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die Datenflusssteuerung (Protokoll) für die in `id` angegebene serielle Schnittstellenverbindung festzulegen. Diese serielle Schnittstellenverbindung muss zuvor mit `OpenSerialPort()` geöffnet worden sein. Sie müssen den gewünschten Datenflusssteuerungs-Modus im Parameter `flow` übergeben. Dies muss eine der folgenden speziellen Konstanten sein:

#FLOW\_OFF:            Verwendet keine Datenflusssteuerung.



**#FLOW\_HARDWARE:**

Verwendet die Hardware-Datenflusssteuerung CTS/RTS.

**#FLOW\_XON\_XOFF:**

Verwendet die Software-Datenflusssteuerung XON/XOFF-Handshaking.

## EINGABEN

<code>id</code>	Identifikator der seriellen Schnittstellenverbindung
<code>baud</code>	gewünschter Datenflusssteuerungs-Modus

## 46.18 SetParity

### BEZEICHNUNG

SetParity – setzt den Paritätsmodus für serielle Schnittstellenverbindung (V8.0)

### ÜBERSICHT

SetParity(`id`, `parity`)

### BESCHREIBUNG

Mit diesem Befehl kann der Paritätsmodus für die in `id` angegebene serielle Schnittstellenverbindung festgelegt werden. Diese serielle Schnittstellenverbindung muss zuvor mit `OpenSerialPort()` geöffnet worden sein. Sie müssen den gewünschten Paritätsmodus im Parameter `parity` übergeben. Dies muss eine der folgenden speziellen Konstanten sein:

**#PARITY\_NONE:**

Verwendet keine Paritätsbit.

**#PARITY\_EVEN:**

Verwendet 1 Bit gerade Parität.

**#PARITY\_ODD:**

Verwendet die 1 Bit ungerade Parität.

## EINGABEN

<code>id</code>	Identifikator der seriellen Schnittstellenverbindung
<code>baud</code>	gewünschter Paritätsmodus

## 46.19 SetRTS

### BEZEICHNUNG

SetRTS – setzt den RTS-Pin-Status für die Verbindung der seriellen Schnittstelle (V8.0)

### ÜBERSICHT

SetRTS(`id`, `state`)

### BESCHREIBUNG

Mit diesem Befehl kann der RTS-Pin-Status für die in `id` angegebene serielle Schnittstellenverbindung festgelegt werden. Diese serielle Schnittstellenverbindung muss zuvor mit

`OpenSerialPort()` geöffnet worden sein. Beachten Sie, dass das manuelle Setzen des RTS-Pins nicht auf jeder Plattform unterstützt wird. Sie müssen den gewünschten Zustand im Parameter `state` übergeben. Dies muss eine der folgenden speziellen Konstanten sein:

`#RTS_ON`: Setzt den RTS-Pin.

`#RTS_OFF`:  
Löscht den RTS-Pin.

#### EINGABEN

`id`            Identifikator der seriellen Schnittstellenverbindung  
`baud`        gewünschter RTS-Pin-Status

## 46.20 SetStopBits

### BEZEICHNUNG

`SetStopBits` – setzt die Stoppbits für die serielle Schnittstellenverbindung (V8.0)

### ÜBERSICHT

`SetStopBits(id, bits)`

### BESCHREIBUNG

Mit diesem Befehl kann die Anzahl der Stoppbits für die in `id` angegebene serielle Schnittstellenverbindung festgelegt werden. Diese serielle Schnittstellenverbindung muss zuvor mit `OpenSerialPort()` geöffnet worden sein. Sie müssen die gewünschten Stoppbits im Parameter `bits` übergeben. Dies muss eine der folgenden speziellen Konstanten sein:

`#STOP_1`: Verwendet 1 Stoppbit.

`#STOP_2`: Verwendet 2 Stoppbits.

#### EINGABEN

`id`            Identifikator der seriellen Schnittstellenverbindung  
`baud`        gewünschte Stoppbits

## 46.21 WriteSerialData

### BEZEICHNUNG

`WriteSerialData` – sendet Daten an die serielle Schnittstelle (V8.0)

### ÜBERSICHT

`count = WriteSerialData(id, data$[, timeout])`

### BESCHREIBUNG

Mit diesem Befehl können die in `data$` angegebenen Daten an die in `id` angegebene serielle Schnittstellenverbindung gesendet werden. Die Verbindung der seriellen Schnittstelle muss zuvor mit `OpenSerialPort()` geöffnet worden sein. Darüber hinaus können Sie im Argument `timeout` eine Dauer in Millisekunden übergeben, um ein Zeitlimit

für den Sendevorgang festzulegen. Wenn der Parameter `timeout` angegeben ist, wird `WriteSerialData()` niemals länger als die angegebene Dauer blockieren.

`WriteSerialData()` gibt die Anzahl der an die seriellen Schnittstelle gesendeten Bytes in `count` zurück. Beachten Sie, dass dies weniger als die Bytes in `data$` sein kann. Wenn nur Teile von `data$` an die serielle Schnittstelle gesendet wurden, müssen Sie `WriteSerialData()` erneut aufrufen, um den Rest zu senden.

#### EINGABEN

<code>id</code>	Identifikator der seriellen Schnittstellenverbindung
<code>data\$</code>	die Daten, die an die serielle Schnittstelle gesendet werden sollen
<code>timeout</code>	optional: Anzahl der Millisekunden, nach deren Ablauf die Operation abgebrochen wird (Standardeinstellung ist 0, d.h. es wird solange blockiert, bis Daten gesendet werden können)

#### RÜCKGABEWERTE

<code>count</code>	Anzahl der Bytes, die erfolgreich gesendet wurden
--------------------	---

#### BEISPIEL

Siehe [Abschnitt 46.11 \[OpenSerialPort\]](#), Seite 1010.



## 47 Soundbibliothek

### 47.1 Übersicht

Hollywoods Soundbibliothek bietet zwei grundlegende Objekttypen: Samples und Musikobjekte. Beispiele sind typischerweise kurze Töne wie Spiel- oder Feedback-Effekte, während Musikobjekte für längere Melodien verwendet werden, die im Hintergrund spielen. Der größte Unterschied zwischen Samples und Musikobjekten ist, dass Samples vollständig in den Speicher geladen werden, während Musikobjekte kontinuierlich von der Festplatte geladen werden. So sollten Sie Samples nur für kurze Töne/Klänge verwenden, da das Laden einer 4-minütigen Musik als Sample etwa 40 Megabyte Speicherplatz einnimmt. Die Samples sind für die sofortige Wiedergabe optimiert, weshalb sie oft zum Audio-Hardware-Speicher hochgeladen werden, so dass sie mit sehr geringer Latenz gespielt werden können. Musikobjekte brauchen aber ein bisschen länger, um mit dem Abspielen anzufangen.

Samples können über die Befehle `@SAMPLE` und `LoadSample()` geladen werden. Sie können auch eigene Samples mit dem Befehl `CreateSample()` erstellen. Um ein Sample abzuspielen, verwenden Sie `PlaySample()`.

Musikobjekte können über die Befehle `@MUSIC` und `OpenMusic()` geladen werden. Sie können auch eigene Musikobjekte mit dem Befehl `CreateMusic()` erstellen. Um ein Musikobjekt abzuspielen, verwenden Sie `PlayMusic()`.

Standardmäßig benutzt die Soundbibliothek von Hollywood 8 Audiokanäle für die Tonwiedergabe. Dies bedeutet, dass Hollywood die Kanäle ausgehen, falls Sie versuchen, mehr als 8 verschiedene Samples, Musikobjekte oder Videos gleichzeitig abzuspielen. Wenn Ihr Skript aus bestimmten Gründen mehr als 8 Kanäle benötigt, müssen Sie die Anzahl der Kanäle mit dem Konsolenargument `-numchannels` erhöhen.

### 47.2 CloseAudio

#### BEZEICHNUNG

CloseAudio – schließt die Audio-Hardware (V8.0)

#### ÜBERSICHT

CloseAudio()

#### BESCHREIBUNG

Dieser Befehl kann zum Schließen der Audio-Hardware verwendet werden. Es ist normalerweise nicht erforderlich, diesen Befehl aufzurufen, da Hollywood die Audio-Hardware automatisch schließt, wenn sie nicht mehr benötigt wird. Unter AmigaOS und kompatiblen Systemen gibt es jedoch Situationen, in denen Sie möglicherweise eine fein abgestimmte Steuerung der Audio-Hardware benötigen. Zum Beispiel weil ein anderes Programm versucht, exklusiven Zugriff auf die Audio-Hardware zu erhalten, was bedeutet, dass Ihr Skript sie zuerst freigeben muss. In solchen Situationen möchten Sie `CloseAudio()` möglicherweise manuell aufrufen. Abgesehen von dieser speziellen Situation besteht keine Notwendigkeit, diesen Befehl überhaupt aufzurufen.

Beim Aufrufen von `CloseAudio()` wird nicht nur die gesamte Audioausgabe angehalten, sondern es werden auch alle Samples aus dem Speicher gelöscht, da die Samples beim Laden normalerweise in den Speicher der Soundkarte geladen werden. Daher

ist es unmöglich, sie im Speicher zu halten, während die Audio-Hardware geschlossen ist. Musik- und Video-Objekte werden von `CloseAudio()` jedoch nicht gelöscht; `CloseAudio()` stoppt nur deren Wiedergabe.

#### EINGABEN

keine

#### BEISPIEL

Siehe [Abschnitt 47.29 \[OpenAudio\]](#), Seite 1044.

## 47.3 CloseMusic

#### BEZEICHNUNG

`CloseMusic` – schließt eine Musik (V2.0)

#### ÜBERSICHT

`CloseMusic(id)`

#### BESCHREIBUNG

Dieser Befehl schließt die in `id` angegebene Musik und löscht den zugehörigen Buffer im Arbeitsspeicher. Dies ist normalerweise nicht notwendig, da Hollywood den gesamten benutzten Arbeitsspeicher wieder freigibt, wenn es beendet wird. Wenn jedoch der Arbeitsspeicher knapp wird und Sie wollen die Musik selbst aus dem Speicher entfernen, dann benutzen Sie diesen Befehl.

#### EINGABEN

`id`            Identifikator der Musik, die Sie schließen wollen

## 47.4 CopySample

#### BEZEICHNUNG

`CopySample` – kopiert ein Sample (V5.0)

#### ÜBERSICHT

`[id] = CopySample(source, dest)`

#### BESCHREIBUNG

Dieser Befehl erstellt eine Kopie von dem in `source` angegebenen Sample und weist ihm die ID von `dest` zu. Das neue Sample ist unabhängig vom alten. Daher kann das alte Sample aus dem Arbeitsspeicher gelöscht werden.

Wenn Sie `Nil` in `dest` angegeben haben, wird `CopySample()` eine freie ID auswählen und Ihnen zurückgeben. Andernfalls benutzt das neue Sample die in `dest` angegebene ID.

#### EINGABEN

`source`        ID des Samples, von dem eine Kopie erstellt werden soll

`dest`           ID des neuen Samples oder `Nil` für die [automatische ID-Auswahl](#)

#### RÜCKGABEWERTE

`id`            optional: ID des Samples; wird nur zurückgegeben, wenn Sie ihn beim Argument `dest Nil` angegeben haben.

**BEISPIEL**

```
CopySample(1, 10)
FreeSample(1)
```

Der obige Code erstellt ein neues Sample 10, der die gleichen Audiodaten wie das Sample 1 enthält. Dann wird das Sample 1 aus dem Arbeitsspeicher gelöscht, weil er nicht mehr benötigt wird.

**47.5 CreateMusic****BEZEICHNUNG**

CreateMusic – erstellt einen dynamischen Musik-Stream (V5.0)

**ÜBERSICHT**

```
[id] = CreateMusic(id, pitch, fmt)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um einen dynamischen Musik-Stream zu erstellen, der ständig mit neuen PCM-Daten über eine benutzerdefinierte Callback-Funktion zugeführt werden muss. Auf diese Weise können Sie im laufenden Betrieb durch eine Callback-Funktion lückenlos Audio mit PCM-Daten abspielen. Die Musik wird zu Hollywoods Musikliste hinzugefügt und kann über die angegebene ID zugegriffen werden. Wenn Sie in `id` `Nil` übergeben, wird `CreateMusic()` automatisch eine freie ID auswählen und sie zurückgeben. Sie müssen auch die gewünschte Wiedergabefrequenz für die Musik im Argument `pitch` sowie die Codierung der PCM-Daten im Argument `fmt` angeben. Derzeit werden folgende Formate unterstützt: `#MONO8`, `#STEREO8`, `#MONO16` und `#STEREO16`.

Bevor Sie diesen Befehl aufrufen, müssen Sie eine Callback-Funktion vom Typ `FillMusicBuffer` mit dem Befehl `InstallEventHandler()` installieren. Diese Callback-Funktion wird dann aufgerufen, wenn der Audio-Server neue PCM-Daten benötigt. Um die neuen PCM-Daten an den Audio-Server zu liefern, hat Ihre Callback-Funktion den Befehl `FillMusicBuffer()` aufzurufen. Siehe [Abschnitt 47.7 \[FillMusicBuffer\]](#), [Seite 1026](#), für Details.

Sobald Sie die Musik mit `CreateMusic()` erstellt haben, können Sie dann alle regulären Befehle aus der Musikbibliothek verwenden, um mit der neuen Musik zu arbeiten. Zum Beispiel können Sie mit `PlayMusic()` die Wiedergabe starten und mit `PauseMusic()` das Musikobjekt anhalten.

Stellen Sie sicher, dass Sie immer eine Hauptschleife verwenden, die `WaitEvent()` aufruft, wenn Sie diesen Befehl verwenden, da die Callback-Funktion von `CreateMusic()` immer von `WaitEvent()` aufgerufen wird! Wenn Sie `WaitEvent()` nicht in einer Schleife verwenden, wird Ihre Callback-Funktion nie aufgerufen werden und somit wird nie ein Ton gespielt!

Bitte beachten Sie, dass dies ein Lowlevel-Befehl ist, der ganz in der Nähe der Hardware-Ebene läuft. Daher sollte Ihre Callback-Funktion Ihr Skript niemals für eine längere Zeit blockieren. Es sollte so schnell wie möglich wieder in die Hauptschleife zurückkehren. Sie dürfen niemals Befehle aufrufen, die das Skript in der Callback-Funktion von `CreateMusic()` blockieren könnten. Zum Beispiel ist der Aufruf

von `Wait()` oder `SystemRequest()` in einer Musik-Callback-Funktion eine sehr schlechte Idee.

#### EINGABEN

`id`            Identifikator der neuen Musik `Nil` für die automatische ID-Auswahl  
`pitch`        gewünschte Wiedergabefrequenz für die Musik  
`fmt`           gewünschtes Format für die Musik

#### RÜCKGABEWERTE

`id`            optional: ID der neuen Musik; wird nur zurückgegeben, wenn Sie beim Argument `id Nil` angegeben haben.

## 47.6 CreateSample

#### BEZEICHNUNG

CreateSample – erstellt ein Sample (V2.0)

#### ÜBERSICHT

```
[id] = CreateSample(id, table, pitch[, fmt, length])
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um ein neues Sample aus benutzerdefinierten PCM-Daten zu erstellen. Das Sample wird der Sampleliste von Hollywood hinzugefügt werden und es kann auf ihn mit der angegebenen ID zugegriffen werden. Wenn Sie `Nil` in `id` übergeben, wird `CreateSample()` automatisch eine freie ID auswählen und Ihnen zurückgeben.

Sie müssen auch die gewünschte Wiedergabefrequenz für dieses Sample im Argument `pitch` angeben. Im optionalen Argument `fmt` können Sie das Format der PCM-Daten angeben, die Sie diesem Befehl übergeben. Derzeit werden folgende Formate unterstützt: `#MONO8` (das ist die Standardeinstellung), `#STEREO8`, `#MONO16` und `#STEREO16`. Das optionale Argument `length` gibt die gewünschte Länge in PCM-Daten für das neuen Sample an.

Die Sampledaten müssen als signierte Ganzzahlen übergeben werden. Für 8-Bit-Samples läuft der gültige Samplebereich von -128 bis 127, und für 16-Bit-Samples geht der gültige Samplebereich von -32768 bis 32767.

Wenn Sie ein Stereo-Sample erstellen möchten, müssen Sie verschachtelte PCM-Daten übergeben, das heißt, linker Kanalsample gefolgt vom rechten Kanalsample gefolgt durch den linken Kanalsample und so weiter.

Ab Hollywood 5.0 können die PCM-Daten für diesen Befehl in einer Reihe von verschiedenen Möglichkeiten übergeben werden. `CreateSample()` kann eine Reihe von PCM-Samples, eine ID einer geöffneten Datei oder einen Speicherblock als Quelle für das neuen Sample verwenden. Welche Quelle verwendet wird, hängt von der Einstellung im Argument `table` ab, die folgende Tags akzeptiert:

**Source:**    Dieser Tag gibt an, von welcher Quelle `CreateSample()` die Audiodaten für das Sample holen soll. Dies muss als Zeichenfolge festgelegt werden, die die Audiodatenquelle identifiziert. Die folgenden Quellen sind möglich:



<b>PCM</b>	Holt die Audiodaten direkt von einer Reihe von PCM-Daten. Wenn Sie dieses Feld angeben, müssen Sie die PCM-Daten in der gleichen Tabelle beginnend bei Index 0 übergeben. <code>CreateSample()</code> wird dann die in <code>length</code> angegebenen PCM-Daten aus dieser Tabelle lesen. Wenn <code>length</code> nicht angegeben wurde, dann liest <code>CreateSample()</code> alle PCM-Daten aus der Tabelle.
<b>File</b>	Holt die Audiodaten aus einer geöffneten Datei. Wenn Sie diesen Quelltyp verwenden, müssen Sie auch eine gültige DateiID im Tag <code>ID</code> angeben. Es ist auch notwendig, das optionale Argument <code>length</code> anzugeben, so dass <code>CreateSample()</code> weiß, wie viele Daten er aus der angegebenen Datei holen soll.
<b>Memory</b>	Holt die Audiodaten aus einem Speicherblock. Wenn Sie diesen Quelltyp verwenden, müssen Sie auch eine gültige Speicherblock-ID im Tag <code>ID</code> angeben. Es ist auch notwendig, das optionale Argument <code>length</code> anzugeben, so dass <code>CreateSample()</code> weiß, wie viele Daten er aus dem angegebenen Speicherblock holen soll.

Der Standardwert für den Tag `Source` ist `PCM`, so dass die Audiodaten aus einer Reihe von PCM-Daten aus der gleichen Tabelle wie diese Optionen holt.

<b>ID:</b>	Dieser Tag ist nur für die Quelltypen <code>File</code> und <code>Memory</code> erforderlich. In diesem Fall müssen Sie hier eine gültige ID der Datei oder Speicherblocks übergeben.
<b>Offset:</b>	Dieser Tag kann nur in Verbindung mit dem Quelltyp <code>Memory</code> verwendet werden. In diesem Fall springt <code>CreateSample()</code> an die Position im Speicherblock, ab dem die Audiodaten abgerufen werden sollen. Der Sprung wird in Bytes angegeben.
<b>Swap:</b>	Dieser Tag kann nur in Verbindung mit Quelltypen <code>File</code> und <code>Memory</code> und einem 16-Bit-Sample verwendet werden. In diesem Fall kann der Tag <code>Swap</code> benutzt werden, ob <code>CreateSample()</code> die beiden Bytes eines 16-Bit-Sample tauschen sollen oder nicht. Dies ist erforderlich, wenn die Sampledaten in der Datei oder im Speicherblock im Little-Endian-Format codiert ist (LSB zuerst). <code>CreateSample()</code> erfordert jedoch 16-bit Sampledaten im Big-Endian-Format (MSB zuerst). Also, wenn Ihre Quelle nur Sampledaten im LSB-Format zur Verfügung stellen kann, einfach den Tag <code>Swap</code> auf <code>True</code> setzen und es sollte nun funktionieren. Dieser Tag ist standardmäßig auf <code>False</code> gesetzt, was bedeutet, nichts wird getauscht.

Bitte beachten Sie, dass das neue Sample mindestens 1000 PCM-Daten verwenden soll. Wenn Sie weniger Daten verwenden, wird die Wiedergabe im Schleifenmodus sehr viel CPU-Kapazität in Anspruch nehmen. Selbst wenn Ihr Sample nur 32 verschiedene Wellenformen hat, sollten Sie sie verketteten, bis das Sample aus Leistungsgründen mindestens 1000 Daten hat.

Wenn Sie große Sampletabellen diesem Befehl übergeben, vergessen Sie bitte nicht, diese Tabellen auf `Nil` zu setzen, wenn Sie diese nicht mehr benötigen. Andernfalls werden Sie große Mengen an Arbeitsspeicher verschwenden.

Ab Hollywood 5.0 können Sie mit `CreateSample()` auch leere Samples erstellen, wenn Sie eine leere Tabelle übergeben oder eine Länge von Null in `length` angeben. In diesem Fall können Sie Befehle wie `InsertSample()` verwenden, um später das Sample mit Audiodaten zu füllen.

#### EINGABEN

<code>id</code>	ID des neuen Samples oder <code>Nil</code> für die automatische ID-Auswahl
<code>table</code>	Tabelle mit weiteren Parametern für das neue Sample
<code>pitch</code>	gewünschte Wiedergabefrequenz für das Sample
<code>fmt</code>	optional: Format des Samples (voreingestellt ist <code>#MONO8</code> ) (V5.0)
<code>length</code>	optional: gewünschte Länge des neuen Samples in PCM-Daten (V5.0)

#### RÜCKGABEWERTE

<code>id</code>	optional: ID des Samples; wird nur zurückgegeben, wenn Sie beim Argument <code>id</code> <code>Nil</code> angegeben haben.
-----------------	--

#### BEISPIEL

```
smpdata = {}
slen = 32
For k = 0 To 30
  For i = 0 To (slen\2)-1
    smpdata[k*slen+i] = -128
    smpdata[k*slen+i+(slen\2)] = 127
  Next
Next
CreateSample(1, smpdata, 6982)
PlaySample(1)
```

Der obige Code erstellt einen einfachen Piepton.

## 47.7 FillMusicBuffer

#### BEZEICHNUNG

`FillMusicBuffer` – beliefert den Sound-Server mit neuen Audiodaten (V5.0)

#### ÜBERSICHT

```
FillMusicBuffer(id, type$, samples[, table])
```

#### BESCHREIBUNG

Dieser Befehl wird in Verbindung mit dynamischen Musik-Streams verwendet, die mit `CreateMusic()` initialisiert wurden. Diese dynamischen Musik-Streams müssen ständig neue Audiodaten zugeführt werden. Dies wird durch `FillMusicBuffer()` gehandhabt. `FillMusicBuffer()` wird die angegebenen Audiodaten an den Sound-Server von Hollywood senden, der diese wiederum an das Audio-Gerät senden wird. Diese Kette von Prozessen ermöglicht es, ununterbrochen dynamisch generierte Audiodaten von Ihrem Skript zu spielen.

Der Sound-Server von Hollywood entscheidet, wenn es mehr Audiodaten benötigt und somit ist der Sound-Server auch derjenige, der entscheidet, wenn Sie `FillMusicBuffer()`

aufrufen müssen. Somit wird Hollywood Sie benachrichtigen, wenn es mehr Audiodaten benötigt und erhöht das `FillMusicBuffer` Ereignis, so dass die Callback-Funktion aufgerufen wird, die Sie mit dem Befehl `InstallEventHandler()` zur Verfügung gestellt haben. Innerhalb dieser Callback-Funktion müssen Sie jetzt den Befehl `FillMusicBuffer()` aufrufen, um neue Daten zu dem Sound-Server zu schicken. Es ist nicht erlaubt, an anderen Stellen im Skript den Befehl `FillMusicBuffer()` aufzurufen! Sie dürfen es nur innerhalb einer Callback-Funktion vom Typ `FillMusicBuffer` verwenden.

Dieser Befehl hat vier Argumente: In `id` geben Sie das zu verwendende Musik-Objekt an. Ihr Callback wird diese Information im Tag `ID` der Ereignismeldung anzeigen. Das dritte Argument `samples` gibt an, wie viele Samples (in PCM-Daten) Sie dem Audio-Server bereitstellen. Dies muss auf genau die gleiche Anzahl von PCM-Daten festgelegt werden, die von Ihnen durch den Callback-Handler angefordert werden. Sie erhalten die Anzahl der PCM-Daten im Tag `Samples` der Ereignismeldung. Das Argument `table` ist optional und muss nur für bestimmte Arten verwendet werden (siehe unten). Das Argument `type$` gibt an, wie Sie die neuen PCM-Daten dem Audio-Server zur Verfügung stellen. Dies kann einer der folgenden Zeichenketten sein:

- |               |  |
|---------------|--|
| <b>PCM</b>    | Sie stellen direkt die neuen PCM-Daten zur Verfügung. In diesem Fall müssen Sie eine Aufstellung übergeben, welches die neuen PCM-Daten im Tag <code>Data</code> des optionalen Argument <code>table</code> enthält (siehe unten).   |
| <b>Sample</b> | Sie werden die neuen PCM-Datengruppen in Form eines Samples zur Verfügung stellen. In diesem Fall müssen Sie die ID des Samples im Tag <code>ID</code> des optionalen Tabellenarguments setzen (siehe unten). Darüber hinaus können Sie die Tags <code>Start</code> , <code>End</code> , <code>Offset</code> und <code>Loop</code> zur Feinabstimmung des Verfahrens verwenden, sollte der Audio-Server neue PCM-Daten holen. Siehe unten für weitere Informationen. |
| <b>Mute</b>   | Wenn Sie diesen Typ angeben, wird der Audio-Server den Audio-Ausgang stumm schalten, während die PCM-Daten übertragen werden. Wenn Sie <code>Mute</code> im <code>type\$</code> Tag übergeben, müssen Sie nichts anderes angeben.  |
| <b>End</b>    | Geben Sie diesen Typ an, wenn Sie die Wiedergabe Ihrer Musik stoppen möchten. Sobald der Audio-Server ein <code>End</code> -Paket empfängt, wird er warten, bis alle Pakete der Warteschlange gespielt wurden und die Wiedergabe Ihrer Musik danach stoppen.   |

Die Tags in dem optionalen Argument `table` hängen von dem in `type$` angegebenen Typ ab. Die folgenden Tags können benutzt werden:

- |              |  |
|--------------|--|
| <b>Data:</b> | Wenn das Argument <code>type\$</code> auf <code>PCM</code> eingestellt ist, müssen Sie eine Reihe von PCM-Daten in diesem Tag angeben. Die Aufstellung sollten so viele Daten enthalten, wie er durch den Audio-Server im Tag <code>Samples</code> anfordert. Die PCM-Daten müssen als signierte ganze Zahlen übergeben werden. Für 8-Bit-Daten läuft der gültige Samplebereich von -128 bis 127 und für 16-Bit-Daten reicht der gültige Samplebereich von -32768 bis 32767. Wenn Sie den Stereo-Modus verwenden, müssen Sie verschachtelte PCM-Daten übergeben, d.h. dem linken Kanalsample folgt das rechte Kanalsample und diesem folgt wieder das linke Kanalsample und so weiter. |
|--------------|--|

**ID:** Wenn `type$` auf `Sample` gesetzt ist, müssen Sie die ID eines Samples übergeben, von dem der Audio-Server die PCM-Daten holen soll. Sie können die Art und Weise der Feinabstimmung, wie der Audioserver die PCM-Daten von diesem Sample holt, mit den Tags `Start`, `End`, `Offset` und `Loop` festlegen. Siehe unten für weitere Informationen.

**Start, End:**

Wenn `type$` auf `Sample` eingestellt ist, können Sie mit diesen beiden Tags den Bereich im Quellensample angeben, die der Audio-Server zum holen der Daten verwenden soll. Dies ist nützlich, wenn Sie vom Audio-Server nur einen Teil der Daten des Samples benutzen wollen. Beide Werte müssen in PCM-Daten festgelegt werden. Diese Tags sind standardmäßig auf 0 für `Start` und auf die Länge des angegebenen Samples für `End` eingestellt. Dies bedeutet, dass für das Abrufen der PCM-Daten standardmäßig das gesamte Sample verwendet wird.

**Offset:** In `Offset` können Sie den Versatz des Samples angeben, ab dem der Audio-Server den PCM-Datenabruf starten soll. Diesen Versatz müssen Sie in PCM-Daten festlegen und bezieht sich auf die Position, welche in `Start` angegeben wurde. Zum Beispiel wird, wenn Sie 10000 in `Start` und 100 in `Offset` übergeben, der Audio-Server die PCM-Daten ab der Position 10100 abrufen. Standardmäßig ist dieser Tag auf 0 eingestellt, womit die PCM-Daten ab dem Anfang des Samples abgerufen werden.

**Loop:** Gibt an, ob `FillMusicBuffer()` an den Anfang der PCM-Daten des Quellensamples springen soll, wenn das Ende erreicht worden ist oder nicht. Der Standardwert ist `True`, was bedeutet, `FillMusicBuffer()` wird automatisch an den Anfang des Samples zurückspringen, wenn ihr Ende erreicht worden ist und mehr PCM-Daten erforderlich sind. Der Beginn des Samples wird durch den Wert im Tag `Start` definiert.

Bitte beachten Sie, dass Sie den Befehl `FlushMusicBuffer()` verwenden müssen, wenn Sie die Audiodaten mit einer sehr geringen Latenz aktualisieren müssen. `FillMusicBuffer()` wird immer etwa 1 Sekunde Musikdaten puffern. Dies bedeutet, dass es etwa 1 Sekunde seit dem Aufruf von `FillMusicBuffer()` dauert, bis Sie tatsächlich die Audiodaten hören, die Sie gerade gesendet haben. Wenn Sie die Audiodaten in Echtzeit aktualisieren möchten, z.B. um auf eine neue Position im Stream zu springen, werden Sie den Musik-Puffer zuerst leeren müssen. Siehe [Abschnitt 47.8 \[FlushMusicBuffer\]](#), [Seite 1029](#), für Details.

## EINGABEN

<code>id</code>	Identifikator des Musikobjekts
<code>type\$</code>	gewünschte Weise, wie die neuen Audiodaten dem Audio-Server zur Verfügung stehen (siehe oben)
<code>samples</code>	Anzahl der angebotenen Samples in PCM-Daten; muss mit der Anzahl Samples identisch sein, die vom Ereignis <code>FillMusicBuffer</code> verlangt wird
<code>table</code>	optional: Tabellenargument für weitere Optionen (siehe oben)

## 47.8 FlushMusicBuffer

### BEZEICHNUNG

FlushMusicBuffer – leert den Puffer des dynamischen Musik-Streams (V6.0)

### ÜBERSICHT

FlushMusicBuffer(id)

### BESCHREIBUNG

Dieser Befehl wird in Verbindung mit dynamischen Musik-Streams verwendet, die mit dem Befehl **CreateMusic()** initialisiert wurden. Diese dynamischen Musik-Streams müssen ständig mit neuen Audiodaten gefüttert werden, die immer wieder mit dem Aufruf des Befehls **FillMusicBuffer()** übertragen werden. **FlushMusicBuffer()** kann verwendet werden, um alle Musik Puffer zu leeren und sie mit neuen Daten aufzufüllen. Dies kann nützlich sein, wenn Sie sofort die Audiodaten aktualisieren müssen, die gerade abgespielt werden. Zum Beispiel weil der Benutzer im Musik-Stream an eine neue Position gesprungen ist. Nach dem Aufruf von **FlushMusicBuffer()** wird Hollywood sofort ein **FillMusicBuffer**-Ereignis aufrufen, so dass Ihr Skript den Audio-Puffer mit aktuellen Daten füllen kann.

Standardmäßig wird es immer eine Verzögerung von etwa 1 Sekunde zwischen dem Aufruf von **FillMusicBuffer()** und der Zeit geben, wo Sie die Audiodaten auf Ihrer Soundkarte hören können. Wenn Sie **FlushMusicBuffer()** aufrufen, können zunächst die Daten mit einer geringeren Latenz an die Soundkarte gesendet werden.

Siehe [Abschnitt 47.7 \[FillMusicBuffer\]](#), [Seite 1026](#), für Details.

### EINGABEN

id            Identifikator des Musikobjektes, das Sie leeren möchten

## 47.9 ForceSound

### BEZEICHNUNG

ForceSound – meldet einen Fehler, wenn Audio-Hardware nicht zugewiesen werden kann (V8.0)

### ÜBERSICHT

ForceSound(fail)

### BESCHREIBUNG

Wenn ein Skript versucht, einen Sound abzuspielen und die Audio-Hardware nicht zugewiesen werden kann, läuft Hollywood normal weiter, nur ohne Sound. Wenn Sie dies nicht möchten, d.h. Hollywood soll einen Fehler melden, falls die Audio-Hardware nicht zugewiesen werden kann, rufen Sie diesen Befehl auf und übergeben Sie im Parameter **fail True**. In diesem Fall gibt Hollywood einen Fehler aus, falls die Audio-Hardware nicht zugewiesen werden kann.

Beachten Sie, dass Sie alternativ auch das Ergebnis von **IsSound()** überprüfen können, um festzustellen, ob die Audio-Hardware zugeordnet werden kann.

**EINGABEN**

`fail`            gibt an, ob Hollywood fehlschlagen soll oder nicht, wenn die Audio-Hardware nicht zugewiesen werden kann (die Standardeinstellung ist `False`, was bedeutet, dass Hollywood nicht fehlschlägt)

## 47.10 FreeModule

**BEZEICHNUNG**

FreeModule – free a module / VERALTET

**ÜBERSICHT**

`FreeModule(id)`

**WICHTIGER HINWEIS**

Dieser Befehl ist veraltet. Bitte verwenden Sie stattdessen `CloseMusic()`.

**BESCHREIBUNG**

This function frees the memory of the module specified by `id`. This is normally not necessary because Hollywood frees all memory when it quits. However, if you are running out of memory and want to free the sample by yourself, use this function.

**EINGABEN**

`id`            identifier of the module

## 47.11 FreeSample

**BEZEICHNUNG**

FreeSample – löscht ein Sample aus dem Arbeitsspeicher

**ÜBERSICHT**

`FreeSample(id)`

**BESCHREIBUNG**

Dieser Befehl löscht das in `id` angegebenen Sample aus dem Arbeitsspeicher. Das ist normalerweise nicht notwendig, da Hollywood allen Arbeitsspeicher freigibt, wenn es beendet wird. Wenn Ihnen allerdings der Arbeitsspeicher knapp wird und Sie das Sample selbst entfernen möchten, benutzen Sie diesen Befehl.

**EINGABEN**

`id`            Identifikation des Samples

**BEISPIEL**

Siehe [Abschnitt 47.26 \[LoadSample\]](#), Seite 1040.

## 47.12 GetChannels

### BEZEICHNUNG

GetChannels – gibt die Anzahl der verfügbaren Kanäle zurück (V6.1)

### ÜBERSICHT

```
n = GetChannels()
```

### BESCHREIBUNG

Dieser Befehl gibt die Anzahl der verfügbaren Kanäle für die Audioausgabe zurück. Der Standardwert ist 8, kann aber mit Hilfe der "Konsolenargumente" geändert werden. Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), [Seite 33](#), für Details.

Beachten Sie, dass dieses Argument nicht die freien Audiokanäle zurückgibt, sondern die Gesamtanzahl der Audiokanäle. Um zu überprüfen, ob es einen freien Kanal für die Audioausgabe hat, verwenden Sie den Befehl [HaveFreeChannel\(\)](#). Siehe [Abschnitt 47.16 \[HaveFreeChannel\]](#), [Seite 1033](#), für Details.

Beachten Sie auch, dass wenn der veraltete Audiotreiber auf AmigaOS aktiv ist (aus Leistungsgründen auf AmigaOS 3.x voreingestellt), die ersten vier Kanäle für die Protracker-Wiedergabe reserviert sind. Das Konsolenargument `-nolegacyaudio` kann verwendet werden, um den veralteten Audiotreiber auf AmigaOS 3.x zu deaktivieren. Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), [Seite 33](#), für Details.

### EINGABEN

keine

### RÜCKGABEWERTE

n            Gesamtzahl der verfügbaren Kanälen

## 47.13 GetPatternPosition

### BEZEICHNUNG

GetPatternPosition – gibt aktuelle Patternposition zurück (V1.9)

### ÜBERSICHT

```
pos = GetPatternPosition()
```

### BESCHREIBUNG

Dieser Befehl gibt die Patternposition des gerade spielenden Protracker-Moduls zurück. Wenn kein Modul spielt, wird -1 zurückgegeben. Sie können diesen Befehl benutzen, um Ihr Skript auf die Musik abzustimmen.

Sie können auch [WaitPatternPosition\(\)](#) benutzen, welcher das Programm anhält, bis die angegebene Patternposition erreicht ist.

### EINGABEN

keine

### RÜCKGABEWERTE

pos            aktuelle Patternposition oder -1

## 47.14 GetSampleData

### BEZEICHNUNG

GetSampleData – ruft die Rohdaten eines Samples ab (V5.0)

### ÜBERSICHT

```
table, count = GetSampleData(id)
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die rohen PCM-Daten in dem in `id` angegebenen Sample abzurufen. Die PCM-Daten werden in einer Tabelle zurückgegeben werden. Das Format der Sampledaten muss als signierte Ganzzahlen entweder in 8-Bit (im Bereich von -128 bis +127) oder 16-Bit (im Bereich von -32768 bis 32767) angegeben werden. Sie können das Sampleformat herausfinden, indem Sie dem Befehl `GetAttribute()` das Attribut `#ATTRTYPE` für die Sampleabfrage übergeben. Wenn das Sample zwei Kanäle verwendet (das heißt Stereo), werden die PCM-Daten in verschachtelter Reihenfolge zurückgegeben werden, das heißt linker Kanalsample gefolgt vom rechten Kanalsample gefolgt durch das linke Kanalsample und so weiter.

Der zweite Rückgabewert `count` von diesem Befehl ist ein Zählerwert, der die Anzahl von PCM-Daten in der Tabelle angibt. Vorsicht, dieser Wert gibt nicht die tatsächliche Gesamtzahl der Aufstellungs-Elemente zurück, aber die Anzahl der PCM-Daten. Bei Stereo-Samples bilden die linken und rechten Kanalsamples zusammen eine Datengruppe. Wenn Sie also Stereodaten erhalten, wird es doppelt so viele Samples in der Tabelle haben als durch `count` beziffert, weil diese Zählungen in PCM-Daten stattfindet.

Wenn Sie große Sampledatentabellen von diesem Befehl erhalten, bitte vergessen Sie nicht, diese Tabellen auf `Nil` zu setzen, wenn Sie sie nicht mehr benötigen. Andernfalls werden Sie große Mengen an Arbeitsspeicher verschwenden.

Um eine Tabelle von PCM-Daten zurück in ein Sample zu konvertieren, können Sie den Befehl `CreateSample()` verwenden.

### EINGABEN

`id`            Identifikator des Samples

### RÜCKGABEWERTE

`table`        eine Tabelle, die die rohen PCM-Daten vom angegebenen Sample enthält,  
`count`        Anzahl der Daten innerhalb der Tabelle

## 47.15 GetSongPosition

### BEZEICHNUNG

GetSongPosition – gibt aktuelle Musikposition zurück (V1.9)

### ÜBERSICHT

```
pos = GetSongPosition()
```

### BESCHREIBUNG

Dieser Befehl gibt die Musikposition des gerade spielenden Protracker-Moduls zurück. Wenn kein Modul läuft, wird -1 zurückgegeben. Sie können diesen Befehl benutzen, um Ihr Skript auf die Musik abzustimmen.



Sie können auch `WaitSongPosition()` benutzen, welcher das Programm anhält bis die angegebene Musikposition erreicht ist.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`pos`            aktuelle Songposition oder -1

## 47.16 HaveFreeChannel

**BEZEICHNUNG**

`HaveFreeChannel` – überprüft, ob ein freier Kanal verfügbar ist (V6.1)

**ÜBERSICHT**

`n = HaveFreeChannel()`

**BESCHREIBUNG**

Dieser Befehl überprüft, ob ein freier Kanal für die Audioausgabe zur Verfügung steht. Falls einer frei ist, wird `HaveFreeChannel()` den Index des freien Kanals zurückgeben, andernfalls gibt er 0 zurück.

Beachten Sie auch, dass wenn der veraltete Audiotreiber auf AmigaOS aktiv ist (aus Leistungsgründen auf AmigaOS 3.x voreingestellt), die ersten vier Kanäle für die Protracker-Wiedergabe reserviert sind. Das Konsolenargument `-nolegacyaudio` kann verwendet werden, um den veralteten Audiotreiber auf AmigaOS 3.x zu deaktivieren. Siehe [Abschnitt 3.2 \[Konsolenargumente\]](#), [Seite 33](#), für Details.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`n`            Index des freien Kanals oder 0, falls alle Kanäle besetzt sind

## 47.17 InsertSample

**BEZEICHNUNG**

`InsertSample` – fügt ein Sample in ein anderes Sample ein (V5.0)

**ÜBERSICHT**

`InsertSample(src, dst, pos[, len, table])`

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die in `len` angegebenen PCM-Daten des Samples `src` an der Position `pos` im Sample `dst` einzufügen. Wenn das optionale Argument `len` nicht angegeben ist, wird das gesamte Sample in die in `pos` angegebene Position eingefügt werden. Wenn die beiden Samples nicht das gleiche Format verwenden, wird dieser Befehl automatisch eine geeignete Umwandlung der Audiodaten vornehmen, so dass die Sampletiefe, das Kanallayout und die Abtastrate der beiden Samples passen.

Mit dem optionalen Argument **table** können Sie erweiterte Optionen konfigurieren. Die folgenden Tags werden derzeit erkannt:

**Start, End:**

Mit diesen beiden Tags können Sie einen Bereich in der Quelldatei angeben, die in das Ziel eingefügt werden soll. Dies ist nützlich, wenn Sie nur einen Teil des Quellensamples in das Ziel einfügen wollen. Beide Werte müssen als PCM-Daten festgelegt werden. Diese Tags sind standardmäßig auf 0 für **Start** und die Länge des Quellensamples für **End** gesetzt. Dies bedeutet, dass standardmäßig das gesamte Sample eingefügt wird.

**Offset:** Gibt die Position im Quellensample an, bei der **InsertSample()** beginnen soll, die PCM-Daten für das Zielsample auszulesen. Diese Position muss in PCM-Daten festgelegt werden und ist in Bezug auf die Position in **Start** anzugeben. Zum Beispiel, wenn Sie 10000 in **Start** übergeben und 100 in **Offset**, dann startet **InsertSample()** den Abruf von PCM-Daten an der Position 10100. Dieser Tag ist standardmäßig auf 0 gesetzt, womit die PCM-Daten von Anfang an des Quellensamples geholt werden.

**Loop:** Gibt an, ob **InsertSample()** an den Anfang der PCM-Daten des Quellensamples springen soll, wenn das Ende erreicht worden ist oder nicht. Der Standardwert ist **True**, was bedeutet, **InsertSample()** wird automatisch an den Anfang des Samples zurückspringen, wenn ihr Ende erreicht worden ist und mehr PCM-Daten erforderlich sind. Der Beginn des Sample wird durch den Wert im Tag **Start** definiert.

Bitte beachten Sie, dass dieser Befehl die Länge des Zielsamples verlängert. Bestehende Audiodaten werden nicht überschrieben. Sie werden nur durch den Einfügeprozess nach vorne verschoben werden.

## EINGABEN

<b>src</b>	Identifikator des Quellensamples
<b>dst</b>	Identifikator des Zielsamples
<b>pos</b>	Position in PCM-Daten, wo in <b>dst</b> das <b>src</b> eingefügt werden soll
<b>len</b>	optional: Anzahl der PCM-Daten, die von <b>src</b> in <b>dst</b> eingefügt werden sollen (voreingestellt ist die Länge von <b>src</b> )
<b>table</b>	optional: Tabelle für weitere Parameter (siehe oben)

## BEISPIEL

```
InsertSample(src, dst, pos[, len, table])
InsertSample(1, 2, 44100, 44100, {Start = 25000, End = 30000})
```

Der obige Code fügt eine Sekunde Audiodaten vom Sample 1 in Sample 2 ein. Das Sample 1 wird an der Position 44100 ins Sample 2 eingefügt, aber nur die PCM-Daten im Bereich von 25000 bis 30000 werden geholt.

## 47.18 IsChannelPlaying

### BEZEICHNUNG

IsChannelPlaying – prüft, ob ein Kanal wiedergegeben wird (V6.1)

### ÜBERSICHT

```
playing[, type, id] = IsChannelPlaying(n)
```

### BESCHREIBUNG

Dieser Befehl überprüft, ob der in **n** angegebene Kanal gerade abgespielt wird und liefert **True**, falls es so ist, sonst **False**. Wenn der Kanal gerade abgespielt wird, gibt **IsChannelPlaying()** auch den Typ und die ID des Objekts zurück, das derzeit auf diesem Kanal abgespielt wird. Dies kann **#MUSIC**, **#SAMPLE** oder **#VIDEO** sein.

### EINGABEN

**n**                der zu überprüfende Kanalindex; Kanalnummerierung beginnt bei 1 und geht bis zur Anzahl der verfügbaren Kanäle

### RÜCKGABEWERTE

**playing**    **True**, wenn der Kanal abgespielt wird, andernfalls **False**

**type**        optional: Objekttyp, der gerade auf diesem Kanal spielt; wird nur zurückgegeben, wenn der Kanal gerade abgespielt wird

**id**          optional: ID des Objekts, das gerade auf diesem Kanal spielt. wird nur zurückgegeben, wenn der Kanal gerade abgespielt wird

## 47.19 IsModule

### BEZEICHNUNG

IsModule – determine if a module is in a supported format / VERALTET

### ÜBERSICHT

```
ret = IsModule(file$)
```

### WICHTIGER HINWEIS

Dieser Befehl ist veraltet. Bitte verwenden Sie stattdessen **IsMusic()**.

### BESCHREIBUNG

This function will check if the file specified **file\$** is in a supported module format. If it is, this function will return **True**, otherwise **False**. If this function returns **True**, you can load the module by calling **LoadModule()**.

### EINGABEN

**file\$**        file to check

### RÜCKGABEWERTE

**ret**            **True** if the module is in a supported format, **False** otherwise

## 47.20 IsMusicPlaying

### BEZEICHNUNG

IsMusicPlaying – prüft, ob eine Musik gerade gespielt wird (V4.5)

### ÜBERSICHT

```
playing = IsMusicPlaying(id)
```

### BESCHREIBUNG

Dieser Befehl überprüft, ob die in `id` angegebene Musik gerade abgespielt wird. Wenn ja, wird `True` zurückgegeben, andernfalls `False`.

### EINGABEN

`id`            Identifikator der zu überprüfenden Musik

### RÜCKGABEWERTE

`playing`    `True`, wenn die Musik gerade spielt, andernfalls `False`

## 47.21 IsMusic

### BEZEICHNUNG

IsMusic – prüft, ob eine Musikdatei in einem unterstützten Format vorliegt (V2.0)

### ÜBERSICHT

```
ret, fmt$ = IsMusic(file$[, table])
```

### BESCHREIBUNG

Dieser Befehl prüft, ob die Datei `file$` in einem unterstützten Musikformat vorliegt. Wenn ja, wird dieser Befehl in `ret` mit `True` zurückgeben, andernfalls `False`. Wenn dieser Befehl `True` zurückgibt, können Sie die Musik mit `OpenMusic()` öffnen.

Der zweite Rückgabewert `fmt$` ist eine Zeichenfolge, die das Musikformat der Datei enthält.

Ab Hollywood 6.0 hat dieser Befehl das optionale Argument `table`, womit Sie weitere Optionen konfigurieren können:

**Loader:** Mit diesem Tag können Sie einen oder mehrere Formatlademodule angeben, die die Datei laden soll. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Lademodule beinhaltet. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, der die angegebene Datei öffnet. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen beinhaltet. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die

die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Siehe [Abschnitt 47.30 \[OpenMusic\]](#), [Seite 1045](#), für eine Liste mit den unterstützten Musikformaten.

#### EINGABEN

`file$`      die zu überprüfende Datei

`table`      optional: Tabelle für weitere Optionen (V6.0)

#### RÜCKGABEWERTE

`ret`          `True`, wenn das Musikobjekt in einem unterstützten Format vorliegt, andernfalls `False`

`fmt$`          Format der Musikdatei

## 47.22 IsSamplePlaying

#### BEZEICHNUNG

`IsSamplePlaying` – prüft, ob ein Sample gespielt wird

#### ÜBERSICHT

`playing = IsSamplePlaying(id)`

#### BESCHREIBUNG

Dieser Befehl prüft, ob das durch `id` angegebene Sample im Moment gespielt wird und gibt `True` zurück für den Fall, dass es gespielt wird - `False` für den Fall, dass es nicht gespielt wird.

#### EINGABEN

`id`            Identifikator des Samples

#### RÜCKGABEWERTE

`playing`      `True`, falls das durch `id` angegebene Sample gespielt wird, andernfalls `False`

#### BEISPIEL

```
LoadSample(1, "Sound/Samples/ChurchOrgan.wav")
PlaySample(1)
Repeat
  Wait(2)
Until IsSamplePlaying(1) = False
FreeSample(1)
```

Der obige Code lädt das Sample "Sound/Samples/ChurchOrgan.wav", spielt es und wartet dann auf das Ende. Danach wird das Sample aus dem Arbeitsspeicher entfernt. Wenn Sie es wie oben tun wollen, ist es einfacher für Sie, den Befehl `WaitSampleEnd()` zu verwenden. Aber wenn Sie einige Dinge erledigen wollen, während das Sample abgespielt wird, müssen Sie es auf diese Weise tun (mit `IsSamplePlaying()` und einer Schleife).

## 47.23 IsSample

### BEZEICHNUNG

IsSample – prüft, ob ein Sample in einem unterstützten Format vorliegt

### ÜBERSICHT

```
ret = IsSample(file$[, table])
```

### BESCHREIBUNG

Dieser Befehl überprüft, ob die in `file$` angegebene Datei in einem Format vorliegt, welches für Samples unterstützt wird. Wenn ja, gibt dieser Befehl **True** zurück, andernfalls **False**. Wenn der Befehl **True** zurückgibt, können Sie das Sample durch Aufruf von `LoadSample()` laden und später abspielen.

Ab Hollywood 6.0 hat dieser Befehl ein optionales Argument `table`, womit Sie weitere Optionen konfigurieren können:

**Loader:** Mit diesem Tag können Sie ein oder mehrere Formatlademodule angeben, die die Datei laden soll. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Lademodule beinhaltet. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Datei-Adaptermodul angeben, der die angegebene Datei öffnet. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen beinhaltet. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

### EINGABEN

`file$`      die zu überprüfende Datei

`table`      optional: Tabelle für weitere Optionen (V6.0)

### RÜCKGABEWERTE

`ret`      **True**, wenn das Sample in einem unterstützen Format ist, andernfalls **False**

## 47.24 IsSound

### BEZEICHNUNG

IsSound – prüft, ob Hollywood einen Ton ausgeben kann

### ÜBERSICHT

```
ret = IsSound()
```

**BESCHREIBUNG**

Dieser Befehl prüft, ob Hollywood einen Ton ausgeben kann. Sie können diesen Befehl benutzen, falls Ihre Anwendung ohne Tonausgaben nicht verwendet werden kann. Normalerweise überspringt Hollywood einfach allen Code, der mit Audio-Ausgaben zu tun hat. Wenn Hollywood keinen Ton ausgeben kann, fährt Hollywood mit der Ausführung des Skripts fort. Wenn Sie das nicht wollen, benutzen Sie `IsSound()`, um zu erfahren, ob Töne ausgegeben werden können.

Ab Hollywood 8.0 können Sie auch den Befehl `ForceSound()` verwenden, damit Hollywood einen Fehler meldet, wenn die Audio-Hardware nicht zugewiesen werden kann. Siehe [Abschnitt 47.9 \[ForceSound\]](#), [Seite 1029](#), für Details.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`ret`            `True`, falls Sound gespielt werden kann; andernfalls `False`

**BEISPIEL**

```
If IsSound() = False
  SystemRequest("My App", "Sorry, sound is required!", "OK")
End
EndIf
```

Obiger Code prüft ob Sound ausgegeben werden kann und beendet das Programm mit einer Fehlermeldung, wenn das nicht möglich ist.

## 47.25 LoadModule

**BEZEICHNUNG**

LoadModule – load a module / VERALTET

**ÜBERSICHT**

```
LoadModule(id, filename$)
```

**WICHTIGER HINWEIS**

Dieser Befehl ist veraltet. Bitte verwenden Sie stattdessen `OpenMusic()`.

**BESCHREIBUNG**

This function loads the module specified by `filename$` into memory and gives it the identifier `id`. The module must be in Protracker format.

**EINGABEN**

`id`            identifier for the module

`filename$`    file to load

**BEISPIEL**

```
LoadModule(5, "Modules/StardustMemories.mod")
```

The above declaration assigns module number 5 to the module "StardustMemories.mod" located in a subdrawer named "Modules".

## 47.26 LoadSample

### BEZEICHNUNG

LoadSample – lädt ein Sample

### ÜBERSICHT

```
[id] = LoadSample(id, filename$[, table])
```

### BESCHREIBUNG

Dieser Befehl lädt das durch `filename$` angegebene Sample in den Arbeitsspeicher und weist ihm den Identifikator `id` zu. Wenn Sie `Nil` in `id` übergeben, wird `LoadSample()` automatisch eine freie ID auswählen und Ihnen zurückgeben.

Sampleformate, die auf allen Plattformen unterstützt werden, sind RIFF WAVE, IFF 8SVX, IFF 16SV und Sampleformate, für die Sie ein Plugin haben. Je nach Plattform, auf der Hollywood ausgeführt wird, können mehr Sampleformate unterstützt werden. Zum Beispiel auf Amigakompatiblen Systemen wird Hollywood in der Lage sein, alle Sampleformate über entsprechende Datatypes zu öffnen.

Ab Hollywood 6.0 hat dieser Befehl ein optionales Argument `table`, mit dem Sie zusätzliche Parameter übergeben können:

**Loader:** Mit diesem Tag können Sie einen oder mehrere Formatlademodule angeben, die die Datei laden soll. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Lademodule beinhaltet. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Datei-Adaptermodul angeben, der die angegebene Datei öffnet. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen beinhaltet. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Dieser Befehl ist auch als Präprozessor-Anweisung vorhanden: `@SAMPLE` lädt ein Sample vor!

Bitte beachten Sie, dass dieser Befehl Sampledaten vollständig in den Arbeitsspeicher lädt. Wenn Sie planen, lange Samples zu spielen, sollten Sie besser `OpenMusic()` nutzen, womit nur kleine Teile der Tondaten im Arbeitsspeicher gepuffert werden.

### EINGABEN

`id` ID des Sample oder `Nil` für die [automatische ID-Auswahl](#)

`filename$` Datei, die geladen wird



`table` optional: Tabelle für weitere Optionen (siehe oben) (V6.0)

## RÜCKGABEWERTE

`id` optional: ID des Samples; wird nur zurückgegeben, wenn Sie beim Argument `id` `Nil` angegeben haben

## BEISPIEL

```
LoadSample(1, "Sound/Samples/WahWah.wav")
PlaySample(1)
WaitSampleEnd(1)
FreeSample(1)
```

Der obige Code lädt das Sample "Sound/Samples/WahWah.wav", spielt es ab und wartet, bis es zu Ende ist. Dann wird es aus dem Arbeitsspeicher gelöscht.

## 47.27 MixSample

### BEZEICHNUNG

MixSample – mixt existierende Samples in ein neues Sample (V5.0)

### ÜBERSICHT

```
[id] = MixSample(id, len, pitch, fmt, smp1, opt1, ...)
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um eines oder mehrere bestehende Samples in ein neues Sample zu mixen. Das neue Sample wird der Sampleliste von Hollywood hinzugefügt und man kann mit der angegebenen `id` darauf zugreifen. Wenn Sie `Nil` in `id` übergeben, wird `MixSample()` automatisch eine freie ID auswählen und Ihnen zurückgeben. Das zweite Argument `len` gibt die gewünschte Länge des neuen Samples in PCM-Daten an. Das dritte Argument `pitch` gibt an, wie viele Daten pro Sekunde dem Audio-Gerät gesendet werden soll. Für CD-Qualität würden Sie 44100 dem Argument `pitch` übergeben, aber in vielen Fällen ist 22050 auch ausreichend. Im Argument `fmt` geben Sie das gewünschte Sampleformat für das neuen Sample an. Derzeit werden folgende Formate unterstützt: `#MONO8`, `#STEREO8`, `#MONO16` und `#STEREO16`. Die beiden erstgenannten sind 8-Bit-PCM-Codierungen, während die beiden letztgenannten 16 Bit pro Kanal verwenden.

Die Samples, die in das neue Sample gemixt werden sollen, werden ab Argument 5 übergeben. Für jedes Sample, welches in das neue Sample gemixt wird, muss man eine ID sowie eine Tabelle übergeben, die weitere Parameter für den Mischvorgang enthält. Sie können diese Muster so oft wiederholen, wie Sie möchten. Die Tabelle, die für jedes Sample übergeben wird, unterstützt die folgenden Tags:

**Pitch:** Gibt die Frequenz an, die verwendet werden soll, wenn dieses Sample gemischt wird. Dieser Tag ist standardmäßig die aktuelle Frequenz, die mit dem Befehl `SetPitch()` definiert wurde.

**Offset:** Gibt die Position im Sample an, bei der `MixSample()` beginnen soll, die PCM-Daten auszulesen. Diese Position muss in PCM-Daten festgelegt werden. Dieser Tag ist standardmäßig auf 0 gesetzt, womit die Daten vom Anfang des Samples an geholt werden.

- Length:** Gibt die maximale Anzahl von PCM-Daten an, die gemischt werden sollen. Der Standardwert ist -1, womit alle zur Verfügung stehenden Daten in das neue Sample gemixt werden.
- Loop:** Gibt an, ob der Mixer weiterhin PCM-Daten am Anfang des Samples holen soll oder nicht, sobald das Ende erreicht ist. Der Standardwert ist **True**, womit der Mixer automatisch an den Anfang des Samples zurück springt, wenn das Ende erreicht ist und mehr PCM-Daten erforderlich sind.
- Scale:** Gibt einen Skalierungsfaktor an, der auf den Mixvorgang angewendet werden soll. Wenn Sie viele Samples zusammen mixen, bekommen Sie wahrscheinlich ein paar Lärmefekte, die Sie nicht wollen. Sie können diese Effekte durch die Reduzierung der Lautstärke vermindern. Dies kann mit Hilfe von diesem Tag erreicht werden. Jede PCM-Daten wird mit dem Skalierungsfaktor, den Sie hier übergeben, multiplizieren. Damit das Volumen um 50% reduziert wird, übergeben Sie einfach 0.5. Dieser Tag ist standardmäßig auf 1.0 gesetzt, womit keine Skalierung angewandt wird.
- Threshold:** Mit diesem Tag können Sie einen Schwellenwert angeben, mit dem dieses Sample in das neue Sample gemischt werden soll. Zum Beispiel, wenn Sie dieses Sample erst nach 10000 PCM-Daten in das neue Sample gemischt haben wollen, können Sie hier 10000 angeben. Der hier angegebene Wert muss in PCM-Daten übergeben werden. Dieser Tag ist standardmäßig auf 0 gesetzt, womit dieses Sample von Beginn an in das neue Sample gemischt wird.

Dieser Befehl ist mächtig. Es wird automatische die Konvertierung zwischen den verschiedenen Kodierungen, Abtastraten und Kanal-Layouts durchführen. Außerdem können Sie so viele Samples in das neue Sample mixen, wie Sie möchten. Sie dürfen auch dasselbe Sample mehrmals mit unterschiedlichen Mixparameter wie variierende Geschwindigkeit oder verschiedenen Schwellwerten in das neue Sample mischen. Wenn Sie unerwünschte Geräuscheffekte erhalten, versuchen Sie die Lautstärke einzelner Samples mit dem Tag **Scale** zu reduzieren (siehe oben).

## EINGABEN

- |              |   |
|--------------|---|
| <b>id</b>    | ID des neuen Samples oder <b>Nil</b> für die automatische ID-Auswahl  |
| <b>len</b>   | gewünschte Länge des neuen Samples in PCM-Daten   |
| <b>pitch</b> | gewünschte Frequenz des neuen Samples   |
| <b>fmt</b>   | gewünschtes Format für das neue Sample  |
| <b>smp1</b>  | erstes Sample für den Mix   |
| <b>opt1</b>  | Optionstabelle des ersten Samples für den Mix   |
| <b>...</b>   | optional: Sie können die ID/Optionssequenz so oft wiederholen, wie Sie so wollen, so können Sie so viele Samples hineinmischen, wie Sie möchten |

## RÜCKGABEWERTE

- |           |   |
|-----------|---|
| <b>id</b> | optional: ID des neuen Samples; wird nur zurückgegeben, wenn Sie beim Argument <b>id</b> <b>Nil</b> angegeben haben |
|-----------|---|

**BEISPIEL**

```
MixSample(1, 10 * 44100, 44100, #STEREO16, 2, {}, 3,
          {Threshold = 3 * 44100}, 4, {6 * 44100})
```

Der obige Code erstellt ein neues Sample im 44.1-Format mit 16 Bit pro PCM-Daten und zwei Kanäle. Die Länge des Samples wird genau 10 Sekunden betragen. Es beginnt Sample 2 in das neue Sample zu mixen, nach 3 Sekunden wird Sample 3 gemixt und nach 6 Sekunden wird angefangen, Sample 4 zu spielen.

**47.28 MUSIC****BEZEICHNUNG**

MUSIC – lädt eine Musik vor (V2.0)

**ÜBERSICHT**

```
@MUSIC id, filename$[, table]
```

**BESCHREIBUNG**

Verwenden Sie diese Präprozessor-Anweisung, um eine Musik vorab zu laden, die Sie später mit `PlayMusic()` abspielen wollen. Die Musikdatei kann in einem beliebigen von Hollywood unterstützten Format sein. Bitte werfen Sie einen Blick auf die Informationen von `OpenMusic()` in dieser Dokumentation zu den unterstützten Musikformaten. Wenn die Musikdatei in einem Streaming-Format ist, wird diese Präprozessor-Anweisung nur das Musikobjekt für die spätere Wiedergabe initialisieren. Es wird nicht große Musikobjekte vollständig in den Arbeitsspeicher laden, aber sie werden als Sound-Stream gepuffert von der Festplatte abgespielt werden.

Das dritte Argument `table` ist optional. Es ist eine Tabelle, die weitere Möglichkeiten für den Ladevorgang bereitstellt. Die folgenden Felder der Tabelle können verwendet werden:

- Link:** Setzen Sie dieses Feld auf `False`, wenn Sie dieses Musikobjekt nicht in die ausführbare Datei/das Applet einbinden wollen, wenn Sie Ihr Skript kompilieren. Dieses Feld ist standardmäßig auf `True` gesetzt, was bedeutet, dass das Musikobjekt mit der ausführbaren Datei/dem Applet beim Kompilieren verknüpft wird.
- Loader:** Mit diesem Tag können Sie einen oder mehrere Formatlademodule angeben, die die Musikdatei laden soll. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Lademodulen beinhaltet. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)
- Adapter:** Mit diesem Tag können Sie ein oder mehreren Datei-Adaptermodul angeben, der die angegebene Datei öffnet. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen beinhaltet. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Musikformate, die auf allen Plattformen unterstützt werden, sind RIFF WAVE, IFF 8SVX, IFF 16SV, Protracker-Module und -Formate, für die Sie ein Plugin haben. Je nach Plattform, auf der Hollywood ausgeführt wird, können mehr Musikformate unterstützt werden. Zum Beispiel auf Windows unterstützt Hollywood alle Formate, die Directshow laden können und unter macOS, alle Formate die von Apples Audiodatei-Schnittstelle erkannt werden.

Wenn Sie eine Musik manuell öffnen wollen, dann benutzen Sie den Befehl [OpenMusic\(\)](#).

**EINGABEN**

**id** ein Wert, der dieses Sample später im Code identifiziert

**filename\$** die Musikdatei, welche sie vorladen möchten

**table** optional: eine Tabelle mit weiteren Optionen (siehe oben)

**BEISPIEL**

```
@MUSIC 1, "TurricanII_Remix.mod"
```

Der obige Code öffnet "TurricanII\_Remix.mod", so dass Sie es später mit [PlayMusic\(\)](#) abspielen können.

## 47.29 OpenAudio

**BEZEICHNUNG**

OpenAudio – öffnet die Audio-Hardware (V8.0)

**ÜBERSICHT**

[OpenAudio\(\)](#)

**BESCHREIBUNG**

Dieser Befehl kann zum Öffnen der Audio-Hardware verwendet werden. Es ist normalerweise nicht erforderlich, diesen Befehl aufzurufen, da Hollywood die Audio-Hardware automatisch öffnet, sobald sie benötigt wird. Unter AmigaOS und kompatiblen Systemen gibt es jedoch Situationen, in denen Sie möglicherweise eine fein abgestimmte Steuerung der Audio-Hardware benötigen. Zum Beispiel weil ein anderes Programm versucht, exklusiven Zugriff auf die Audio-Hardware zu erhalten, was bedeutet, dass Ihr Skript sie zuerst freigeben muss. In diesen Situationen können Sie [OpenAudio\(\)](#) und [CloseAudio\(\)](#) manuell aufrufen. Abgesehen von dieser speziellen Situation besteht keine Notwendigkeit, diesen Befehl überhaupt aufzurufen.

**EINGABEN**

keine

**BEISPIEL**

```

OpenAudio()
OpenMusic(1, "Turrican2_Remix.mod")
PlayMusic(1)
WaitLeftMouse
StopMusic(1)
CloseAudio()

```

Der obige Code spielt "Turrican2\_Remix.mod" und schließt die Audiohardware. Dadurch können andere Programme auf AmigaOS exklusiven Zugriff auf Audiogeräte reservieren.

**47.30 OpenMusic****BEZEICHNUNG**

OpenMusic – öffnet eine Musikdatei (V2.0)

**ÜBERSICHT**

```
[id] = OpenMusic(id, filename$[, table])
```

**BESCHREIBUNG**

Dieser Befehl öffnet die Musikdatei `filename$` und weist ihr den Identifikator `id` zu. Wenn Sie in `id` **Nil** übergeben, wird `OpenMusic()` automatisch eine freie ID auswählen und sie zurückgeben. Die in `filename$` angegebene Datei wird für die Wiedergabe geöffnet und vorbereitet werden. Bitte beachten Sie, dass mit `OpenMusic()` geöffnete Dateien unter Verwendung von Audio-Streaming abgespielt werden. Der Befehl `LoadSample()` auf der anderen Seite, wird zunächst die gesamte Sound-Datei in den Arbeitsspeicher laden. Daher sollten Sie `LoadSample()` für kurze Stücke verwenden und `OpenMusic()` für größere Sounds und Hintergrundmusik aufrufen.

Musikformate, die auf allen Plattformen unterstützt werden, sind RIFF WAVE, IFF 8SVX, IFF 16SV, Protracker-Module und -Formate, für die Sie ein Plugin haben. Je nach Plattform, auf der Hollywood ausgeführt wird, können mehr Musikformate unterstützt werden. Zum Beispiel auf Windows unterstützt Hollywood alle Formate, die Directshow laden können und unter macOS, alle Formate die von Apples Audiodatei-Schnittstelle erkannt werden.

Ab Hollywood 6.0 hat dieser Befehl ein optionales Argument `table`, womit Sie zusätzliche Parameter übergeben können:

**Loader:** Mit diesem Tag können Sie ein oder mehrere Formatlademodule angeben, die diese Musik laden sollen. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehreren Lademodulen enthält. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen sollen. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehreren Adaptermodulen enthält. Standardmäßig wird der mit `SetDefaultAdapter()` beinhaltete. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwen-

det. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), Seite 96, für Details. (V6.0)

#### UserTags:

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), Seite 100, für Details. (V10.0)

Dieser Befehl ist auch als Präprozessor-Anweisung vorhanden: Mit **@MUSIC** können Sie Musikobjekte vorladen!

### EINGABEN

**id**            Identifikator der Musikdatei oder **Nil** für die **automatische ID-Auswahl**

**filename\$**  
Musikdatei die geladen werden soll

**table**       optional: Tabellenargument mit weiteren Optionen (siehe oben) (V6.0)

### RÜCKGABEWERTE

**id**            optional: Identifikator der Musik; wird nur zurückgegeben, wenn Sie **Nil** als **id** angegeben haben (siehe oben)

### BEISPIEL

```
OpenMusic(1, "Turrican2_Remix.mod")
PlayMusic(1)
```

Der obige Code spielt "Turrican2\_Remix.mod" ab.

## 47.31 PauseModule

### BEZEICHNUNG

PauseModule – pause the currently playing module / VERALTET

### ÜBERSICHT

PauseModule()

### WICHTIGER HINWEIS

Dieser Befehl ist veraltet. Bitte verwenden Sie stattdessen **PauseMusic()**.

### BESCHREIBUNG

This function pauses the currently playing module. It can be resumed with the **ResumeModule()**.

### EINGABEN

Keine

## 47.32 PauseMusic

### BEZEICHNUNG

PauseMusic – hält eine Musik an (V2.0)

### ÜBERSICHT

PauseMusic(id)

### BESCHREIBUNG

Dieser Befehl hält die in `id` abspielende Musik an. Diese Musik muss spielen, wenn Sie diesen Befehl aufrufen. Sie können die Wiedergabe fortsetzen, indem Sie den Befehl `ResumeMusic()` aufrufen.

### EINGABEN

`id`            Identifikator der laufenden Musik, die angehalten werden soll

## 47.33 PlayModule

### BEZEICHNUNG

PlayModule – start playing a music module / VERALTET

### ÜBERSICHT

PlayModule(number)

### WICHTIGER HINWEIS

Dieser Befehl ist veraltet. Bitte verwenden Sie stattdessen `PlayMusic()`.

### BESCHREIBUNG

Starts playing the music module with the number `number`. This music module must have been loaded with `LoadModule()`. If there is already a music module playing, it will be stopped automatically.

### EINGABEN

`number`        number of the music module to start

### BEISPIEL

```
LoadModule(1,"StardustMemories.mod")
PlayModule(1)
```

## 47.34 PlayMusic

### BEZEICHNUNG

PlayMusic – spielt eine Musik ab (V2.0)

### ÜBERSICHT

PlayMusic(id[, table])

### BESCHREIBUNG

Dieser Befehl startet die Wiedergabe der in `id` angegebene Musik. Dieses Musikobjekt muss entweder mit der Präprozessor-Anweisung `@MUSIC` oder dem Befehl `OpenMusic()` zuvor geöffnet worden sein.

Bitte beachten Sie, dass vor Hollywood 6.0 nur eine Musik abgespielt werden konnte. Ab 6.0 ist diese Einschränkung zwar nicht mehr vorhanden, wird aber immer noch erzwungen, um mit älteren Skripten kompatibel zu bleiben. Wenn also ein Musikobjekt bereits abgespielt wird und Sie diesen Befehl aufrufen, wird die erste Musik gestoppt, bevor die Wiedergabe der neuen Musik beginnt. Wenn Sie mehrere MusikObjekte zur gleichen Zeit abspielen wollen, müssen Sie explizit diese Einschränkung deaktivieren, indem Sie den Befehl `LegacyControl()` aufrufen und den Tag `SingleMusic` auf `False` setzen. `PlayMusic()` stoppt nun nicht mehr jede Musik. Siehe [Abschnitt 50.22 \[LegacyControl\]](#), [Seite 1104](#), für Details.

Vor Hollywood 4.5 war das zweite Argument optional und gab an, wie oft die Musik abgespielt werden soll. Ab Hollywood 4.5 ist das zweite Argument nun ein optionales Tabellenargument. Natürlich wird die alte Syntax noch wegen der Kompatibilität unterstützt. Neue Skripte sollten allerdings die neue Syntax verwenden. Das optionale Argument `table` erkennt die folgenden Tags:

- Times:** Diesen Tag können Sie verwenden, um anzugeben, wie oft die Musik abgespielt werden soll. Dieser Tag ist auf 1 voreingestellt, womit das Musikobjekt nur einmal abgespielt wird. Wenn Sie ein Musikobjekt als Endlosschleife spielen wollen, geben Sie hier 0 ein.
- Volume:** Geben Sie hier die gewünschte Lautstärke an. Der Bereich geht von 0 bis 64. Wenn nichts angegeben wurde, wird die Standardlautstärke der Musikobjekte verwendet werden. (Sie können die Standardlautstärke eines Musikobjekts ändern, indem Sie den Befehl `SetMusicVolume()` benutzen).
- Channel:** Hier können Sie den Kanal für die Wiedergabe des Musikobjekts einstellen. In der Standardeinstellung wählt `PlayMusic()` automatisch einen freien Kanal und wird fehlschlagen, wenn es keinen freien Kanal mehr hat. Um dieses Verhalten zu ändern, können Sie dieses Feld verwenden. Wenn ein Kanal angegeben wurde, wird Hollywood immer die Wiedergabe auf dem hier angegebenen Kanal erzwingen. Wenn der Kanal bereits verwendet wird, wird er zunächst gestoppt werden. (V6.1)

## EINGABEN

- `id` Identifikator der Musik, die gespielt werden soll
- `table` optional: Tabellenargument für weitere Optionen (V4.5)

## BEISPIEL

Siehe [Abschnitt 47.30 \[OpenMusic\]](#), [Seite 1045](#).

## 47.35 PlaySample

### BEZEICHNUNG

`PlaySample` – spielt ein Sample (V1.5)

### ÜBERSICHT

`PlaySample(id[, table], ...)`



**BESCHREIBUNG**

Beginnt das in `id` angegebene Sample abzuspielen. Sie können Samples entweder mit dem Befehl `LoadSample()` laden oder mit der Präprozessor-Anweisung `@SAMPLE` vorladen. Ab Hollywood 2.0 können Sie auch die Tabelle `table` als zweites Argument übergeben, die Parameter für die Wiedergabe des Samples definiert. Die Tabelle kann folgende Felder enthalten:

- Times:** Dieses Feld kann verwendet werden, um festzulegen, wie oft das Sample gespielt werden soll. Der Standardwert ist 1, was bedeutet, dass das Sample einmal abgespielt wird. Wenn Sie das Sample als Endlosschleife laufen lassen wollen, setzen **Times** auf 0.
- Volume:** Setzen Sie dieses Feld auf die gewünschte Wiedergabelautstärke. Sie können Zahlen von 0 bis 64 übergeben. Wenn nichts angegeben wurde, wird die Standardlautstärke des Samples verwendet werden. (Sie können die Standardlautstärke eines Samples ändern, indem Sie den Befehl `SetVolume()` benutzen).
- Pitch:** Damit setzen Sie die gewünschte Tonhöhe. Dieser Wert muss in Hertz übergeben werden. Wenn nichts angegeben wurde, wird die Standardtonhöhe des Samples verwendet werden (Sie können die Standardtonhöhe eines Samples ändern, indem Sie den Befehl `SetPitch()` verwenden).
- Time:** Dieses Feld kann verwendet werden, um festzulegen, wie lange das Sample gespielt werden soll. Hollywood wird das Sample so lange abspielen, bis die Zeit in **Time** abgelaufen ist. **Time** muss in Millisekunden angegeben werden. Dieser und der Tag **Times** schließen sich gegenseitig aus.
- Panning:** Mit diesem Feld können sie festlegen, wo ein Sample gespielt werden soll (Balance). Der Bereich geht von 0 bis 255, wobei 0 bedeutet, dass das Sample nur über die linke Box ausgegeben wird, bei 128 werden beide Lautsprecher gleichmäßig benutzt und 255 bedeutet, dass nur über die rechte Box ausgegeben wird. Wenn nichts angegeben wurde, werden die Standardeinstellungen des Samples verwendet werden (Sie können die Standardbalance des Samples ändern, indem Sie den Befehl `SetPanning()` aufrufen). (V4.5)
- Channel:** Hier können Sie den Kanal für die Wiedergabe des Samples einstellen. In der Standardeinstellung wählt `PlaySample()` automatisch einen freien Kanal und wird fehlschlagen, wenn es keinen freien Kanal mehr hat. Um dieses Verhalten zu ändern, können Sie dieses Feld verwenden. Wenn ein Kanal angegeben wurde, wird es immer die Wiedergabe auf dem hier angegebenen Kanal erzwingen. Wenn der Kanal bereits abgespielt wird, wird er zunächst gestoppt werden. (V6.1)

Ebenfalls neu ab Hollywood 2.0 ist die Möglichkeit, mehrere Samples auf einmal, mit dem Aufruf `PlaySample()` abzuspielen. Wiederholen Sie einfach die Argumentenliste so oft, wie Sie möchten und `PlaySample()` wird alle angegebenen Samples zusammen spielen - perfekt synchronisiert. Bitte beachten Sie, dass für jedes weitere Sample dort auch ein zusätzliches optionales Argument zur Verfügung steht, dass entweder die Anzahl der Male für das Abspielen des Samples angibt oder es ist eine Tabelle, die weitere Attribute für die Samplewiedergabe enthält. Siehe oben, für alle Möglichkeiten.

**EINGABEN**

<b>id</b>	Identifikator des abzuspielenden Samples
<b>table</b>	eine Tabelle, die Wiedergabeparameter für das Sample enthält (V2.0)
<b>...</b>	Sie können die Argumentenliste so oft wiederholen, wie Sie wollen und können so viele Samples gleichzeitig abspielen, wie Sie möchten

**BEISPIEL**

```
PlaySample(1)
```

Der obige Code beginnt mit der Wiedergabe des Samples 1. Das Sample wird nicht als Schleife abgespielt.

```
PlaySample(1, {Time = 10000})
```

Der obige Code spielt das Sample 1 für genau 10 Sekunden (= 10000 Millisekunden).

```
PlaySample(1, {Times = 2}, 2, {Times = 4}, 3, {Time=5000})
```

Der obige Code spielt Sample 1 zweimal, Sample 2 viermal und Sample 3 wird für 5 Sekunden gespielt. Alle drei Samples werden sofort gestartet.

## 47.36 PlaySubsong

**BEZEICHNUNG**

PlaySubsong – beginnt, einen Subsong eines Musikobjekts zu spielen

**ÜBERSICHT**

```
PlaySubsong(number[, id, table])
```

**BESCHREIBUNG**

Beginnt, die Subsong-Nummer des momentan gespielten Musikobjektes zu spielen. Wenn Sie das optionale Argument **id** weglassen, wird die aktuelle Musik gespielt.

Das optionale Argument **table** kann für weitere Optionen verwendet werden. Dieses Argument unterstützt die gleichen Felder wie der Befehl **PlayMusic()**. Siehe [Abschnitt 47.34 \[PlayMusic\]](#), [Seite 1047](#), für Details.

Bitte beachten Sie, dass nur einige Musikformate Subsongs unterstützen. Zum Beispiel können alte Tracker-Modul-Formate oft mehrere Subsongs enthalten. Wenn ein Protracker-Modul verwendet wird, wird dieser Befehl an die angegebene Songposition springen.

**EINGABEN**

<b>number</b>	Nummer des zu spielenden Subsongs
<b>id</b>	optional: ID des Musikobjekts, das verwendet wird (Standard ist die derzeit abgespielte Musik) (V5.3)
<b>table</b>	optional: Tabellenargument für weitere Optionen (V5.3)

**BEISPIEL**

```
PlaySubsong(5, 1)
```

Dieses Beispiel beginnt das Protracker-Modul Nummer 1 zu spielen und springt dann zum Subsong Nummer 5.

## 47.37 ResumeModule

### BEZEICHNUNG

ResumeModule – resume the paused module / VERALTET

### ÜBERSICHT

ResumeModule()

### WICHTIGER HINWEIS

Dieser Befehl ist veraltet. Bitte verwenden Sie stattdessen **ResumeMusic()**.

### BESCHREIBUNG

This function resumes the currently paused module. It can be paused with the **PauseModule()**.

### EINGABEN

Keine

## 47.38 ResumeMusic

### BEZEICHNUNG

ResumeMusic – setzt eine angehaltenes Musikobjekt fort (V2.0)

### ÜBERSICHT

ResumeMusic(id)

### BESCHREIBUNG

Dieser Befehl nimmt die Wiedergabe des in **id** angegebenen und unterbrochenes Musikobjekt wieder auf. Sie können die Wiedergabe eines Musikobjekts mit dem Befehl **PauseMusic()** anhalten.

### EINGABEN

**id**            Identifikator des Musikobjekts, dessen Wiedergabe fortgeführt werden soll

## 47.39 SAMPLE

### BEZEICHNUNG

SAMPLE – lädt ein Sample vor (V2.0)

### ÜBERSICHT

@SAMPLE id, filename\$[, table]

### BESCHREIBUNG

Verwenden Sie diese Präprozessor-Anweisung, um ein Sample vorzuladen, den Sie später mit **PlaySample()** abspielen wollen.

Das dritte Argument **table** ist optional. Es ist eine Tabelle, die weitere Möglichkeiten für den Ladevorgang zur Verfügung stellt. Die folgenden Felder können verwendet werden:

- Link:** Setzen Sie dieses Feld auf **False**, wenn Sie dieses Sample nicht in die ausführbare Datei/das Applet einbinden wollen, wenn Sie Ihr Skript kompilieren. Dieses Feld ist standardmäßig auf **True** gesetzt, was bedeutet, dass das Sample mit der ausführbaren Datei/dem Applet beim Kompilieren verknüpft wird.
- Loader:** Mit diesem Tag können Sie einen oder mehrere Formatlademodule angeben, die die Datei laden soll. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehrerer Lademodulen beinhaltet. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)
- Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen sollen. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen enthält. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)
- UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Beispielformate, die auf allen Plattformen unterstützt werden, sind RIFF WAVE, IFF 8SVX, IFF 16SV und Sampleformate, für die Sie ein Plugin haben. Je nach Plattform, auf der Hollywood ausgeführt wird, können mehr Sampleformate unterstützt werden. Zum Beispiel auf Amigakompatiblen Systemen wird Hollywood in der Lage sein, alle Sampleformate über entsprechende Datatypes zu öffnen.

Wenn Sie ein Sample manuell laden möchten, nutzen Sie bitte den Befehl `LoadSample()`.

## EINGABEN

- id** ein Wert, der dieses Sample später im Code identifiziert
- filename\$** die Sampledatei, welche sie vorladen möchten
- table** optional: eine Tabelle mit weiteren Optionen (siehe oben)

## BEISPIEL

```
@SAMPLE 1, "Gunshot.8svx"
```

Der obige Code weist die Samplenummer 1 dem Sample "Gunshot.8svx" zu.

```
@SAMPLE 1, "Sound/Samples/Gunshot.wav", {Link=False}
```

Macht das gleiche wie der Code vorher, aber das Sample wird beim kompilieren nicht eingebunden werden.

## 47.40 SaveSample

### BEZEICHNUNG

SaveSample – speichert ein Sample als Datei ab (V5.0)

### ÜBERSICHT

```
SaveSample(id, f$[, fmt, t])
```

### BESCHREIBUNG

Dieser Befehl speichert das in `id` angegebene Sample in der Datei `f$` ab. Im optionalen Argument `fmt` können Sie das Format angeben, in dem das Sample exportiert werden soll. Derzeit wird nur `#SMPFMT_WAVE` unterstützt, womit das Sample im Format RIFF WAVE gespeichert wird.

Ab Hollywood 10.0 akzeptiert `SaveSample()` ein optionales Tabellenargument, mit dem Sie zusätzliche Argumente an den Befehl übergeben können. Die folgenden Tags werden derzeit vom optionalen Tabellenargument unterstützt:

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, die die angegebene Datei speichern sollen. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen enthält. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

### EINGABEN

<code>id</code>	Identifikator des Samples, welcher abgespeichert wird
<code>f\$</code>	Pfad/Dateiname
<code>fmt</code>	optional: Format, in dem das Sample gespeichert wird (voreingestellt ist <code>#SMPFMT_WAVE</code> )
<code>t</code>	optional: Tabelle mit weiteren Optionen (siehe oben) (V10.0)

### BEISPIEL

```
@SAMPLE 1, "ouch.8svx"
```

```
SaveSample(1, "ouch.wav")
```

Der obige Code lädt ein Sample im Format IFF 8SVX und speichert ihn als RIFF WAVE-Sample.

## 47.41 SeekMusic

### BEZEICHNUNG

SeekMusic – springt an eine bestimmte Position im Musikobjekt (V2.0)

### ÜBERSICHT

SeekMusic(id, pos)

### BESCHREIBUNG

Sie können diesen Befehl verwenden, um die angegebene Position **pos** in der in **id** angegebene Musik zu springen. Das Musikobjekt muss nicht spielen. Wenn die Musik spielt und Sie rufen **SeekMusic()** auf, wird es sofort an die angegebene Position springen. Die Position wird in Millisekunden angegeben. Wenn Sie also an die Position 03.24 springen möchten, würden Sie den Wert 204000 in **pos** angeben, weil  $3 * 60 * 1000 + 24 * 1000 = 204000$  übergeben müssen.

Bitte beachten Sie, dass dieser Befehl nicht mit Protracker-Modulen funktioniert.

### EINGABEN

<b>id</b>	Identifikator des Musikobjekts
<b>pos</b>	neue Position in der Musik

## 47.42 SetChannelVolume

### BEZEICHNUNG

SetChannelVolume – ändert die Lautstärke eines Kanals (V6.1)

### ÜBERSICHT

SetChannelVolume(n, volume)

### BESCHREIBUNG

Dieser Befehl ändert die Lautstärke des in **n** angegebenen Kanals. Die Numerierung der Kanäle beginnt bei 1 und geht bis zu der Anzahl der verfügbaren Kanäle. **volume** kann eine Zahl von 0 (still) bis 64 (volle Lautstärke) sein oder eine Zeichenkette mit einer prozentualen Angabe, z.B. "50%"

### EINGABEN

<b>n</b>	Kanalindex
<b>volume</b>	neue Lautstärke für den Kanal

## 47.43 SetMasterVolume

### BEZEICHNUNG

SetMasterVolume – stellt die Gesamtlautstärke ein / VERALTET (V1.5)

### ÜBERSICHT

SetMasterVolume(vol)

**BESCHREIBUNG**

Bitte beachten Sie: Dieser Befehl ist veraltet und wird früher oder später entfernt. Darum sollten Sie besser `SetVolume()` für Samples und `SetMusicVolume()` für Musikdateien und Protracker-Module benutzen.

Dieser Befehl stellt die Gesamtlautstärke auf den in `vol` angegebenen Wert ein. Mit diesem Befehl können Sie "Ton ausblenden" oder "Ton einblenden" realisieren.

Ab Hollywood 2.0 kann in `vol` auch eine Zeichenfolge mit prozentualer Angabe übergeben werden, z.B. "50%".

**EINGABEN**

`vol`            neue Gesamtlautstärke (Bereich von 0 bis 64 oder prozentuale Angabe)

**BEISPIEL**

```
For k = 64 To 0 Step -5
    SetMasterVolume(k)
Next
```

Der obige Code blendet alle abspielenden Sounds aus, indem die Gesamtlautstärke geändert wird.

## 47.44 SetMusicVolume

**BEZEICHNUNG**

`SetMusicVolume` – ändert die Lautstärke eines Musikobjekts (V2.0)

**ÜBERSICHT**

`SetMusicVolume(id, volume)`

**BESCHREIBUNG**

Dieser Befehl ändert die Lautstärke des in `id` angegebenen Musikobjekts. Falls die Musik gerade spielt, ändert dieser Befehl die Lautstärke on-the-fly, womit Sie dies für Ausblendungen der Musik verwenden können usw.

Ab Hollywood 2.0 kann dem Argument `volume` auch eine Zeichenfolge mit einer prozentualen Angabe übergeben werden, z.B. "50%".

**EINGABEN**

`id`            Identifikator des Musikobjekts

`volume`       neue Lautstärke des Musikobjekts (Bereich: 0=still bis 64=volle Lautstärke oder prozentuale Angabe)

## 47.45 SetPanning

**BEZEICHNUNG**

`SetPanning` – legt die Sampleausrichtung (Balance) fest (V1.9)

**ÜBERSICHT**

`SetPanning(id, pan)`

**BESCHREIBUNG**

Mit diesem Befehl können Sie angeben, wo das Sample mit dem Identifikator `id` gespielt werden soll. Der Parameter `pan` liegt zwischen 0 und 255, wobei 0 bedeutet, dass das Sample nur über die linke Box ausgegeben wird und 255, dass es nur über die rechte Box ausgegeben wird. Wenn Sie 128 angeben, dann wird das Sample über beide Boxen gleichmäßig ausgegeben. Dies ist auch die Standardeinstellung für jedes Sample.

Sie können auch die speziellen Konstanten `#LEFT`, `#CENTER` und `#RIGHT` angeben, welche auf 0, 128 und 255 jeweils korrespondieren.

Wenn das Sample gerade spielt, wird die Ausrichtung sofort ("on-the-fly") geändert, was für nette Effekte benutzt werden kann.

**EINGABEN**

<code>id</code>	ID des zu benutzenden Samples
<code>pan</code>	neuer Ausrichtungswert (von 0 bis 255)

**BEISPIEL**

```
SetPanning(1, 255)
PlaySample(1)
```

Der obige Code spielt das Sample 1 nur durch den rechten Lautsprecher.

## 47.46 SetPitch

**BEZEICHNUNG**

`SetPitch` – stellt die Tonhöhe eines Samples ein

**ÜBERSICHT**

```
SetPitch(id,pitch)
```

**BESCHREIBUNG**

Dieser Befehl verstellt die Tonhöhe des durch `id` angegebenen Samples. Wenn das Sample gerade spielt, wird die Tonhöhe sofort ("on-the-fly") verstellt, was für einige nette Soundeffekte verwendet werden kann. Sie müssen den Wert in Hertz (Hz) angeben.

**EINGABEN**

<code>id</code>	Identifikator des Samples
<code>pitch</code>	neue Tonhöhe des Samples in Hertz

## 47.47 SetVolume

**BEZEICHNUNG**

`SetVolume` – stellt die Lautstärke eines Samples ein

**ÜBERSICHT**

```
SetVolume(id,volume)
```



**BESCHREIBUNG**

Dieser Befehl stellt die Lautstärke des durch `id` angegebenen Samples ein. Wenn das Sample gerade spielt, wird die Lautstärke "on-the-fly" verstellt, was für einige nette Soundeffekte verwendet werden kann.

Ab Version 2.0 kann bei `volume` eine prozentige Angabe in Form einer Zeichenkette angegeben werden, z.B. "50%".

**EINGABEN**

<code>id</code>	Identifikator eines Samples
<code>volume</code>	neue Lautstärke des Samples (Wertebereich: 0=Still bis 64=volle Lautstärke oder prozentige Angabe)

**BEISPIEL**

```
LoadSample(1, "Sound/Samples/GroovyLoop.wav")
PlaySample(1)
Wait(100)
For k = 64 To 0 Step -1
    SetVolume(1,vol)
Next
```

Dieser Code lädt das Sample "dh0:Sound/Samples/GroovyLoop.8svx", spielt es ab, wartet 2 Sekunden und blendet es dann langsam aus.

## 47.48 StopChannel

**BEZEICHNUNG**

StopChannel – stoppt das Abspielen über diesen Kanal (V6.1)

**ÜBERSICHT**

```
StopChannel(n)
```

**BESCHREIBUNG**

Dieser Befehl stoppt das Abspielen über den in `n` angegebenen Kanal. Der Kanalbereich geht von 1 bis zu der Anzahl der verfügbaren Kanäle.

**EINGABEN**

<code>n</code>	Index des Kanals, der gestoppt wird
----------------	-------------------------------------

## 47.49 StopModule

**BEZEICHNUNG**

StopModule – stop the currently playing module / VERALTET

**ÜBERSICHT**

```
StopModule()
```

**WICHTIGER HINWEIS**

Dieser Befehl ist veraltet. Bitte verwenden Sie stattdessen `StopMusic()`.

**BESCHREIBUNG**

Stops the module that is currently playing and frees all used audio channels.

**EINGABEN**

Keine

## 47.50 StopMusic

**BEZEICHNUNG**

StopMusic – stoppt ein abgespieltes Musikobjekt (V2.0)

**ÜBERSICHT**

StopMusic(id)

**BESCHREIBUNG**

Dieser Befehl stoppt das in `id` angegebene Musikobjekt. Dies wird nicht fehlschlagen, wenn die angegebene Musik nicht abgespielt wird.

**EINGABEN**

`id`            Identifikator des Musikobjekts, das gestoppt wird

## 47.51 StopSample

**BEZEICHNUNG**

StopSample – stoppt ein Sample

**ÜBERSICHT**

StopSample(id)

**BESCHREIBUNG**

Stoppt das Sample mit der angegebenen `id` und gibt alle dafür benutzten Kanäle frei. Dies wird nicht fehlschlagen, wenn das angegebene Sample zur Zeit nicht abgespielt wird.

**EINGABEN**

`id`            ID des Samples, welches gestoppt werden soll

## 47.52 WaitMusicEnd

**BEZEICHNUNG**

WaitMusicEnd – hält das Skript an, bis die Musik beendet ist (V10.0)

**ÜBERSICHT**

WaitMusicEnd(id)

**BESCHREIBUNG**

Dieser Befehl hält die Ausführung des Skripts an, bis die durch `id` angegebene Musik beendet ist. Danach wird die Ausführung Ihres Skripts fortgesetzt. Wenn Sie etwas tun müssen, während Ihre Musik abgespielt wird, verwenden Sie den Befehl `IsMusicPlaying()` in Verbindung mit einer Schleife.

**EINGABEN**

`id` ID einer Musik, die gerade gespielt wird

## 47.53 WaitPatternPosition

**BEZEICHNUNG**

`WaitPatternPosition` – hält das Programm an, bis das Modul eine bestimmte Pattern-Position erreicht hat

**ÜBERSICHT**

`WaitPatternPosition(pos)`

**BESCHREIBUNG**

Dieser Befehl hält die Ausführung des Programms an, bis das momentan gespielte Modul die angegebene Position `pos` erreicht hat. Vor Benutzung dieses Befehls müssen Sie `PlayModule()` aufrufen. Das ist sinnvoll, um Ihre Applikation an den zeitlichen Ablauf der Musik anzupassen.

**EINGABEN**

`pos` Pattern-Position, auf die gewartet werden soll

**BEISPIEL**

```
PlayModule(1)
WaitPatternPosition(63)
```

Obiges Beispiel beginnt, Modul Nummer 1 abzuspielen und wartet dann, bis das Ende des ersten Patterns erreicht ist.

## 47.54 WaitSampleEnd

**BEZEICHNUNG**

`WaitSampleEnd` – hält das Programm an, bis das Sample fertig abgespielt ist

**ÜBERSICHT**

`WaitSampleEnd(id)`

**BESCHREIBUNG**

Dieser Befehl unterbricht die Ausführung des Programms, bis das durch `id` angegebene Sample zu Ende gespielt ist. Danach wird die Abarbeitung Ihres Skripts fortgesetzt. Falls Sie etwas tun müssen, solange Ihr Sample abgespielt wird, benutzen Sie den `IsSamplePlaying()` Befehl in Verbindung mit einer Schleife.

**EINGABEN**

`id` ID des Samples, das momentan abgespielt wird

**BEISPIEL**

Siehe [Abschnitt 47.26 \[LoadSample\]](#), Seite 1040.

## 47.55 WaitSongPosition

### BEZEICHNUNG

WaitSongPosition – hält das Programm an, bis das Modul eine bestimmte Songposition erreicht hat

### ÜBERSICHT

WaitSongPosition(pos)

### BESCHREIBUNG

Dieser Befehl hält die Ausführung des Programms an, bis das momentan gespielte Modul die angegebene Songposition **pos** erreicht hat. Vor Benutzung dieses Befehls müssen Sie **PlayModule()** aufrufen. Das ist sinnvoll, um Ihre Applikation an den zeitlichen Ablauf der Musik anzupassen.

### EINGABEN

**pos**            Songposition, auf die gewartet werden soll

### BEISPIEL

PlayModule(1)

WaitSongPosition(2)

Obiges Beispiel beginnt Modul Nummer 1 abzuspielen und wartet dann auf Songposition 2.

## 48 Speicherblockbibliothek

### 48.1 AllocMem

#### BEZEICHNUNG

AllocMem – weist einen neuen Speicherblock zu (V2.0)

#### ÜBERSICHT

```
[id] = AllocMem(id, size)
```

#### BESCHREIBUNG

Dieser Befehl weist einen neuen Speicherblock der angegebenen Größe zu und macht ihn unter *id* verfügbar. Falls Sie hingegen **Nil** als *id* übergeben haben, wählt **AllocMem()** automatisch eine ID und gibt sie zurück. Der Arbeitsspeicher wird nicht initialisiert und ist daher mit zufälligen Daten gefüllt. Wenn Sie diese auf Null initialisieren möchten, verwenden Sie den Befehl **FillMem()**.

#### EINGABEN

*id* ID für den Speicherblock oder **Nil** für die automatische ID-Zuweisung  
*size* Größe für den Speicherblock in Byte

#### RÜCKGABEWERTE

*id* optional: ID des Speicherblocks; Wird nur zurückgegeben, wenn Sie **Nil** in *id* eingetragen haben (siehe oben)

### 48.2 AllocMemFromPointer

#### BEZEICHNUNG

AllocMemFromPointer – initialisiert den Speicherblock vom Zeiger (V6.0)

#### ÜBERSICHT

```
[id] = AllocMemFromPointer(id, ptr, size)
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen Zeiger des Typs **#LIGHTUSERDATA** in einen Speicherblock zu konvertieren, den Sie unter Verwendung des Speicherblockbefehls lesen und schreiben können. Das neue Speicherblockobjekt wird unter der *id* verfügbar gemacht, oder wenn Sie **Nil** als *id* angeben, wählt **AllocMemFromPointer()** automatisch eine ID aus und gibt sie zurück.

Beachten Sie, dass **AllocMemFromPointer()** keine lokale Kopie des Arbeitsspeichers erstellt, auf den durch *ptr* verwiesen wird. Es wird nur ein Containerobjekt zugewiesen, damit Sie mit den Speicherblockbefehlen auf die Speicherdaten zugreifen können. Das Argument *size* wird nur verwendet, um Lese- oder Schreiboperationen außerhalb der Speicherblockgrenzen zu verhindern. Wenn Sie die Größe des Speicherblocks nicht kennen, können Sie auch im Argument *size* 0 übergeben. In diesem Fall verbietet Hollywood keine Lese- und Schreiboperationen auf diesem Speicherblockobjekt.

Seien Sie gewarnt, dass dies ein gefährlicher Befehl ist und sollte nur von Programmierern verwendet werden, die wissen, was sie tun. Lesen oder Schreiben in nicht zugewiesenen Arbeitsspeicher kann leicht zum Abstürzen Ihres Programms führen.

#### EINGABEN

**id** ID für den Speicherblock oder **Nil** für die automatische ID-Zuweisung  
**ptr** **#LIGHTUSERDATA**, der auf einen Speicherblock zeigt  
**size** Größe des Speicherblocks in Bytes oder 0, wenn Sie die Größe nicht kennen

#### RÜCKGABEWERTE

**id** optional: ID des Speicherblocks; Wird nur zurückgegeben, wenn Sie **Nil** in **id** eingetragen haben (siehe oben)

## 48.3 AllocMemFromVirtualFile

#### BEZEICHNUNG

**AllocMemFromVirtualFile** – initialisiert den Speicherblock aus einer virtuellen Datei (V6.1)

#### ÜBERSICHT

```
[id] = AllocMemFromVirtualFile(id, vf$)
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um auf den Rohspeicherinhalt einer virtuellen Zeichenkettendatei zuzugreifen, die mit **DefineVirtualFileFromString()** erstellt wurde. Das neue Speicherblockobjekt wird unter der **id** zur Verfügung gestellt, oder wenn Sie **Nil** als **id** angeben haben, wählt **AllocMemFromVirtualFile()** automatisch eine ID aus und gibt sie zurück.

Beachten Sie, dass **AllocMemFromVirtualFile()** keine lokale Kopie des Arbeitsspeichers machen wird, die zur virtuellen Zeichenkettendatei gehört. Es wird nur einem Containerobjekt zugewiesen, so dass Sie über die Speicherblockbefehle auf die Speicherdaten zugreifen können.

Beachten Sie außerdem, dass beim Umgang mit schreibbaren virtuellen Zeichenketten-dateien ihre Speicherdarstellung mit jeder einzelnen Schreiboperation, die an der virtuellen Zeichenkettendatei ausgeführt wird, geändert werden kann. Somit ist es nicht sicher, auf den Arbeitsspeicher der virtuellen Zeichenkettendatei durch einen Container mit **AllocMemFromVirtualFile()** zuzugreifen, nachdem eine Schreiboperation in dieser virtuellen Zeichenkettendatei durchgeführt wurde. Stattdessen müssen Sie nach jedem Schreibvorgang einen neuen Container erstellen und den alten mit **FreeMem()** zuerst löschen. Alles andere wird früher oder später zum Absturz führen.

Beachten Sie auch, dass es verboten ist, in den von diesem Befehl zugewiesenen Speicherblock zu schreiben, es sei denn, die virtuelle Zeichenkettendatei wurde als schreibbar erstellt.

Seien Sie gewarnt, dass dies ein gefährlicher Befehl ist und sollte nur von Programmierern verwendet werden, die wissen, was sie tun. Lesen oder Schreiben in nicht zugeteilten Arbeitsspeicher kann Ihr Programm leicht zum abstürzen bringen und alle möglichen Probleme verursachen.

**EINGABEN**

<code>id</code>	ID für den Speicherblock oder <code>Nil</code> für die automatische ID-Zuweisung
<code>vf\$</code>	Virtuelle Zeichenkette, welche durch <code>DefineVirtualFileFromString()</code> zugeordnet wurde

**RÜCKGABEWERTE**

<code>id</code>	optional: ID des Speicherblocks; Wird nur zurückgegeben, wenn Sie <code>Nil</code> in <code>id</code> eingetragen haben (siehe oben)
-----------------	--

## 48.4 CopyMem

**BEZEICHNUNG**

`CopyMem` – kopiert Daten zwischen Speicherblöcken (V2.0)

**ÜBERSICHT**

`CopyMem(src, dst, size[, src_offset, dst_offset])`

**BESCHREIBUNG**

Dieser Befehl kopiert die Anzahl Bytes in `size` aus dem Speicherblock mit der ID `src` in den Block mit der ID `dst`. Mit dem optionalen Argument `src_offset` können Sie den Startpunkt fürs Auslesen im Quellblock festlegen. Mit dem zweiten optionalen Argument `dst_offset` können Sie den Startpunkt im Zielblock definieren.

Bitte beachten Sie, dass `src` und `dst` nicht die gleichen Blöcke sein dürfen.

**EINGABEN**

<code>src</code>	Quellspeicherblock zum auslesen von Daten
<code>dst</code>	Zielspeicherblock, in dem die Daten kopiert werden
<code>size</code>	Größe in Byte der Daten, die kopiert werden
<code>src_offset</code>	optional: von wo aus im Quellblock mit dem Lesen begonnen wird (standardmäßig auf 0 = Anfang des Blocks)
<code>dst_offset</code>	optional: von wo aus im Zielblock geschrieben wird (standardmäßig auf 0 = Anfang des Blocks)

## 48.5 DecreasePointer

**BEZEICHNUNG**

`DecreasePointer` – verringert einen Zeiger (V6.0)

**ÜBERSICHT**

`ptr = DecreasePointer(ptr, amount)`

**BESCHREIBUNG**

Dieser Befehl verringert den in `ptr` angegebenen Zeiger des Typs `#LIGHTUSERDATA` durch die Menge der Bytes, die im Argument 2 `amount` angegeben sind. Da Sie keine Zeiger

in Hollywood verwenden sollten, ist dieser Befehl wirklich nur beim Debuggen von Code nützlich oder bei einigen experimentellen Sachen, die Sie mit Hollywood machen.

Um einen Zeiger zu erhöhen, können Sie den Befehl `IncreasePointer()` benutzen. Siehe [Abschnitt 48.11 \[IncreasePointer\]](#), [Seite 1067](#), für Details.

#### EINGABEN

`ptr` Zeiger, der als `#LIGHTUSERDATA`-Variabele übergeben wird

`amount` Anzahl der zu verringernden Bytes

#### RÜCKGABEWERTE

`ptr` Neuer Zeiger des Typs `#LIGHTUSERDATA`

## 48.6 DumpMem

#### BEZEICHNUNG

DumpMem – gibt einen Speicherblock aus (V2.0)

#### ÜBERSICHT

`DumpMem(id[, len, offset])`

#### BESCHREIBUNG

Dieser Befehl gibt den Inhalt des durch `id` angegebenen Speicherblocks einer Debug-Vorrichtung aus (normalerweise ein Konsolenfenster). Als optionales Argument können Sie die Länge in Bytes in `len` angeben, die ausgegeben werden soll. Wenn es weggelassen wird, wird der ganze Block ausgegeben. Das optionale Argument `offset` kann verwendet werden, die Ausgabe um einen Versatz innerhalb des Blocks zu starten.

Wegen der Rohdaten, die normalerweise in den Speicherblöcken sind, wird Hollywood einen Hex-Dump einschließlich ASCII-Schreibweise (wenn möglich) erstellen. Das Format ist folgender:

```
xxxxxxx: bb bb bb bb bb bb bb bb bb bb bb bb bb bb cccccccccccccccc
```

x: Versatz in hexadezimaler Schreibweise

b: 16 Bytes pro Zeile

c: Die 16 Bytes in ASCII-Schreibweise oder '.', wenn das Zeichen nicht grafisch ist

#### EINGABEN

`id` ID des Speicherblocks, der ausgegeben wird

`len` optional: Länge in Byte (Standardwert 0, was bedeutet, dass der komplette Block ausgegeben wird)

`offset` optional: von wo aus in dem Block die Ausgabe gestartet werden soll (Standardwert ist 0 = Anfang des Blocks)



## 48.7 FillMem

### BEZEICHNUNG

FillMem – füllt einen Speicherblock (V2.0)

### ÜBERSICHT

FillMem(id, val, size[, offset, type])

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen Teil oder den gesamten Speicherblock mit einem vorgegebenen Wert zu füllen. Dabei gibt **id** den zu verwendenden Speicherblock an, **val** ist der zu füllende Wert und **size** gibt die Größe in Byte für den Füllvorgang an. Sie können das optionale Argument **offset** zur Feinabstimmung des Füllvorgangs verwenden, in dem Sie den Startpunkt vorgeben (in Bytes). Das optionale Argument **type** gibt den Typ des Werts an und kann **#BYTE** (1 Byte), **#SHORT** (2 Bytes) oder **#INTEGER** (4 Bytes) sein.

Wenn Sie **#SHORT** oder **#INTEGER** als Fülltyp benutzen, muss die Argumentgröße ein Vielfaches von 2 oder 4 sein. Auch der Versatz, falls angegeben, muss ein Vielfaches von 2 oder 4 sein.

### EINGABEN

<b>id</b>	zu verwendender Speicherblock
<b>val</b>	Wert, mit dem der Block gefüllt wird
<b>size</b>	Größe des Füllvorgangs in Byte; muss ein vielfaches von 2 oder 4 sein, wenn der Typ <b>#SHORT</b> oder <b>#INTEGER</b> ist
<b>offset</b>	optional: Versatz im Block, an dem die Füllung beginnen soll (standardmäßig 0, was den Anfang des Blocks bedeutet); müssen ein Vielfaches von 2 oder 4 sein, wenn der Typ <b>#SHORT</b> oder <b>#INTEGER</b> ist
<b>type</b>	optional: Typ des Wertes; derzeit unterstützt werden <b>#BYTE</b> , <b>#SHORT</b> und <b>#INTEGER</b> (standardmäßig auf <b>#BYTE</b> )

### BEISPIEL

```
AllocMem(1, 65536)
```

```
FillMem(1, 0, 65536)
```

Weist einen Block von 64kb zu und initialisiert ihn mit 0.

## 48.8 FreeMem

### BEZEICHNUNG

FreeMem – gibt einen Speicherblock frei (V2.0)

### ÜBERSICHT

FreeMem(id)

### BESCHREIBUNG

Dieser Befehl gibt den Arbeitsspeicher frei, der von dem durch **id** festgelegten Block belegt wird.

**EINGABEN**

**id** Speicherblock, der freizugeben ist

**48.9 GetMemPointer****BEZEICHNUNG**

GetMemPointer – gibt die Rohadresse des Speicherblocks zurück (V6.0)

**ÜBERSICHT**

```
ptr = GetMemPointer(id[, offset])
```

**BESCHREIBUNG**

Dieser Befehl gibt die Rohadresse des in **id** übergebenen Speicherblocks zurück. Optional können Sie im Argument **offset** einen Versatz in Byte angeben, der der Adresse hinzugefügt werden soll, bevor sie zurückgegeben wird. Der Zeiger wird als Variable vom Typ **#LIGHTUSERDATA** zurückgegeben. Er bleibt gültig, bis Sie **FreeMem()** auf das Speicherblockobjekt aufrufen.

Dieser Befehl ist nur in Verbindung mit Befehlen sinnvoll, die Parameter vom Typ **#LIGHTUSERDATA** erwarten. Zur Zeit gibt es keine Hollywood-Befehle, welche **#LIGHTUSERDATA**-Parameter verarbeiten können. Aber Plugins können **#LIGHTUSERDATA**-Parameter für bestimmte Aufgaben verwenden, falls Tabellen zu langsam sind.

**EINGABEN**

**id** ID des Speicherblocks, dessen Adresse zurückgegeben werden soll

**offset** optional: Versatz in Bytes, die der Adresse hinzugefügt wird (Standardwert 0)

**RÜCKGABEWERTE**

**ptr** Zeiger auf die Rohdaten des angegebenen Speicherblocks

**48.10 GetMemString****BEZEICHNUNG**

GetMemString – gibt eine Zeichenkette vom Speicherblock zurück (V7.1)

**ÜBERSICHT**

```
s$ = GetMemString(id[, offset, length])
```

**BESCHREIBUNG**

Dieser Befehl liefert die in **length** angegebene Anzahl Bytes ab dem in **offset** angegebenen Versatz aus dem in **id** genannten Speicherblock. Sowohl der Parameter **offset** als auch **length** müssen in Bytes angegeben werden. Wenn nichts übergeben wurde, wird **offset** auf 0 (d.h. der Anfang des Speicherblocks) und **length** ebenfalls auf 0 gesetzt, was bedeutet, dass alle verbleibenden Bytes ab dem angegebenen Versatz zurückgegeben werden.

Beachten Sie, dass Zeichenketten von Hollywood auch Binärdaten enthalten können, so dass die von `GetMemString()` zurückgegebene Zeichenkette nicht unbedingt eine gültige UTF-8-Zeichenfolge ist, sondern die aus dem angegebenen Speicherblock kopierten Binärdaten enthält.

#### EINGABEN

<code>id</code>	ID des Speicherblocks
<code>offset</code>	optional: Versatz in Bytes, der festlegt, wo mit dem Lesen von Bytes begonnen werden soll (voreingestellt ist 0)
<code>length</code>	optional: Anzahl der Bytes zum Auslesen oder 0 zum Auslesen aller verbleibenden Bytes im Speicherblock (voreingestellt ist 0)

#### RÜCKGABEWERTE

<code>s\$</code>	Inhalt des angegebenen Speicherblockbereichs
------------------	--

## 48.11 IncreasePointer

#### BEZEICHNUNG

IncreasePointer – erhöht einen Zeiger (V6.0)

#### ÜBERSICHT

```
ptr = IncreasePointer(ptr, amount)
```

#### BESCHREIBUNG

Dieser Befehl erhöht den in `ptr` angegebenen Zeiger des Typs `#LIGHTUSERDATA` durch die in `amount` angegebenen Bytes. Da Sie keine Zeiger in Hollywood verwenden sollten, ist dieser Befehl nur nützlich, wenn Sie Code debuggen oder einige experimentelle Sachen mit Hollywood machen.

Um einen Zeiger zu verringern, können Sie die den Befehl `DecreasePointer()` benutzen. Siehe [Abschnitt 48.5 \[DecreasePointer\]](#), Seite 1063, für Details.

#### EINGABEN

<code>ptr</code>	Zeiger, der als <code>#LIGHTUSERDATA</code> -Variable angegeben wird
<code>amount</code>	Anzahl der zu erhöhenden Bytes

#### RÜCKGABEWERTE

<code>ptr</code>	Neuer Zeiger des Typs <code>#LIGHTUSERDATA</code>
------------------	---

## 48.12 MemToTable

#### BEZEICHNUNG

MemToTable – gibt den Speicherblockinhalt als Tabelle zurück (V6.0)

#### ÜBERSICHT

```
t = MemToTable(id, type[, table])
```

## BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Inhalt eines Speicherblocks als Tabelle (oder einen Teil eines Speicherblocks) zurückzugeben. Das Argument **type** gibt den Datentyp der Elemente an, die aus dem Speicherblock gelesen werden sollen. Das kann **#BYTE** (1 Byte), **#SHORT** (2 Bytes), **#INTEGER** (4 Bytes), **#FLOAT** (4 Bytes) oder **#DOUBLE** (8 Bytes) sein.

Mit dem optionalen Argument **table** können weitere Parameter für die Operation gesetzt werden. Folgende Tabellenfelder werden derzeit erkannt:

- Items:** Die Anzahl der zu lesenden Elemente aus dem Speicherblock. Beachten Sie, dass dies nicht eine Größe in Bytes ist, sondern eine Elementanzahl. Wenn Sie also das Argument **type** auf **#INTEGER** und **Items** auf 4 setzen, werden 16 Byte aus dem Speicherblock gelesen. Standardmäßig werden alle Elemente im Speicherblock gelesen.
- Offset:** Dieses Feld kann verwendet werden, um einen Versatz in Byte innerhalb des Arbeitsspeichers festzulegen, wo **MemToTable()** Elemente zu lesen beginnt. Die Voreinstellung ist 0, womit vom Anfang des Speicherblocks gelesen wird.
- Signed:** Wenn dieser Tag auf **True** gesetzt ist, wird **MemToTable()** alle Elemente des Typs **#BYTE**, **#SHORT**, und **#INTEGER** als Werte mit Vorzeichen behandeln. Vorgabe ist **False**.
- EndianSwitch:** Wenn dieser Tag auf **True** gesetzt ist, wird **MemToTable()** die Byte-Reihenfolge für alle Multi-Byte-Datentypen wechseln. Dies kann nützlich sein, wenn Sie zwischen Big- und Little-Endian-Werte konvertieren müssen. Vorgabe ist **False**.

Um eine Tabelle wieder in einen Speicherblock zu konvertieren, verwenden Sie den Befehl **TableToMem()**. Siehe [Abschnitt 48.16 \[TableToMem\]](#), Seite 1071, für Details.

## EINGABEN

- id** ID des Speicherblocks
- type** Datentyp der zu lesenden Elemente (siehe oben)
- table** optional: weitere Tabellen Parameter (siehe oben)

## RÜCKGABEWERTE

- t** eine Tabelle, die so viele Elemente enthält, wie im Tag **Items** angegeben ist

## BEISPIEL

```
AllocMem(1, 26)
For Local k = 0 To 25 Do Poke(1, k, 'A' + k, #BYTE)
Local t = MemToTable(1, #BYTE)
For Local k = 0 To 25 Do Print(Chr(t[k]))
```

Dies gibt das Alphabet einer Speicherblockquelle aus.

## 48.13 Peek

### BEZEICHNUNG

Peek – betrachtet einen Speicherblock (V2.0)

### ÜBERSICHT

```
val = Peek(id, offset[, type, len, endian])
```

### BESCHREIBUNG

Mit diesem Befehl können Sie den in `id` angegebenen Speicherblock mit dem in `offset` angegebenen Versatz betrachten. `type` gibt den Datentyp an, den Sie betrachten möchten. Das kann `#BYTE`, `#SHORT`, `#INTEGER`, `#FLOAT`, `#DOUBLE` oder `#STRING` sein. `#BYTE` wird ein Byte aus dem Block lesen, `#SHORT` liest zwei Bytes, `#INTEGER` sowie `#FLOAT` vier Bytes, `#DOUBLE` acht Bytes und `#STRING` liest aus dem Speicherblock, bis er auf ein nicht-grafisches oder ein Nullzeichen trifft.

Ab Hollywood 2.5 können Sie das optionalen Argument `len` angeben. Dieses Argument kann nur beim Typ `#STRING` verwendet werden. Wenn angegeben, liest `Peek()` genau `len` Bytes aus dem angegebenen Speicherblock und gibt es als Zeichenkette zurück. Sie können dies verwenden, um Rohdaten aus Speicherblöcken zu lesen, da `Peek()` bei nicht-grafischen oder Nullzeichen nicht mehr endet, wenn `len` angegeben wird. Wenn `len` 0 ist, was der Standardeinstellung entspricht, liest `Peek()` Bytes, bis es ein nicht-grafisches oder Nullzeichen findet.

Ab Hollywood 6.0 gibt es einen neuen Parameter `endian`, mit dem Sie die Byte-Reihenfolge angeben können, die beim Lesen der Daten aus dem Speicherblock verwendet werden soll. Dies kann auf die folgenden Typen festgelegt werden:

#### #BIGENDIAN:

Big-Endian-Byte-Reihenfolge, MSB zuerst. Dies ist die Standardeinstellung.

#### #NATIVEENDIAN:

Native-Endian-Byte-Reihenfolge. Wenn Sie diesen Typ verwenden, hängt die Byte-Reihenfolge von der Standard-Byte-Reihenfolge auf dem Hostsystem ab, d.h. Big Endian auf Big-Endian-Systemen, Little Endian auf Little-Endian-Systemen. Seien Sie vorsichtig mit diesem Typ, da er die Portabilität einschränkt.

#### #LITTLEENDIAN:

Little-Endian-Byte-Reihenfolge, zuerst LSB. (V8.0)

### EINGABEN

<code>id</code>	ID des zu verwendenden Speicherblocks
<code>offset</code>	Versatz, ab dem gelesen wird (in Bytes)
<code>type</code>	optional: Datentyp, der betrachtet wird (standardmäßig auf <code>#INTEGER</code> )
<code>len</code>	optional: Anzahl der zu lesenden Bytes (funktioniert nur in Verbindung mit dem Typ <code>#STRING</code> ) (Voreinstellung 0, wobei gelesen wird bis der Befehl auf ein nicht-grafisches oder Nullzeichen trifft) (V2.5)
<code>endian</code>	optional: Reihenfolge der Bytes (Standardeinstellung ist <code>#BIGENDIAN</code> ) (V6.0)

**RÜCKGABEWERTE**

**val**            Inhalt des Speicherblocks beim angegebenen Versatz; kann ein Wert oder eine Zeichenfolge sein (wenn **type** auf **#STRING** gesetzt wurde)

**48.14 Poke****BEZEICHNUNG**

Poke – schreibt in einen Speicherblock (V2.0)

**ÜBERSICHT**

`Poke(id, offset, val[, type, endian])`

**BESCHREIBUNG**

Dieser Befehl schreibt den in **val** angegebenen Wert oder Zeichenfolge in den Speicherblock mit der **id** im angegebenen Versatz **offset**. **type** gibt den Typ von **val** an (Standardwert ist **#INTEGER**). Sie können auch folgende Typen verwenden: **#BYTE** (1 Byte), **#SHORT** (2 Byte), **#FLOAT** (4 Byte Einfach-Präzisions-Fließkommazahl), **#DOUBLE** (8 Byte Doppel-Präzisions-Fließkommazahl) oder **#STRING**. Das Einfügen einer Zeichenfolge in einem Speicherblock belegt die Zahl der Zeichen in der Zeichenfolge plus 1 Byte.

Ab Hollywood 6.0 gibt es einen neuen Parameter **endian**, mit dem Sie die Byte-Reihenfolge angeben können, die beim Lesen der Daten aus dem Speicherblock verwendet werden soll. Dies kann auf die folgenden Typen festgelegt werden:

**#BIGENDIAN:**

Big-Endian-Byte-Reihenfolge, MSB zuerst. Dies ist die Standardeinstellung.

**#NATIVEENDIAN:**

Native-Endian-Byte-Reihenfolge. Wenn Sie diesen Typ verwenden, hängt die Byte-Reihenfolge von der Standard-Byte-Reihenfolge auf dem Hostsystem ab, d.h. Big Endian auf Big-Endian-Systemen, Little Endian auf Little-Endian-Systemen. Seien Sie vorsichtig mit diesem Typ, da er die Portabilität einschränkt.

**#LITTLEENDIAN:**

Little-Endian-Byte-Reihenfolge, zuerst LSB. (V8.0)

**EINGABEN**

**id**            ID des zu verwendenden Speicherblocks  
**offset**       Versatz, ab dem geschrieben wird (in Bytes)  
**val**           Daten, die geschrieben werden; kann eine Zeichenfolge oder Zahl sein  
**type**        optional: Datentyp, der geschrieben wird (Voreingestellt ist **#INTEGER**)  
**endian**      optional: Reihenfolge der Bytes (Standardeinstellung ist **#BIGENDIAN**) (V6.0)

**BEISPIEL**

```
AllocMem(1, 1024)
Poke(1, 0, "Hello World!", #STRING)
Print(Peek(1, 0, #STRING))
```

Dies wird "Hello World!" auf den Bildschirm ausgeben.

## 48.15 ReadMem

### BEZEICHNUNG

ReadMem – liest Rohdaten aus einer Datei (V2.0)

### ÜBERSICHT

ReadMem(file\_id, blk\_id, len[, offset])

### BESCHREIBUNG

Dieser Befehl erlaubt Ihnen, `len` Bytes Rohdaten aus der in `file_id` angegebenen offenen Datei zu lesen und in den Speicherblock `blk_id` abzulegen (um eine Datei zu öffnen, benutzen Sie den Befehl `OpenFile()`). Zusätzlich können Sie optional das Argument `offset` angeben, um zu definieren, wo im Speicherblock die Rohdaten gespeichert werden. Die Daten aus der Quelldatei werden von der aktuellen Cursorposition der Datei gelesen, die Sie mit dem Befehl `Seek()` ändern können.

### EINGABEN

<code>file_id</code>	ID der geöffneten Datei
<code>blk_id</code>	ID des Speicherblocks
<code>len</code>	Bytes, die aus der Datei gelesen werden
<code>offset</code>	optional: Versatz, wo die Daten im Block gespeichert werden (Voreinstellung 0 = Anfang des Blocks)

### BEISPIEL

```
len = FileSize("C:SetPatch")
OpenFile(1, "C:SetPatch", #MODE_READ)
AllocMem(1, len)
ReadMem(1, 1, len)
CloseFile(1)
OpenFile(1, "Ram:Copy_of_SetPatch", #MODE_WRITE)
WriteMem(1, 1, len)
CloseFile(1)
FreeMem(1)
```

Erstellt eine Kopie des SetPatch-Programms im RAM: unter Verwendung der beiden Rohdaten-I/O-Befehle `ReadMem()` und `WriteMem()`.

## 48.16 TableToMem

### BEZEICHNUNG

TableToMem – schreibt den Inhalt der Tabelle in den Speicherblock (V6.0)

### ÜBERSICHT

TableToMem(t, id, type[, table])

### BESCHREIBUNG

Dieser Befehl schreibt den Inhalt der in `t` angegebenen Tabelle zu dem durch `id` angegebenen Speicherblockobjekt. In `type` geben Sie den Datentyp der Elemente an, die in den

Speicherblock geschrieben werden sollen. Das kann `#BYTE` (1 Byte), `#SHORT` (2 Bytes), `#INTEGER` (4 Bytes), `#FLOAT` (4 Bytes), oder `#DOUBLE` (8 Bytes) sein.

Mit dem optionalen Argument `table` können weitere Optionen für die Operation gesetzt werden. Folgende Tabellenfelder werden derzeit erkannt:

**Offset:** Dieser Tag kann verwendet werden, um einen Versatz in Byte innerhalb des Arbeitsspeichers festzulegen, der definiert, wo `TableToMem()` beginnen soll Elemente zu schreiben. Die Voreinstellung ist 0, womit das Schreiben am Anfang des Speicherblocks beginnt.

**EndianSwitch:**

Wenn dieses Feld auf `True` gesetzt ist, wechselt `TableToMem()` die Byte-Reihenfolge für alle Multi-Byte-Datentypen, wenn sie in den Speicherblock geschrieben werden. Das kann nützlich sein wenn Sie zwischen Big- und Little-Endian-Werte konvertieren müssen. Standardwert ist `False`.

Wenn es mehr Elemente in der Tabelle gibt, als der Speicherblock speichern kann, wird dieser Befehl einen Fehler ausgeben.

Um einen Speicherblock wieder in eine Tabelle zu konvertieren, verwenden Sie den Befehl `MemToTable()` Siehe [Abschnitt 48.12 \[MemToTable\]](#), [Seite 1067](#), für Details.

## EINGABEN

<code>t</code>	Tabelle, deren Inhalt in den Speicherblock geschrieben werden soll
<code>id</code>	der zu verwendender Speicherblock
<code>type</code>	Datentyp der zu schreibenden Elemente (siehe oben)
<code>table</code>	optional: Tabelle, um weiteren Optionen zu konfigurieren (siehe oben)

## 48.17 WriteMem

### BEZEICHNUNG

WriteMem – schreibt Rohdaten in eine Datei (V2.0)

### ÜBERSICHT

`WriteMem(file_id, blk_id, len[, offset])`

### BESCHREIBUNG

Dieser Befehl schreibt `len` Bytes des Speicherblocks `blk_id` in die Datei `file_id`. Mit dem optionalen Argument `offset` geben Sie an, ab wo die Daten aus dem Speicherblock gelesen werden. Die Daten werden ab der aktuellen Cursorposition in die Datei geschrieben. Möchten Sie die Cursorposition der Datei ändern, benutzen Sie dafür den Befehl `Seek()`. Um eine Datei zu öffnen, benutzen Sie den Befehl `OpenFile()`.

## EINGABEN

<code>file_id</code>	ID der geöffneten Datei
<code>blk_id</code>	ID des Speicherblocks
<code>len</code>	schreibt diese Anzahl Bytes in die Datei



**offset**      optional: Versatz, ab wo die Daten aus dem Block gelesen werden sollen  
(Voreinstellung 0 = Anfang des Blocks)

**BEISPIEL**

Siehe [Abschnitt 48.15 \[ReadMem\]](#), Seite 1071.



## 49 Spritebibliothek

### 49.1 Übersicht

Sprites sind elementare Teile vieler Anwendungen. Sie können für viele verschiedene Zwecke verwendet werden und Hollywood geht sehr flexibel mit ihnen um, weil sie vollständig in die Software implementiert werden. Somit gibt es keine Beschränkungen für Spritegröße, -farben, -transparenz und so weiter. Traditionell werden Sprites für Spieler und Feind Grafiken in Spielen verwendet, aber sie kommen auch praktisch in vielen anderen Situationen vor.

Allgemein kann man sagen, Sprites haben drei verschiedene Attribute, die sie von Pinseln unterscheidet:

1. Sprites sind immer an der vordersten Front vom Display.
2. Jedes Sprite kann nur einmal auf dem Bildschirm vorkommen.
3. Ein Sprite kann mehrere Einzelbilder haben.

Eine detailliertere Erläuterung der Sprite-Befehle folgt weiter unten.

1. Sprites sind immer an der vordersten Front vom Display. Schauen Sie sich den folgenden Code an:

```
LoadBrush(1, "test.iff")
LoadSprite(1, "test2.iff")
DisplaySprite(1, 0, 0)
DisplayBrush(1, 0, 0)
```

Sie sehen, dass wir zwei Bilder laden: Pinsel 1 und Sprite 1. Nun zeigen wir Sprite 1 und danach Pinsel 1. Normalerweise müsste Pinsel 1 über dem Sprite gezeichnet werden, weil `DisplayBrush()` nach `DisplaySprite()` aufgerufen wurde. Sprites sind jedoch immer auf der Vorderseite des Displays und deshalb wird in diesem speziellen Fall Pinsel 1 hinter Sprite 1 gezeichnet werden.

Dies gilt für alle normalen Grafikbefehle von Hollywood: Sie können nie Grafiken über ein Sprite zeichnen! Sprites sind immer auf der Vorderseite und die normalen Befehle, das heißt Nicht-Sprite-Befehle können nie ein Sprite übermalen. Sie können nur Sprites über neue Sprites zeichnen.

2. Jedes Sprite kann nur einmal auf dem Bildschirm vorkommen. Schauen Sie sich den folgenden Code an:

```
LoadSprite(1, "test.iff")
DisplaySprite(1, 0, 0)
DisplaySprite(1, 100, 100)
```

Sie sehen, dass wir Sprite 1 zwei Mal anzeigen. Zunächst wird es bei 0 : 0 und ein zweites Mal bei 100 : 100 angezeigt. Da jedes Sprite nur einmal auf dem Bildschirm sein kann, wird der zweite `DisplaySprite()` Sprite 1 nicht wieder zeichnen, aber es von 0 : 0 bis 100: 100 bewegen. `DisplaySprite()` überprüft, ob Sprite 1 auf dem Bildschirm ist. Wenn ja, wird es abgeholt und an die neue Position verschoben. So können Sie einfach Ihre Sprites auf dem Bildschirm bewegen.

3. Ein Sprite kann mehrere Einzelbilder haben. Weil Sprites oft für Animation verwendet werden, kann jedes Sprite mehrere Einzelbilder wie ein Animationsobjekt von Hollywood tragen. Der Befehl `DisplaySprite()` akzeptiert ein optionales Argument, womit Sie angeben können, welches Einzelbild gezeigt werden soll.

Weitere Informationen über die Sprite-Umsetzung in Hollywood:

- Sprites sind an Clip-Regionen gebunden: Wenn ein Sprite zum ersten Mal angezeigt wird, wird es an die derzeitig aktive Clip-Region gebunden. Der Sprite wird auch in dieser Clip-Region bleiben, wenn Sie die Clip-Region später deaktivieren. Um ein Sprite in einer Clip-Region zu entfernen, können Sie entweder die gesamte Clip-Region mit `FreeClipRegion()` aus dem Speicher löschen oder den Sprite mit `RemoveSprite()` entfernen und ihn dann wieder anzuzeigen, wenn keine Clip-Region mehr aktiv ist.
- Wenn Sie ein neues Hintergrundbild anzeigen, werden alle Sprites vom alten Hintergrundbild automatisch entfernt.
- Sprites können nur auf Ihrem Display gezeichnet werden. Sie können nicht Sprites auf Pinsel, Masken oder Alphakanäle zeichnen.
- Ebenen können nicht mit Sprites zusammen verwendet werden. Diese beiden Konzepte funktionieren nicht miteinander.
- Die Doppelpuffer-Befehle können nicht zusammen mit Sprites verwendet werden. Wenn Sie Doppelpuffer verwenden, brauchen Sie in der Regel sowieso keine Sprites.

## 49.2 CopySprite

### BEZEICHNUNG

CopySprite – kopiert einen Sprite (V2.0)

### ÜBERSICHT

[id] = CopySprite(source, dest)

### BESCHREIBUNG

Dieser Befehl erstellt eine Kopie des in `source` angegebenen Sprite und weist ihm die ID `dest` zu. Wenn Sie beim Argument `dest` `Nil` angeben, wird `CopySprite()` für den kopierten Sprite automatisch eine Zahl auswählen und Ihnen übergeben. Der neue Sprite ist unabhängig vom alten, so dass Sie den alten Sprite aus dem Arbeitsspeicher löschen können, nachdem er kopiert wurde.

Wenn Sie einen neuen Sprite mit der gleichen Grafik wie dem alten Sprite haben wollen, sollten Sie stattdessen `CreateSprite()` verwenden; es kann Sprite-Links erstellen, die sehr wenig Arbeitsspeicher verbrauchen und damit, wann immer möglich, den Vorzug zu `CopySprite()` gegeben werden sollte.

### EINGABEN

<code>source</code>	Identifikator des Quellsprites
<code>dest</code>	ID für den neuen Sprite oder <code>Nil</code> für die automatische ID-Zuweisung

### RÜCKGABEWERTE

<code>id</code>	optional: Identifikator des kopierten Sprites; Wird nur zurückgegeben werden, wenn Sie <code>Nil</code> als Argument 2 angegeben haben (siehe oben)
-----------------	---

## 49.3 CreateSprite

### BEZEICHNUNG

CreateSprite – erstellt einen Sprite (V2.0)

### ÜBERSICHT

```
[id] = CreateSprite(id, type, ...)
[id] = CreateSprite(id, #ANIM, source_id)
[id] = CreateSprite(id, #BRUSH, source_id[, width, height, frames,
                                fr_per_row, source_x, source_y])
[id] = CreateSprite(id, #SPRITE, source_id1, source_id2, ...)
[id] = CreateSprite(id, #TEXTOBJECT, source_id)
```

### BESCHREIBUNG

Dieser Befehl erstellt einen neuen Sprite aus der angegebenen Quelle. Die Sprite-Quelle kann eine Animation, ein Pinsel, ein Sprite oder ein Textobjekt sein. Der neue Sprite wird unter der angegebenen *id* gespeichert werden. Wenn Sie beim Argument *id* **Nil** angeben, wird `CreateSprite()` automatisch eine Zahl auswählen und Ihnen übergeben. Die Argumente von `CreateSprite()` hängen davon ab, welchen Quellentyp Sie angeben. Wenn der Typ `#ANIM` ist, übergeben Sie einfach die ID des Animationsobjektes in *source\_id*.

Wenn der Typ `#BRUSH` ist, müssen Sie die ID in *source\_id* angeben. Mit `CreateSprite()` können Sie auch mehrere Einzelbilder aus einem Pinsel lesen. Wenn Sie das wollen, müssen Sie auch die Argumente *width*, *height* und *frames* übergeben. *width* und *height* definieren die Dimensionen für den Sprite und *frames* gibt an, wie viele Einzelbilder `CreateSprite()` aus dem Pinsel lesen soll. Falls die Einzelbilder in mehreren Reihen im Pinsel abgelegt sind, werden Sie mit dem Argument *fr\_per\_row* angeben müssen, wie viele Bilder es in jeder Reihe hat. Schließlich kann man `CreateSprite()` mit den Argumenten *source\_x* und *source\_y* (beide sind standardmäßig auf 0 gesetzt) mitteilen, ab welchem Einzelbild im Pinsel das Auslesen starten soll. `CreateSprite()` wird dann an der Position *source\_x* und *source\_y* beginnen, die in *frames* definierte Anzahl Einzelbilder in der Abmessung von *width* und *height* aus dem in *source\_id* angegebenen Pinsel zu lesen. Nachdem es *fr\_per\_row* Einzelbilder gelesen hat, wird in die nächste Zeile vorgerückt. Wenn Sie nur drei Argumente angeben, wird `CreateSprite()` einfach den in *source\_id* angegebenen Pinsel in einen Sprite konvertieren.

Wenn der Typ `#SPRITE` ist, wird `CreateSprite()` einen neuen Sprite aus einer unbegrenzten Anzahl von Quellen-Sprites erstellen. Sie können so viele Quellen-Sprites angeben, wie Sie wollen. Selbstverständlich kann jeder dieser Sprites auch mehrere Einzelbilder aufweisen. Wenn Sie `#SPRITE` verwenden, wird `CreateSprite()` nie die Grafikdaten des angegebenen Quellen-Sprites kopieren. Aus Leistungsgründen werden die Quellen-Sprites nur referenziert und somit wird Ihr neuer Sprite von ihnen abhängen. Durch die Verwendung von einem Verweis auf den vorhandenen Sprite, ist `CreateSprite()` sehr schnell und braucht nur sehr wenig Arbeitsspeicher. Das ist sehr praktisch, wenn Sie verschiedene Animationssequenzen von immer dem gleichen Quellen-Sprite erstellen möchten. Bitte beachten Sie aber, dass das Erstellen eines Sprites von einem Unter-Sprite eine Abhängigkeit erzeugt. Wird ein Unter-Sprite aus dem Arbeitsspeicher entfernt, werden automatisch alle Unter-Sprites auch aus dem Arbeitsspeicher gelöscht, die eine Re-

ferenz auf den erstgenannten Unter-Sprite hatten. Sie sollten also nicht die Unter-Sprites aus dem Arbeitsspeicher entfernen, bevor Sie mit dem neu erstellten Sprite fertig sind.

Wenn der Typ `#TEXTOBJECT` ist, wird `CreateSprite()` einen Sprite aus dem in `source_id` angegebenen Textobjekt erstellen. Sie müssen nur die ID des Quelltext-Objekts übergeben.

#### EINGABEN

<code>id</code>	Identifikator des neuen Sprites oder <code>Nil</code> für die automatische ID-Zuweisung
<code>type</code>	Typ des Quellenobjekts
<code>...</code>	weitere Argumente, die abhängig vom jeweiligen Typ sind (siehe oben)

#### RÜCKGABEWERTE

<code>id</code>	optional: Identifikator des neuen Sprites; Wird nur zurückgegeben werden, wenn Sie <code>Nil</code> als Argument 1 angegeben haben (siehe oben)
-----------------	---

## 49.4 DisplaySprite

#### BEZEICHNUNG

DisplaySprite – zeigt einen Sprite (V2.0)

#### ÜBERSICHT

`DisplaySprite(id, x, y[, frame])`

#### BESCHREIBUNG

Dieser Befehl zeigt den in `id` angegebenen Sprite an der definierten Position an. Wenn der Sprite bereits auf dem Bildschirm ist, wird er an die neue Position bewegt werden. Das optionale Argument `frame` kann verwendet werden, um festzulegen, welches Einzelbild angezeigt werden soll. Wenn es weggelassen wird, zeigt `DisplaySprite()` das nächste Bild des Sprites an (wenn der Sprite mehrere Einzelbilder hat).

#### EINGABEN

<code>id</code>	Identifikator des Sprites
<code>x</code>	gewünschte x-Position
<code>y</code>	gewünschte y-Position
<code>frame</code>	optional: Einzelbild, welches angezeigt werden soll (voreingestellt ist 0, womit das nächste Einzelbild angezeigt wird)

## 49.5 FlipSprite

#### BEZEICHNUNG

FlipSprite – dreht einen Sprite (V2.0)

#### ÜBERSICHT

`FlipSprite(id, xflip)`

**BESCHREIBUNG**

Dieser Befehl dreht (spiegelt) den in `id` angegebene Sprite um die eigene Achse. Wenn `xflip` auf `True` gesetzt ist, wird der Sprite auf seiner Achse in x-Richtung, sonst in y-Richtung gedreht.

Dieser Befehl kann nur auf Sprites verwendet werden, die nicht mit irgendwelchen anderen Sprites referenziert sind. Es kann auch nicht mit Sprite-Links verwendet werden, die mit dem Befehl `CreateSprite()` mit Quelltyp `#SPRITE` erstellt wurden.

**EINGABEN**

`id`            Identifikator des Sprites, welcher gedreht (gespiegelt) wird  
`xflip`        `True` für horizontale Achsdrehung (x), `False` für vertikale Achsdrehung (y)

## 49.6 FreeSprite

**BEZEICHNUNG**

`FreeSprite` – löscht den Sprite aus dem Speicher (V2.0)

**ÜBERSICHT**

`FreeSprite(id)`

**BESCHREIBUNG**

Dieser Befehl löscht den in `id` angegebenen Sprite aus dem Speicher. Um den Speicherverbrauch zu reduzieren, sollten Sie Sprites aus dem Speicher löschen, wenn Sie sie nicht mehr benötigen.

Wenn der Sprite noch auf dem Bildschirm ist und Sie rufen `FreeSprite()` auf, wird er entfernt werden, bevor der Speicher freigegeben wird.

**EINGABEN**

`id`            Identifikator des Sprites, welcher aus dem Speicher gelöscht wird

## 49.7 LoadSprite

**BEZEICHNUNG**

`LoadSprite` – lädt einen Sprite (V2.0)

**ÜBERSICHT**

`[id] = LoadSprite(id, filename$[, args])`

**BESCHREIBUNG**

Dieser Befehl lädt den in `filename$` angegebenen Sprite in den Arbeitsspeicher und weist ihm die `id` zu. Wenn Sie `Nil` angeben, wird `LoadSprite()` automatisch eine freie ID für Sie auswählen und Ihnen zurückgeben.

Unterstützte Bildformate sind PNG, JPEG, BMP, IFF ILBM und einige mehr, je nachdem auf welcher Plattform Hollywood läuft. Ab Hollywood 4.5, kann `LoadSprite()` auch Animationsformate (IFF ANIM, GIF ANIM, unkomprimierte AVIs oder AVIs mit Motion-JPEG-Kompression) öffnen und wandelt diese Animationen direkt in ein Sprite um.

Das dritte Argument **args** ist optional. Es ist eine Tabelle, die weitere Möglichkeiten für den Ladevorgang verwendet. Die folgenden Felder der Tabelle können benutzt werden:

**Transparency:**

Dieses Feld kann verwendet werden, um eine Farbe mit einem **RGB-Wert** anzugeben, die im Sprite transparent sein soll.

**LoadAlpha:**

Setzen Sie dieses Feld auf **True**, wenn der Alphakanal des Bildes auch geladen werden soll. Bitte beachten Sie, dass nicht alle Bilder einen Alphakanal haben und dass nicht alle Bildformate in der Lage sind, Alphakanaldata zu speichern. Es wird vorgeschlagen, dass Sie das PNG-Format verwenden, wenn Sie Alphakanaldata benötigen. Dieser Tag ist standardmäßig **False**.  
(4.5)

**X, Y, Width, Height, Frames, FPR:**

Diese Felder können für die Feinabstimmung des Ladevorgangs verwendet werden. Mit diesen Feldern erstellen Sie einen Sprite mit mehreren Einzelbildern aus einem einzigen Bild. **Width** und **Height** definieren die Dimensionen für den Sprite und **Frames** gibt an, wie viele Einzelbilder **LoadSprite()** aus dem Bild lesen wird. Wenn die Einzelbilder in mehreren Reihen im Quellbild abgelegt sind, werden Sie auch das Argument **FPR** übergeben müssen (steht für Frames per row (Einzelbild pro Zeile)), um **LoadSprite()** mitzuteilen, wie viele Bilder es in jeder Reihe hat. schließlich können Sie **LoadSprite()** anweisen, wo in der Bilddatei das Scannen starten soll, indem Sie die Felder **X** und **Y** angeben (beide standardmäßig auf 0 gesetzt). **LoadSprite()** wird dann an der Position **X/Y** beginnen und mit **Frames** die Anzahl der Bilder mit der Abmessungen von **Width** und **Height** aus der Bilddatei **filename\$** lesen. Nachdem es die Anzahl der in **FPR** gesetzten Bilder gelesen hat, wird es in die nächste Zeile vorrücken. Alle diese Felder können nur verwendet werden, wenn Sie eine Bilddatei in **filename\$** angeben. Wenn Sie eine Animationsdatei angeben, werden sie ignoriert.

**Loader:** Mit diesem Tag können Sie einen oder mehrere Formatlademodule angeben, die die Datei laden soll. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Lademodulen beinhaltet. Standardmäßig wird der mit **SetDefaultLoader()** eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details.  
(V6.0)

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, der die angegebene Datei öffnet. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Adaptermodulen beinhaltet. Standardmäßig wird der mit **SetDefaultAdapter()** eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details.  
(V6.0)

**LoadTransparency:**

Ist dieser Tag auf **True** gesetzt, wird die monochrome Transparenz des Bildes geladen. Bitte beachten Sie, dass dieser Tag speziell für monochrome



Transparenzkanäle ausgelegt sind, die einem transparenten Stift in einem palettenbasierten Bild haben. Wenn Sie den Alphakanal eines Bildes laden möchten, stellen Sie den Tag `LoadAlpha` auf `True`. Dieser `LoadTransparency` Tag ist standardmäßig auf `False` gesetzt. (V6.0)

#### **LoadPalette:**

Wenn dieser Tag auf `True` gesetzt ist, lädt Hollywood den Sprite als Paletten-Sprite. Dies bedeutet, dass Sie die Palette des Sprites abrufen und ändern können, was für bestimmte Effekte wie Farbwechsel nützlich ist. Sie können Stifte auch mit dem Tag `TransparentPen` (siehe unten) oder dem Tag `LoadTransparency` (siehe oben) transparent machen. Paletten-Sprites haben auch den Vorteil, dass sie weniger Speicher benötigen, da 1 Pixel nur 1 Byte Speicherplatz anstelle von 4 Byte für 32-Bit-Bilder benötigt. Dieser Tag ist standardmäßig `False`. (V9.0)

#### **TransparentPen:**

Wenn der Tag `LoadPalette` auf `True` gesetzt wurde (siehe oben), kann mit dem Tag `TransparentPen` ein Stift definiert werden, der transparent gemacht werden soll. Stifte werden ab 0 gezählt. Alternativ können Sie auch den Tag `LoadTransparency` auf `True` setzen, um Hollywood zu zwingen, den in der Datei gespeicherten transparenten Stift zu verwenden (sofern das Dateiformat die Speicherung transparenter Stifte unterstützt). Dieser Tag ist standardmäßig auf `#NOPEN` gesetzt. (V9.0)

#### **UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Bitte beachten Sie, dass die Felder `Transparency`, `LoadTransparency` und `LoadAlpha` sich gegenseitig ausschließen. Eine Animation kann nicht eine Maske und einen Alphakanal haben!

Dieser Befehl ist auch als Präprozessor-Anweisung vorhanden: mit `@SPRITE` können Sie Sprites vorladen.

### **EINGABEN**

<code>id</code>	Identifikator des neuen Sprites oder <code>Nil</code> für die automatische ID-Zuweisung
<code>filename\$</code>	Datei, die geladen wird
<code>args</code>	optional: weitere Optionen zum Konfigurieren des Ladevorgangs

### **RÜCKGABEWERTE**

<code>id</code>	optional: Identifikator des neuen Sprites; Wird nur zurückgegeben werden, wenn Sie <code>Nil</code> als Argument 1 angegeben haben (siehe oben)
-----------------	---

### **BEISPIEL**

```
LoadSprite(1, "MySprite.png", {Transparency = #RED})
```

Dieser Code lädt "MySprite.png" als Sprite 1 und die Farbe Rot wird transparent dargestellt.

```
LoadSprite(1, "PlayerSprites.png", {Width = 32, Height = 32,
    Frames = 16, FPR = 8, Transparency = #BLACK})
```

Der obige Code erzeugt den Sprite 1 aus der Datei "PlayerSprites.png". Sprite 1 wird die Dimensionen 32x32 haben und 16 verschiedene Einzelbilder enthalten. Die Einzelbilder sind mit 8 Bildern pro Zeile in dem Bild "PlayerSprites.png" abgespeichert. So muss LoadSprite() zwei Zeilen scannen, um die 16 Einzelbilder alle zu lesen.

## 49.8 MoveSprite

### BEZEICHNUNG

MoveSprite – bewegt einen Sprite von a nach b (V2.0)

### ÜBERSICHT

```
MoveSprite(id, xa, ya, xb, yb[, table])
```

### BESCHREIBUNG

Dieser Befehl bewegt (scrollt) den durch `id` angegebenen Sprite ruckelfrei von dem durch `xa/ya` angegebenen Ort zu der durch `xb/yb` angegebene Position.

Weitere Konfigurationen sind mit der optionalen Tabelle `table` möglich. Sie können die Bewegungsgeschwindigkeit sowie Spezialeffekte angeben und ob die Bewegung asynchron sein soll. Siehe [Abschnitt 43.46 \[MoveBrush\]](#), [Seite 948](#), für weitere Informationen über das optionalen Tabellenargument. Neben den Tabellenelementen, welche beim Befehl `MoveBrush()` erklärt werden, akzeptiert `MoveSprite()` das zusätzliche Tabellenargument `AnimSpeed`: Dieser Geschwindigkeitswert definiert, nach wie vielen Bewegungsschritten das nächste Einzelbild angezeigt werden soll. Eine höhere Zahl bedeutet also eine geringere Wiedergabegeschwindigkeit der Animation.

### EINGABEN

<code>id</code>	ID des Sprites
<code>xa</code>	Anfangs-X-Position
<code>ya</code>	Anfangs-Y-Position
<code>xb</code>	End-X-Position
<code>yb</code>	End-Y-Position
<code>table</code>	optional: weitere Konfigurationen für die Bewegung

### BEISPIEL

```
MoveSprite(1, 100, 50, 0, 50, {Speed = 5, AnimSpeed = 4})
```

Verschiebt den Sprite 1 von 100:50 nach 0:50 mit der Geschwindigkeit 5 und der Abspielgeschwindigkeit 4.

## 49.9 RemoveSprite

### BEZEICHNUNG

RemoveSprite – entfernt einen Sprite vom Display (V2.0)

### ÜBERSICHT

RemoveSprite(id)

### BESCHREIBUNG

Dieser Befehl entfernt den in `id` angegebenen Sprite vom Display. Beachten Sie, dass der Sprite nicht aus dem Arbeitsspeicher gelöscht wird. So können Sie ihn jederzeit wieder anzeigen.

### EINGABEN

`id` ID des Sprites, der entfernt werden soll

## 49.10 RemoveSprites

### BEZEICHNUNG

RemoveSprites – entfernt alle Sprites eines Displays (V2.0)

### ÜBERSICHT

RemoveSprites([keep])

### BESCHREIBUNG

Mit diesem Befehl werden alle Sprites aus dem Display entfernt. Wenn Sie das optionale Argument `keep` auf `True` setzen, werden die Sprites entfernt, werden aber als normale Grafiken auf dem Display wiedergegeben. Das bedeutet, dass Sie sie jetzt mit anderen Grafiken übermalen könnten (zum Beispiel mit einem "Game Over" Pinsel).

Wenn `keep` auf `True` gesetzt ist, werden Sie nicht sehen, dass dieser Befehl etwas bewirkt. Dieser Eindruck ist jedoch falsch. Die Sprites sind in der Tat entfernt, aber Sie sehen keinen Unterschied, weil sie sofort als normale Grafiken auf dem Display wiedergegeben werden. So können Sie sie jetzt übermalen.

### EINGABEN

`keep` optional: `True`, wenn die Sprites als normale Grafiken auf dem Bildschirm bleiben sollen (voreingestellt ist `False`)

## 49.11 ScaleSprite

### BEZEICHNUNG

ScaleSprite – skaliert einen Sprite (V2.0)

### ÜBERSICHT

ScaleSprite(id, width, height)

### BESCHREIBUNG

Dieser Befehl skaliert den in `id` angegebenen Sprite auf die Breite `width` und Höhe `height`.

Dieser Befehl kann nur mit Sprites verwendet werden, die nicht mit irgendwelchen anderen Sprites referenziert sind. Es kann auch nicht mit Sprite-Links verwendet werden, die mit `CreateSprite()` und dem als Quellentyp `#SPRITE` erstellt wurden.

Bitte beachten Sie: Sie sollten Skalierungen immer mit einer frischen Kopie des Original Sprite durchführen. Zum Beispiel, wenn Sie Sprite 1 auf 12x8 skalieren und es dann wieder zurück auf 640x480 vergrößern, erhalten Sie ein verwirrtes Bild. Deshalb sollten Sie immer den Original Sprite beibehalten und nur Kopien davon skalieren.

Sie können `#KEEPASPRAT` entweder bei der Breite `width` oder der Höhe `height` angeben. Hollywood wird dann die Größe automatisch berechnen, indem das Seitenverhältnis des Sprites berücksichtigt wird.

Als Alternative können Sie bei `width` und `height` auch eine prozentige Angabe in Form einer Zeichenkette angeben, z.B. "50%".

#### EINGABEN

<code>id</code>	ID des Sprites, welcher skaliert werden soll
<code>width</code>	gewünschte neue Breite des Sprites
<code>height</code>	gewünschte neue Höhe des Sprites

## 49.12 SetSpriteZPos

#### BEZEICHNUNG

`SetSpriteZPos` – ändert die z-Position eines Sprites (V7.0)

#### ÜBERSICHT

`SetSpriteZPos(id, zpos)`

#### BESCHREIBUNG

Mit diesem Befehl kann die z-Position eines Sprites geändert werden. Die z-Position eines Sprites ist seine Position in der Hierarchie der Sprites. Der erste (d.h. hinterste) Sprite hat eine z-Position von 1, der letzte (d.h. vorderste) Sprite hat die z-Position, welche gleich der Anzahl der derzeit vorhandenen Sprites ist. Sie müssen die neue gewünschte z-Position für den angegebenen Sprite an diesen Befehl übergeben. Der Sprite wird dann genau diese z-Position annehmen. Bestehende Sprites, die auf oder nach dieser z-Position sind, werden nach unten verschoben. Um einen Sprite ganz nach vorne zu bewegen (d.h. höchste z-Position), können Sie den speziellen Wert 0 für das `zpos`-Argument übergeben. Um einen Sprite ganz nach hinten zu bewegen, übergeben Sie 1 im `zpos`-Argument.

#### EINGABEN

<code>id</code>	ID des Sprites, dessen z-Position geändert werden soll
<code>zpos</code>	neue z-Position für den Sprite oder 0, um den Sprite auf die höchste z-Position zu bewegen

## 49.13 SPRITE

### BEZEICHNUNG

SPRITE – lädt einen Sprite vor (V2.0)

### ÜBERSICHT

@SPRITE id, filename\$[, table]

### BESCHREIBUNG

Diese Präprozessor-Anweisung lädt den in **filename\$** angegebene Sprite vor und weist ihm den Identifikator **id** zu.

Unterstützte Bildformate sind PNG, JPEG, BMP, IFF ILBM und einige mehr, je nachdem auf welcher Plattform Hollywood läuft. Ab Hollywood 4.5, @SPRITE kann auch Animationsformate (IFF ANIM, GIF ANIM, unkomprimierte AVIs oder AVIs mit Motion-JPEG-Kompression) öffnen und wandelt diese Animationen direkt in ein Sprite um.

Das dritte Argument ist optional. Es ist eine Tabelle, die weitere Möglichkeiten für den Ladevorgang verwendet. Die folgenden Felder der Tabelle können benutzt werden:

#### Transparency:

Dieses Feld kann verwendet werden, um eine Farbe mit einem **RGB-Wert** anzugeben, die im Sprite transparent sein soll.

#### LoadAlpha:

Setzen Sie dieses Feld auf **True**, wenn der Alphakanal des Bildes auch geladen werden soll. Bitte beachten Sie, dass nicht alle Bilder einen Alphakanal haben und dass nicht alle Bildformate in der Lage sind, Alphakanaldaten zu speichern. Es wird vorgeschlagen, dass Sie das PNG-Format verwenden, wenn Sie Alphakanaldaten benötigen. Dieser Tag ist standardmäßig **False**. (4.5)

#### Link:

Setzen Sie dieses Feld auf **False**, wenn Sie diesen Sprite beim Kompilieren nicht in die ausführbare Datei oder ins Applet einbinden möchten. Dieses Feld ist standardmäßig auf **True** gesetzt, womit der Sprite in Ihre Datei oder mit in ihrem Applet im Kompilierungsmodus eingebunden wird.

#### X, Y, Width, Height, Frames, FPR:

Diese Felder können für die Feinabstimmung des Ladevorgangs verwendet werden. Mit diesen Feldern erstellen Sie einen Sprite mit mehreren Einzelbilder aus einem einzigen Bild. **Width** und **Height** definieren die Dimensionen für den Sprite und **Frames** gibt an, wie viele Einzelbilder @SPRITE aus dem Bild lesen wird. Wenn die Einzelbilder in mehreren Reihen im Quellbild abgelegt sind, werden Sie auch das Argument **FPR** übergeben müssen (steht für Frames per row (Einzelbild pro Zeile)), um @SPRITE mitzuteilen, wie viele Bilder es in jeder Reihe hat. schließlich können Sie @SPRITE anweisen, wo in der Bilddatei das Scannen starten soll, indem Sie die Felder **X** und **Y** angeben (beide standardmäßig auf 0 gesetzt). @SPRITE wird dann an der Position **X/Y** beginnen und mit **Frames** die Anzahl der Bilder mit den Abmessungen von **Width** und **Height** aus der Bilddatei **filename\$** lesen. Nachdem es die Anzahl der in **FPR** gesetzten Bilder gelesen hat, wird es in die nächste Zeile vorrücken. Alle diese Felder können nur verwendet werden, wenn Sie eine

Bilddatei in `filename$` angeben. Wenn Sie eine Animationsdatei angeben, werden sie ignoriert.

**Loader:** Mit diesem Tag können Sie einen oder mehrere Formatlademodule angeben, die die Datei laden soll. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Lademodulen beinhaltet. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, der die angegebene Datei öffnet. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Adaptermodulen beinhaltet. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**LoadTransparency:**

Ist dieser Tag auf `True` gesetzt, wird die monochrome Transparenz des Bildes geladen. Bitte beachten Sie, dass dieser Tag speziell für monochrome Transparenzkanäle ausgelegt ist, die einen transparenten Stift in einem palettenbasierten Bild haben. Wenn Sie den Alphakanal eines Bildes laden möchten, stellen Sie den Tag `LoadAlpha` auf `True`. Dieser `LoadTransparency` Tag ist standardmäßig auf `False` gesetzt. (V6.0)

**LoadPalette:**

Wenn dieser Tag auf `True` gesetzt ist, lädt Hollywood den Sprite als Paletten-Sprite. Dies bedeutet, dass Sie die Palette des Sprites abrufen und ändern können, was für bestimmte Effekte wie Farbwechsel nützlich ist. Sie können Stifte auch mit dem Tag `TransparentPen` (siehe unten) oder dem Tag `LoadTransparency` (siehe oben) transparent machen. Paletten-Sprites haben auch den Vorteil, dass sie weniger Speicher benötigen, da 1 Pixel nur 1 Byte Speicherplatz anstelle von 4 Byte für 32-Bit-Bilder benötigt. Dieser Tag ist standardmäßig `False`. (V9.0)

**TransparentPen:**

Wenn der Tag `LoadPalette` auf `True` gesetzt wurde (siehe oben), kann mit dem Tag `TransparentPen` ein Stift definiert werden, der transparent gemacht werden soll. Stifte werden ab 0 gezählt. Alternativ können Sie auch den Tag `LoadTransparency` auf `True` setzen, um Hollywood zu zwingen, den in der Datei gespeicherten transparenten Stift zu verwenden (sofern das Dateiformat die Speicherung transparenter Stifte unterstützt). Dieser Tag ist standardmäßig auf `#NOPEN` gesetzt. (V9.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Bitte beachten Sie, dass die Felder **Transparency**, **LoadTransparency** und **LoadAlpha** sich gegenseitig ausschließen. Eine Animation kann nicht eine Maske und einen Alpha-kanal haben!

Möchten Sie Sprites manuell laden, benutzen Sie den Befehl **LoadSprite()**.

#### EINGABEN

<b>id</b>	ein Wert, der diesen Sprite später im Code identifiziert
<b>filename\$</b>	die Bilddatei, die Sie laden möchten
<b>table</b>	optional: eine Tabelle, die eine Kombination der oben erwähnten Felder enthält

#### BEISPIEL

```
@SPRITE 1, "MySprite.png", {Transparency = #RED}
```

Dieser Code lädt "MySprite.png" als Sprite 1 und die Farbe Rot wird transparent dargestellt.

```
@SPRITE 1, "PlayerSprites.png", {Width = 32, Height = 32,  
    Frames = 16, FPR = 8, Transparency = #BLACK}
```

Der obige Code erzeugt den Sprite 1 aus der Datei "PlayerSprites.png". Sprite 1 wird die Dimensionen 32x32 haben und 16 verschiedene Einzelbilder enthalten. Die Einzelbilder sind mit 8 Bildern pro Zeile in dem Bild "PlayerSprites.png" abgespeichert. So muss **@SPRITE** zwei Zeilen scannen, um die 16 Einzelbilder alle zu lesen.





## 50 Systembibliothek

### 50.1 Beep

#### BEZEICHNUNG

Beep – spielt den System-Piep ab (V8.0)

#### ÜBERSICHT

Beep([type])

#### BESCHREIBUNG

Mit diesem Befehl können Sie den Standard-Piepton des Systems wiedergeben. Bei Systemen, die unterschiedliche Pieptöne für verschiedene Kontexte unterstützen, können Sie das optionale Argument `type` angeben, um festzulegen, welcher Piepton gespielt werden soll.

Die folgenden Pieptypen werden derzeit unterstützt:

#BEEPSYSTEM:

System-Piepton. Dies ist die Voreinstellung.

#BEEPERROR:

Fehler-Piepton.

#BEEPWARNING:

Warn-Piepton.

#BEEPQUESTION:

Frage-Piepton.

#BEEPINFORMATION:

Information-Piepton.

Beachten Sie, dass nicht alle Betriebssysteme alle Arten von Pieptönen unterstützen. Auf jeder Plattform wird nur `#BEEPSYSTEM` unterstützt.

#### EINGABEN

`type` optional: Art des Pieptons, der abgespielt werden soll (Voreingestellt ist `#BEEPSYSTEM`)

#### BEISPIEL

Beep()

Der obige Code spielt den Standard-Piepton ab.

### 50.2 CollectGarbage

#### BEZEICHNUNG

CollectGarbage – erzwingt eine Speicherbereinigung oder setzt neuen Grenzwert (V2.0)

#### ÜBERSICHT

CollectGarbage([threshold])

**BESCHREIBUNG**

Mit diesem Befehl können Sie eine Speicherbereinigung erzwingen. Dazu müssen Sie `CollectGarbage()` aufrufen, ohne das optionale Argument `threshold` anzugeben. Wenn Sie hingegen `threshold` übergeben, wird dieser Wert als neuer Grenzwert festgelegt. Das bedeutet, dass immer dann, wenn die Speicherbelastung größer als der angegebene Grenzwert in Kilobytes ist, Hollywood automatisch den Speicherbereiniger laufen lässt.

Wenn Sie das optionale Argument auslassen, wird Hollywood sofort den Speicherbereiniger ausführen und einen neuen Grenzwert festlegen, der doppelt so groß ist wie der gerade freigegebene Arbeitsspeicher.

Um Informationen über den Zustand des Müllsammlers zu erhalten, rufen Sie den Befehl `GCInfo()` auf.

**EINGABEN**

`threshold`

Grenzwert in Kilobyte, der angibt, wann Hollywood den Speicherbereiniger ausführen sollte (standardmäßig 0, was bedeutet, dass die Speicherbereinigung sofort ausgeführt wird)

## 50.3 DisableLineHook

**BEZEICHNUNG**

`DisableLineHook` – deaktiviert die Hollywood-Funktion "LineHook" (V6.0)

**ÜBERSICHT**

`DisableLineHook()`

**BESCHREIBUNG**

Mit diesem Befehl kann die interne Funktion "LineHook" von Hollywood deaktiviert werden. "LineHook" ist eine Funktion, die nach jeder Codezeile automatisch von Hollywood aufgerufen wird. Sie ist verantwortlich für die Steuerung mehrerer Aufgaben, die Hollywood für seine internen Abläufe braucht, zum Beispiel Ereignis-Handling und Videowiedergabe. Daher ist es sehr wichtig, dass der "LineHook" mehrmals pro Sekunde aufgerufen wird, sonst wird Ihre Anwendung nicht mehr reagieren und die Videowiedergabe fängt an zu stottern. Das Deaktivieren der Funktion "LineHook" für eine kurze Zeitspanne kann jedoch die rohe Leistung von Hollywood erheblich steigern, da der gesamte Code überhaupt nicht mehr über die Funktion "LineHook" ausgeführt wird. Daher sollten Sie sie vorübergehend deaktivieren, wenn Sie einige komplexe Berechnungen so schnell wie möglich erledigen müssen. Stellen Sie jedoch sicher, dass die Funktion "LineHook" so bald wie möglich wieder aktiviert wird, um zu verhindern, dass Ihre Anwendung nicht mehr reagiert.

Um den "LineHook" wieder zu aktivieren, müssen Sie `EnableLineHook()` aufrufen.

**EINGABEN**

keine

## 50.4 ELSE

### BEZEICHNUNG

ELSE – Block, der abgearbeitet wird, wenn alle Bedingungen fehlgeschlagen sind (V7.0)

### ÜBERSICHT

@ELSE

### BESCHREIBUNG

Diese Präprozessor-Anweisung signalisiert den Anfang eines Blocks, der abgearbeitet werden soll, wenn alle vorherigen @IF und @ELSEIF Präprozessor-Anweisungen nicht zutrafen.

Siehe [Abschnitt 50.17 \[IF\]](#), [Seite 1099](#), für Details und einem Beispiel.

### EINGABEN

keine

### BEISPIEL

Siehe [Abschnitt 50.17 \[IF\]](#), [Seite 1099](#).

## 50.5 ELSEIF

### BEZEICHNUNG

ELSEIF – prüft eine andere Bedingung (V7.0)

### ÜBERSICHT

@ELSEIF val

### BESCHREIBUNG

Diese Präprozessor-Anweisung prüft die in val angegebene Bedingung. Wenn sie True ist, wird der Präprozessor die Abarbeitung des Skripts in diesem Block fortsetzen. Wenn val False ist, wird @ELSEIF zur nächsten @ELSEIF, @ELSE oder @ENDIF-Anweisung verzweigen. Beachten Sie, dass die Bedingung ein konstanter Ausdruck sein muss, da @ELSEIF auf Präprozessor-Ebene arbeitet, d.h. Skriptvariablen sind zu diesem Zeitpunkt nicht verfügbar.

Siehe [Abschnitt 50.17 \[IF\]](#), [Seite 1099](#), für Details und einem Beispiel.

### EINGABEN

val            die zu überprüfende Bedingung; muss ein konstanter Ausdruck sein

### BEISPIEL

Siehe [Abschnitt 50.17 \[IF\]](#), [Seite 1099](#).

## 50.6 EnableLineHook

### BEZEICHNUNG

EnableLineHook – aktiviert die Hollywood-Funktion "LineHook" (V6.0)

### ÜBERSICHT

EnableLineHook()

**BESCHREIBUNG**

Mit diesem Befehl kann die interne Funktion "LineHook" von Hollywood aktiviert werden. "LineHook" ist eine Funktion, die nach jeder Codezeile automatisch von Hollywood aufgerufen wird. Sie ist verantwortlich für die Steuerung mehrerer Aufgaben, die Hollywood für seine internen Abläufe braucht, zum Beispiel Ereignis-Handling und Videowiedergabe. Daher ist es sehr wichtig, dass der "LineHook" mehrmals pro Sekunde aufgerufen wird, sonst wird Ihre Anwendung nicht mehr reagieren und die Videowiedergabe fängt an zu stottern. Das Deaktivieren der Funktion "LineHook" für eine kurze Zeitspanne kann jedoch die rohe Leistung von Hollywood erheblich steigern, da der gesamte Code überhaupt nicht mehr über die Funktion "LineHook" ausgeführt wird. Daher sollten Sie sie vorübergehend deaktivieren, wenn Sie einige komplexe Berechnungen so schnell wie möglich erledigen müssen. Stellen Sie jedoch sicher, dass die Funktion "LineHook" so bald wie möglich wieder aktiviert wird, um zu verhindern, dass Ihre Anwendung nicht mehr reagiert.

Um den "LineHook" zu deaktivieren, müssen Sie `DisableLineHook()` aufrufen.

**EINGABEN**

keine

**50.7 End****BEZEICHNUNG**

End – beendet Hollywood

**ÜBERSICHT**

`End([code])`

**BESCHREIBUNG**

Dieser Befehl beendet sofort Ihr Skript, schließt Hollywood, gibt allen Arbeitsspeicher frei und schließt alle Bibliotheken.

Ab Hollywood 7.0 können Sie den optionalen Parameter `code` übergeben, um den Rückgabecode festzulegen, den der Hollywood-Prozess an seinen übergeordneten Prozess übergeben soll. Dies ist nützlich, wenn Sie Hollywood-Programme von einer Konsole ausführen.

**EINGABEN**

<code>code</code>	optional: Code, der dem übergeordneten Prozess übergeben wird (voreingestellt ist 0) (V7.0)
-------------------	---

**50.8 ENDIF****BEZEICHNUNG**

ENDIF – definiert das Ende des bedingten Blocks (V7.0)

**ÜBERSICHT**

`@ENDIF`

**BESCHREIBUNG**

Diese Präprozessor-Anweisung signalisiert das Ende eines bedingten Blocks, der zuvor von einer **@IF**-Präprozessor-Anweisung eingeleitet wurde.

Siehe [Abschnitt 50.17 \[IF\]](#), [Seite 1099](#), für Details und einem Beispiel.

**EINGABEN**

keine

**BEISPIEL**

Siehe [Abschnitt 50.17 \[IF\]](#), [Seite 1099](#).

## 50.9 GCInfo

**BEZEICHNUNG**

GCInfo – fragt den Status des Speicherbereinigers ab (V2.0)

**ÜBERSICHT**

```
count, threshold = GCInfo()
```

**BESCHREIBUNG**

Dieser Befehl gibt Informationen zum aktuellen Status des Speicherbereinigers (Garbage Collector) zurück. Der erste Rückgabewert **count** gibt an, wie viele Kilobyte im Arbeitsspeicher derzeit von Hollywood belegt sind, wohingegen der zweite Rückgabewert **threshold** die Grenze in Kilobyte angibt, die die Speicherbereinigung auslösen soll. Wenn der Speicherverbrauch den festgelegten Grenzwert überschreitet, führt Hollywood automatisch die Speicherbereinigung aus.

Sie können die Speicherbereinigung auch manuell ausführen oder den Grenzwert ändern, indem Sie den Befehl **CollectGarbage()** aufrufen.

**EINGABEN**

keine

**RÜCKGABEWERTE**

**count**            Speichermenge in Kilobyte, die derzeit von Hollywood verwendet wird

**threshold**       aktueller Grenzwert für die Speicherbereinigung

## 50.10 GetConstant

**BEZEICHNUNG**

GetConstant – gibt den konstanten Wert aus einer Zeichenfolge zurück (V4.5)

**ÜBERSICHT**

```
val = GetConstant(c$)
```

**BESCHREIBUNG**

Dieser Befehl liefert den Wert einer Konstanten, die als Zeichenfolge übergeben wird.

**EINGABEN**

`c$` Innere Zeichenfolge-Konstante, deren Wert abgerufen werden soll

**RÜCKGABEWERTE**

`val` Wert der Konstante

**BEISPIEL**

```
DebugPrint(#HOLLYWOOD, "=", GetConstant("#HOLLYWOOD"))
```

Ermittelt die Konstante `#HOLLYWOOD` zuerst über die direkte Deklaration und dann über `GetConstant()`. Das Ergebnis sollte das gleiche sein.

## 50.11 GetDefaultAdapter

**BEZEICHNUNG**

`GetDefaultAdapter` – ermittelt den Standardadapter für den Typ (V10.0)

**ÜBERSICHT**

```
adapter$ = GetDefaultAdapter(type)
```

**BESCHREIBUNG**

Dieser Befehl ermittelt den Standardadapter für den durch `type` angegebenen Typ und gibt ihn in `adapter$` zurück. Welche Adaptertypen in `type` übergeben werden können, ist im Handbuch zum Befehl `SetDefaultAdapter()` beschrieben. Siehe [Abschnitt 50.26 \[SetDefaultAdapter\]](#), [Seite 1110](#), für Details.

Beachten Sie, dass wenn kein Standardadapter mit `SetDefaultAdapter()` für einen bestimmten Typ festgelegt wurde, eine leere Zeichenkette zurückgegeben wird.

**EINGABEN**

`type` abzufragender Adaptertyp

## 50.12 GetDefaultLoader

**BEZEICHNUNG**

`GetDefaultLoader` – ermittelt den Standardlader für den Typ (V10.0)

**ÜBERSICHT**

```
loader$ = GetDefaultLoader(type)
```

**BESCHREIBUNG**

Dieser Befehl ermittelt den Standardlader für den durch `type` angegebenen Typ und gibt ihn in `loader$` zurück. Welche Lader-Typen in `type` übergeben werden können, ist im Handbuch zum Befehl `SetDefaultLoader()` beschrieben. Siehe [Abschnitt 50.27 \[SetDefaultLoader\]](#), [Seite 1111](#), für Details.

Beachten Sie, dass wenn kein Standard-Lader mit `SetDefaultLoader()` für einen bestimmten Typ festgelegt wurde, eine leere Zeichenkette zurückgegeben wird.

**EINGABEN**

`type` abzufragender Ladetyp

## 50.13 GetMemoryInfo

### BEZEICHNUNG

GetMemoryInfo – gibt Speicherinformationen zurück

### ÜBERSICHT

```
space = GetMemoryInfo(type)
```

### BESCHREIBUNG

Dieser Befehl gibt Informationen über den Speicherplatz in Ihrem System zurück. Die folgenden Konstanten können in **type** angegeben werden:

**#CHIPMEMORY:**

gibt die Größe des Chip-RAM zurück

**#FASTMEMORY:**

gibt die Größe des Fast-RAM zurück

### EINGABEN

**id** eine der oben aufgeführten Konstanten

### RÜCKGABEWERTE

**space** Speicherplatz

### BEISPIEL

```
chip=GetMemoryInfo(#CHIPMEMORY)
fast=GetMemoryInfo(#FASTMEMORY)
Print("You have", chip, "bytes of chip memory and", fast,
      "bytes of fast memory!")
```

Der obige Code wird den Chip- und Fast-RAM ausgeben.

## 50.14 GetSystemInfo

### BEZEICHNUNG

GetSystemInfo – gibt OS-spezifische Informationen zurück (V4.5)

### ÜBERSICHT

```
t = GetSystemInfo()
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um bestimmte Informationen von dem Betriebssystem abzufragen, auf dem Hollywood gerade ausgeführt wird. **GetSystemInfo()** gibt eine Tabelle zurück, die mehrere Felder enthält, die je nach Betriebssystem unterschiedlich sind.

Die folgenden Felder werden in der Rückgabetabelle **t** initialisiert:

**UserHome:**

Pfad zum Home-Verzeichnis des Benutzers. Wird auf Windows, macOS und Linux unterstützt. (V5.3)

**UserName:**

Name des aktuellen Benutzers. Wird auf Windows, macOS und Linux unterstützt. (V5.3)

**ProgramFiles:**

Pfad zum Verzeichnis der Programmdateien auf diesem Computer. Wird auf Windows (seit V4.5) und auf macOS (seit V5.3) unterstützt.

**AppData:** Pfad zum Anwendungsdatenordner für den aktuellen Benutzer auf diesem Computer. Wird auf Windows (seit V4.5), auf macOS (seit V5.3) und iOS (seit 7.0) unterstützt.

**CommonAppData:**

Pfad zum Anwendungsordner für alle Benutzer auf diesem Computer. Wird auf Windows und macOS unterstützt. (V6.1)

**LocalAppData:**

Pfad zum lokalen, Nicht-Roaming-Anwendungsdatenordner für den aktuellen Benutzer auf diesem Computer. Unterstützt wird das auf Windows und macOS. Unter macOS wird dies auf den gleichen Pfad wie **CommonAppData** gesetzt. (V9.0)

**MyDocuments:**

Pfad zum Ordner "Meine Dokumente" auf diesem Computer. Wird auf Windows (seit V4.5), unter macOS (seit V5.3) sowie iOS (seit 7.0) unterstützt.

**Windows:** Pfad zum Windows-Verzeichnis auf diesem Computer. Wird nur unter Windows unterstützt. (V4.5)

**SDCard:** Pfad zum externen Speichergerät. Dies wird aus früheren Gründen als **SDCard** bezeichnet, da das externe Speichergerät in den Anfängen von Android typischerweise eine SD-Karte war. Dieser Tag wird nur von Android unterstützt. Beachten Sie außerdem, dass Apps ab Android 6.0 nicht mehr ohne explizite Benutzererlaubnis aus diesem Ordner lesen und in diesen Ordner schreiben dürfen. Wenn Sie den Hollywood Player verwenden, fordert der Benutzer beim Start automatisch eine solche Berechtigung an. Wenn Sie eigenständige APKs mit dem Hollywood APK Compiler kompilieren, müssen Sie jedoch die Lese- und/oder Schreibberechtigung für diesen Ordner manuell anfordern, indem Sie den Befehl **PermissionRequest()** verwenden. Siehe [Abschnitt 23.7 \[PermissionRequest\]](#), [Seite 353](#), für Details. (V5.1)

**ExternalStorage:**

Pfad zum externen Speicherordner auf diesem Gerät. Wird nur auf Android unterstützt. (V5.1)

**InternalStorage:**

Pfad zum internen Speicherordner auf diesem Gerät. Wird nur auf Android unterstützt. (V5.1)

**AppBundle:**

Pfad zum Programmpaket des aktuellen Programms. Wird nur unter macOS (V6.1) und iOS (V7.0) unterstützt.

**Preferences:**

Der Pfad, den Hollywood verwendet, um die mit dem Befehl **SavePrefs()** Einstellungen zu speichern. (V7.1)



TempFiles:

Ein Pfad, in den Sie temporäre Dateien schreiben können. (V7.1)

#### EINGABEN

keine

#### RÜCKGABEWERTE

t eine Tabelle mit den oben beschriebenen Feldern

## 50.15 GetType

#### BEZEICHNUNG

GetType – untersucht eine Variable (V2.0)

#### ÜBERSICHT

```
type = GetType(var)
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Typ einer Variablen oder einem Wert herauszufinden. Mögliche Rückgabewerte sind #NUMBER, #STRING, #TABLE, #FUNCTION, #USERDATA, #LIGHTUSERDATA, #THREAD und #NIL.

Dieser Befehl wird häufig verwendet, um herauszufinden, ob eine Variable **Nil** ist. Ab Hollywood 6.0 gibt es aber auch einen neuen Komfortbefehl namens **IsNil()**, die auch zum prüfen von Variablen auf **Nil** verwendet werden kann.

Siehe [Abschnitt 8.1 \[Data types\]](#), [Seite 103](#), für Details.

#### EINGABEN

var Variable, die untersucht wird

#### BEISPIEL

```
type = GetType("Hello World")
```

Dies wird #STRING zurückgeben.

```
type = GetType({1, 2, 3, 4})
```

Das gibt #TABLE zurück.

```
type = GetType(Function() DebugPrint("Hello") EndFunction)
```

Dadurch wird #FUNCTION zurückgegeben.

## 50.16 GetVersion

#### BEZEICHNUNG

GetVersion – gibt Informationen über die Hollywood-Version und Plattform zurück (V3.0)

#### ÜBERSICHT

```
t = GetVersion()
```

## BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einige Informationen über die Hollywood-Version zu erhalten, die aktuell Ihr Skript oder Applet ausführt. Sie können auch Informationen über die Plattform abrufen, auf der das Skript ausgeführt wird.

Dieser Befehl gibt eine Tabelle zurück, wobei die folgenden Felder initialisiert wurden:

### Application:

Dies ist "Hollywood" oder der "HollywoodPlayer" je nachdem, welche Version verwendet wird. Wenn Sie Ihr Skript als ausführbare Datei speichern, ist dies immer der "HollywoodPlayer".

**Version:** Eine Zeichenfolge, die die Version von Hollywood enthält, z.B. "3.0".

### Version\_Date:

Eine Zeichenfolge, die das Erstellungsdatum dieser Hollywood-Version enthält.

**Kernel:** Eine Zeichenfolge, die die Kernelversion von Hollywood enthält, z.B. "3.0".

### Kernel\_Date:

Ein Zeichenfolge mit dem Erstellungsdatum des Hollywood-Kernel.

**Beta:** Dieses Feld wird auf **True** gesetzt, wenn eine Beta-Version von Hollywood verwendet wird, ansonsten **False**.

### Platform:

Dies ist wahrscheinlich das nützlichste Feld, denn es enthält die Plattform, auf der Hollywood derzeit läuft. Dieses Feld kann "AmigaOS3", "MorphOS", "WarpOS", "AmigaOS4", "AROS", "Win32", "MacOS", "Linux", "iOS" oder "Android" sein. Beachten Sie, dass "Win32" auch für 64-Bit-Windows zurückgegeben wird. Es wird aus historischen Gründen "Win32" genannt.

**Demo:** Dieses Feld wird auf **True** gesetzt, wenn der Benutzer eine Demoversion von Hollywood ausführt. (V4.71)

**Plugins:** Dieses Feld wird auf eine Tabelle festgelegt, die Informationen zu allen Plugins enthält, die von Hollywood für dieses Skript geladen wurden. Siehe [Abschnitt 44.3 \[GetPlugins\]](#), [Seite 985](#), für Details. (V5.1)

**CPU:** Dieses Feld wird auf die CPU-Architektur eingestellt, auf der Hollywood gerade ausgeführt wird. Dieses Feld kann "m68k", "m68k/ppc" (für WarpOS), "ppc", "arm", "arm64", "i386" oder "x64" beinhalten. (V5.2)

### BigEndian:

Dieses Feld wird auf **True** gesetzt, wenn Hollywood auf einer Big-Endian-CPU (z.B. 68000 oder PowerPC) oder **False** für eine Little-Endian-CPU läuft (x86, x64, und ARM). (V6.0)

## EINGABEN

keine

## RÜCKGABEWERTE

**t** eine Tabelle mit Informationen über die verwendete Hollywood-Version

**BEISPIEL**

```
t = GetVersion()  
If t.platform = "Win32" Then Error("Sorry, Win32 is not supported yet!")
```

Der obige Code überprüft, welche Version läuft und wird mit einem Fehler beendet, wenn Hollywood auf Windows läuft.

**50.17 IF****BEZEICHNUNG**

IF – prüft die Bedingung (V7.0)

**ÜBERSICHT**

@IF val

**BESCHREIBUNG**

Diese Präprozessor-Anweisung prüft für die in `val` angegebene Bedingung. Wenn sie `True` ist, wird der Präprozessor das Abarbeiten des Skripts in diesem Block fortsetzen. Wenn `val` hingegen `False` ist, wird @IF zum nächsten @ELSEIF, @ELSE, oder @ENDIF-Anweisung verzweigen, um bestimmte Zeilen des Skripts zu überspringen, wenn Bedingungen nicht erfüllt sind.

Beachten Sie, dass die Bedingung ein konstanter Ausdruck sein muss, da @IF auf Präprozessor-Ebene arbeitet, womit Skriptvariablen zu diesem Zeitpunkt nicht verfügbar sind. Sie können numerische Konstanten oder Hollywood-Konstanten verwenden, die entweder mit der Anweisung `Const` oder dem Konsolenargument `-setconstants` definiert wurden. Es gibt auch einige integrierte Konstanten, die es Ihnen erlauben, die Plattform (und Hollywood-Version) zu ermitteln, auf welcher Hollywood/Ihr Programm gerade läuft. Bitte siehe Liste unten.

Beachten Sie auch, dass die Präprozessor-Anweisung @IF und @ELSEIF nicht melden, wenn Sie eine Konstante verwenden, die überhaupt nicht deklariert wurde. Sie erhalten in diesem Fall keine "Konstante nicht gefunden!" Fehlermeldung. Stattdessen werden die nicht deklarierten Konstanten einfach auf 0 ausgewertet. Der Grund für diesen Entwurf ist, dass die plattformspezifischen Konstanten (siehe unten) nur bei der Ausführung eines Hollywood-Skripts auf der jeweiligen Plattform oder beim Kompilieren definiert werden. Wenn Sie also auf einer bestimmten Plattform auf eine andere Plattform testen, erhalten Sie keinen Fehler, wenn die Plattformkonstante nicht definiert wurde. Stattdessen wird die nicht existierende Konstante einfach auf 0 ausgewertet.

Im Gegensatz zur normalen **If-Anweisung** arbeitet @IF auf Präprozessor-Ebene. Somit können Sie @IF verwenden, um den Präprozessor zu zwingen, bestimmte Routen zu nehmen oder bestimmte Teile Ihres Codes zu ignorieren. Dies ist bei der normalen **If-Anweisung** überhaupt nicht möglich, denn sobald diese **If-Anweisung** ausgeführt wird, hat der Präprozessor seinen Job bereits beendet.

Zum Beispiel könnten Sie den Präprozessor zwingen, verschiedenen Code zu analysieren, je nachdem, auf welcher Plattform Hollywood läuft. Sie können dem Präprozessor auch sagen, dass er bestimmte Teile des Codes ignoriert. Diese Teile werden nicht einmal auf syntaktische Korrektheit geprüft. Sie werden völlig übersprungen, genau wie Kommentare, so könnte alles Mögliche in diesen Blöcken sein.

Hier ist ein Beispiel, das ein anderes Hintergrundbild und einen anderen Fenstertitel für die einzelnen Plattformen verwendet, die von Hollywood unterstützt werden:

```
@IF #HW_AMIGA
    @BGPIC 1, "bg_amiga.png"
    @DISPLAY {Title = "My project (Amiga)"}
@ELSEIF #HW_MACOS
    @BGPIC 1, "bg_macos.png"
    @DISPLAY {Title = "My project (macOS)"}
@ELSEIF #HW_LINUX
    @BGPIC 1, "bg_linux.png"
    @DISPLAY {Title = "My project (Linux)"}
@ELSEIF #HW_WINDOWS
    @BGPIC 1, "bg_windows.png"
    @DISPLAY {Title = "My project (Windows)"}
@ELSE
    @BGPIC 1, "bg_default.png"
    @DISPLAY {Title = "My project (Default)"}
@ENDIF
```

Für diesen Zweck könnten Sie die normale *If-Anweisung* nicht verwenden, weil, wie der Name schon sagt, der Präprozessor das Skript analysiert, bevor es ausgeführt wird. Wenn Sie also die normale *If-Anweisung* verwendet haben, werden alle Präprozessor-Anweisung im obigen Code analysiert und ausgeführt, weil die Präprozessor-Anweisungen die Laufzeitanweisungen wie die normale *If-Anweisung* ignorieren würde.

Wie Sie oben sehen können, gibt es einige integrierte Konstanten, die es Ihnen erlauben, die Plattform zu überprüfen, auf der Hollywood gerade läuft oder kompiliert. Folgende integrierte Konstanten stehen zur Verfügung:

```
#HW_AMIGA
    Definiert AmigaOS oder kompatible Plattformen, z.B. AmigaOS, MorphOS,
    AROS, WarpOS.

#HW_AMIGAOS3
    Definiert AmigaOS 3.

#HW_AMIGAOS4
    Definiert AmigaOS 4.

#HW_ANDROID
    Definiert Android.

#HW_AROS
    Definiert AROS.

#HW_IOS
    Definiert iOS.

#HW_LINUX
    Definiert Linux.

#HW_MACOS
    Definiert macOS.

#HW_MORPHOS
    Definiert MorphOS.
```

**#HW\_WARPOS**  
Definiert WarpOS.

**#HW\_WINDOWS**  
Definiert Windows.

**#HW\_LITTLE\_ENDIAN**  
Definiert alle Little-Endian-Systeme.

**#HW\_64BIT**  
Definiert 64-Bit Systeme.

**#HW\_VERSION**  
Enthält den Ganzzahlenteil der Hollywood-Versionsnummer.

**#HW\_REVISION**  
Enthält die Nachkommazahl der Hollywood-Versionsnummer.

Beachten Sie, dass Hollywood bei der Kompilierung Ihres Skripts in eine ausführbare Datei automatisch die Konstanten der Zielarchitektur festlegt. Zum Beispiel wenn Sie Ihr Skript unter Windows für AmigaOS 3 kompilieren, wird Hollywood die Konstanten **#HW\_AMIGA** und **#HW\_AMIGAOS3** setzen. Die Konstanten **#HW\_WINDOWS** und **#HW\_LITTLE\_ENDIAN** werden nicht gesetzt. Sie werden nur gesetzt, wenn Sie für Windows kompilieren oder Ihr Skript auf einem Windows-Rechner ausführen.

Beachten Sie auch, dass bei der Erstellung von Applets keine der oben genannten architektonischen Konstanten gesetzt werden. Applets sind vollplattform-agnostisch, also wenn man ein Applet kompiliert, wird keine der architektonischen Konstanten gesetzt. Eine Ausnahme ist, wenn Sie ein Applet explizit für Android oder iOS kompilieren, indem Sie "android" oder "ios" an das Konsolenargument **-exetype** übergeben. In diesem Fall werden **#HW\_ANDROID** bzw. **#HW\_IOS** gesetzt. Wenn Sie ein plattformunabhängiges Applet kompilieren, indem Sie "applet" an das Argument **-exetype** übergeben, wird keine der architektonischen Konstanten gesetzt.

## EINGABEN

**val**            Bedingung, welche geprüft wird; muss ein konstanter Ausdruck sein

## BEISPIEL

```
@IF #HW_VERSION >= 7
...
@ENDIF
```

Der oben genannte Code teilt dem Präprozessor nur den folgenden Code zu, wenn wir mindestens Version 7 von Hollywood haben.

## 50.18 IIf

### BEZEICHNUNG

IIf – gibt einen Wert zurück, abhängig von der Bedingung (V2.0)

### ÜBERSICHT

```
ret = IIf(expr, true_expr, false_expr)
```

**BESCHREIBUNG**

Dieser Befehl prüft, ob der angegebene Ausdruck in **expr** **True** (ungleich Null) ist. Wenn dies der Fall ist, gibt der Befehl den in **true\_expr** angegebenen Ausdruck zurück, andernfalls wird **false\_expr** zurückgegeben. Es ist wichtig zu beachten, dass sowohl **TRUE**-als auch **FALSE**-Ausdrücke in jedem Fall ausgewertet werden, unabhängig davon, ob der Ausdruck **True** oder **False** ist. So wird z.B. **IIf(a <> 0, 5 / a, 0)** nicht funktionieren, wenn **a=0** ist, da Hollywood versucht **5/0** auszuwerten.

**EINGABEN**

**expr**            Quell-Ausdruck

**true\_expr**        Ausdruck, der zurückgegeben werden soll, wenn **expr** **True** ist

**false\_expr**        Ausdruck, der zurückgegeben werden soll, wenn **expr** **False** ist

**RÜCKGABEWERTE**

**ret**            Ergebnis

## 50.19 INCLUDE

**BEZEICHNUNG**

**INCLUDE** – importiert Code aus externen Skriptdateien oder Applets (V2.0)

**ÜBERSICHT**

**@INCLUDE file\$**

**BESCHREIBUNG**

Diese Präprozessor-Anweisung importiert den kompletten Code aus der in **file\$** angegebene Datei in das aktuelle Projekt. Der Code wird an der Stelle eingefügt, wo **@INCLUDE** im Skript definiert wurde. Normalerweise verwenden Sie **@INCLUDE** am Anfang des Projekts.

Diese Präprozessor-Anweisung ist nützlich, wenn Sie größere Projekte haben und diese über mehrere Dateien verteilen möchten. Die Include-Dateien beinhalten in der Regel Funktionen, die Sie dann von Ihrem Hauptskript aus aufrufen können. Include-Dateien können auch Präprozessor-Anweisungen beinhalten.

Seit Hollywood 4.0 können Sie auch Hollywood-Applets mit dieser Anweisung einfügen. Dies ist für den Import von Code aus Libraries nützlich.

Siehe [Abschnitt 7.6 \[Code einbinden\]](#), [Seite 92](#), für Details.

**EINGABEN**

**file\$**            die zu importierende Skript-Datei

**BEISPIEL**

```

;---Datei: script2.hws---
Function p_Print(t$)
    Print(t$)
EndFunction

```

```

;EOF script2.hws

;---Datei: Hauptskript.hws
@INCLUDE "script2.hws"

p_Print("Hello World!")
WaitLeftMouse
End
;EOF Hauptskript.hws

```

Der obige Code besteht aus zwei Skripten: script2.hws enthält die Funktion p\_Print(), die einfach den Befehl `Print()` von Hollywood aufruft. script2.hws wird dann in das Hauptskript aufgenommen, welches dann auch die Funktion p\_Print() aufruft.

## 50.20 IsNil

### BEZEICHNUNG

IsNil – prüft, ob eine Variable Nil ist (V6.0)

### ÜBERSICHT

```
bool = IsNil(var)
```

### BESCHREIBUNG

Dieser Befehl prüft, ob die in `var` angegebene Variable Nil ist. Zusammen mit `GetType()` ist `IsNil()` die einzige zuverlässige Möglichkeit, um herauszufinden, ob eine Variable Nil ist oder nicht. Eine Überprüfung der Variablen gegen Nil mit "`a = Nil`" ist nicht zuverlässig, da dies auch zu `True` führen wird, wenn die Variable null anstatt Nil ist. Beispiel:

```

a = 0
b = Nil
DebugPrint(IsNil(a), a = Nil) ; prints "0 1"
DebugPrint(IsNil(b), b = Nil) ; prints "1 1"

```

Sie sehen dass "`a = Nil`" `True` zurückgibt, obwohl es Null ist und nicht Nil. Das ist, weil Nil immer als Null betrachtet wird, wenn es in Ausdrücken verwendet wird. Also, wenn Sie herausfinden wollen, ob eine Variable wirklich Nil ist, verwende immer `IsNil()` oder `GetType()`.

Siehe [Abschnitt 8.1 \[Datentypen\]](#), [Seite 103](#), für Details.

### EINGABEN

`var` Variable, welche untersucht wird

### RÜCKGABEWERTE

`bool` True, wenn die Variable Nil ist, andernfalls False

## 50.21 IsUnicode

### BEZEICHNUNG

IsUnicode – prüft, ob Hollywood im Unicode-Modus ist (V7.0)

**ÜBERSICHT**

```
bool = IsUnicode()
```

**BESCHREIBUNG**

Dieser Befehl gibt **True** zurück, wenn Hollywood derzeit im Unicode-Modus ist, ansonsten **False**. Da Skripte immer im Unicode-Modus laufen sollten, ist dieser Befehl wahrscheinlich nicht oft benutzt.

**EINGABEN**

keine

**RÜCKGABEWERTE**

bool          **True**, wenn Hollywood im Unicode-Modus ist, ansonsten **False**

## 50.22 LegacyControl

**BEZEICHNUNG**

LegacyControl – aktiviert oder deaktiviert bestimmte alte Funktionalitäten (V6.0)

**ÜBERSICHT**

```
LegacyControl(feature$, enable)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um bestimmte alte Funktionalitäten zu aktivieren oder deaktivieren, die Hollywood noch aus Kompatibilitätsgründen unterstützt. Sie müssen den Namen der Funktionalität, die Sie ansprechen möchten, sowie **True** übergeben, um sie zu aktivieren, oder **False**, um sie zu deaktivieren.

Die folgenden Zeichenfolgen können derzeit in **feature\$** übergeben werden:

**"SingleMusic":**

Vor Hollywood 6.0 konnte nur eine Musik gleichzeitig abgespielt werden. Diese Einschränkung wurde mit Hollywood 6.0 entfernt, aber standardmäßig stoppt **PlayMusic()** alle Musikwiedergaben, um voll kompatibel mit früheren Versionen zu sein. Wenn Sie das nicht wollen, müssen Sie **LegacyControl()** aufrufen und **SingleMusic** auf **False** setzen. Hollywood wird dann in der Lage sein, mehrere Musikobjekte zur gleichen Zeit abzuspielen. Dieser Tag ist standardmäßig auf **True** gesetzt.

**"LineBasedShapes":**

Vor Hollywood 6.0 wurden alle durch die Befehle **Arc()**, **Circle()**, **Ellipse()** und **Box()** gezeichneten Formen (bei Verwendung des optionalen Parameters, um ein Rechteck mit runden Ecken zu zeichnen) als Polygone gezeichnet, die sie ziemlich quadratisch aussehen liessen. Ab Hollywood 6.0 werden diese runden Formen als echte Bézier-Splines gezeichnet, wenn Antialiasing eingeschaltet ist. Dies wird ein wenig langsamer als der Polylinie-basierte Ansatz aus früheren Versionen sein, aber es wird viel besser aussehen. Wenn Sie möchten, dass Hollywood den polygonbasierten Ansatz aus früheren Versionen verwendet, können Sie diesen Tag auf **True** setzen. In diesem Fall werden die oben aufgeführten Befehle wie früher als Polygone gezeichnet.



Dieser Tag wird standardmäßig auf **False** gesetzt, außer wenn ein Applet ausgeführt wird, welches von Hollywood-Versionen älter als 6.0 kompiliert wurde. In diesem Fall wird standardmäßig dieser Tag auf **True** gesetzt.

## EINGABEN

**feature\$** Name der Fähigkeit, die aktiviert oder deaktiviert werden soll (siehe oben)  
**enable** **True** um die Fähigkeit zu aktivieren, **False** zu deaktivieren

## 50.23 LINKER

### BEZEICHNUNG

LINKER – übergibt Optionen an den Linker (V8.0)

### ÜBERSICHT

@LINKER table

### BESCHREIBUNG

Diese Präprozessor-Anweisung kann verwendet werden, um Optionen an Hollywoods Linker zu übergeben. Daher wird sie nur verwendet, wenn Skripts zu Applets oder ausführbaren Programmen kompiliert werden. Wenn Sie nur Skripts mit dem Hollywood-Interpreter ausführen, wird diese Präprozessor-Anweisung ignoriert.

Sie müssen eine Tabelle an die Präprozessor-Anweisung @LINKER übergeben. Folgende Tags werden derzeit erkannt:

**Files:** Mit diesem Tag können Sie eine Liste von Dateien übergeben, die mit dem Applet oder der ausführbaren Datei verlinkt (eingebunden) werden sollen. Dies ähnelt dem Konsolenargumenten **-linkfiles**, jedoch können Sie die zu verknüpfenden Dateien direkt in Ihrem Skript speichern. Dies ist möglicherweise praktischer, als dass Sie zu diesem Zweck eine externe Datenbankdatei verwalten müssen. Sie müssen eine Tabelle mit einer Liste von Dateien übergeben, die mit diesem Tag verknüpft werden sollen. Beachten Sie, dass es sehr wichtig ist, dass die in **Files** übergebenen Dateinamensangaben genau mit den an Hollywood-Befehlen übergebenen Dateipfade übereinstimmen, die dann die verknüpften Dateien laden. Wenn Sie keine identischen Dateipfade verwenden, kann Hollywood die mit dem Applet oder der ausführbaren Datei verknüpften Dateien nicht den entsprechenden Befehlen zuordnen. Siehe [Abschnitt 4.3 \[Einbinden von Datendateien\]](#), [Seite 58](#), für Details.

**Fonts:** Mit diesem Tag können Sie eine Liste von Schriftarten übergeben, die mit dem Applet oder der ausführbaren Datei verlinkt (eingebunden) werden sollen. Dies ähnelt dem Konsolenargumenten **-linkfonts**, jedoch können Sie die zu verknüpfenden Schriftarten direkt in Ihrem Skript speichern. Dies ist möglicherweise praktischer, als dass Sie dazu eine externe Datenbankdatei verwalten müssen. Ihr Skript lädt dann automatisch die verknüpften Schriftarten aus Ihrem Applet oder der ausführbaren Datei, wenn Sie **SetFont()** aufrufen. Die Verwendung des Tags **Fonts** der Präprozessor-Anweisung @LINKER zum Verknüpfen von Schriftarten in Applets oder ausführbaren Dateien ist

eine Alternative zur Verwendung der Präprozessor-Anweisung **@FONT**. Normalerweise sollte die Verwendung von **@FONT** jedoch wesentlich einfacher sein als die Verwendung von **Fonts**. Daher sollten Sie **Fonts** nur aus guten Gründen verwenden. Siehe [Abschnitt 4.4 \[Einbinden von Schriftarten\]](#), [Seite 60](#), für Details.

**Wichtiger Hinweis:** Bitte beachten Sie, dass die meisten Schriften urheberrechtlich geschützt sind und es nicht erlaubt ist, sie in Ihre Programme einzubinden/zum Verknüpfen, ohne eine Lizenz zu erwerben. Siehe [Abschnitt 4.4 \[Einbinden von Schriftarten\]](#), [Seite 60](#), für Details.

## EINGABEN

**table**      Tabelle mit Optionen, die an den Linker übergeben werden sollen

## BEISPIEL

```
@LINKER {Files = {"test.jpg", "title.png"},
         {Fonts = {"Arial", "Times New Roman"}}
```

```
LoadBGPic(1, "test.jpg")
LoadBrush(1, "title.png")
SetFont("Arial", 36)
NPrint("Hello World")
SetFont("Times New Roman", 72)
NPrint("Hello Hollywood")
```

Der obige Code verknüpft die Dateien `test.jpg` und `title.png` mit dem Applet oder der ausführbaren Datei. `LoadBGPic()` und `LoadBrush()` laden dann die Dateien direkt aus dem Applet oder der ausführbaren Datei anstelle einer externen Quelle. Darüber hinaus werden die Zeichensätze `Arial` und `Times New Roman` in das Applet oder in die ausführbare Datei eingebunden. `SetFont()` öffnet diese Zeichensätze auch direkt vom Applet oder der ausführbaren Datei aus.

## 50.24 OpenURL

### BEZEICHNUNG

`OpenURL` – öffnet die URL im Standard-Webbrowser (V4.5)

### ÜBERSICHT

`OpenURL(url$)`

### BESCHREIBUNG

Dieser Befehl öffnet die in `url$` angegebene Adresse im Standard-Webbrowser.

### EINGABEN

**url\$**      URL, welche geöffnet werden soll

### BEISPIEL

```
OpenURL("http://www.airsoftsoftwair.com/")
```

Der obige Code führt Sie direkt auf die Hauptseite von Airsoft Softwair.

## 50.25 OPTIONS

### BEZEICHNUNG

OPTIONS – konfiguriert verschiedene Optionen (V4.5)

### ÜBERSICHT

@OPTIONS table

### BESCHREIBUNG

Mit dieser Präprozessor-Anweisung können Sie verschiedene allgemeine Optionen konfigurieren. Sie müssen diesem Befehl eine Tabelle übergeben, die definiert, welche Optionen Sie konfigurieren möchten.

Die folgenden Tags werden derzeit erkannt:

#### LockSettings:

Dieser Tag hat die gleiche Funktionalität wie die **Konsolenargumente** mit demselben Namen. Wenn Sie **LockSettings** auf **True** setzen, kompiliert Hollywood Programme, wird aber keine Argumente von der **Konsole** oder der **Pseudo-Konsole** akzeptieren. Der einzige Unterschied zum Konsolenargument ist, dass wenn Sie **LockSettings** in dieser Präprozessor-Anweisung verwenden, es auch alle Benutzeränderungen beim Ausführen von Hollywood-Skripten verbieten wird. D.h. wenn Sie hier **LockSettings** benutzt haben, verwendet Ihr Skript immer den Stil, wie er in den Präprozessor-Anweisungen definiert wurde. Sie können den Stil nicht ändern, indem Sie Argumente wie **Borderless** oder **Sizeable** an das Skript übergeben.

#### SoftTimer:

Wenn Sie diesen Tag auf **True** setzen, verwendet Hollywood einen Software-Timer mit niedriger Auflösung anstelle des hochauflösenden Hardware-Timers. Dies ist manchmal notwendig, da bei bestimmter älterer Windows XP-Hardware der Timer gelegentlich springen kann, was zu unerwartetem Verhalten führt. Dieser Tag wird nur auf der Windows-Plattform unterstützt. (V5.3)

#### NoCommodity:

Wenn Sie diesen Tag auf **True** setzen, wird sich Hollywood unter AmigaOS nicht selber zur Liste der Commodities in Exchange hinzufügen. Dieser Tag wird nur auf AmigaOS und kompatiblen Systemen unterstützt. Voreinstellung ist **False**. (V6.0)

#### RegisterApplication:

Wenn Sie diesen Tag auf **True** setzen, registriert sich Hollywood beim Start als AmigaOS 4 Anwendung durch die `application.library`. Dies ist notwendig, wenn Sie Befehle aufrufen möchten, die die spezifischen Programmfunktionen von AmigaOS 4 verwenden wie **SendApplicationMessage()**, oder wenn Sie wollen, dass Ihre Anwendung in AmiDock erscheint. Um das Piktogramm zu ändern, welches in AmiDock angezeigt wird, verwenden Sie die Präprozessor-Anweisung **@APPICON**. Dieser Tag ist nur auf AmigaOS 4 verfügbar. Voreingestellt ist **False**. (V6.0)

**DockyContextMenu:**

Mit diesem Tag können Sie die ID einer Menüleiste angeben, die in AmigaOS 4 als Kontextmenü für das Docky Ihrer Anwendung verwendet werden soll. Die hier angegebene Menüleiste muss nur einen einzelnen Menübaum ohne Untermenüs enthalten. Tastenkürzel in Menüleisten werden nicht unterstützt, weil sie in einem nicht immer sichtbaren Kontextmenü keinen Sinn machen. Bitte beachten Sie, dass die Einstellung dieses Tags automatisch Ihre Anwendung als App-Docky in AmiDock erscheinen lässt. Weitere Informationen zum Unterschied zwischen App- und Standard-Dockies finden Sie unter [AmiDock-Informationen](#). Dieser Tag wird nur berücksichtigt, wenn Sie `RegisterApplication` auf `True` gesetzt haben. Wird nur auf AmigaOS 4 unterstützt. (V6.0)

**DockyBrush:**

Mit diesem Tag können Sie die ID eines Pinsels angeben, der auf AmigaOS 4 als Piktogramm Ihrer Anwendung im AmiDock angezeigt werden soll. Normalerweise würden Sie die Präprozessor-Anweisung `@APPICON` verwenden, um das AmiDock-Piktogramm Ihrer Anwendung zu konfigurieren, aber der Tag `DockyBrush` kann in einer der folgenden Situationen nützlich sein: Erstens können Sie mit `DockyBrush` einen beliebigen Pinsel angeben und sind daher nicht auf die vordefinierten Größen von `@APPICON` beschränkt. Stattdessen kann das Docky-Piktogramm Ihrer Anwendung eine beliebige Größe haben. Zweitens, wenn Sie `DockyBrush` verwenden, wird Hollywood automatisch eine App-Docky für Sie erstellen, während mit der Präprozessor-Anweisung `@APPICON` eine Standard-Docky erstellt würde (solange Ihr Docky kein Kontextmenü enthält). Siehe [Abschnitt 16.1 \[AmiDock information\], Seite 173](#), für mehr Details zum Unterschied zwischen App- und Standard-Dockies. Dieser Tag wird nur berücksichtigt, wenn `RegisterApplication` auf `True` gesetzt ist und wird offensichtlich nur auf AmigaOS 4 unterstützt. (V6.0)

**NoDocky:** Wenn dieser Tag auf `True` gesetzt ist, wird Hollywood Ihre Anwendung nicht im AmiDock anzeigen. Dieser Tag ist nützlich, wenn Sie eine unsichtbare Anwendung haben möchten, die alle Anwendungsfunktionalitäten wie das Nachrichtensystem und Ringhio nutzen kann, aber nicht in AmiDock erscheint. Dieser Tag wird nur auf AmigaOS 4 unterstützt und verwendet, wenn `RegisterApplication` auf `True` gesetzt wurde. (V6.0)

**Encoding:**

Mit diesem Feld können Sie die Zeichencodierung des Skripts festlegen. Beachten Sie, dass Sie diese Anweisung am Anfang Ihres Skripts setzen müssen, da es sonst Probleme geben wird. Folgende Zeichencodierungen werden derzeit unterstützt:

**#ENCODING\_UTF8:**

Die Zeichencodierung vom Skript ist UTF-8 (mit oder ohne BOM (Byte Order Mark/Bytereihenfolge-Markierung)). Dies ist auch die Voreinstellung und sollte wann immer möglich verwendet werden.

**#ENCODING\_ISO8859\_1:**

Die Zeichencodierung des Skripts ist ISO 8859-1. Beachten Sie, dass aus historischen Gründen Hollywood die Zeichencodierung ISO 8859-1 nicht auf AmigaOS und kompatible verwendet, sondern die System-Standard-Zeichencodierung. **#ENCODING\_ISO8859\_1** wird Hollywood in den Legacy-Modus versetzen und somit sollte Ihr Skript vollständig kompatibel mit Hollywood-Versionen älter als 7.0 sein. Da jedoch der ISO 8859-1-Modus mehrere Nachteile aufweist, wird nicht empfohlen, diesen Legacy-Modus dauerhaft zu verwenden. Stattdessen sollten Sie Ihre Skripte anpassen, um im Unicode-Modus korrekt zu arbeiten.

Beachten Sie, dass nicht empfohlen wird, **#ENCODING\_ISO8859\_1** zu verwenden, da Hollywood nur dann korrekt auf Systemen läuft, die mit westeuropäischen Sprachen kompatibel sind. Sie sollten immer **#ENCODING\_UTF8** verwenden, denn das wird Hollywood in den Unicode-Modus versetzen und dafür sorgen, dass Hollywood korrekt auf allen Systemen läuft. Da **#ENCODING\_UTF8** auch die Voreinstellung ist, müssen Sie das Feld **Encoding** normalerweise nicht verwenden.

Die hier angegebene Codierung wird automatisch als Standardcodierung für die Text- und Zeichenketten-Bibliothek mit dem Befehl **SetDefaultEncoding()** gesetzt. Dies bedeutet, dass alle Befehle der Zeichenketten- und Textbibliotheken auf diese Codierung zurückgreifen. (V7.0)

**NoChDir:** Standardmäßig wechselt Hollywood immer das aktuelle Verzeichnis des Skripts oder Applets, das gerade ausgeführt wird. Übergeben Sie dieses Argument, wenn Sie dieses Verhalten nicht wollen. In diesem Fall wird Hollywood das aktuelle Verzeichnis nicht ändern, wenn ein Skript ausgeführt wird. (V7.1)

**EnableDebug:**

Wenn dieser Tag auf **False** gesetzt ist, werden die Befehle **DebugPrint()**, **DebugPrintNR()**, **Assert()**, **DebugOutput()** und **@WARNING** ignoriert. So wird es Ihnen ermöglicht, Debugging-Befehle mit nur einem einzigen Aufruf global zu deaktivieren. Beim Kompilieren von Skripten wird Hollywood **EnableDebug** standardmäßig auf **False** setzen. Dies ist die empfohlene Einstellung, da sie Leute daran hindert, Ihre Projekte rückzuentwickeln. Sie können somit die Debug-Ausgabe nicht aktivieren, indem sie das Konsolenargument **-debugoutput** angeben. Wenn Sie Skripte ausführen, wird **EnableDebug** standardmäßig auf **True** gesetzt, damit Sie Ihre Skripte debuggen können. (V7.1)

**GlobalPlugins:**

Auf AmigaOS und kompatiblen Systemen können Plugins auch global in **LIBS:Hollywood** installiert werden. Von Hollywood kompilierte Programme laden jedoch nur die Plugins, die neben Hollywood in ihrem Verzeichnis gespeichert sind. Wenn Sie möchten, dass Ihr Programm auch alle Plugins in

**LIBS:Hollywood** lädt, müssen Sie den Tag **GlobalPlugins** auf **True** setzen. Offensichtlich wird dieser Tag nur auf AmigaOS und kompatiblen Plattformen unterstützt. (V9.0)

**DPIAware:**

Dieser Tag wird nur unter Windows unterstützt. Wenn Sie ihn auf **True** setzen, startet Hollywood im DPI-Awareness-Modus. Das bedeutet, dass es das Betriebssystem nicht auffordert, Hollywood automatisch zu skalieren, um es an den DPI-Wert des Monitors anzupassen. Wenn **DPIAware** auf **False** gesetzt ist (was auch die Voreinstellung ist), wendet Hollywood die Skalierung auf Monitoren mit hohem DPI-Wert automatisch an, damit das Display auf ihnen nicht zu klein erscheint. Ein Display mit 640 x 480 Pixeln wird beispielsweise auf einem Monitor mit hohem DPI-Wert wirklich winzig erscheinen. Durch die automatische Anpassung des Displays an die DPI des Monitors versucht Hollywood dies zu vermeiden. Diese Skalierung kann jedoch dazu führen, dass Displays auf Monitoren mit hohem DPI unscharf erscheinen. Wenn Sie das nicht möchten, setzen Sie **DPIAware** auf **True**. Beachten Sie jedoch, dass Sie darauf achten müssen, dass Ihr Display dann auf Monitoren mit hoher DPI korrekt angezeigt wird. Dies können Sie beispielsweise durch Setzen des Tags **SystemScale** in der Präprozessor-Anweisung **@DISPLAY** tun. Beachten Sie zudem, dass **DPIAware** nur unter Windows unterstützt wird. Auf allen anderen Plattformen ist Hollywood immer DPI-fähig. (V9.0)

**ConsoleMode:**

Wenn Sie diesen Tag auf **True** setzen, kompiliert Hollywood ein Programm, das unter Windows im Konsolenmodus ausgeführt wird. Unter Windows wird zwischen Konsolen- und Windows-Programmen unterschieden. Wenn Sie also ein Programm für die Konsole kompilieren möchten, müssen Sie Hollywood explizit dazu auffordern. Sie können dies tun, indem Sie diesen Tag auf **True** setzen. Beachten Sie, dass dieser Tag offensichtlich nur verarbeitet wird, wenn Programme für Windows mit Hollywood kompiliert werden. Ansonsten wird es einfach ignoriert. Die Voreinstellung ist **False**. (V9.0)

**Quiet:**

Wenn Sie diesen Tag auf **True** setzen, zeigt Hollywood nicht seine traditionelle Startausgabe an, sondern startet ohne. Wenn Sie Hollywood-Applets mit **Quiet** auf **True** setzen und den Hollywood Player für Amiga und kompatible Systeme benutzen, öffnet der Player auch nicht sein Startfenster. Die Voreinstellung ist **False**. (V9.0)

## EINGABEN

**table**      Tabelle mit den gewünschten Optionen (siehe oben)

## 50.26 SetDefaultAdapter

### BEZEICHNUNG

SetDefaultAdapter – stellt den Standardadapter ein (V10.0)

## ÜBERSICHT

`SetDefaultAdapter(type, adapter$)`

## BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Standardadapter für den durch `type` angegebenen Typ auf den durch `adapter$` angegebenen Adapter zu setzen. Alle Hollywood-Befehle, die Adapter des angegebenen Typs unterstützen, verwenden dann standardmäßig den Adapter, der dieser Befehl verwendet.

Die folgenden Adaptertypen werden derzeit von Hollywood unterstützt und können in `type` übergeben werden:

### #ADAPTER\_FILE:

Datei-, Verzeichnis- und Dateisystemadapter. Sie werden verwendet, um Dateien und Verzeichnisse zu öffnen und das Dateisystem zu handhaben. Sie können an allen Befehlen übergeben werden, die sich mit Dateien befassen.

### #ADAPTER\_NETWORK:

Netzwerkadapter. Sie können verwendet werden, um Hollywoods integrierten Netzwerk-Handler zu ersetzen. Zu den Befehlen, die Netzwerkadapter unterstützen, gehören `OpenConnection()` und `DownloadFile()`.

### #ADAPTER\_SERIALIZE:

Serialisierungs-Adapter. Sie werden verwendet, um Daten zu serialisieren und zu deserialisieren. Befehle, die Serialisierungs-Adapter unterstützen, sind unter anderem `SerializeTable()` und `DeserializeTable()`.

Der Standardadapter für alle oben aufgeführten Adaptertypen ist `default`, was bedeutet, dass Plugin-Adapter immer vor Hollywoods integrierten Adapter abgefragt werden.

Die Zeichenkette, die Sie in `adapter$` übergeben, kann einen oder mehrere Adapter enthalten. Werden mehrere Adapter angegeben, müssen diese durch `|`-Zeichen getrennt werden. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details.

## EINGABEN

`type` zu verwendender Adaptertyp (siehe oben für mögliche Werte)

`adapter$` neuer Standardwert für den angegebenen Adapter

## BEISPIEL

```
SetDefaultAdapter(#ADAPTER_FILE, "inbuilt")
OpenFile(1, "test.txt")
```

Der obige Code setzt den Standarddateiadapter auf `inbuilt`. Das bedeutet, dass alle Hollywood-Befehle, die sich mit Dateien befassen, keine Plugin-Dateiadapter mehr unterstützen, weil Sie Hollywood angewiesen haben, nur den integrierten Dateiadapter zu verwenden.

## 50.27 SetDefaultLoader

### BEZEICHNUNG

`SetDefaultLoader` – stellt den Standardlader ein (V10.0)



## ÜBERSICHT

`SetDefaultLoader(type, loader$)`

## BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den Standardlader für den durch `type` angegebenen Typ auf den durch `loader$` angegebenen Lader zu setzen. Alle Hollywood-Befehle, die Laderprogramme des angegebenen Typs unterstützen, verwenden dann standardmäßig das Laderprogramm, das dieser Befehl verwendet.

Die folgenden Ladertypen werden derzeit von Hollywood unterstützt und können in `type` übergeben werden:

### #LOADER\_IMAGE:

Bildlader, die beispielsweise von Befehlen wie `LoadBrush()` verwendet werden.

### #LOADER\_ANIM:

Animationslader, die zum Beispiel von Befehlen wie `OpenAnim()` verwendet werden.

### #LOADER\_SOUND:

Soundlader, die zum Beispiel von Befehlen wie `OpenMusic()` verwendet werden.

### #LOADER\_VIDEO:

Videolader, die beispielsweise von Befehlen wie `OpenVideo()` verwendet werden.

### #LOADER\_ICON:

Piktogrammmlader, die beispielsweise von Befehlen wie `LoadIcon()` verwendet werden.

### #LOADER\_FONT:

Schriftartenlader, die zum Beispiel von Befehlen wie `OpenFont()` verwendet werden.

Der Standardlader für alle oben aufgeführten Ladertypen ist `default`, was bedeutet, dass Hollywood zuerst Plugin-, dann integrierte und zum Schluss native Lader abfragt, um eine Datei zu öffnen. Eine Ausnahme ist `#LOADER_FONT`, wo aus Kompatibilitätsgründen nur native Ladeprogramme abgefragt werden, die Schriftart zu öffnen.

Die Zeichenkette, die Sie in `loader$` übergeben, kann einen oder mehrere Lader enthalten. Werden mehrere Lader angegeben, müssen diese durch `|`-Zeichen getrennt werden. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details.

## EINGABEN

`type`        zu verwendender Ladertyp (siehe oben für mögliche Werte)  
`loader$`    neuer Standardwert für den angegebenen Lader

## BEISPIEL

```
SetDefaultLoader(#LOADER_IMAGE, "plugin")
LoadBrush(1, "test.jpg")
```

Der obige Code setzt den Standard-Bildlader auf `plugin`. Das bedeutet, dass Befehle wie `LoadBrush()` nicht mehr den integrierten Bildlader von Hollywood oder den vom



Betriebssystem bereitgestellten Bildlader verwenden. Nur Plugin-Bidlader werden abgefragt, Bilddateien zu laden.

## 50.28 SetVarType

### BEZEICHNUNG

SetVarType – Variable angeben / VERALTET

### ÜBERSICHT

SetVarType(var, type[, arraysize])

### BESCHREIBUNG

Ab Hollywood 2.0 ist dieser Befehl veraltet und nur aus Kompatibilitätsgründen noch enthalten.

Dieser Befehl kann zum Deklarieren von Variablen benutzt werden. Sie müssen nur Variablen deklarieren, die vom voreingestellten Wert (**#LONG**) verschieden sind und keinen Identifikator in ihrem Namen tragen, wie das "\$" für Zeichenketten oder das "!" für Fließkommazahlen. Felder müssen immer durch Angabe des optionalen Arguments "arraysize" deklariert werden.

### EINGABEN

<b>var</b>	Variablenname, der deklariert werden soll
<b>type</b>	gewünschter Typ (kann <b>#LONG</b> , <b>#FLOAT</b> oder <b>#STRING</b> sein)
<b>arraysize</b>	optional: Wenn Sie dieses Argument angeben, wird anstelle einer einzelnen Variable ein Feld der angegebenen Länge angelegt

## 50.29 ShowNotification

### BEZEICHNUNG

ShowNotification – zeigt Systembenachrichtigung an (V8.0)

### ÜBERSICHT

ShowNotification(title\$, msg\$[, table])

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um eine Systembenachrichtigungsbox aufzurufen, in der die in **msg\$** an den Benutzer übergebene Nachricht angezeigt wird. Das Benachrichtigungsfeld verwendet den in **title\$** angegebenen Titel. Wenn Sie im Argument **title\$** eine leere Zeichenkette ("" ) übergeben, verwendet die Benachrichtigung den in der Präprozessor-Anweisung **@APPTITLE** angegebenen Titel.

Mit dem optionalen Tabellenargument **table** können weitere Parameter für die Systembenachrichtigung konfiguriert werden. Die folgenden Tags werden derzeit vom Argument **table** erkannt:

<b>Icon:</b>	Dieser Tag kann verwendet werden, um das Piktogramm festzulegen, das in der Benachrichtigungsbox angezeigt werden soll, wenn das Hostsystem
--------------	---

dies unterstützt. Derzeit können hier folgende Piktogrammtypen angegeben werden:

```
#REQICON_NONE:
    Kein Piktogramm
#REQICON_INFORMATION:
    Ein Informationszeichen
#REQICON_ERROR:
    Ein Fehlerzeichen
#REQICON_WARNING:
    Ein Warnungszeichen
#REQICON_QUESTION:
    Ein Fragezeichen
```

Beachten Sie, dass nicht alle Plattformen diesen Tag unterstützen.

**Timeout:** Mit diesem Tag kann ein Timeout für die Benachrichtigung in Millisekunden angegeben werden (nicht auf Android: Hier werden nur spezielle Konstanten erkannt. Siehe unten) Wenn dieser Tag gesetzt ist, wird die Benachrichtigung nach Ablauf der angegebenen Zeit (in Millisekunden) ausgeblendet. Andernfalls verwendet die Benachrichtigung den Standard-Timeout-Wert des Hostsystems für Benachrichtigungen. Sie können hier auch eine der folgenden speziellen Konstanten übergeben:

```
#DURATION_SHORT:
    Verwenden Sie dies für eine kurze Benachrichtigung.
#DURATION_LONG:
    Verwenden Sie dies für eine lange Benachrichtigung.
```

Beachten Sie, dass Sie unter Android eine der oben aufgeführten speziellen Konstanten übergeben müssen. Das direkte Übergeben eines Millisekundenwerts wird auf Android nicht unterstützt. Beachten Sie auch, dass nicht alle Plattformen den Tag **Timeout** unterstützen.

**X:** Gewünschte horizontale Position für die Benachrichtigung. Dies kann auch eine von Hollywoods speziellen Koordinatenkonstanten wie **#CENTER** sein. Dieser Tag wird nur unter Android und iOS unterstützt.

**Y:** Gewünschte vertikale Position für die Benachrichtigung. Dies kann auch eine von Hollywoods speziellen Koordinatenkonstanten wie **#CENTER** sein. Dieser Tag wird nur unter Android und iOS unterstützt.

**NoSound:** Setzen Sie diesen Tag auf **True**, wenn die Benachrichtigung nicht mit einem Ton begleitet werden soll. Dieser Tag wird derzeit nur von macOS unterstützt.

Beachten Sie, dass Benachrichtigungen unter Windows nur angezeigt werden können, wenn Ihr Programm ein Piktogramm in der Windows-Taskleiste hat. Daher fügt **ShowNotification()** automatisch ein Piktogramm in der Taskleiste für Ihr Programm hinzu, wenn noch keines vorhanden ist. Wenn Sie ein Taskleisten-Piktogramm für

Ihr Programm unter Windows manuell installieren möchten, rufen Sie den Befehl `SetTrayIcon()` auf, bevor Sie `ShowNotification()` verwenden. Siehe [Abschnitt 42.12 \[SetTrayIcon\]](#), [Seite 900](#), für Details.

Beachten Sie auch, dass `ShowNotification()` auf AmigaOS 4 über das Ringhio-System von der `application.library` implementiert wird und daher nur verwendet werden kann, wenn sich Ihr Skript als AmigaOS 4-Programm mit dem Tag `RegisterApplication` in `@OPTIONS` registriert hat. Wenn Sie eine genaue Steuerung der Ringhio-Nachrichten benötigen und nicht portabel sein müssen, können Sie stattdessen den Befehl `ShowRinghioMessage()` unter AmigaOS 4 verwenden, da dies noch mehr Konfigurationsoptionen bietet. Siehe [Abschnitt 16.13 \[ShowRinghioMessage\]](#), [Seite 182](#), für Details.

Unter MorphOS verwendet `ShowNotification()` das Benachrichtigungssystem von MagicBeacon. Unter AmigaOS 3 wird Ranchero zum Anzeigen von Benachrichtigungen verwendet. Beachten Sie, dass sich Apps registrieren müssen, bevor sie Benachrichtigungen mit Ranchero anzeigen können. Darum wird empfohlen, dass wenn Sie `ShowNotification()` auf AmigaOS 3 mit Ranchero verwenden möchten, auch `@APPIDENTIFIER` verwenden sollten, um eine eindeutige ID für Ihre App festzulegen. Andernfalls wird der generische App-Name "Hollywood" verwendet.

#### EINGABEN

<code>title\$</code>	gewünschter Titel für die Benachrichtigung oder leere Zeichenkette für den Standardtitel
<code>msg\$</code>	gewünschter Nachrichtentext für die Benachrichtigung
<code>table</code>	optional: Tabelle mit weiteren Konfigurationsparametern (siehe oben)

### 50.30 Sleep

#### BEZEICHNUNG

Sleep – hält die Skriptaussführung für eine bestimmte Zeit an (V10.0)

#### ÜBERSICHT

`Sleep(time)`

#### BESCHREIBUNG

Dieser Befehl hält die Skriptaussführung für das in `time` angegebene Zeitintervall an. Dieses Zeitintervall muss in Millisekunden angegeben werden.

`Sleep()` macht dasselbe wie `Wait()`, außer dass `Sleep()` immer in Millisekunden arbeitet, während `Wait()` standardmäßig Ticks verwendet.

#### EINGABEN

<code>time</code>	Zeitintervall in Millisekunden
-------------------	--------------------------------

#### BEISPIEL

```
Sleep(4000)
```

Der obige Code hält die Skriptaussführung für 4 Sekunden an.

## 50.31 VERSION

### BEZEICHNUNG

VERSION – definiert, welche Hollywood-Version benötigt wird (V2.0)

### ÜBERSICHT

@VERSION version, revision

### BESCHREIBUNG

Mit dieser Präprozessor-Anweisung können Sie die Version von Hollywood definieren, die Ihr Skript für die Ausführung mindestens braucht. Sie sollten diese Präprozessor-Anweisung immer als erste in Ihrem Skript verwenden.

### EINGABEN

version erforderliche Hollywood-Version

revision erforderliche Hollywood-Reversion

### BEISPIEL

@VERSION 2,0

Definiert, dass dieses Skript mindestens Hollywood 2.0 benötigt.

## 50.32 Wait

### BEZEICHNUNG

Wait – wartet einen bestimmten Zeitraum

### ÜBERSICHT

Wait(time[, unit])

### BESCHREIBUNG

Dieser Befehl wartet die durch **time** angegebene Zeit und fährt dann mit der Abarbeitung des Skriptes fort. Die Standardeinheit ist Ticks. Ein Tick ist 1/50 Sekunde. Wenn Sie also eine Sekunde warten möchten, müssen Sie **time** auf 50 setzen.

Ab Hollywood 1.9 können Sie das optionale Argument **unit** angeben, um eine Einheit zu definieren, in der die Variable **time** vorliegt. Die folgenden Einheiten sind möglich:

#MILLISECONDS:

Zeit in Millisekunden (1/1000 sekunde)

#SECONDS:

Zeit in Sekunden

#TICKS: Zeit in Ticks (Standard) (1/50 Sekunde)

Sie können auch den Befehl **Sleep()** verwenden, der das Gleiche wie **Wait()** tut, aber in Millisekunden arbeitet, sodass Sie etwas Tipparbeit sparen.

### EINGABEN

time Zeit, die gewartet werden soll

unit optional: in welcher Einheit die Zeitangabe vorliegt (Standardeinstellung ist #TICKS) (V1.9)

**BEISPIEL**

`Wait(200)`

Wartet 4 Sekunden.



## 51 Tabellenbibliothek

### 51.1 Concat

#### BEZEICHNUNG

Concat – verkettet Tabellenzeichenketten in einer einzigen Zeichenfolge (V5.0)

#### ÜBERSICHT

```
s$ = Concat(table[, sep$, start, end])
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die Zeichenfolgen von Tabellenindizes **start** bis **end** in eine einzelne Zeichenfolge zu verketten. Optional durch die Zeichenfolge **sep\$** können die Tabellenzeichenketten in der Zeichenfolge begrenzt werden. Wenn **start** und **end** nicht angegeben sind, werden standardmäßig alle Tabellenzeichenketten verkettet werden. Wenn **sep\$** nicht angegeben ist, wird kein Trennzeichen zwischen den Tabellenzeichenketten gesetzt.

Bitte beachten Sie, dass dieser Befehl nur Zeichenketten mit Ganzzahlen-Indizes berücksichtigen kann. Zeichenfolgen mit nicht ganzzahligen Indizes werden nicht berücksichtigt.

#### EINGABEN

<b>table</b>	Tabelle, die als Quelle verwendet wird
<b>sep\$</b>	optional: Trennzeichenfolge (Standard "")
<b>start</b>	optional: Tabellenindex der ersten Zeichenfolge die zu verketten ist (Standard ist 0, was bedeutet, das die Tabelle bei Index 0 startet)
<b>end</b>	optional: Tabellenindex der letzten Zeichenfolge die zu verketten ist (standardmäßig ist die Anzahl der Tabellenelemente minus eins, was das Ende der Tabelle bedeutet)

#### BEISPIEL

```
t = {"Hello", "this", "is", "a", "test!"}
DebugPrint(Concat(t, " "))
```

Der obige Code verkettet alle Zeichenfolgen in der Tabelle und trennt sie durch ein eingefügtes Leerzeichen.

### 51.2 CopyTable

#### BEZEICHNUNG

CopyTable – erstellt eine unabhängige Kopie einer Tabelle (V4.6)

#### ÜBERSICHT

```
t = CopyTable(src[, shallow])
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um eine unabhängige Kopie der angegebenen Quelltable zu erstellen. Wie Sie wahrscheinlich bemerkt haben, wird nur ein Verweis

auf die Tabelle erstellt, wenn die Quelltable mit dem Gleichheitszeichen (=) einer neuen Variablen zugewiesen wird. Dies wird aus Effizienzgründen gemacht, weil die Herstellung vollständiger Kopien einer Tabelle in den meisten Fällen nicht notwendig ist. In einigen Fällen jedoch müssen Sie eine völlig unabhängige Kopie der Tabelle haben. Dies kann mit diesem Befehl durchgeführt werden.

Beginnend mit Hollywood 6.0 akzeptiert dieser Befehl das optionale Argument **shallow**. Wenn Sie dieses Argument auf **True** setzen, wird **CopyTable()** anstelle einer unabhängigen Kopie alle Untertabellen nur durch einen Verweis kopieren. Das bedeutet, wenn die Quelltable geändert wird, alle Kopien die in Bezug stehen auch modifiziert werden. Das hat den Vorteil, dass Ressourcen geschont werden und kann auch nützlich für den Fall sein, falls eine vollständige Kopie zu einem Stapelüberlauf führen würde.

Siehe [Abschnitt 8.4 \[Tabellen\]](#), [Seite 106](#), für Details.

#### EINGABEN

**src**           Tabelle, die zu kopieren ist

**shallow**       optional: gibt an, ob eine vollständige (**False**) oder eine Kopie mit Verweis erstellt werden soll (**True**) (Standard ist **False**) (V6.0)

#### RÜCKGABEWERTE

**t**            neue Tabelle

#### BEISPIEL

```
t1 = {1, 2, 3, 4, 5}
t2 = t1
t2[0] = 10
DebugPrint(t1[0]) ; -> gibt 10 aus, weil t2 ein Verweis auf t1 ist
t3 = CopyTable(t1)
t3[0] = 20
DebugPrint(t1[0]) ; -> gibt 10 aus!
```

Dieser Code zeigt zunächst das Verhalten bei einer Kopie einer Tabelle, welche nur einen Verweis auf eine vorhandene Tabelle erstellt. Danach erfolgt über **CopyTable()** eine vollständige Kopie.

## 51.3 CreateList

#### BEZEICHNUNG

**CreateList** – erstellt eine optimierte Liste (V9.0)

#### ÜBERSICHT

```
list = CreateList()
```

#### BESCHREIBUNG

Dieser Befehl erzeugt eine optimierte Liste und gibt sie als leere Tabelle zurück.

Der Vorteil bei der Verwendung optimierter Listen anstelle von normalen Hollywood-Tabellen ist, dass Befehle wie **InsertItem()**, **RemoveItem()**, **ListItems()** und **GetItem()** viel schneller sind.



Der Nachteil ist, dass das Hinzufügen oder Entfernen von Elementen nur über `InsertItem()` und `RemoveItem()` erfolgen kann. Sie dürfen keine Elemente zu optimierten Listen hinzufügen oder daraus entfernen, indem Sie die Tabelle direkt ändern. Es ist notwendig, die oben genannten Befehlen zu verwenden.

Um eine vorhandene Hollywood-Tabelle in eine optimierte Liste zu konvertieren, können Sie den Befehl `SetListItems()` verwenden. Siehe [Abschnitt 51.20 \[SetListItems\]](#), [Seite 1132](#), für Details.

## EINGABEN

keine

## RÜCKGABEWERTE

`list` eine leere optimierte Liste

## BEISPIEL

```
t = CreateList()
;t = {}
StartTimer(1)
For Local k = 1 To 10000
    InsertItem(t, k)
Next
NPrint(ListItems(t))
NPrint("This took", GetTimer(1), "ms")
```

Der obige Code erstellt eine leere optimierte Liste, fügt 10000 Elemente hinzu und gibt die dafür benötigte Zeit aus. Deaktivieren Sie die erste Zeile und kommentieren Sie die zweite Zeile aus, um zu sehen, wie viel schneller optimierte Listen im Vergleich zu normalen Hollywood-Tabellen sind.

## 51.4 ForEach

### BEZEICHNUNG

ForEach – durchläuft alle Elemente einer Tabelle (V5.0)

### ÜBERSICHT

```
[v] = ForEach(table, func[, userdata])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um alle Elemente der Tabelle zu durchlaufen, die im ersten Argument `table` angegeben ist. Für jedes Tabellenelement ruft dieser Befehl die Benutzerfunktion auf, welche in `func` angegeben ist. Die Benutzerfunktion erhält zwei Argumente: Das erste Argument wird den Index der Tabellenelemente enthalten, während das zweite Argument den Wert in diesem Index enthält. Wenn die Benutzerfunktion einen Wert zurückgibt, wird die Schleife unterbrochen und dieser Wert wird als Ergebnis von `ForEach()` zurückgegeben.

Bitte beachten Sie, dass dieser Befehl die gesamte Tabelle durchläuft. Wenn Sie nur die Ganzzahlen-Indizes durchlaufen möchten, verwenden Sie stattdessen den Befehl `ForEachI()`. Siehe [Abschnitt 51.5 \[ForEachI\]](#), [Seite 1122](#), für Details.

Beginnend mit Hollywood 6.1 akzeptiert dieser Befehl das optionale Argument **userdata**. Der Wert, den Sie hier angeben, wird als drittes Argument an Ihre Callback-Funktion weitergeleitet. Der Wert kann von beliebiger Art sein.

#### EINGABEN

**table**      Tabelle, die durchlaufen werden soll

**func**        Benutzerfunktion, welche für jedes Tabellenelement aufgerufen wird

**userdata**   optional: Benutzerdaten, die an die Callback-Funktion weitergegeben wird (V6.1)

#### RÜCKGABEWERTE

**v**            optional: Rückgabewert, wenn der Durchlauf von einer Benutzerfunktion unterbrochen wird (siehe oben).

#### BEISPIEL

```
t = {1, 2, 3, 4, Test$="Hello", Value=9.2}
ForEach(t, DebugPrint)
```

Der obige Code bildet den Inhalt der Tabelle in **t** ab, bei Verwendung von **ForEach()**.

## 51.5 ForEachI

#### BEZEICHNUNG

ForEachI – durchläuft alle Ganzzahlen-Indizes einer Tabelle (V5.0)

#### ÜBERSICHT

```
[v] = ForEachI(table, func[, userdata])
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um alle Ganzzahlen-Indizes der Tabelle zu durchlaufen, die im Argument **table** angegeben ist. Für jedes Tabellenelement mit einem ganzzahligen Index wird die Benutzerfunktion aufgerufen, die im Argument **func** angegeben ist. Die Benutzerfunktion wird zwei Argumente erhalten: Das erste Argument wird den Index des Tabellenelements enthalten, während das zweite Argument den Wert von diesem Index enthält. Wenn die Benutzerfunktion einen Wert zurückgibt, wird die Schleife unterbrochen und dieser Wert wird als Ergebnis von **ForEach()** zurückgegeben.

Bitte beachten Sie, dass dieser Befehl nur die Ganzzahlen-Indizes durchläuft. Wenn Sie die gesamte Tabelle durchlaufen möchten, verwenden Sie stattdessen dafür den Befehl **ForEach()**. Siehe [Abschnitt 51.4 \[ForEach\]](#), [Seite 1121](#), für Details.

Beginnend mit Hollywood 6.1 akzeptiert dieser Befehl das optionale Argument **userdata**. Der Wert, den Sie hier angeben, wird als drittes Argument an Ihre Callback-Funktion weitergeleitet. Der Wert kann von beliebiger Art sein.

#### EINGABEN

**table**      Tabelle, die durchlaufen werden soll

**func**        Benutzerfunktion, welche für jedes Tabellenelement aufgerufen wird

**userdata**   optional: Benutzerdaten, die an die Callback-Funktion weitergegeben wird (V6.1)

**RÜCKGABEWERTE**

`v` optional: Rückgabewert, wenn der Durchlauf von einer Benutzerfunktion unterbrochen wird (siehe oben).

**BEISPIEL**

```
t = {1, 2, 3, 4, Test$="Hello", Value=9.2}
ForEachI(t, DebugPrint)
```

Der obige Code bildet den Inhalt der Tabelle in `t` ab. Das bedeutet, dass die Indizes `Test$` und `Value` nicht beachtet werden.

## 51.6 GetItem

**BEZEICHNUNG**

`GetItem` – ruft ein Listenelement ab (V9.0)

**ÜBERSICHT**

```
item = GetItem(list, idx)
```

**BESCHREIBUNG**

Dies gibt das Element am Index `idx` in der durch `list` angegebenen Liste zurück. Wenn `idx -1` ist, wird das letzte Listenelement, wenn `idx` außerhalb des Bereichs liegt, wird `Nil` zurückgegeben.

Beachten Sie, dass Sie diesen Befehl normalerweise nicht verwenden sollten, da Sie mit dem `[]`-Operator schneller auf Listenelemente zugreifen können. Der einzige Fall, in dem die Verwendung von diesem Befehl einen echten Geschwindigkeitsvorteil hat, ist, wenn Sie das letzte Element einer optimierten Liste abrufen möchten und die Anzahl der Elemente in der Liste nicht kennen. In diesem Fall ist die Verwendung von `GetItem()` schneller als zuerst mit `ListItems()` herauszufinden, wieviele Elemente in der Liste sind.

**EINGABEN**

`list` zu verwendende Liste

`idx` Index des abzurufenden Listenelements; übergeben Sie `-1`, um das letzte Element zu erhalten

**RÜCKGABEWERTE**

`item` Element vom angegebenen Index

## 51.7 GetMetaTable

**BEZEICHNUNG**

`GetMetaTable` – gibt die Metatabelle einer Tabelle zurück (V2.0)

**ÜBERSICHT**

```
mt = GetMetaTable(t)
```

**BESCHREIBUNG**

Dieser Befehl ruft die Metatabelle der angegebenen Tabelle auf und gibt diese zurück. Wenn die angegebene Tabelle keine Metatabelle besitzt, wird `Nil` zurückgegeben. Siehe

[Abschnitt 9.8 \[Metamethods\]](#), [Seite 117](#), für mehr Informationen über Metatabellen und Metamethoden.

#### EINGABEN

`t`           Tabelle, deren Metatabelle Sie abrufen möchten

#### RÜCKGABEWERTE

`mt`           Metatabelle der angegebenen Tabelle oder `Nil` wenn die Tabelle keine Metatabelle hat

## 51.8 HaveItem

#### BEZEICHNUNG

`HaveItem` – überprüft, ob ein Tabellenelement vorhanden ist (V6.0)

#### ÜBERSICHT

```
bool = HaveItem(t, key)
bool = HasItem(t, key)
```

#### BESCHREIBUNG

Dieser Befehl überprüft, ob die angegebene Tabelle ein Element im Index `key` hat oder nicht. Wenn ein Element in diesem Index ist, gibt `HaveItem()` `True` zurück. Dies ist ein Komfortbefehl für `RawGet()`.

Beachten Sie, dass wenn Sie eine Zeichenfolge im Parameter `key` übergeben, diese automatisch in Kleinbuchstaben umgewandelt wird. Wenn Sie das nicht möchten, verwenden Sie stattdessen `RawGet()`.

Ab Hollywood 9.1 hat dieser Befehl ein Synonym namens `HasItem()`, das dasselbe tut, aber grammatikalisch besser ist.

#### EINGABEN

`t`           Tabelle, welche abgefragt wird  
`key`       Index, welcher vorhanden ist oder nicht

#### RÜCKGABEWERTE

`bool`       True oder False je nachdem, ob das Element vorhanden ist oder nicht

#### BEISPIEL

```
t = {x = 10, y = 20}
NPrint(HaveItem(t, "x"), HaveItem(t, "y"), HaveItem(t, "z"))
```

Der obige Code druckt 1 / 1 / 0 (= True, True, False) aus.

## 51.9 InsertItem

#### BEZEICHNUNG

`InsertItem` – fügt ein Element in eine Liste ein (V2.0)

#### ÜBERSICHT

```
InsertItem(list, item[, pos])
```

**BESCHREIBUNG**

Dieser Befehl fügt das angegebene Element `item` in den in `pos` angegebenen Bereich. Ein Element kann von beliebiger Art sein. Wenn Sie nicht das optionale Argument `pos` angeben, wird das Element am Ende der Liste angehängt werden. Wenn Sie das Argument `pos` benutzen, wird das Element an dieser Stelle eingefügt werden und alle folgenden Elemente werden eine Position nach oben verschoben werden. Der Positionszähler startet bei 0, welches das erste Element ist.

Beachten Sie, dass dieser Befehl bei Verwendung mit normalen Hollywood-Tabellen ziemlich langsam ist. Um `InsertItem()` zu beschleunigen, müssen Sie ihn mit optimierten Listen verwenden, die von `CreateList()` erstellt wurden. Siehe [Abschnitt 51.3 \[CreateList\]](#), [Seite 1120](#), für Details.

**EINGABEN**

<code>list</code>	Tabelle, wo das Element eingefügt wird
<code>item</code>	Element, welches eingefügt wird (kann von jeder Art sein)
<code>pos</code>	optional: Einfügeposition (Standard ist -1 was bedeutet, an das Ende der Liste)

**BEISPIEL**

```
a = {1, 2, 3, 4, 5, 7, 8, 9, 10}
InsertItem(a, 6, 5)
For k = 1 To ListItems(a) Do Print(a[k] - 1) .. " "
```

Druckt aus "1 2 3 4 5 6 7 8 9 10". Das Element "6" wird an der Position 5 eingefügt, so dass die Reihe abgeschlossen ist.

## 51.10 IPairs

**BEZEICHNUNG**

IPairs – durchläuft die gesamte Ganzzahlen-Indizes einer Tabelle (V2.0)

**ÜBERSICHT**

```
func, state, val = IPairs(table)
```

**BESCHREIBUNG**

Dieser Befehl kann in Verbindung mit der generischen Anweisung `For` verwendet werden, um alle Ganzzahlen-Indizes einer Tabelle zu durchlaufen. Bei Bedarf kann `IPairs()` durch die generische `For`-Anweisung drei Werte zurückgeben: Eine Iterator-Funktion in `func`, eine private Statusinformation in `state` und einen Anfangswert für den Durchlauf in `val`. Die von `IPairs()` zurückgegebene Iterator-Funktion stoppt den Durchlauf, wenn ein Indiz auftritt, dessen Wert auf `Nil` gesetzt ist.

Wenn Sie alle Felder einer Tabelle durchlaufen wollen, anstatt nur die Ganzzahl-Indizes, verwenden Sie stattdessen den Befehl `Pairs()`.

Siehe [Abschnitt 11.4 \[Generische For-Anweisung\]](#), [Seite 135](#), für Details.

**EINGABEN**

<code>table</code>	Tabelle, welche durchlaufen wird
--------------------	----------------------------------

**RÜCKGABEWERTE**

**func**        Iterator-Funktion  
**state**       private Statusinformationen  
**val**         Anfangswert des Durchlaufs

**BEISPIEL**

```
a = {"one", "two", "three"}
For i, v In IPairs(a)
    DebugPrint(i, v)
Next
```

Der obige Code gibt "0 one", "1 two" und "2 three" aus.

**51.11 IsTableEmpty****BEZEICHNUNG**

IsTableEmpty – überprüft, ob die Tabelle leer ist (V6.1)

**ÜBERSICHT**

```
b = IsTableEmpty(t)
```

**BESCHREIBUNG**

Dieser Befehl überprüft, ob die angegebene Tabelle leer ist und gibt **True** (leer) beziehungsweise **False** (nicht leer) zurück.

**EINGABEN**

**t**            Tabelle überprüfen

**RÜCKGABEWERTE**

**b**            **True** (leer) oder **False** (nicht leer)

**BEISPIEL**

```
Print(IsTableEmpty({}))
Print(IsTableEmpty({0}))
```

Der erste Aufruf druckt "1" (**True**), der zweite "0" (**False**) aus.

**51.12 ListItems****BEZEICHNUNG**

ListItems – zählt die Elemente einer Liste (V2.0)

**ÜBERSICHT**

```
c = ListItems(list)
```

**BESCHREIBUNG**

Dieser Befehl durchläuft alle Elemente in der in **list** angegebenen Liste und gibt an, wie viele Elemente in der Liste sind. Das Zählen stoppt, wenn ein Element der Art **Nil** in der Liste gefunden wird.

Beachten Sie, dass dieser Befehl nur Elemente in aufeinanderfolgenden ganzzahligen Indizes zählt. Es beginnt bei Index 0 und zählt alle Elemente in aufeinanderfolgenden ganzzahligen Indizes bis ein **Nil** Element angetroffen wird. Zum Zählen aller Elemente einer Tabelle, verwenden Sie stattdessen den Befehl **TableItems()**. Siehe [Abschnitt 51.23 \[TableItems\]](#), [Seite 1134](#), für Details.

Beachten Sie außerdem, dass dieser Befehl bei Verwendung mit normalen Hollywood-Tabellen ziemlich langsam ist. Um **ListItems()** zu beschleunigen, müssen Sie ihn mit optimierten Listen verwenden, die von **CreateList()** erstellt wurden. Siehe [Abschnitt 51.3 \[CreateList\]](#), [Seite 1120](#), für Details.

#### EINGABEN

**list**           Tabelle, deren Elemente gezählt werden

#### RÜCKGABEWERTE

**c**               Zähler

#### BEISPIEL

```
Print(ListItems({1, 2, 3, 4, 5, 6, 7, 8, 9, 10}))
```

Dies gibt 10 zurück.

## 51.13 NextItem

#### BEZEICHNUNG

**NextItem** – durchläuft Felder einer Tabelle (V2.0)

#### ÜBERSICHT

```
next, item = NextItem(table[, start])
```

#### BESCHREIBUNG

**NextItem()** gibt das Element zurück, das nach dem Element **start** in der angegebenen Tabelle folgt. Wenn **start Nil** ist, wird das erste Tabellen Element zurückgegeben, Wenn nach **start** kein Element kommt, wird **Nil** zurückgegeben.

Dieser Befehl wird hauptsächlich verwendet, um alle Felder der Tabelle in Argument **table** zu durchlaufen. Dazu übergeben Sie nur die Tabelle in Argument **table** ohne das zweite Argument **start**. **NextItem()** gibt dann den nächsten Index in der Tabelle und den Wert an diesem Tabellenindex in **item** zurück. Um alle Felder zu durchlaufen, müssen Sie beim nächsten Aufruf von **NextItem()** als zweites Argument den **next**-Wert übergeben, bis der **next**-Wert **Nil** ist.

Wenn es keine weiteren Elemente mehr in der Tabelle gibt, wird **Nil** zurückgegeben und Sie können Ihre Schleife beenden. Seien Sie aber beim Überprüfen von Variablen gegen **Nil** vorsichtig, da **0=Nil** in Hollywood tatsächlich wahr ist. **GetType()** ist der einzige zuverlässige Weg um herauszufinden, ob eine Variable wirklich **Nil** ist. Ein einfaches Überprüfen mit **Nil** würde auch zu **True** führen, wenn die Variable 0 wäre.

Erwarten Sie nicht, dass dieser Befehl die Tabellenfelder in der Reihenfolge zurückgibt, wie sie zugewiesen wurden. Hollywood speichert sie oft in einer anderen Reihenfolge ab.

#### EINGABEN

**table**           Tabelle, welche durchlaufen wird

**start** optional: wo der Durchlauf startet (Standard ist **Nil**, was bedeutet am Anfang beginnen)

## RÜCKGABEWERTE

**next** Index des nächsten Tabellenelements nach **start** oder **Nil**, wenn es keine Elemente mehr gibt

**item** Tabellenwert des Elements neben **start**

## BEISPIEL

```
t = {1, 2, 3, 4, 5, "Hello World", {100, 200, 300}, [-1.5] = -1.5,
      b = 66, Function(s) DebugPrint(s) EndFunction}
a, b = NextItem(t)
While GetType(a) <> #NIL
  DebugPrint(b)
  a, b = NextItem(t, a)
Wend
```

Der obige Code durchläuft eine heterogene Tabelle. Die Ausgabe wird folgende sein:

```
2
3
4
5
Hello World
Table: 74cbd42c
Function: 74cbd3c8
1
-1.5
66
```

Sie sehen, dass die Felder in einer anderen Reihenfolge zurückgegeben werden, als sie zugeordnet waren.

## 51.14 Pack

### BEZEICHNUNG

Pack – packt Parameter in eine Tabelle (V8.0)

### ÜBERSICHT

```
t = Pack(a[, b, ...])
```

### BESCHREIBUNG

Dieser Befehl übernimmt alle übergebenen Parameter und speichert sie in einer Tabelle, die zurückgegeben wird. Die Tabelle enthält so viele Elemente, wie Sie dem Befehl Parameter übergeben haben.

Verwenden Sie zum Entpacken aller Tabellenelemente den Befehl **Unpack()**.

### EINGABEN

**a** erster Parameter, der in die Tabelle gepackt werden soll

**b** zweiter Parameter, welcher in die Tabelle gepackt werden soll



... unbegrenzte Anzahl weiterer Parameter, welche in die Tabelle gepackt werden sollen

## RÜCKGABEWERTE

`t` eine Tabelle mit allen an diesen Befehl übergebenen Parametern

## BEISPIEL

```
t = Pack(1, 2, 3)
Print(t[0], t[1], t[2])
```

Dies wird "1 2 3" ausgeben.

## 51.15 Pairs

### BEZEICHNUNG

Pairs – durchläuft alle Felder einer Tabelle (V2.0)

### ÜBERSICHT

```
func, state, val = Pairs(table)
```

### BESCHREIBUNG

Dieser Befehl kann in Verbindung mit der generischen Anweisung `For` verwendet werden, um alle Felder in der Tabelle zu durchlaufen. Bei Bedarf kann `Pairs()` durch die generischen `For`-Anweisung drei Werte zurückgeben: Eine Iterator-Funktion in `func`, eine private Statusinformation in `state` und einen Anfangswert für den Durchlauf in `val`. Die Iterator-Funktion wird dann die Schlüssel/Wertkombination aller Tabellenfelder zurückgeben.

Wenn Sie nur die ganzzahligen Indizes einer Tabelle durchlaufen möchten, verwenden Sie stattdessen den Befehl `IPairs()`.

Siehe [Abschnitt 11.4 \[Generische For-Anweisung\]](#), [Seite 135](#), für Details.

### EINGABEN

`table` Tabelle, welche durchlaufen wird

## RÜCKGABEWERTE

`func` Iterator-Funktion

`state` Privatstatusinformation

`val` Anfangswert des Durchlaufs

## BEISPIEL

```
a = {"one", "two", "three"}
For i, v In IPairs(a)
    DebugPrint(i, v)
Next
```

Der obige Code gibt "0 one", "1 two" und "2 three" aus.

## 51.16 RawEqual

### BEZEICHNUNG

RawEqual – vergleicht zwei Tabellen ohne Metamethoden (V2.0)

### ÜBERSICHT

```
eq = RawEqual(t1, t2)
```

### BESCHREIBUNG

Dieser Befehl vergleicht die Tabelle **t1** mit **t2** und liefert **True**, wenn sie gleich sind, ansonsten **False**. Dies ist im Grunde das gleiche wie wenn Sie folgendes schreiben würden:

```
eq = (t1 = t2)
```

Der Unterschied ist jedoch, dass **RawEqual()** den Vergleich ohne die **Metamethoden** aufruft, welche in den Tabellen definiert werden könnten. Siehe [Abschnitt 9.8 \[Metamethoden\]](#), [Seite 117](#), für Details.

### EINGABEN

**t1**            Tabelle 1 zum Vergleichen

**t2**            Tabelle 2 zum Vergleichen

### RÜCKGABEWERTE

**eq**            **True** wenn die Tabellen gleich sind, andernfalls **False**

## 51.17 RawGet

### BEZEICHNUNG

RawGet – liest einen Wert aus der Tabelle ohne Metamethoden (V2.0)

### ÜBERSICHT

```
v = RawGet(t, key)
```

### BESCHREIBUNG

Dieser Befehl liest den Wert aus dem Index **key** aus der angegebenen Tabelle **t** und gibt ihn zurück. Grundsätzlich ist dieser Befehl das Selbe wie folgender Ausdruck:

```
v = t[key]
```

Der Unterschied besteht darin, dass **RawGet()** nie die **Metamethoden** aufrufen und somit nicht fehlschlagen wird, wenn der angegebene Index nicht existiert. Somit ist es nützlich, um zu überprüfen, ob ein bestimmter Tabellenindex vorhanden ist oder zum Lesen von Werten aus der Tabelle ohne die Metamethoden aufzurufen. Siehe [Abschnitt 9.8 \[Metamethoden\]](#), [Seite 117](#), für Details.

Bitte beachten Sie, dass Zeichenketten-Indizes in der Regel in Kleinbuchstaben geschrieben sind, außer wenn Sie den Tabellenindex mit Klammern initialisieren. Betrachten Sie den folgenden Code:

```
t1 = {TEST = 1}
DebugPrint(RawGet(t1, "TEST"), RawGet(t1, "test")) ; prints Nil/1
t2 = {}
t2.TEST = 1
```

```

DebugPrint(RawGet(t2, "TEST"), RawGet(t2, "test")) ; prints Nil/1
t3 = {}
t3["TEST"] = 1
DebugPrint(RawGet(t3, "TEST"), RawGet(t3, "test")) ; prints 1/Nil

```

Wie Sie sehen, wird beim Initialisieren des `TEST-Elements` mit geschweiften Klammern oder dem Punktoperator der Zeichenketten-Index `TEST` automatisch in Kleinbuchstaben umgewandelt. Bei Verwendung von Klammern zur Initialisierung des `TEST-Elements` findet keine Konvertierung statt. Dies hat zur Folge, dass Sie nicht auf nicht-kleingeschriebene Zeichenketten-Indizes zugreifen können, die mit dem Punktoperator der Klammer-Syntax initialisiert wurden, da der Punktoperator den Index immer in Kleinbuchstaben umsetzt.

Ab Hollywood 6.0 können Sie auch den Komfortbefehl `HaveItem()` nutzen, um zu überprüfen, ob ein Tabellenelement existiert. Siehe [Abschnitt 51.8 \[HaveItem\]](#), [Seite 1124](#), für Details.

## EINGABEN

`t`            Tabelle abfragen  
`key`         Index für die Suche

## RÜCKGABEWERTE

`v`            Wert des angegebenen Index oder `Nil`, wenn der Index in der Tabelle nicht existiert.

## BEISPIEL

```

t = {x = 10, y = 20}
NPrint(RawGet(t, "x"))
NPrint(RawGet(t, "y"))
NPrint(RawGet(t, "z"))

```

Der obige Code gibt 10 / 20 / `Nil` aus.

## 51.18 RawSet

### BEZEICHNUNG

`RawSet` – schreibt Werte in die Tabelle ohne Metamethoden (V2.0)

### ÜBERSICHT

`RawSet(t, key, val)`

### BESCHREIBUNG

Dieser Befehl schreibt den angegebenen Wert `val` in der angegebenen Tabelle `t` mit dem Index `key`. Der angegebene Wert kann von jeder Art (Zahl, Zeichenkette, Tabelle, Funktion, usw.) sein. Grundsätzlich ist dieser Befehl das Selbe wie folgender Ausdruck:

```
t[key] = v
```

Der einzige Unterschied besteht darin, dass `RawSet()` nie `Metamethoden` aufrufen wird, so dass Sie den vollen und sofortigen Zugriff auf alle Tabellenfelder haben. Siehe [Abschnitt 9.8 \[Metamethoden\]](#), [Seite 117](#), für Details.

**EINGABEN**

<b>t</b>	Tabelle, die Sie ändern möchten
<b>key</b>	Index der Tabelle
<b>val</b>	Wert, der gesetzt werden soll

**51.19 RemoveItem****BEZEICHNUNG**

RemoveItem – entfernt ein Element aus einer Liste (V2.0)

**ÜBERSICHT**

```
e = RemoveItem(list[, pos])
```

**BESCHREIBUNG**

Dieser Befehl entfernt ein Element aus der in **list** angegebene Liste und gibt sie zurück. Wenn Sie das optionale Argument **pos** weglassen, wird das letzte Element von der Liste entfernt. Andernfalls wird das angegebene Element entfernt. Position 0 ist das erste Element in der Liste. Nach dem Entfernen des Elements wird die Aufstellung neu organisiert und die Lücken geschlossen.

Beachten Sie, dass dieser Befehl bei Verwendung mit normalen Hollywood-Tabellen ziemlich langsam ist. Um **RemoveItem()** zu beschleunigen, müssen Sie ihn mit optimierten Listen verwenden, die von **CreateList()** erstellt wurden. Siehe [Abschnitt 51.3 \[CreateList\]](#), [Seite 1120](#), für Details.

**EINGABEN**

<b>list</b>	Tabelle, aus der das Element entfernt werden soll
<b>pos</b>	optional: Element, welches entfernt werden soll (Standardwerte ist -1 was bedeutet, dass das letzte Element entfernt wird)

**RÜCKGABEWERTE**

**e** das Element, das gerade entfernt wurde

**BEISPIEL**

```
a = {1, 2, 3, 4, 5, 6, 7, 8, 8, 9, 10}
e = RemoveItem(a, 7)
For k = 1 To ListItems(a) Do Print(a[k - 1] .. " ")
```

Entfernt die Nummer 8, weil es zweimal in der Liste ist. Die Variable **e** hat den Wert 8. Nach dem Entfernen des Elements, wird die richtige Zeile ausgegeben: "1 2 3 4 5 6 7 8 9 10".

**51.20 SetListItems****BEZEICHNUNG**

SetListItems – wandelt eine Tabelle in eine optimierte Liste um (V9.0)

**ÜBERSICHT**

`SetListItems(t, n)`

**BESCHREIBUNG**

Mit diesem Befehl können Sie eine bestehende Tabelle in eine optimierte Liste konvertieren. Optimierte Listen haben den Vorteil, dass die Befehle `InsertItem()`, `RemoveItem()`, `ListItems()` und `GetItem()` viel schneller sind als bei der Verwendung mit normalen Hollywood-Tabellen.

Sie müssen die zu konvertierende Tabelle im Argument `t` und die Anzahl der Listeneinträge in `n` übergeben. Beachten Sie, dass der Wert, den Sie in `n` übergeben, mit der Anzahl der aktuell in der Tabelle enthaltenen Listeneinträge übereinstimmen muss, d.h. er muss mit dem Rückgabewert von `ListItems()` übereinstimmen.

Beachten Sie auch, dass es bei der Verwendung von optimierten Listen einige Einschränkungen gibt. Siehe [Abschnitt 51.3 \[CreateList\]](#), [Seite 1120](#), für Details.

**EINGABEN**

`t`            Tabelle, welche in eine optimierte Liste umgewandelt wird  
`n`            Anzahl der Einträge in der Liste

**BEISPIEL**

```
t = {}
For Local k = 1 To 10000 Do t[k-1] = k
SetListItems(t, 10000)
Print(ListItems(t))
```

Der obige Code erstellt eine normale Hollywood-Tabelle, füllt sie mit 10000 Elementen und wandelt sie dann in eine optimierte Liste um.

## 51.21 SetMetaTable

**BEZEICHNUNG**

`SetMetaTable` – weist eine Metatabelle einer Tabelle zu (V2.0)

**ÜBERSICHT**

`SetMetaTable(t, mt)`

**BESCHREIBUNG**

Dieser Befehl weist die in `mt` angegebene Metatabelle der Tabelle `t` zu. Wenn Sie `Nil` in `mt` übergeben, wird die Metatabelle der Tabelle entfernt. Siehe [Abschnitt 9.8 \[Metamethoden\]](#), [Seite 117](#), für mehr Information über Metatabellen und Metamethoden.

**EINGABEN**

`t`            Tabelle, die geändert werden soll  
`mt`          Metatabelle, die Sie der Tabelle zuweisen möchten

**BEISPIEL**

Siehe [Abschnitt 9.8 \[Metamethods\]](#), [Seite 117](#).

## 51.22 Sort

### BEZEICHNUNG

Sort – sortiert ein Feld

### ÜBERSICHT

Sort(array[, sortfunc])

### BESCHREIBUNG

Dieser Befehl sortiert das Feld (Array), das durch die variable **array** angegeben wird. Es unterstützt Felder vom Typ **Zahlen**, Typ **Zeichenkette** oder einen beliebigen Datentyp über eine benutzerdefinierte Callback-Funktion (siehe nächsten Absatz). Dieser Befehl stoppt die Sortierung, wenn er ein **Nil**-Element oder eine leere Zeichenfolge ("" ) in Zeichenkettenfelder findet. Zeichenkettenfelder werden alphabetisch und Zahlenfelder in aufsteigender Reihenfolge sortiert.

Ab Hollywood 4.5 können Sie den Sortiervorgang mithilfe einer benutzerdefinierten Callback-Funktion anpassen. Diese Funktion muss zwei Parameter akzeptieren und gibt sie zurück, wenn der erste Parameter vor dem zweiten eingefügt werden soll oder nicht. Dies bietet Ihnen eine große Flexibilität beim Einrichten von benutzerdefinierten Sortiervorgängen, da Sie beliebige Werte vergleichen und Sie auch die Sortierreihenfolge anpassen können.

### EINGABEN

**array**        Feld, welches sortiert werden soll

**sortfunc**    optional: benutzerdefinierte Funktion, welche Hollywood mitteilt, wie es sortieren soll (V4.5)

### BEISPIEL

```
names = {"Joey", "Dave", "Mark", "Stephen", "Al", "Jefferson"}
Sort(names)
For k = 0 To 5
    NPrint(names[k])
Next
```

Der obige Code definiert ein Feld, fügt einige Namen hinzu und sortiert sie dann. Die Ausgabe ist "Al, Dave, Jefferson, Joey, Mark, Stephen".

```
nums = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10}
Sort(nums, Function(a, b) Return(a > b) EndFunction)
For k = 0 To 9
    NPrint(nums[k])
Next
```

Der obige Code verwendet eine benutzerdefinierte Sortierfunktion und sortiert die Tabelle "nums" in absteigender Reihenfolge. Das Ergebnis wird sein: 10, 9, 8, 7, 6, 5, 4, 3, 2, 1.

## 51.23 TableItems

### BEZEICHNUNG

TableItems – zählt alle Tabellenelemente (V6.1)

**ÜBERSICHT**

```
c = TableItems(t)
```

**BESCHREIBUNG**

Dieser Befehl zählt alle Elemente in der angegebenen Tabelle **t**. Im Gegensatz zu **ListItems()** welche nur die Elemente in aufeinanderfolgenden Ganzzahlen-Indizes berücksichtigt, zählt **TableItems()** wirklich alle Tabellenelemente, einschließlich jene aus Zeichenketten oder Fließkomma-Indizes.

**EINGABEN**

**t**            Tabelle, deren Elemente gezählt werden sollen

**RÜCKGABEWERTE**

**c**            Anzahl der Elemente in der Tabelle

**BEISPIEL**

```
Print(TableItems({x=5, y=6, [6]=1, 3, 4, 5, 6}))
```

Dies gibt 7 aus.

## 51.24 Unpack

**BEZEICHNUNG**

Unpack – entpackt eine Tabelle (V2.0)

**ÜBERSICHT**

```
a, ... = Unpack(t)
```

**BESCHREIBUNG**

Dieser Befehl entpackt die von **t** angegebene Tabelle und gibt alle ihre Elemente zurück. **Unpack()** gibt so viele Werte zurück, wie Elemente in der Tabelle vorhanden sind.

Verwenden Sie zum Packen von Parametern in eine Tabelle den Befehl **Pack()**.

**EINGABEN**

**t**            Tabelle, welche entpackt werden soll

**RÜCKGABEWERTE**

**a**            erster Wert

**...**        zusätzliche Rückgabewerte, je nachdem wie viele Elemente in der Tabelle sind

**BEISPIEL**

```
a, b, c = Unpack({1, 2, 3})
```

Der obige Code entpackt die angegebene Tabelle. **a** erhält den Wert 1, **b** wird 2 zugewiesen und **c** erhält den Wert 3.

```
a = {1, 2, 3, 4, 5, 6}
```

```
Print(Unpack(a))
```

Dies gibt "1 2 3 4 5 6" aus.





## 52 Textbibliothek

### 52.1 Übersicht

Hollywoods Textbibliothek enthält Befehle, die sich mit der Schriftverwaltung, Textmessung, Textzeichnung und Texttransformation beschäftigen. Sie können Schriftarten mit dem Befehlen `SetFont()`, `OpenFont()`, oder `@FONT` öffnen. Wichtig ist zu wissen, dass Hollywood zwei verschiedene Schriftartenmodule unterstützt:

1. Integriertes Schriftartenmodul: Dies ist das empfohlene Modul, obwohl es aus historischen Gründen nicht das Standardmodul ist. Das integrierte Modul arbeitet vollständig unabhängig vom Schriftartenmodul des Host-Betriebssystems und garantiert so, dass der Text auf jeder Plattform exakt gleich aussieht. Die von Befehlen wie `Print()` und `TextOut()` gezeichneten Pixel sind mit dem eingebauten Schriftartenmodul auf jeder Plattform exakt gleich. Sie können auf das eingebaute Schriftartenmodul zugreifen, indem Sie den Tag `Engine` bei den Befehlen `SetFont()`, `OpenFont()` oder `@FONT` auf `#FONTENGINE_INBUILT` setzen.
2. Natives Schriftartenmodul: Hier wird das Schriftartenmodul des Host-Betriebssystems verwendet. Aus historischen Gründen ist dies auch das Standardmodul. Aber es wird nicht empfohlen, es zu verwenden, da Text mit diesem Modul auf jeder Plattform leicht unterschiedlich aussehen. Wenn dies kein Problem für Sie ist, können Sie es verwenden. Aber wenn auf jeder Plattform ein identisches Aussehen erzielt werden soll, müssen Sie stattdessen das integrierte Schriftartenmodul verwenden.

Das integrierte Schriftartenmodul kann \*.ttf-Schriftarten auch direkt öffnen, sodass Sie nicht einmal diese Schriftarten installieren müssen, um sie mit dem integrierten Schriftartenmodul zu verwenden. Sie könnten einfach einen Code wie folgt verwenden:

```
OpenFont(1, "c:/Windows/Fonts/Arial.ttf", 36, {Engine = #FONTENGINE_
INBUILT})
```

Siehe [Abschnitt 52.29 \[Schriftdekleration\]](#), Seite 1160, für Details.

Siehe [Abschnitt 52.4 \[Arbeiten mit Schriften\]](#), Seite 1139, für Details.

### 52.2 AddFontPath

#### BEZEICHNUNG

AddFontPath – fügt zusätzlich einen Suchpfad für Schriftarten hinzu (V5.0)

#### ÜBERSICHT

AddFontPath(path\$)

#### BESCHREIBUNG

Dieser Befehl fügt den in `path$` angegebenen Pfad zu den Suchpfaden von Hollywoods eingebauten Schriftartenmodul (Font-Engine) hinzu. Das eingebaute Schriftartenmodul sucht standardmäßig nur in einem Unterverzeichnis mit dem Namen "Fonts" vom aktuellen Verzeichnis nach Zeichensätzen. Auf Amiga-Systemen sucht es auch in "FONTS:". Wenn Sie möchten, dass Hollywood auch andere Pfade verwendet, müssen Sie diese mit `AddFontPath()` hinzufügen.

Beachten Sie, dass die hier angegebenen Suchpfade nur das eingebaute Schriftartenmodul betreffen, d.h. sie werden nur verwendet, wenn Sie bei den Befehlen `OpenFont()`, `SetFont()` und der Präprozessor-Anweisung `@FONT #FONTENGINE_INBUILT` angeben. Die hier angegebenen Suchpfade werden bei Verwendung von `#FONTENGINE_NATIVE` nicht beachtet.

#### EINGABEN

`path$`      der Pfad, welcher dem Suchpfad des Schriftartenmoduls von Hollywood hinzugefügt werden soll

#### BEISPIEL

```
AddFontPath("Data/Fonts")
```

Fügt den Pfad "Data/Fonts" dem Suchpfad des Schriftartenmoduls hinzu.

## 52.3 AddTab

#### BEZEICHNUNG

AddTab – fügt einen Tabulator hinzu

#### ÜBERSICHT

```
AddTab(pos, ...)
```

#### BESCHREIBUNG

Dieser Befehl fügt den durch `pos` angegebenen Tabulator in Hollywoods Tabulatorliste ein. Tabulatoren können nur mit dem Befehl `Print()` benutzt werden. Falls sich ein Tabulator in der Zeichenkette befindet, die an `Print()` übergeben wird, wird zur nächstliegenden Tabulatorposition gesprungen. Sie können die Tabulatoreinstellungen mit `ResetTabs()` zurücksetzen.

Neu in V2.0: Sie können beliebig viele Tabulatorpositionen an diesen Befehl übergeben.

#### EINGABEN

`pos`            Position des neuen Tabulators (in Pixeln)

`...`            mehr Tabulatorpositionen (V2.0)

#### BEISPIEL

```
AddTab(100, 200, 300, 400)
SetFontStyle(#UNDERLINED)
NPrint("Last name\tFirst name\tAge\tGender\n")
SetFontStyle(#NORMAL)
NPrint("Doe\tJon\t34\tMale")
NPrint("Smith\tMaggie\t25\tFemale")
NPrint("...\t...\t...\t...")
```

Obiger Code stellt mit Hilfe von Tabulatoren eine Tabelle dar.

## 52.4 Arbeiten mit Schriften

Wenn Hollywood ausführbare Dateien für verschiedene Plattformen kompiliert, ist das häufigste Problem in der Regel die Frage, welche Schriften für Ihr Skript erforderlich sind. Die einfachste Lösung für dieses Problem ist, einfach alle erforderlichen Schriften von Ihrem Skript in mit der ausführbaren Datei zu verknüpfen. Sie können dies entweder mit der Präprozessor-Anweisung **@FONT** oder mit dem Konsolenargument **-linkfonts** erledigen. Viele Schriftarten sind jedoch urheberrechtlich geschützt und es ist nicht erlaubt, sie in die ausführbare Datei einzubinden. So müssen Sie möglicherweise Schriftarten manuell laden, statt sie zu verknüpfen. Wie das funktioniert, hängt von der Art der Schrift ab. Hollywood unterstützt zwei Schrifttypen: Bitmap-Fonts im AmigaOS-Format und TrueType-Schriftarten. Bitte lesen Sie unten aufgeführten Informationen über den Umgang mit diesen beiden Schrifttypen.

### 1) Der Umgang mit Amiga Bitmap Schriften:

Amiga Bitmap-Schriften werden nativ von Hollywood auf allen Plattformen unterstützt. Der Vorteil ist, dass sie beim ersten Mal nicht installiert werden müssen. Sie können sofort verwendet werden. Erstellen Sie einfach ein **Fonts** Unterverzeichnis im Verzeichnis von Ihrer ausführbaren Datei und kopieren Sie alle Amiga Bitmap-Schriften, die sie benötigt. Beachten Sie, dass eine Amiga Bitmap-Schrift nicht eine einzelne Datei, sondern drei Komponenten erfordert:

- a. \*.font Datei enthält Informationen über die Schrift
- b. Verzeichnis mit dem Namen der Schrift
- c. Ein oder mehrere Bitmaps, die die Rastergrafiken für die verschiedenen Schriftgrößen enthält

Wenn Sie also beispielsweise die Schrift goudyb in der Größe 23 unter Windows zu verwenden wollen, benötigen Sie die folgenden Dateien:

```
C:/Program Files/MyProg/MyCoolProgram.exe ; exe erstellt mit Hollywood
C:/Program Files/MyProg/Fonts/goudyb.font ; Infodatei der Schrift
C:/Program Files/MyProg/Fonts/goudyb/23 ; Bitmap im der Größe 23
```

Auf AmigaOS würde es wie folgt aussehen:

```
dh0:Programs/MyProg/MyCoolProgram ; exe erstellt mit Hollywood
dh0:Programs/MyProg/Fonts/goudyb.font ; Informationsdatei der Schrift
dh0:Programs/MyProg/Fonts/goudyb/23 ; Bitmap im der Größe 23
```

Unter macOS müssen Sie die Aufmerksamkeit auf die Tatsache richten, dass alle Datendateien in den Ordner **Resources** innerhalb des Anwendungspakets ihres Programmes gespeichert werden müssen. So würde es wie folgt aussehen:

```
/Programs/MyProg.app ; exe erstellt mit Hollywood
/Programs/MyProg.app/Contents/Resources/Fonts/goudyb.font
/Programs/MyProg.app/Contents/Resources/Fonts/goudyb/23
```

**Wichtig (AmigaOS):** Schriften, die eine zusätzliche \*.otag Datei haben, sind keine Bitmap Schriften! Das sind in der Regel Vektorschriften im TrueType-Format. TrueType-Schriftarten können nicht durch einfaches Kopieren in ein Unterverzeichnis relativ zu Ihrem Programm verwendet werden. TrueType-Schriften müssen immer zuerst installiert werden! Bitte siehe unten für weitere Informationen.

## 2) Der Umgang mit TrueType Schriften:

Das Arbeiten mit TrueType-Schriftarten unterscheidet sich von dem mit Bitmap-Schriften in der Art und Weise. Die TrueType-Schriften müssen immer installiert werden, bevor Sie sie verwenden können. Der einzige Weg, TrueType-Schriftarten zu verwenden, ohne sie zu installieren ist, sie in die ausführbare Datei einzubinden. Dies ist jedoch häufig wegen den Urheberrechten nicht möglich. TrueType-Schriftarten kommen als eine einzelne Datei daher, die in der Regel die Erweiterung `*.ttf` trägt. Um eine solche `*.ttf` Datei auf Ihrem System zu installieren, müssen Sie Folgendes tun:

AmigaOS3/MorphOS/AROS:

Verwenden Sie das Programm FTManager. Beachten Sie, dass FTManager standardmäßig eine ziemlich peinliche Schriftnamen verwendet, die Sie ändern sollten, wenn Sie Ihr Skript für mehrere Plattformen kompilieren. Siehe [Abschnitt 52.29 \[Schriftdekleration\]](#), [Seite 1160](#), für Details.

AmigaOS4:

Verwenden sie das Programm TypeManager vom Verzeichnis `SYS:System`.

Windows und macOS:

Klicken sie doppelt auf die Datei `*.ttf` und klicken sie auf Installieren.

Sobald Sie die neue Schriftart installiert haben, ist sie für die Verwendung mit Hollywood bereit.

## 52.5 CloseFont

### BEZEICHNUNG

CloseFont – schließt eine aktuelle Schriftart (V4.5)

### ÜBERSICHT

CloseFont(id)

### BESCHREIBUNG

Mit diesem Befehl wird der gesamte Arbeitsspeicher gelöscht, der durch die in `id` angegebene Schriftart belegt ist. Um den Speicherverbrauch zu reduzieren, sollten Sie Schriften schließen, wenn Sie sie nicht mehr benötigen.

### EINGABEN

`id`            Identifikator der zu schließende Schriftart

## 52.6 CopyTextObject

### BEZEICHNUNG

CopyTextObject – kopiert ein Textobjekt (V4.0)

### ÜBERSICHT

[id] = CopyTextObject(source, dest)

### BESCHREIBUNG

Dieser Befehl kopiert das in `source` angegebene Textobjekt und erstellt daraus das neue Textobjekt `dest`. Wenn Sie `Nil` im Argument `dest` angeben, wählt dieser Befehl automatisch eine ID für das kopierte Textobjekt aus und gibt sie an Sie zurück. Das neue

Textobjekt ist unabhängig vom alten, so dass Sie das Quelltextobjekt freigeben können, nachdem es kopiert wurde.

#### EINGABEN

**source**      Identifikator des Textobjekts, welches Sie kopieren möchten  
**dest**          Identifikator des neuen Textobjekts oder **Nil** für die **automatische ID-Auswahl**

#### RÜCKGABEWERTE

**id**            optional: ID des Textobjekts; wird nur zurückgegeben, wenn Sie beim Argument **id** **Nil** angegeben haben (siehe oben)

#### BEISPIEL

```
CopyTextObject(1, 10)
FreeTextObject(1)
```

Der obige Code erzeugt ein neues Textobjekt 10, das die gleichen Grafikdaten wie das Textobjekt 1 enthält. Dann wird das Textobjekt 1 aus dem Arbeitsspeicher gelöscht, da es nicht mehr benötigt wird.

## 52.7 CreateFont

#### BEZEICHNUNG

CreateFont – erstellt eine Schriftart aus einem Pinsel (V10.0)

#### ÜBERSICHT

```
[id] = CreateFont(id, brushid, charmap, width, height, cols[, t])
```

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um eine neue Schriftart aus einer Pinselquelle zu erstellen. Dies kann nützlich sein, wenn Sie mit benutzerdefinierten Schriftarten arbeiten müssen, die als Bilddateien verteilt werden, anstatt mit gängigen Schriftartformaten, wie es häufig in Spielen aus den 1980er und frühen 1990er Jahren und in Szenendemos der Fall war. Sie müssen den gewünschten Identifikator für die neue Schriftart im Argument **id** und den Identifikator der Pinselquelle im Argument **brushid** übergeben. Wenn Sie **Nil** in **id** übergeben, wählt **CreateFont()** automatisch einen Identifikator und gibt ihn zurück. Nachdem die Schriftart erfolgreich erstellt wurde, können Sie sie mit **UseFont()** als aktuelle Schriftart festlegen.

Der Parameter **charmap** muss auf eine Zeichenkette gesetzt werden, der die einzelnen Zeichen im Pinsel beschreibt. Die Zeichenabmessungen müssen in den Parametern **width** und **height** übergeben werden. Alle Zeichen müssen die gleichen Abmessungen haben. Die Anzahl der Zeichen pro Zeile muss im Argument **cols** übergeben werden.

Beispielsweise könnte eine Schriftart, die die Großbuchstaben A-Z unterstützt und deren Zeichen jeweils 32 x 32 Pixel groß sind, in einem Pinsel angeordnet werden, der 4 Reihen mit 8 Zeichen pro Reihe hat, mit Ausnahme der letzten Reihe, die nur 2 enthält, weil das englische Alphabet nur 26 Zeichen hat, also reichen 3 Zeilen mit 8 Zeichen plus eine letzte Zeile mit 2 Zeichen. Sie könnten also eine solche Schriftart aus einem 256x128 großen Pinsel erstellen und 32 für **width** und **height**, 8 für **cols** und "ABCDEFGH-IJKLMNOPQRSTUVWXYZ" in **Charmap** übergeben.

Das optionale Tabellenargument kann verwendet werden, um einige zusätzliche Optionen festzulegen. Die folgenden Tags werden derzeit erkannt:

**Name:** Auf diese Weise können Sie Ihrer Schriftart einen Namen geben. Standardmäßig wird der Name der Schriftart auf "Font" gesetzt.

**RowSpacing:** Wenn es einen gewissen Abstand zwischen den verschiedenen Zeichenreihen im Pinsel gibt, können Sie dies `CreateFont()` mit diesem Tag mitteilen. Setzen Sie diesen Tag einfach auf die Anzahl der Abstandspixel zwischen jeder Zeile und `CreateFont()` überspringt den Abstand, wenn es eine Schriftart erstellt. Der Standardwert ist 0, was bedeutet, dass kein vertikaler Abstand vorhanden ist.

**ColSpacing:** Wenn zwischen den einzelnen Zeichen im Pinsel etwas Abstand ist, können Sie dies `CreateFont()` mit diesem Tag mitteilen. Setzen Sie diesen Tag einfach auf die Anzahl der Abstandspixel zwischen jedem Zeichen und `CreateFont()` überspringt den Abstand, wenn er eine Schriftart erstellt. Der Standardwert ist 0, was bedeutet, dass kein horizontaler Abstand vorhanden ist.

**Ascender:** Mit diesem Tag können Sie die gewünschte Oberlänge für die Schriftart festlegen. Die Oberlänge einer Schriftart ist der maximale Zeichenumfang von der Grundlinie bis zum oberen Rand der Zeile. Hollywood verwendet den Oberlängenwert, um beispielsweise zu bestimmen, wo die Linie für den Textstil der Unterstreichung gezeichnet werden soll. Dies ist standardmäßig die in `height` übergebene Zeichenhöhe minus 1.

`CreateFont()` unterstützt sowohl Palettenpinsel als auch Pinsel mit Maske oder Alpha-Kanal. Wenn der in `brushid` übergebene Pinsel ein 1-Bit-Palettenpinsel ist, können Sie auch die Farbe der Schriftart mit `SetFontColor()` und anderen Befehlen ändern, genau wie Sie es für normale Schriftarten tun können. Wenn die Pinseltiefe jedoch mehr als 1 Bit beträgt, wird die Schriftart als Farbschrift behandelt, die immer dieselbe Farbe verwendet, unabhängig davon, wie die aktuelle Schriftfarbe eingestellt ist.

Beachten Sie, dass `CreateFont()` ziemlich flexibel ist und auch als Kachelkarte verwendet werden könnte. Ordnen Sie einfach jede Kachel einem Zeichen zu und zeichnen Sie dann die gesamte Kachelkarte mit einem einzigen Aufruf von `TextOut()`. Dies sollte viel schneller sein, als die Kacheln einzeln zu zeichnen.

## EINGABEN

<code>id</code>	Identifikator für die Schriftart oder <code>Nil</code> für automatische ID-Auswahl
<code>brushid</code>	Identifikator des Quellpinsels
<code>charmap</code>	Zeichenkette, die alle Zeichen im Pinsel beschreibt
<code>width</code>	Breite jedes Schriftzeichens
<code>height</code>	Höhe jedes Schriftzeichens
<code>cols</code>	Anzahl Zeichen pro Zeile

`t` optional: Tabelle mit weiteren Optionen (siehe oben)

## RÜCKGABEWERTE

`id` optional: Identifikator der Schriftart; wird nur zurück gegeben wenn Sie `Nil` als Argument 1 übergeben (siehe oben)

## BEISPIEL

```
CreateFont(1, 2, "ABCDEFGHJKLMNOPQRSTUVWXYZ0123456789!-.:?", 30, 32, 10)
UseFont(1)
NPrint("HELLO WORLD!")
```

Der obige Code erstellt eine neue Schriftart aus Pinsel 2. Es gibt 41 Zeichen im Quellpinsel und sie sind als 10 Zeichen pro Zeile und jeweils 30 x 32 Pixel angelegt. Das bedeutet, dass der Quellpinsel mindestens 300 x 160 Pixel groß sein muss. Nach dem Erstellen der Schriftart wird diese als aktuelle ausgewählt und der Text "HELLO WORLD!" wird ausgegeben.

## 52.8 CreateTextObject

### BEZEICHNUNG

`CreateTextObject` – erzeugt ein Textobjekt

### ÜBERSICHT

```
[id] = CreateTextObject(id, text$[, table])
```

### BESCHREIBUNG

Dieser Befehl erzeugt ein neues Textobjekt, das die durch den in `text$` angegebenen Daten enthält und weist ihm die in `id` eingetragene ID zu. Wenn Sie `Nil` in `id` übergeben, wird `CreateTextObject()` automatisch eine freie ID auswählen und zurückgeben. Der Text wird in der aktuellen Farbe und mit der aktuell ausgewählten Schrift dargestellt.

Der Vorteil von Textobjekten gegenüber dem Standardtext (Ausgabe zum Beispiel über den Befehl `Print()`) ist, dass Sie Textobjekte einfach auf dem Bildschirm positionieren, entfernen oder mit dem Befehl `MoveTextObject()` verschieben/scrollen können.

Ab Hollywood 2.5 können Sie die **Textformatierungen** von Hollywood in der Zeichenkette `text$` an `CreateTextObject()` übergeben. Mit diesen Formatmarkierungen können Sie die Schriftart und Farbe Ihres Textes on-the-fly ändern. Formatmarkierungen beginnen und enden immer mit einer eckigen Klammer (`'[']`). Wenn Sie nur eine eckige Klammer ausgeben möchten, müssen Sie zwei eckige Klammern (`'[[']`) verwenden. Bei nur einer eckigen Klammer erwartet Hollywood immer eine Formatmarkierung. Weitere Informationen zu diesem Thema finden Sie im Kapitel **Textformatierungen**.

In Hollywood 5.0 hat sich die Syntax von diesem Befehl etwas verändert. Während die alte Syntax noch wegen der Kompatibilität unterstützt wird, sollten neue Skripts die neue Syntax verwenden, die die Tabelle `table` als Argument akzeptiert. Die Tabelle kann die folgenden Elemente enthalten:

**Align:** Ermöglicht die Angabe der Ausrichtung des Textes nach einem Zeilenumbruchzeichen. Folgende Ausrichtungen werden unterstützt:

`#LEFT` Linksbündig.

**#RIGHT**      Rechtsbündig.

**#CENTER**    Zentriert.

**#JUSTIFIED**

Gibt den Text in Blocksatz aus. (V7.0)

Für **Align** ist **#LEFT** voreingestellt.

#### WordWrap:

Wenn Sie diesen zusätzlichen Parameter angeben, kann der Befehl **CreateTextObject()** ein automatischer Wortumbruch für Sie durchführen. Mit diesem Tag können Sie eine maximale Breite für Ihren Text angeben. **CreateTextObject()** wird dann einen Wortumbruch verwenden, um sicherzustellen, dass kein Text über dieses Limit hinausgeht. Wenn Sie dieses Argument nicht oder auf 0 setzen (was auch voreingestellt ist), ist der Text so breit wie er benötigt wird. Ab Hollywood 9.1 können Sie auch bedingte Bindestriche oder Leerzeichen mit Nullbreite verwenden, um den Wortumbruch anzupassen. Da es sich jedoch um Unicode-Zeichen handelt, müssen Sie sicherstellen, dass Sie in diesem Fall die UTF-8-Codierung verwenden.

#### Encoding:

Dieser Parameter kann verwendet werden, um die Zeichencodierung von **text\$** anzugeben. Voreingestellt ist die Standardcodierung für die Textbibliothek, welche mit dem Befehl **SetDefaultEncoding()** festgelegt ist. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details.

#### Color:

Mit diesem Tag können Sie die Textfarbe angeben. Die Farbe muss als **ARGB-Wert** angegeben werden. Wenn Sie diesen Tag nicht angeben, verwendet **CreateTextObject()** die Farbe, die mit dem Befehl **SetFontColor()** festgelegt wurde.

#### Pen:

Wenn der Palettenmodus **#PALETTEMODE\_PEN** ist, kann dieser Tag verwendet werden, um den Stift festzulegen, der zum Zeichnen des Textes verwendet werden soll. Wenn der Palettenmodus **#PALETTEMODE\_PEN** ist und **Pen** nicht angegeben ist, wird stattdessen der Stiftsatz verwendet, der **SetDrawPen()** verwendet. (V9.0)

#### Linespacing:

Dieser Tag kann verwendet werden, um die Abstandspixel zwischen Zeilen anzupassen. Er kann auf einen positiven oder negativen Wert eingestellt werden. Ein negativer Wert schiebt die Zeilen näher zusammen, während ein positiver Wert den Abstand zwischen den Zeilen vergrößert. Ein Wert von 0 bedeutet kein benutzerdefinierter Zeilenabstand. Voreingestellt ist 0. (V10.0)

#### Charspacing:

Ermöglicht Ihnen, den Abstand zwischen den Zeichen anzupassen. Sie können diesen auf einen positiven oder negativen Wert setzen. Ein positiver Wert vergrößert den Abstand zwischen den Zeichen, ein negativer Wert verringert ihn. (V10.0)



**NoAdjust:**

Beim Zeichnen von Textobjekten mit `DisplayTextObject()` positioniert Hollywood sie so, dass sie aussehen, als wären sie mit `TextOut()` gezeichnet worden, was bedeutet, dass sie gegebenenfalls nach links und oben versetzt werden könnten. Teile einiger Zeichen sind so konzipiert, dass sie im Bereich der vorherigen Zeichen erscheinen. Dies ist häufig bei Zeichen wie "j" der Fall. Wenn Sie das nicht möchten, setzen Sie `NoAdjust` auf `True`. In diesem Fall führt der Aufruf von `DisplayTextObject()` niemals zu einer Anpassung der Positionierung, sondern das Textobjekt wird strikt an der angegebenen Position gezeichnet. Die von Hollywood auf ein Textobjekt angewendeten Anpassungsoffsets für den Fall, dass `NoAdjust False` ist, können durch Abfragen der Tags `#ATTRADJUSTX`- und `#ATTRADJUSTY` ermittelt werden. Voreingestellt ist `False`. (V10.0)

Beachten Sie auch, dass Hollywood nur westlichen Standard von links nach rechts basierten Text auf horizontalen Linien ausgerichtet unterstützt. Von rechts nach links und vertikalem Text wird derzeit nicht unterstützt.

Beachten Sie außerdem, dass beim Zeichnen auf ein palettenbasiertes Ziel und wenn der Palettenmodus auf `#PALETTEMODE_PEN` eingestellt ist, dieser Befehl mit dem Stift über `SetDrawPen()` zeichnet anstelle der Farbe, welcher mit dem Befehl `SetFontColor( )` oder dem Tag `Color` von oben gesetzt wurde.

**EINGABEN**

`id` ID des neuen Textobjekts oder `Nil` für die automatische ID-Auswahl  
`text$` Text für das Textobjekt  
`table` optional: Eine Tabelle mit weiteren Optionen

**RÜCKGABEWERTE**

`id` optional: ID des neuen Textobjekts; wird nur zurückgegeben, wenn Sie beim Argument `id Nil` angegeben haben (siehe oben)

**BEISPIEL**

```
SetFontColor(#RED)
SetFont("times.font", 18)
CreateTextObject(1, "Hello World!")
DisplayTextObject(1, #CENTER, #CENTER)
```

Der obige Code erzeugt ein Textobjekt mit der Schriftart "times" (Größe 18) und mit der Farbe Rot. Der Text ist "Hello World!". Nach der Erstellung wird das Textobjekt in der Mitte des Bildschirms angezeigt.

## 52.9 DisplayTextObject

**BEZEICHNUNG**

`DisplayTextObject` – zeigt ein Textobjekt an

**ÜBERSICHT**

```
DisplayTextObject(id,x,y)
```

**BESCHREIBUNG**

Dieser Befehl zeigt das durch `id` angegebene Textobjekt an den Koordinaten `x/y` an.

Wenn Ebenen aktiviert sind, fügt dieser Befehl eine neue Ebene des Typs `#TEXTOBJECT` zum Ebenenstapel hinzu.

**EINGABEN**

<code>id</code>	Identifikator des Textobjekts
<code>x</code>	X-Position für das Textobjekt auf dem Bildschirm
<code>y</code>	Y-Position für das Textobjekt auf dem Bildschirm

**BEISPIEL**

Siehe [Abschnitt 52.8 \[CreateTextObject\]](#), Seite 1143.

## 52.10 DisplayTextObjectFX

**BEZEICHNUNG**

`DisplayTextObjectFX` – lässt ein Textobjekt mit einem Übergangseffekt erscheinen

**ÜBERSICHT**

```
[handle] = DisplayTextObjectFX(id, x, y[, table])
```

**BESCHREIBUNG**

Dieser Befehl ist eine erweiterte Version des Befehls `DisplayTextObject()`. Sie stellt das durch `id` angegebene Textobjekt an der durch `x/y` angegebenen Position dar und benutzt dazu einen der vielen Übergangseffekte, die Hollywood besitzt. Sie müssen außerdem die Übergangsgeschwindigkeit angeben.

Wenn Ebenen aktiviert sind, fügt dieser Befehl eine neue Ebene des Typs `#TEXTOBJECT` dem Ebenenstapel hinzu.

Ab Hollywood 4.0 verwendet dieser Befehl eine neue Syntax mit nur einer einzigen Tabelle als optionales Argument. Die alte Syntax wird aus Kompatibilitätsgründen weiterhin unterstützt. Das optionale Tabellenargument `table` kann verwendet werden, um den Übergangseffekt zu konfigurieren. Folgende Optionen sind möglich:

**Type:** Gibt den gewünschten Effekt für den Übergang an. Siehe [Abschnitt 30.11 \[DisplayTransitionFX\]](#), Seite 630, für eine Liste aller unterstützten Übergangseffekte. (Voreingestellt ist `#RANOMEFFECT`)

**Speed:** Legt die gewünschte Geschwindigkeit für den Übergang fest. Je höher der Wert, den Sie hier angeben, desto schneller wird der Effekt angezeigt werden. (Standardeinstellung ist `#NORMALSPEED`)

**Parameter:** Einige Übergangseffekte akzeptieren einen zusätzlichen Parameter, der hier angegeben werden kann. (Standardeinstellung ist `#RANDOMPARAMETER`)

**Async:** Sie können dieses Feld verwenden, um ein asynchrones Zeichnungsobjekt für diesen Übergang zu erstellen. Wenn Sie hier `True` angeben, wird `DisplayTextObjectFX()` sofort verlassen und es wird ein Handler für

das asynchrone Zeichnungsobjekt kreiert, den Sie dann mit dem Befehl `AsyncDrawFrame()` verwenden können. Ein Beispieldskript finden Sie unter dem Befehl `AsyncDrawFrame()`. Siehe [Abschnitt 19.1 \[AsyncDrawFrame\]](#), [Seite 231](#), für weitere Informationen über asynchrone Zeichnungsobjekte.

#### EINGABEN

<code>id</code>	Identifikator des darzustellenden Textobjekts
<code>x</code>	gewünschte X-Position für das Textobjekt
<code>y</code>	gewünschte Y-Position für das Textobjekt
<code>table</code>	optional: Einstellungen für den Übergangseffekt

#### RÜCKGABEWERTE

<code>handle</code>	optional: Handler auf ein asynchrones Zeichnungsobjekt; wird nur zurückgegeben, wenn <code>Async</code> auf <code>True</code> gesetzt wurde (siehe oben)
---------------------	--

#### BEISPIEL

```
DisplayTextObjectFX(1, 0, 0, #VLINES, 10) ; alte Syntax
```

oder

```
DisplayTextObjectFX(1, 0, 0, {Type = #VLINES, Speed = 10}) ; neue Syntax
```

Diese Zeile stellt Textobjekt 1 an Position 0:0 dar und benutzt zum Einblenden einen `#VLINES`-Effekt mit Geschwindigkeit 10.

## 52.11 FONT

#### BEZEICHNUNG

`FONT` – lädt eine Schriftart vor (V4.5)

#### ÜBERSICHT

```
@FONT id, fontname$, size[, table]
```

#### BESCHREIBUNG

Diese Präprozessor-Anweisung lädt die in `fontname$` angegebene Schriftart in der gewünschten Größe `size` (in Pixel) vor und weist ihr den Identifikator `id` zu. Sie können sie dann aus Ihrem Skript aus aktivieren, indem Sie den Befehl `UseFont()` benutzen.

Die in `fontname$` angegebene Schriftart muss der [Schriftdekleration](#) entsprechen. Siehe [Abschnitt 52.29 \[Schriftdekleration\]](#), [Seite 1160](#), für Details.

Siehe [Abschnitt 52.24 \[OpenFont\]](#), [Seite 1156](#), für mehr Informationen über Schriftarten in Hollywood.

Die Verwendung von `@FONT` ist praktisch, wenn Sie alle Schriften, welche von Ihren Skripts verwendet werden, mit Ihrem Applet der ausführbaren Datei eingebunden werden sollen. Standardmäßig werden alle mit `@FONT` angegebenen Schriftarten mit dem Applet der ausführbaren Datei verknüpft. Sie können dies ändern, indem Sie im optionalen Tabellenargument `Link` auf `False` setzen.

Das vierte Argument `table` ist optional. Es ist eine Tabelle, mit der weitere Optionen für den Ladevorgang eingestellt werden können. Diese Tabelle akzeptiert alle von der optionalen Tabelle `SetFont()` unterstützten sowie die folgenden Tags:

**Link:** Setzen Sie dieses Feld auf `False`, wenn Sie diese Schriftart nicht in die ausführbare Datei/das Applet einbinden wollen, wenn Sie Ihr Skript kompilieren. Dieses Feld ist standardmäßig auf `True` gesetzt, was bedeutet, dass die Schriftart mit der ausführbaren Datei/dem Applet beim Kompilieren verknüpft wird.

Möchten Sie Schriftarten manuell öffnen, dann verwenden Sie den Befehl `OpenFont()`.

**Wichtiger Hinweis:** Bitte beachten Sie, dass die meisten Schriftarten urheberrechtlich geschützt sind und es nicht erlaubt ist, sie in Ihre Programme zu integrieren, ohne eine Lizenz zu erwerben. So überprüfen Sie die Lizenz der Schriftart, die Sie in Ihr Programm einbinden möchten! Wenn Sie für Lizenzen nicht bezahlen möchten, ist es ratsam, eine kostenlose Schriftart wie DejaVu oder Bitstream Vera oder eine der TrueType-Schriftarten zu verwenden, die in Hollywood integriert sind (`#SANS`, `#SERIF`, `#MONOSPACE`, vgl. `SetFont()`)

## EINGABEN

<code>id</code>	einen Wert, um die Schriftart später im Code zu identifizieren
<code>fontname\$</code>	Die Schriftart, die Sie öffnen wollen
<code>size</code>	gewünschte Größe der Schriftart in Pixel
<code>table</code>	optional: Tabelle mit weiteren Optionen

## BEISPIEL

```
@FONT 1, "Arial", 36
```

Öffnet die Schriftart Arial in der Größe 36 und weist ihr die ID 1 zu.

## 52.12 FreeGlyphCache

### BEZEICHNUNG

FreeGlyphCache – löscht den Glyphenspeicher (V4.7)

### ÜBERSICHT

```
FreeGlyphCache(mode[, id])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die zwischengespeicherten Glyphen einer bestimmten oder aller Schriftarten im Arbeitsspeicher befindlichen Schriftarten zu löschen. Wenn Sie die zwischengespeicherten Glyphen einer bestimmten Schriftart löschen möchten, müssen Sie das Argument `mode` auf 1 setzen und die ID der Schriftart im optionalen Argument `id` übergeben. Wenn Sie den Glyphenspeicher aller geladenen Schriftarten beenden möchten, geben Sie im Argument `mode` einfach 0 ein.

**EINGABEN**

<b>mode</b>	setzen Sie diesen Wert auf 1, um den Glyphenspeicher der in Argument <b>id</b> angegebene Schriftart oder 0, um den Glyphenspeicher aller Schriftarten zu löschen
<b>id</b>	optional: ID der Schriftart, deren Glyphenspeicher gelöscht werden soll ( <b>mode</b> muss auf 1 gesetzt sein)

**52.13 FreeTextObject****BEZEICHNUNG**

FreeTextObject – löscht ein Textobjekt aus dem Arbeitsspeicher

**ÜBERSICHT**

FreeTextObject(id)

**BESCHREIBUNG**

Dieser Befehl löscht das in **id** angegebene Textobjekt aus dem Arbeitsspeicher. Das ist normalerweise nicht notwendig, da Hollywood allen Arbeitsspeicher freigibt, wenn es beendet wird. Wird Ihnen allerdings der Arbeitsspeicher knapp und Sie wollen das Textobjekt selbst löschen, so benutzen Sie diesen Befehl.

**EINGABEN**

<b>id</b>	Identifikator des Textobjekts
-----------	-------------------------------

**52.14 GetAvailableFonts****BEZEICHNUNG**

GetAvailableFonts – gibt eine Liste der verfügbaren Schriftarten zurück (V4.7)

**ÜBERSICHT**

**t** = GetAvailableFonts()

**BESCHREIBUNG**

Dieser Befehl scannt alle auf dem aktuellen Computer installierten Schriftarten, stellt sie in eine Tabelle und gibt die Informationen in **t** zurück. Dies ist nützlich, um zu überprüfen, ob eine bestimmte Schriftart ohne den Aufruf von **SetFont()** oder **OpenFont()** verfügbar ist.

Die von diesem Befehl zurückgegebene Tabelle **t** besteht aus mehreren Untertabellen, d.h. eine Untertabelle für jede Schriftart. Die Untertabellen haben die folgenden Elemente initialisiert:

<b>Name:</b>	Der vollständige Schriftname (d.h. Familienname plus Stil). Beispiel: "Arial Bold Italic".
<b>Family:</b>	Der Familienname dieser Schriftart, z.B. "Arial".

**Weight:** Die Strichstärke dieser Schriftart. Die wird auf eine der folgenden Strichstärkekonstanten gesetzt, wobei `#FONTWEIGHT_THIN` die dünnste und `#FONTWEIGHT_ULTRABLACK` die dickste Strichstärke ist:

```
#FONTWEIGHT_THIN
#FONTWEIGHT_EXTRALIGHT
#FONTWEIGHT_ULTRALIGHT
#FONTWEIGHT_LIGHT
#FONTWEIGHT_BOOK
#FONTWEIGHT_NORMAL
#FONTWEIGHT_REGULAR
#FONTWEIGHT_MEDIUM
#FONTWEIGHT_SEMIBOLD
#FONTWEIGHT_DEMIBOLD
#FONTWEIGHT_BOLD
#FONTWEIGHT_EXTRABOLD
#FONTWEIGHT_ULTRABOLD
#FONTWEIGHT_HEAVY
#FONTWEIGHT_BLACK
#FONTWEIGHT_EXTRABLACK
#FONTWEIGHT_ULTRABLACK
```

**Slant:** Die Neigung dieser Schriftart. Die wird auf eine der folgenden Neigungskonstanten gesetzt:

```
#FONTSLANT_ROMAN
#FONTSLANT_ITALIC
#FONTSLANT_OBLIQUE
```

**Bitmap:** `True`, wenn diese Schriftart eine Bitmap-Schriftart ist, `False` bei einer Vektorschriftart. Vektorschriftarten können frei transformiert und antialiasiert werden.

**Sizes:** Wenn die Schriftart eine Bitmap-Schriftart ist, wird hier eine Tabelle mit einer Liste der verfügbaren Größen für die Schriftart sein. Wenn die Schriftart eine Vektorschriftart ist, ist diese Tabelle leer.

Beachten Sie, dass es keine Garantie dafür gibt, dass alle Aufrufe mit den Befehlen `OpenFont()` oder `SetFont()` mit den zurückgegebenen Schriftarten gelingen wird. Es kann häufig passieren, dass `OpenFont()` und `SetFont()` mit einer bestimmten Schriftart fehlschlägt, obwohl sie in der verfügbaren Tabelle `t` zurückgegeben wurde. Dies liegt daran, dass `GetAvailableFonts()` die verfügbaren Schriftarten für alle Hollywood-Schriftartenmodule zurückgibt. Wenn Sie `OpenFont()` oder `SetFont()` aufrufen, kann nur ein Schriftartenmodul angegeben werden. Wenn ein Aufruf des Befehls `OpenFont()` fehlschlägt, obwohl die Schriftart von `GetAvailableFonts()` zurückgegeben wurde, dann ist dies ein Zeichen, dass Sie das falsche Schriftartenmodul verwenden, um diese Schriftart zu öffnen. In diesem Fall wechseln Sie einfach das Schriftartenmodul und es sollte funktionieren.

## EINGABEN

keine

**RÜCKGABEWERTE**

t            Eine Tabelle mit allen verfügbaren Schriftarten

**BEISPIEL**

```
t = GetAvailableFonts()
For Local k = 0 To ListItems(t) - 1
    DebugPrint("Family:", t[k].Family, "Weight:", t[k].Weight,
               "Slant:", t[k].Slant, "Bitmap:", t[k].Bitmap)
Next
```

Der obige Code listet alle auf diesem System verfügbaren Schriftarten auf.

**52.15 GetBulletColor****BEZEICHNUNG**

GetBulletColor – ermittelt die aktuelle Farbe des Aufzählungszeichens (V9.0)

**ÜBERSICHT**

```
color = GetBulletColor()
```

**BESCHREIBUNG**

Dieser Befehl gibt die mit **SetFontColor()** oder **SetBulletColor()** festgelegte Farbe des Aufzählungszeichens zurück.

**EINGABEN**

keine

**RÜCKGABEWERTE**

color        aktuelle Farbe des Aufzählungszeichens

**52.16 GetCharMaps****BEZEICHNUNG**

GetCharMaps – gibt die unterstützte Zeichensatztablelle der Schriftart zurück (V9.0)

**ÜBERSICHT**

```
t = GetCharMaps()
```

**BESCHREIBUNG**

Dieser Befehl gibt eine Tabelle zurück, die alle Zeichensatztabellen enthält, die von der derzeit aktiven Schriftart unterstützt werden. Die folgenden Zeichensatztabellen werden derzeit von Hollywood unterstützt:

```
#CHARMAP_DEFAULT
#CHARMAP_MSSYMBOL
#CHARMAP_UNICODE
#CHARMAP_SJIS
#CHARMAP_BIG5
#CHARMAP_WANSUNG
#CHARMAP_JOHAB
```

```
#CHARMAP_ADOBESTANDARD
#CHARMAP_ADOBEEXPERT
#CHARMAP_ADOBECUSTOM
#CHARMAP_ADOBELATIN1
#CHARMAP_OLDLATIN2
#CHARMAP_APPLEROMAN
```

Beachten Sie, dass Zeichentabellen nur für Schriftarten unterstützt werden, die von Hollywoods integriertem Schriftartenmodul verarbeitet werden, d.h. der Tag **Engine** muss beim Öffnen der Schriftart auf **#FONTENGINE\_INBUILT** gesetzt sein. Siehe [Abschnitt 52.32 \[SetFont\]](#), [Seite 1163](#), für Details.

#### EINGABEN

keine

#### RÜCKGABEWERTE

**t**           Tabelle mit allen Zeichensatztabellen, die von der aktuellen Schriftart unterstützt werden

## 52.17 GetDefaultEncoding

#### BEZEICHNUNG

GetDefaultEncoding – ruft die Standard-Zeichencodierungen ab (V7.0)

#### ÜBERSICHT

```
tencoding, sencoding = GetDefaultEncoding()
```

#### BESCHREIBUNG

Dieser Befehl gibt die Standard-Zeichencodierungen für die Textbibliothek in **tencoding** und für die Zeichenketten-Bibliothek in **sencoding** zurück. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details.

#### EINGABEN

keine

#### RÜCKGABEWERTE

**tencoding**  
Standard-Zeichencodierung für die Text-Bibliothek

**sencoding**  
Standard-Zeichencodierung für die Zeichenketten-Bibliothek

## 52.18 GetFontColor

#### BEZEICHNUNG

GetFontColor – gibt die aktuelle Schriftfarbe zurück (V7.1)

#### ÜBERSICHT

```
color = GetFontColor()
```



**BESCHREIBUNG**

Dieser Befehl gibt die Schriftfarbe zurück, die mit `SetFontColor()` gesetzt wurde. Siehe [Abschnitt 52.33 \[SetFontColor\]](#), [Seite 1166](#), für Details.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`color`      aktuelle Schriftfarbe

## 52.19 GetFontStyle

**BEZEICHNUNG**

`GetFontStyle` – gibt den aktuellen Schriftstil zurück (V7.1)

**ÜBERSICHT**

```
style[, t] = GetFontStyle()
```

**BESCHREIBUNG**

Dieser Befehl gibt den aktuellen Schriftstil zurück, der mit `SetFontStyle()` gesetzt wurde. Der Rückgabewert `style` wird auf eine Kombination der Flags `#BOLD`, `#ITALIC`, `#UNDERLINED`, `#ANTIALIAS`, `#SHADOW`, und `#BORDER` gesetzt. Siehe [Abschnitt 52.34 \[SetFontStyle\]](#), [Seite 1166](#), für Details.

Zur Info: Vor Hollywood 9.0 wurde der Stil `#BORDER #EDGE` genannt.

Wenn `#SHADOW` gesetzt ist, gibt `GetFontStyle()` auch eine Tabelle als zweiten Rückgabewert zurück, die die folgenden Felder enthält:

**ShadowColor:**

Die Schattenfarbe.

**ShadowSize:**

Die Grösse bzw. der Abstand des Schattens von der Form in Pixeln.

**ShadowDir:**

Die Richtung des Schattens. Dies ist eine der [Richtungskonstanten](#).

Wenn `#BORDER` gesetzt ist, wird auch eine Tabelle mit den folgenden Feldern zurückgegeben:

**BorderColor:**

Die Farbe des Rahmens.

**BorderSize:**

Die Dicke des Rahmens in Pixel.

Siehe [Abschnitt 52.34 \[SetFontStyle\]](#), [Seite 1166](#), für mehr Informationen über die Schriftstile.

**EINGABEN**

keine

**RÜCKGABEWERTE**

`style`      eine Kombination der Schriftstile-Flags

**t** optional: Tabelle mit zusätzlichen Informationen über den Schriftstil (siehe oben)

## 52.20 GetKerningPair

### BEZEICHNUNG

GetKerningPair – gibt die Einstellung des Kerning für zwei benachbarte Zeichen zurück (V5.0)

### ÜBERSICHT

```
kern = GetKerningPair(a$, b$[, encoding])
```

### BESCHREIBUNG

Dieser Befehl berechnet den Kerning-Wert, der auf den Raum zwischen den beiden Zeichen **a\$** und **b\$** angewendet würde, wenn sie nebeneinander gezeichnet wurden. Kerning wird oft verwendet, um Leerzeichen zwischen zwei Zeichen zu reduzieren. Wenn beispielsweise ein "j"-Zeichen neben einem "i"-Zeichen gezeichnet wird, wird das "j" normalerweise einige Pixel nach links verschoben, so dass sein Unterhang unter dem "i" erscheint, wodurch der Text glatter aussieht. Der von diesem Befehl zurückgegebene Kerning-Wert wird in Pixeln angegeben. Ein negativer Kerning-Wert bedeutet eine Verschiebung nach links, während sich ein positiver Kerning-Wert nach rechts bewegt.

Das optionale Argument **encoding** kann verwendet werden, um die Zeichencodierung innerhalb eines **a\$** und **b\$** anzugeben.

Voreingestellt ist die Standardcodierung für die Textbibliothek, welche mit dem Befehl **SetDefaultEncoding()** festgelegt ist. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details.

Beachten Sie, dass nur ein einzelnes Zeichen in **a\$** und **b\$** sein dürfen, damit **GetKerningPair()** korrekt funktioniert.

### EINGABEN

**a\$** erstes Zeichen

**b\$** zweites Zeichen

**encoding** optional: Zeichencodierung, die von **a\$** und **b\$** verwendet werden (standardmäßig auf die beim letzten Aufruf von **SetDefaultEncoding()** angegebene Codierung der Textbibliothek)

### RÜCKGABEWERTE

**kern** Kerning-Wert für **a\$** und **b\$**

### BEISPIEL

```
SetFont(#SANS, 72)
```

```
SetFontStyle(#ANTIALIAS)
```

```
kern = GetKerningPair("W", "a")
```

Der obige Code berechnet den Kerning-Wert für die Zeichen "W" und "a" unter Verwendung des eingebauten Sans-Serif-Zeichensatzes in der Größe 72. Es wird -3 zurückgegeben, was bedeutet, dass das Zeichen "a" 3 Pixel zum Zeichen "W" verschoben wird.

## 52.21 Locate

### BEZEICHNUNG

Locate – setzt die Cursorposition

### ÜBERSICHT

Locate(*x*, *y*)

### BESCHREIBUNG

Dieser Befehl setzt den Cursor auf *x*/*y*. Die Cursorposition wird vom Befehl **Print** als Anfangsposition benutzt.

Bitte beachten Sie: Sie können keine von Hollywoods speziellen Konstanten für *x* oder *y* angeben, weil es keine Referenzhöhe oder -breite gibt. Deshalb können Werte wie **#CENTER**, **#BOTTOM**, **#RIGHT** etc. nicht funktionieren. Wenn Sie diese speziellen Konstanten benutzen möchten, müssen Sie den Befehl **TextOut()** benutzen, um Ihren Text auszugeben.

### EINGABEN

<i>x</i>	gewünschte neue x-Position
<i>y</i>	gewünschte neue y-Position

## 52.22 MoveTextObject

### BEZEICHNUNG

MoveTextObject – bewegt ein Textobjekt von *a* nach *b*

### ÜBERSICHT

MoveTextObject(*id*, *xa*, *ya*, *xb*, *yb*[, *table*])

### BESCHREIBUNG

Dieser Befehl bewegt (scrollt) das durch *id* angegebene Textobjekt ruckelfrei von der durch *xa*/*ya* zu der durch *xb*/*yb* angegebene Position.

Weitere Konfigurationsmöglichkeiten sind mit der optionalen Tabelle *table* möglich. Sie können die Bewegungsgeschwindigkeit, den Spezialeffekt und die asynchrone Bewegung festlegen. Siehe **Abschnitt 43.46 [MoveBrush]**, **Seite 948**, für weitere Informationen über das optionale Tabellenargument.

### EINGABEN

<i>id</i>	Identifikator des Textobjekts, das bewegt werden soll
<i>xa</i>	Anfangs-X-Position
<i>ya</i>	Anfangs-Y-Position
<i>xb</i>	End-X-Position
<i>yb</i>	End-Y-Position
<i>table</i>	optional: weitere Konfigurationen für das Bewegen

**BEISPIEL**

```
MoveTextObject(1,100,50,0,50,{Speed = 5})
```

Bewegt das Textobjekt 1 mit Geschwindigkeit 5 von 100:50 nach 0:50.

**52.23 NPrint****BEZEICHNUNG**

NPrint – schreibt Daten und hängt einen Zeilenumbruch an

**ÜBERSICHT**

```
NPrint(var, ...)
```

**BESCHREIBUNG**

Macht das gleiche wie **Print**, hängt aber noch einen Zeilenumbruch am Ende an.

**EINGABEN**

<b>var</b>	auszugebende Daten
<b>...</b>	andere Argumente (V2.0), siehe <b>Print</b>

**52.24 OpenFont****BEZEICHNUNG**

OpenFont – öffnet eine neue Schriftart (V4.5)

**ÜBERSICHT**

```
[id] = OpenFont(id, fontname$, size[, table])
```

**BESCHREIBUNG**

Dieser Befehl lädt die in **fontname\$** angegebene Schriftart und stellt sie Ihrem Skript unter dem angegebenen Identifikator **id** zur Verfügung. Wenn Sie **Nil** in **id** übergeben, wählt **OpenFont()** automatisch eine freie ID aus und gibt sie zurück. Nachdem die Schriftart erfolgreich geöffnet wurde, können Sie sie als aktuelle Schriftart mit dem Befehl **UseFont()** verwenden.

Die in **fontname\$** angegebene Schriftart muss der **Schriftdekleration** entsprechen. Siehe **Abschnitt 52.29 [Schriftdekleration]**, **Seite 1160**, für Details.

Mit dem optionalen Argument **table** können weitere Optionen eingestellt werden. Dies ist besonders nützlich, wenn Sie das integrierte Schriftmodul von Hollywood verwenden möchten, die ein pixelgenaues, identisches Erscheinungsbild über verschiedene Plattformen hinweg garantiert. Siehe **Abschnitt 52.32 [SetFont]**, **Seite 1163**, für Informationen, welche Tags im optionalen Tabellenargument verwendet werden können.

Normalerweise ist es bequemer, Schriftarten mit dem Befehl **SetFont()** direkt zu öffnen. Aber unter bestimmten Umständen, wenn Sie zum Beispiel viel zwischen verschiedenen Schriftarten wechseln müssen, ist es praktisch, diese Schriftarten mit dem Befehl **OpenFont()** vorzuladen. Sie sind dann schneller für Ihr Skript verfügbar.

Dieser Befehl ist auch als Präprozessor-Anweisung verfügbar. Verwenden Sie **@FONT**, um Schriften vorzuladen! Der Vorteil von **@FONT** ist, dass die dort angegebenen Zeichensätze beim kompilieren automatisch mit Ihrem Applet/der ausführbaren Datei verknüpft werden.

Siehe **Abschnitt 52.4 [Arbeiten mit Schriften]**, **Seite 1139**, für weitere Informationen plattformunabhängiger Verwendung von Schriftarten.

#### EINGABEN

**id**            Identifikator der neuen Schriftart oder **Nil** für die **automatische ID-Auswahl**

**fontname\$**        Schriftart, die geöffnet wird

**size**            gewünschte Größe der Schriftart in Pixel

**table**          optional: Tabelle mit weiteren Optionen (V4.7)

#### RÜCKGABEWERTE

**id**            optional: ID der neuen Schriftart; wird nur zurückgegeben, wenn Sie beim Argument **id Nil** angegeben haben (siehe oben)

#### BEISPIEL

```
OpenFont(1, "Arial", 36)
```

```
UseFont(1)
```

Öffnet Arial in der Größe 36 und wird zur aktuellen Schriftart.

## 52.25 Print

#### BEZEICHNUNG

Print – schreibt Daten auf den Bildschirm

#### ÜBERSICHT

```
Print(var, ...)
```

#### BESCHREIBUNG

Druckt die durch **var** angegebenen Daten auf den Bildschirm. Dieser Befehl kann mit allen unterschiedlichen Datentypen umgehen: Sie können Zeichenketten, Ganz-, Fließkommazahlen, Tabellen oder gar Funktionen drucken. Die Daten werden an der aktuellen Cursorposition eingefügt, die Sie durch Aufrufen von **Locate()** bestimmen können.

Der Befehl benutzt Wortumbruch, d.h. wenn der Rand erreicht ist und ein Wort nicht mehr in die Zeile passt, wird automatisch ein Zeilenumbruch ausgeführt. Sie können die Ränder manuell Mithilfe von **SetMargins()** festlegen. Ab Hollywood 9.1 können Sie auch bedingte Bindestriche oder Leerzeichen mit Nullbreite verwenden, um den Wortumbruch anzupassen. Da es sich jedoch um Unicode-Zeichen handelt, müssen Sie sicherstellen, dass Sie in diesem Fall die UTF-8-Codierung verwenden.

Dieser Befehl berücksichtigt auch Ihre Tabulatoreinstellungen. Wenn Sie eine Zeichenkette ausgeben möchten, die einen Tabulator enthält ("**\t**"), wird sie zur nächsten Tabulatorposition springen. Die Tabulatoreinstellungen können Sie mit den **AddTab()** und **ResetTabs()** Kommandos definieren.

Sie können hier ebenfalls Escape-Codes angeben: Siehe [Abschnitt 8.3 \[Zeichenketten\]](#), [Seite 104](#), für Details.

Wenn Ebenen aktiviert sind, wird dieser Befehl eine neue Ebene des Typs `#PRINT` dem Ebenenstapel hinzufügen.

Ab Hollywood 2.0 können Sie so viele Argumente übergeben, wie Sie möchten. Wenn Sie mehrere Argumente an diesen Befehl übergeben, müssen Sie mit einem Leerzeichen getrennt werden.

Beginnend mit Hollywood 2.5 können Sie mit [Textformatierungen](#) in Zeichenketten arbeiten und somit die Schriftart und Farbe des Textes on-the-fly steuern. Formatmarkierungen beginnen und enden immer mit einer eckigen Klammer (`'[ ]'`). Falls Sie nur eine eckige Klammer ausgeben möchten, müssen Sie zwei eckigen Klammern verwenden. Bei einer eckigen Klammer erwartet Hollywood immer eine Formatmarkierung. Siehe [Abschnitt 52.37 \[Textformatierungen\]](#), [Seite 1170](#), für Details.

Neben `Print()` können Sie auch die Befehle `NPrint()` und `TextOut()` verwenden, um Text auf dem Bildschirm auszugeben.

Beachten Sie, dass beim Schreiben auf ein palettenbasiertes Ziel und wenn der Palettenmodus auf `#PALETTEMODE_PEN` eingestellt ist, dieser Befehl mit dem Stift über `SetDrawPen()` schreibt anstelle der Schriftfarbe, welche mit dem Befehl `SetFontColor()` gesetzt wurde.

#### EINGABEN

```
var      auszugebende Daten
...      weitere Argumente (V2.0)
```

#### BEISPIEL

```
Print("Hello World!")
```

Gibt an der aktuellen Cursorposition "Hello World!" aus.

## 52.26 ResetTabs

#### BEZEICHNUNG

ResetTabs – löscht die Tabulatoreinstellungen

#### ÜBERSICHT

```
ResetTabs()
```

#### BESCHREIBUNG

Dieser Befehl löscht alle vorherigen Tabulatoreinstellungen und setzt die Voreinstellung (ein Tabulator wird in 8 Leerzeichen umgewandelt). Es gibt keinen Befehl, um einen einzelnen Tabulator zu entfernen. Wenn Sie dies tun möchten, müssen Sie diesen Befehl aufrufen und dann alle Tabulatoren mit dem Befehl [AddTab\(\)](#) hinzufügen, außerdem, den Sie entfernt haben möchten.

#### EINGABEN

```
keine
```

## 52.27 RotateTextObject

### BEZEICHNUNG

RotateTextObject – dreht ein Textobjekt (V4.0)

### ÜBERSICHT

RotateTextObject(id, angle[, smooth])

### BESCHREIBUNG

Dieser Befehl dreht das in `id` festgelegte Textobjekt um den angegebenen Winkel `angle` (in Grad). Ein positiver Winkel dreht sich gegen und ein negativer dreht sich im Uhrzeigersinn. Optional können Sie die Antialiasing-Interpolation aktivieren, indem Sie im Argument `smooth` `True` übergeben.

Beachten Sie, dass `RotateTextObject()` bei Vektortextobjekten immer mit dem nicht transformierten Textobjekt arbeitet. Das bedeutet, dass alle vorherigen Transformationen, die mit `RotateTextObject()`, `TransformTextObject()` oder `ScaleTextObject()` auf das Textobjekt angewendet wurden, beim Aufruf von `RotateTextObject()` rückgängig gemacht werden.

### EINGABEN

<code>id</code>	ID des Textobjekts, welches gedreht wird
<code>angle</code>	gewünschter Rotationswinkel in Grad
<code>smooth</code>	optional: ob Antialiasing-Interpolation verwendet werden soll oder nicht (V9.1)

## 52.28 ScaleTextObject

### BEZEICHNUNG

ScaleTextObject – skaliert ein Textobjekt (V4.0)

### ÜBERSICHT

ScaleTextObject(id, width, height[, smooth])

### BESCHREIBUNG

Dieser Befehl skaliert das in `id` angegebene Textobjekt auf die angegebene Dimension. Wenn das Textobjekt eine Vektorschriftart verwendet, wird es ohne Qualitätsverlust skaliert. Optional können Sie die Antialiasing-Interpolation aktivieren, indem Sie im Argument `smooth` `True` übergeben.

Sie können `#KEEPASPRAT` als Breite `width` oder Höhe `height` übergeben. Hollywood berechnet dann die Größe automatisch, indem das Seitenverhältnis des Textobjekts berücksichtigt wird. Breite und Höhe können auch eine Zeichenkette mit einer prozentualen Angabe sein, z.B. "50%".

Beachten Sie, dass `ScaleTextObject()` bei Vektortextobjekten immer auf dem nicht transformierten Textobjekt arbeitet. Das bedeutet, dass alle vorherigen Transformationen, die mit `ScaleTextObject()`, `TransformTextObject()` oder `RotateTextObject()` auf das Textobjekt angewendet wurden, beim Aufruf von `ScaleTextObject()` rückgängig gemacht werden.

**EINGABEN**

<b>id</b>	Identifikator des Textobjekts, welches skaliert wird
<b>width</b>	gewünschte neue Breite des Textobjekts
<b>height</b>	gewünschte neue Höhe des Textobjekts
<b>smooth</b>	optional: ob Antialiasing-Interpolation verwendet werden soll oder nicht (V9.1)

**BEISPIEL**

```
ScaleTextObject(1, 600, 200)
```

Skaliert das Textobjekt 1 auf die Größe von 600x200.

**52.29 Schriftdeklaration**

Die Hollywood-Schriftdeklaration ist die Notation, die verwendet werden muss, wenn neue Schriftarten mit **SetFont()**, **OpenFont()** oder **@FONT** zu öffnen sind. Diese drei Befehle benötigen eine Zeichenkette mit der Beschreibung der Schriftart, die Sie öffnen möchten. Diese Zeichenfolge muss diese Richtlinien befolgen:

1. Verwenden Sie keine Dateinamen, sondern den Schriftnamen! z.B.

```
SetFont("dh0:Fonts/Goudyb.font", 23)    ; --> falsch!
SetFont("Goudyb", 23)                   ; --> richtig!
OpenFont(1, "c:/Windows/Fonts/Arial.ttf", 36) ; --> falsch!
OpenFont(1, "Arial", 36)                 ; --> richtig!
```

**AUSNAHME:** Ab Hollywood 4.7 gibt es eine neues Schriftartenmodul namens **#FONTENGINE\_INBUILT**. Wenn Sie dieses Modul verwenden, können Sie die Font-Datei direkt angeben, aber nur für \*.ttf Schriften! So würde der folgende Code mit Hollywood 4.7 und höher legal sein:

```
OpenFont(1, "c:/Windows/Fonts/Arial.ttf", 36, {Engine = #FONTENGINE_INBUILT})
```

2. Auf manchen Systemen werden Groß- und Kleinschreibung im Fontnamen beachtet, wenn Sie **#FONTENGINE\_NATIVE** verwenden (zum Beispiel auf dem macOS). Daher sollten Sie immer den Namen in genau der gleichen Weise der Schriftart angeben, wie sie in der Schrift erscheint. Dies kann mögliche Probleme vermeiden.

```
SetFont("arial", 36) ; --> falsch!
SetFont("Arial", 36) ; --> richtig!
```

3. Für TrueType-Schriftarten besteht die Deklaration aus zwei Teilen: 1) Der Name der Schrift und 2) seine Stilparameter. Es muss ein Leerzeichen zwischen dem Namen und dem Stil sein. Wenn es mehrere Stile sind, müssen sie jeweils durch Leerzeichen getrennt werden. Z.B.

```
SetFont("Arial", 36)
SetFont("Arial Bold", 36)
SetFont("Arial Bold Italic", 36)
```

Natürlich könnte man auch "Arial" öffnen und dann **SetFontStyle()** mit **#BOLD** oder **#ITALIC** aufrufen, aber der Vorteil es direkt mit **SetFont()** zu laden ist, dass dies die



entworfenen fett/kursiv Variante der TrueType-Schriftart öffnet. `SetFontStyle()` auf der anderen Seite erstellt fett und kursiv mit einem Algorithmus, der nicht so gut aussieht wie die speziell für fett/kursiv entworfenen Schriften.

4. **Besonderer Hinweis für AmigaOS3, MorphOS, und AROS Benutzer:** FTManager verwendet häufig sehr umständliche Namen für die Schriftarten. Wenn Sie zum Beispiel versuchen, die Schriftart "Adobe Caslon Pro Bold Italic" mit dem FTManager zu installieren, wird diese Schrift standardmäßig mit dem Namen "adobecaslonprobolditalic" versehen. Sie werden dann zwar in der Lage sein, "adobecaslonprobolditalic" auf AmigaOS3, MorphOS oder AROS mit Hollywood zu öffnen, aber natürlich wird es nicht wegen diesem peinlichen Schriftnamen auf AmigaOS4, Windows oder macOS arbeiten. Daher sollten Sie den vorgeschlagenen Namen der Schriftart von FTManager auf folgende Weise bearbeiten:

1. Fügen Sie Leerzeichen zwischen den verschiedenen Komponenten ein:

```
"adobecaslonprobolditalic" -> "adobe caslon pro bold italic"
```

2. Passen Sie die Schreibweise des Namens an (angezeigt in FTManager):

```
"adobe caslon pro bold italic" -> "Adobe Caslon Pro bold italic"
```

3. Nutzen Sie alle Stileinstellungen:

```
"Adobe Caslon Pro bold italic" -> "Adobe Caslon Pro Bold Italic"
```

Wenn Sie diese Richtlinien befolgen, wird die Schriftart auch auf anderen Systemen und nicht nur mit AmigaOS3, MorphOS sowie AROS arbeiten.

## 52.30 SetBulletColor

### BEZEICHNUNG

SetBulletColor – stellt die Farbe des Aufzählungszeichen ein (V9.0)

### ÜBERSICHT

```
SetBulletColor(color)
```

### BESCHREIBUNG

Dieser Befehl legt die Farbe fest, die von Aufzählungszeichen verwendet werden soll, wenn `TextOut()` im Listenmodus verwendet wird. Standardmäßig werden Aufzählungszeichen in der aktuellen Schriftfarbe dargestellt, die mit dem Befehl `SetFontColor()` gesetzt wurde. Wenn Sie möchten, dass sie in einer anderen Farbe gezeichnet werden, können Sie diesen Befehl verwenden. Das Argument `color` muss entweder ein **RGB-** oder ein **ARGB-Wert** für Text mit Alpha-Blending sein.

Siehe [Abschnitt 52.39 \[TextOut\]](#), [Seite 1172](#), für weitere Informationen zu Aufzählungslisten.

### EINGABEN

`color`      **RGB-** oder **ARGB-Farbangabe**

### BEISPIEL

```
SetBulletColor(#GRAY)
```

Dieser Code setzt die Farbe des Aufzählungszeichens auf Grau.

```
SetBulletColor(ARGB(128, #RED))
```

Der obige Code setzt die Aufzählungszeichenfarbe auf Halbrod. Der Hintergrund scheint dann mit einem Verhältnis von 50% (128 = 50% von 255) durch den Text.

## 52.31 SetDefaultEncoding

### BEZEICHNUNG

SetDefaultEncoding – ändert die Standard-Zeichencodierung (V4.7)

### ÜBERSICHT

```
SetDefaultEncoding(tencoding[, sencoding])
```

### BESCHREIBUNG

Mit diesem Befehl kann die Standard-Zeichencodierungen für die Text- und Zeichenketten-Bibliotheken geändert werden. Beachten Sie, dass aus Gründen der Kompatibilität Hollywood zwei verschiedene Standard-Zeichencodierungen unterstützt: Eine für die Text- und eine für die Zeichenketten-Bibliothek. Unter normalen Umständen sollten jedoch beide Standard-Zeichencodierungen auf die gleiche Zeichencodierung gesetzt werden.

Die Standard-Zeichencodierungen für die Textbibliothek wird in **tencoding** angegeben und wirkt sich auf die Befehle wie **Print()**, **NPrint()**, **TextOut()** und **CreateTextObject()** aus.

Die Standard-Zeichencodierungen für die Zeichenketten-Bibliothek muss im Parameter **sencoding** angegeben werden und beeinflusst die meisten Befehle der Zeichenketten-Bibliothek wie z.B. **ReplaceStr()** und **StrLen()**.

Die folgenden Zeichencodierungen werden derzeit von Hollywood unterstützt:

**#ENCODING\_UTF8:**

Verwendet die UTF-8-Codierung. Dies ist der Standard seit Hollywood 7.0.

**#ENCODING\_ISO8859\_1:**

Verwendet die ISO 8859-1-Codierung. Dies war der Standard vor Hollywood 7.0. Beachten Sie, dass aus historischen Gründen **#ENCODING\_ISO8859\_1** auf AmigaOS und kompatiblen nicht wirklich ISO 8859-1 bedeutet, sondern immer die Standard-Zeichencodierung des Systems verwendet. Bei den meisten Amiga-Systemen ist das schon ISO 8859-1, aber osteuropäische Systeme verwenden beispielsweise eine andere Kodierung.

Ab Hollywood 7.0 sollte **#ENCODING\_ISO8859\_1** nicht mehr verwendet werden. Es wird aus Kompatibilitätsgründen noch unterstützt, aber es kann zu Problemen bei Nicht-ISO-8859-1-Systemen führen. Sie sollten ab Hollywood 7.0 immer **#ENCODING\_UTF8** verwenden.

### EINGABEN

**tencoding**

Standard-Zeichencodierung für die Text-Bibliothek

**sencoding**

Standard-Zeichencodierung für die Zeichenketten-Bibliothek (V7.0)

## 52.32 SetFont

### BEZEICHNUNG

SetFont – ändert die aktuelle Schriftart

### ÜBERSICHT

SetFont(font\$, size[, table])

### BESCHREIBUNG

Dieser Befehl ändert die aktuelle Schriftart auf die durch `font$` und `size` angegebenen Werte. Die gewünschte Grösse muss im Argument `size` in Pixel festgelegt werden. Die aktuelle Schriftart wird von Befehlen wie `Print()`, aber auch von `CreateTextObject()` verwendet. Die in `font$` angegebene Schriftart muss sich an die [Schriftdeklaration](#) halten. Siehe [Abschnitt 52.29 \[Schriftdeklaration\]](#), [Seite 1160](#), für Details.

Der Schriftstil wird beim Aufruf dieses Befehls zurückgesetzt.

Ab Hollywood 4.7 gibt es das optionale Argument `table`, mit dem Sie die folgenden erweiterten Optionen konfigurieren können:

**Engine:** Dieser Tag gibt an, welches Schriftartenmodul Hollywood für diese Schriftart verwenden soll. Dies kann entweder `#FONTENGINE_NATIVE` (nutzt das native Schriftartenmodul des Host-Betriebssystems) oder `#FONTENGINE_INBUILT` (verwendet das in Hollywood integrierte Schriftartenmodul) sein. Wenn Sie mit TrueType-Schriftarten in Ihrem Projekt arbeiten und wollen, dass Ihre Texte auf jeder Plattform genau gleich aussehen, müssen Sie sicherstellen, dass Sie das Modul `#FONTENGINE_INBUILT` verwenden, da sonst der Text von Plattform zu Plattform unterschiedlich aussehen wird. Ein weiterer Vorteil des Moduls `#FONTENGINE_INBUILT` ist, dass man direkt eine \*.ttf-Datei als `font$` ohne die Notwendigkeit der Installation auf dem lokalen System angeben kann. Siehe [Abschnitt 52.29 \[Schriftdeklaration\]](#), [Seite 1160](#), für Details. Aus Kompatibilitätsgründen ist dieser Tag standardmäßig `#FONTENGINE_NATIVE`. Beachten Sie, dass der Tag `Engine` seit Hollywood 10.0 veraltet ist. Sie sollten jetzt stattdessen den Tag `Loader` verwenden (siehe unten). Das Übergeben von `native` im Tag `Loader` entspricht dem Festlegen von `Engine` auf `#FONTENGINE_NATIVE` und das Übergeben von `inbuilt` im Tag `Loader` entspricht der `#FONTENGINE_INBUILT`-Engine. (V4.7)

**Cache:** Gibt an, ob der Glyphenspeicher verwendet werden soll. Glyphenspeicher kann die Leistung erheblich steigern, vor allem bei langsameren Systemen wie OS3, aber natürlich braucht es mehr Arbeitsspeicher. Glyphenspeicher wird derzeit nur von dem eingebauten Schriftartenmodul unterstützt (d.h. `#FONTENGINE_INBUILT`). Um diesen Arbeitsspeicher zu deaktivieren, setzen Sie diesen Tag auf `False`. Die Voreinstellung ist `True`. (V4.7)

**UsePoints:**

Setzen Sie dieses Feld auf `True`, wenn Sie eine Punktgröße anstelle einer Pixelgröße im Argument `size` übergeben möchten. Wenn Sie dieses Feld auf `True` setzen, interpretiert `SetFont()` den in `size` übergebenen Wert in Punkten (pt) anstelle von Pixeln. Im Allgemeinen wird nicht empfohlen, dieses Feld zu verwenden, da Punktgrößen immer von den Dots-per-inch (DPI) des Host-Displays abhängen, aber alle anderen Grafiken sind

typischerweise Pixel-Grafiken, die unabhängig von den DPI-Einstellungen des Host-Systems sind. Bei der Integration von Schriftarten, die mit einer Punkthöhe mit Pixelgrafiken geöffnet werden, können diese Schriftarten je nach DPI-Einstellungen des Host-Displays größer oder kleiner erscheinen und das Design so durcheinander bringen. Aus diesem Grund wird in der Regel nicht empfohlen, die Schrifthöhe in Punkten anstelle von Pixeln anzugeben. Voreinstellung ist `False`. (V7.0)

**CharMap:** Wenn `Engine` auf `#FONTENGINE_INBUILT` gesetzt wurde, können Sie mit dem Tag `CharMap` die Zeichensatztabelle angeben, die die Schriftart verwenden soll. Normalerweise ist es nicht notwendig, dies einzustellen, aber einige Schriftarten (z.B. Wingdings, Webdings) verwenden benutzerdefinierte Zeichensatztabelle, die nicht konsistent auf Unicode abgebildet werden können. In diesem Fall kann es hilfreich sein, dem Schriftartenmodul explizit mitzuteilen, welche Zeichensatztabelle sie verwenden soll. `CharMap` kann auf folgende Zeichensatztabellen eingestellt werden:

```
#CHARMAP_DEFAULT
#CHARMAP_MSSYMBOL
#CHARMAP_UNICODE
#CHARMAP_SJIS
#CHARMAP_BIG5
#CHARMAP_WANSUNG
#CHARMAP_JOHAB
#CHARMAP_ADOBESTANDARD
#CHARMAP_ADOBEEXPERT
#CHARMAP_ADOBECUSTOM
#CHARMAP_ADOBELATIN1
#CHARMAP_OLDLATIN2
#CHARMAP_APPLEROMAN
```

Der Standardwert ist `#CHARMAP_DEFAULT`. Um herauszufinden, welche Zeichensatztabelle von einer Schriftart unterstützt werden, verwenden Sie den Befehl `GetCharMaps()`. (V9.0)

**Loader:** Mit diesem Tag können Sie einen oder mehrere Formatlader angeben, die die Schriftart laden sollen. Dies muss auf eine Zeichenkette gesetzt werden, die den Namen eines oder mehrerer Lader enthält. Stellen Sie dies auf `native` ein, wenn Sie möchten, dass Hollywood die nativen Schriftartenmodule des Host-Betriebssystems für die Schriftart verwendet. Sie können `Loader` auch auf `inbuilt` setzen, um die in Hollywood integrierten Schriftartenmodule zu verwenden. Wenn Sie in Ihrem Projekt TrueType-Fonts verwenden und möchten, dass Ihre Texte auf allen Plattformen exakt gleich aussehen, müssen Sie darauf achten, dass Sie hier `inbuilt` übergeben, da sonst das Aussehen der Texte von Plattform zu Plattform unterschiedlich ist. Ein weiterer Vorteil der integrierten Schriftartenmodule ist, dass Sie eine `*.ttf`-Datei direkt als `font$` angeben können, ohne die Schriftart zuerst auf dem lokalen System installieren zu müssen. Siehe [Abschnitt 52.29 \[Schriftdekleration\]](#), [Seite 1160](#), für Details. Standardmäßig wird der Lader verwendet, der mit `SetDefaultLoader()` gesetzt wurde. Denken Sie daran, dass wenn

kein anderer Standard-Lader mit `SetDefaultLoader()` festgelegt wurde, dieser aus Kompatibilitätsgründen standardmäßig auf `native` gesetzt wird. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), Seite 96, für Details. (V10.0)

**Adapter:** Mit diesem Tag können Sie einen oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen. Dies muss auf eine Zeichenkette festgelegt werden, die den Namen eines oder mehrerer Adapter enthält. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), Seite 96, für Details. (V10.0)

**UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), Seite 100, für Details. (V10.0)

Hollywood hat drei integrierten TrueType-Schriften, die Sie verwenden können. Sie können diese mit folgenden speziellen Konstanten öffnen:

**#SANS:** Öffnet die integrierte Schriftart ohne Serifen.

**#SERIF:** Öffnet die integrierte Schriftart mit Serifen.

**#MONOSPACE:** Öffnet die integrierte nichtproportionale Schriftart (alle Zeichen haben dieselbe Breite)

Die Verwendung von eingebauten Schriftarten ist hilfreich, wenn Sie sicherstellen möchten, dass Ihr Skript auf anderen Systemen funktioniert, ohne zuvor einige Schriftarten installieren zu müssen. Wenn Sie die eingebauten Schriftarten von Hollywood verwenden, wird Ihr Skript sofort funktionieren. Beachten Sie, dass wenn Sie eine der eingebauten Schriftarten verwenden, Hollywood automatisch das eingebaute Schriftartenmodul benutzen wird, um sicherzustellen, dass die Schriftart auf jedem System genau gleich dargestellt wird.

Siehe [Abschnitt 52.4 \[Arbeiten mit Schriften\]](#), Seite 1139, für weitere Informationen plattformunabhängiger Verwendung von Schriftarten.

## EINGABEN

<code>font\$</code>	Name der Schriftart, welche geladen werden soll (oder eine der speziellen Konstanten)
<code>size</code>	gewünschte Größe (y-Größe) der Schriftart
<code>table</code>	optional: Tabelle mit weiteren Optionen (siehe oben) (V4.7)

## BEISPIEL

```
SetFont("times",18)
Print("Hello World")
```

Dieser Code setzt die Schriftart auf "times" mit der Größe 18 und gibt "Hello World" aus.

## 52.33 SetFontColor

### BEZEICHNUNG

SetFontColor – ändert die Farbe der aktuellen Schriftart

### ÜBERSICHT

SetFontColor(color)

### BESCHREIBUNG

Dieser Befehl ändert die Farbe der aktuellen Schriftart auf die in `color` angegebene, was ein **RGB-Wert** ist.

Neu in Hollywood 2.5: Die Farbe kann nun auch als **ARGB-Wert** für Text mit Alpha-Transparenz angegeben werden.

Beachten Sie, dass ab Hollywood 9.0 die Farbe, die Sie an diesen Befehl übergeben, auch als aktuelle Farbe für das Aufzählungszeichens festgelegt wird. Wenn Sie eine andere Farbe verwenden möchten, rufen Sie den Befehl **SetBulletColor()** auf.

### EINGABEN

`color`      **RGB-** oder **ARGB-Wert**

### BEISPIEL

SetFontColor(#GRAY)

Das setzt die Schriftfarbe auf ein grau.

SetFontColor(ARGB(128, #RED))

Der obige Code setzt die Schriftfarbe auf Halbtot. Der Hintergrund leuchtet dann mit einem Verhältnis von 50% (128 = 50% von 255) durch den Text.

## 52.34 SetFontStyle

### BEZEICHNUNG

SetFontStyle – ändert den Stil der aktuellen Schriftart

### ÜBERSICHT

SetFontStyle(style[, t])

### FRÜHERE SYNTAX

SetFontStyle(#SHADOW, color, distance, direction) (V2.5)

SetFontStyle(#BORDER, color, size) (V2.5)

### BESCHREIBUNG

Dieser Befehl ändert das Aussehen der aktuellen Schrift in den durch `style` angegebenen Stil, der eine beliebige Kombination der folgenden Textstile sein muss:

**#NORMAL:** Setzt den Schriftstil auf normal zurück. Dies kann nicht mit anderen Stil-Flags kombiniert werden.

**#BOLD:** Setzt den Stil auf fette Schrift.

**#ITALIC:** Setzt den Stil auf kursive Schrift.

**#UNDERLINED:**

Setzt den Stil auf unterstrichene Schrift.

**#ANTIALIAS:**

Benutzt für die Darstellung der Schriftart Antialias; Beachten Sie, dass Antialiasing nur für TrueType-Schriftarten verfügbar ist. (V2.0)

**#SHADOW:** Fügt dem Text einen Schatteneffekt hinzu. Wenn Sie diesen Stil festlegen, können Sie im optionalen Tabellenargument zusätzliche Argumente übergeben, um das Erscheinungsbild des Schatteneffekts zu steuern. Siehe unten für Details. (V2.5)

**#BORDER:** Fügt dem Text einen Rahmeneffekt hinzu. Wenn Sie diesen Stil festlegen, können Sie zusätzliche Argumente im optionalen Tabellenargument übergeben, um das Erscheinungsbild des Rahmeneffekts zu steuern. Siehe unten für Details. (V2.5)

Um mehrere Schriftstile in einem einzigen Aufruf zu kombinieren, verwenden Sie einfach die bit-oder-Verknüpfung ('|'), z.B. ein Aufruf von `SetFontStyle(#BOLD|#ITALIC)` setzt den Schriftstil auf fett und kursiv. Natürlich schließt sich der Stil `#NORMAL` gegenseitig aus und ist mit keinem anderen Stil kombinierbar.

Ab Hollywood 9.0 verwendet `SetFontStyle()` eine neue Syntax, die ein optionales Tabellenargument akzeptiert, das die folgenden Tags unterstützt:

**ShadowDir:**

Gibt die Richtung des Schattens an. Dies muss auf eine von Hollywoods **Richtungskonstanten** gesetzt werden. Dieser Tag wird nur berücksichtigt, wenn der Stil `#SHADOW` gesetzt wurde (siehe oben). (V9.0)

**ShadowColor:**

Gibt die Farbe des Schattens an. Dies muss ein **ARGB-Wert** sein, der eine Transparenzeinstellung enthalten kann. Dieser Tag wird nur berücksichtigt, wenn der Stil `#SHADOW` gesetzt wurde (siehe oben). (V9.0)

**ShadowSize:**

Gibt die Länge des Schattens an. Dieser Tag wird nur berücksichtigt, wenn der Stil `#SHADOW` gesetzt wurde (siehe oben). (V9.0)

**BorderColor:**

Gibt die Farbe des Rahmens an. Dies muss ein **ARGB-Wert** sein, der eine Transparenzeinstellung enthalten kann. Dieser Tag wird nur berücksichtigt, wenn der Stil `#BORDER` gesetzt wurde (siehe oben). (V9.0)

**BorderSize:**

Gibt die Dicke des Rahmens an. Dieser Tag wird nur berücksichtigt, wenn der Stil `#BORDER` gesetzt wurde (siehe oben). (V9.0)

Bitte beachten Sie, dass TrueType-Schriftarten oft separate Schriftarten für die jeweiligen Stile haben. In diesem Fall sollten Sie immer diese speziell gestalteten Schriftarten verwenden, da `SetFontStyle()` fette und kursive Stile mit einem Algorithmus erstellt, der oft nicht so gut aussieht wie handgearbeitete fette oder kursive Schriftarten. Wenn

Sie also vorhaben, Arial in Fettschrift zu verwenden, sollten Sie besser "Arial Bold" verwenden, wenn Sie `SetFont()`, `OpenFont()` oder `@FONT` aufrufen, als zuerst "Arial" und nachher `SetFontStyle()` mit `#BOLD` verwenden.

#### EINGABEN

`style`      Spezielle Stilkonstante (siehe Liste oben)  
`t`            optional: Tabelle mit zusätzlichen Argumenten (siehe oben) (V9.0)

#### BEISPIEL

```
SetFontStyle(#BOLD|#ITALIC)
```

Der obige Code setzt den Schriftstil auf fett und italic.

```
SetFontStyle(#SHADOW, {ShadowColor = ARGB(128, $939393),  
    ShadowSize = 16, ShadowDir = #SHDWSOUTHEAST})
```

Der obige Code ermöglicht einen halbtransparenten grauen Schatten, der 16 Pixel im Südosten des Haupttextes liegt.

## 52.35 SetMargins

#### BEZEICHNUNG

`SetMargins` – definiert die Ränder für ausgegebenen Text

#### ÜBERSICHT

```
SetMargins(left, right[, noclip])
```

#### BESCHREIBUNG

Dieser Befehl erlaubt Ihnen die Ränder zu setzen, die für einen Text benutzt werden sollen, der mit Hilfe von `Print()` ausgegeben wird. Das ist sehr nützlich, falls Sie Text nur in einem bestimmten Bereich Ihres Displays darstellen wollen. Das Argument `left` gibt den linken Rand an und das Argument `right` den rechten. Wörter, die diese Ränder überschreiten, werden automatisch in die nächste Zeile umgebrochen.

Die Voreinstellungen sind für links ist 0 und für rechts minus 1.

Standardmäßig werden die in `left` und `right` angegebenen Ränder an den Grenzen des Displays abgeschnitten. Wenn Sie das nicht möchten, können Sie das optionale Argument `noclip` auf `True` setzen. Wenn `noclip True` ist, kann `left` auch kleiner als 0 und `right` größer als die Display-Breite sein.

#### EINGABEN

`left`            linker Rand (in Pixeln)  
`right`          rechter Rand (in Pixeln)

#### BEISPIEL

```
SetMargins(200, 300)
```

```
Print("Hello World. This is my first program using margins.")
```

Obenstehender Code definiert die Ränder 200 und 300, was bedeutet, dass Textausgaben nur zwischen Pixel 200 und 300 vorgenommen werden. Dann schreibt es einigen Text auf den Bildschirm.



## 52.36 TextExtent

### BEZEICHNUNG

TextExtent – gibt detaillierte Informationen zu einem Textumfang zurück (V2.5)

### ÜBERSICHT

```
extent = TextExtent(string$, t])
```

### BESCHREIBUNG

Dieser Befehl gibt detaillierte Informationen über den Umfang der angegebenen Zeichenfolge `string$` mit den aktuellen Schriftart- und Stileinstellungen zurück. Im Gegensatz zu `TextWidth()`, die nur die Breite der Cursor-Vorwärtsbewegung zurückgibt, berechnet `TextExtent()` das exakte Grenzrechteck für die angegebene Zeichenkette.

Dieser Befehl gibt in den Feldern der Tabelle `extent` folgende Informationen zurück:

<b>MinX:</b>	Der Versatz auf der linken Seite des Rechtecks. Der ist oft negativ.
<b>MinY:</b>	Der Versatz von der Grundlinie bis zum oberen Rand des Rechtecks. Der ist immer negativ.
<b>MaxX:</b>	Der Versatz zur rechten Seite des Rechtecks.
<b>MaxY:</b>	Der Versatz von der Grundlinie bis zum unteren Rand des Rechtecks.
<b>Width:</b>	Dies ist der gleiche Wert wie beim Befehl <code>TextWidth()</code> .
<b>Height:</b>	Dies ist der gleiche Wert wie beim Befehl <code>TextHeight()</code> .

Die Werte in `MinX`, `MinY`, `MaxX` und `MaxY` beziehen sich immer auf die aktuelle Cursorposition. Wenn zum Beispiel `MinX` -10 ist, bedeutet dies, dass `Print()` diese Zeichenkette -10 Pixel von der aktuellen Cursorposition auf der X-Achse wiedergeben würde. Der Wert in `Width` gibt an, wo der Cursor nach dem Wiedergabevorgang enden würde. Dies ist oft weniger als `MaxX`-1. Im Fall von kursivem Text ist das letzte Zeichen normalerweise hinter der abschließenden Cursorposition.

Um die volle Breite der angegebenen Zeichenkette zu berechnen, subtrahiere einfach `MinX` von `MaxX` und addiere 1, d.h. `Volle_Breite = MaxX - MinX + 1`.

Ab Hollywood 4.7 gibt es das optionale Argument `encoding`, das verwendet werden kann, um die Zeichencodierung innerhalb von `string$` anzugeben. Dies ist standardmäßig auf die Zeichencodierung festgelegt, die als Standardcodierung der Textbibliothek mit dem Befehl `SetDefaultEncoding()` festgelegt wurde. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details.

Ab Hollywood 10.0 akzeptiert dieser Befehl ein optionales Tabellenargument, mit dem Sie die folgenden zusätzlichen Optionen angeben können:

#### Encoding:

Dieser Tag kann verwendet werden, um die von `string$` verwendete Zeichencodierung anzugeben. Es wird standardmäßig die Zeichencodierung verwendet, die als Standard der Textbibliothek eingestellt ist, die mit dem Befehl `SetDefaultEncoding()` festgelegt wurde. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details.

**Charspacing:**

Ermöglicht Ihnen, den Abstand zwischen den Zeichen anzupassen. Sie können diesen auf einen positiven oder negativen Wert setzen. Ein positiver Wert vergrößert den Abstand zwischen den Zeichen, ein negativer Wert verringert ihn. (V10.0)

**EINGABEN**

**string\$** Quelltext

**t** optional: Tabellenargument, das weitere Optionen enthält (siehe oben) (V10.0)

**RÜCKGABEWERTE**

**extent** detaillierte Informationen über die Textdimensionen

## 52.37 Textformatierungen

Seit der Version 2.5 ist Hollywood in der Lage, Textformatierung on-the-fly zu erledigen. Die Befehle **Print()**, **CreateTextObject()** und **TextOut()** unterstützen spezielle Formatmarkierungen, die es Ihnen ermöglichen, die Textfarbe und die -art zu ändern, ohne dass die Befehle **SetFontStyle()** oder **SetFontColor()** aufgerufen werden müssen.

Die folgenden Formatmarkierungen sind derzeit verfügbar:

- [b]**: Ändert den Schriftstil zu "fett" (bold). Mit **[/b]** wird "fett" beendet.
- [i]**: Ändert den Schriftstil zu "kursiv" (italic). Mit **[/i]** wird "kursiv" beendet.
- [u]**: Ändert den Schriftstil zu "unterstrichen". Mit **[/u]** wird "unterstrichen" beendet.

**[shadow=color,size,direction]:**

Fügt dem Text einen Schatteneffekt hinzu. Der neue Schatten wird dann die angegebene Farbe in **color** verwenden. Für den Pixelabstand wird der im Argument **size** angegebenen Wert benutzt und er wird nach der in **direction** angegebenen Richtung ausgerichtet. Bitte verwenden Sie eine der 8 **Richtungskonstanten** als Argument. Die Farbe kann in **RGB-** oder **ARGB-**Notation angegeben werden. Schattentransparenz wird voll unterstützt. Verwenden Sie **[/shadow]** um den Schatten-Stil zu beenden. Beachten Sie, dass wenn der Palettenmodus auf **#PALETTEMODE\_PEN** gesetzt ist und der Text auf ein Palettenziel gezeichnet wird, das Argument **color** keine **RGB-Farbe** sein darf, sondern ein Palettenstift.

**[border=color,size]:**

Fügt dem Text einen Rahmeneffekt hinzu. Dieser Rahmen verwendet die in **color** angegebene Farbe und die in **size** angegebene Dicke. Die Farbe kann eine **RGB-** oder **ARGB-Farbangabe** sein. Die Transparenz des Rahmens wird vollständig unterstützt. Verwenden Sie **[/border]**, um den Rahmenstil zu beenden. Beachten Sie, dass wenn der Palettenmodus auf **#PALETTEMODE\_PEN** gesetzt ist und der Text auf ein Palettenziel gezeichnet wird, das Argument **color** keine **RGB-Farbe** sein darf, sondern ein Palettenstift. Vor Hollywood 9.0 war dieser Tag als 'edge' bekannt.

`[color=color]:`

Ändert die Schriftfarbe in die Farbe `color`. Diese Farbe kann in **RGB** oder **ARGB** angegeben werden. Wenn Sie einen ARGB-Wert übergeben, wird der Text mit Überblendung wiedergegeben. Verwenden Sie `[/color]`, um die aktuelle Farbwiedergabe zu beenden und wieder die vorher eingestellte Farbe zu verwenden.

`[pen=pen]:`

Wenn der Palettenmodus auf `#PALETTEMODE_PEN` eingestellt ist und der Text auf ein Palettenziel gezeichnet wird, kann dieser Tag verwendet werden, um den Zeichnungsstift zu ändern. Verwenden Sie `[/pen]`, um den zuvor aktiven Stift wiederherzustellen. (V9.0)

`[bulletcolor=color]:`

Ändert die Farbe des Aufzählungszeichens in die in `color` angegebene. Aufzählungszeichen werden nur verwendet, wenn `TextOut()` im Listenmodus verwendet wird. Siehe **Abschnitt 52.39** [`TextOut`], **Seite 1172**, für Details. Die Farbe, die Sie diesem Tag übergeben, kann in der Notation **RGB** oder **ARGB** sein. Wenn Sie einen ARGB-Wert übergeben, wird das Aufzählungszeichen mit Überblendung wiedergegeben. Beachten Sie, dass dieser Tag im Gegensatz zu allen anderen oben genannten Tags nicht beendet werden darf. Er fungiert lediglich als Anweisung für den Formatprozessor, die aktuelle Aufzählungsfarbe zu ändern. Sie dürfen ihn also niemals mit `[/bulletcolor]` beenden. (V9.0)

`[bulletpen=pen]:`

Ändern Sie den Stift des Aufzählungszeichens in den in `pen` angegebenen. Aufzählungszeichen werden nur verwendet, wenn `TextOut()` im Listenmodus verwendet wird. Siehe **Abschnitt 52.39** [`TextOut`], **Seite 1172**, für Details. Beachten Sie, dass dieser Tag im Gegensatz zu allen anderen oben genannten Tags nicht beendet werden darf. Er fungiert lediglich als Anweisung für den Formatprozessor, die aktuelle Aufzählungsfarbe zu ändern. Sie dürfen ihn also niemals mit `[/bulletpen]` beenden. (V9.0)

Bitte beachten Sie, dass Sie wegen diesen Format-Tags zwei eckige Klammern ([`]`) verwenden müssen, wenn Sie eine eckige Klammer in Ihrem Text haben wollen. Wenn nur eine eckige Klammer im Text steht, erwartet Hollywood immer eine Formatmarkierung.

Hier ist ein Beispiel, wie Sie diese Formatmarkierungen mit den Befehlen der Textbibliothek verwenden können:

```
Print("Normal [b]Bold[/b] [i]Italic[/i] [u]Underlined[/u]")
```

Wie Sie sehen können, sind Formatmarkierungen wirklich einfach zu gebrauchen.

## 52.38 TextHeight

### BEZEICHNUNG

`TextHeight` – gibt die Höhe einer Zeichenkette zurück (V1.5)

### ÜBERSICHT

```
height = TextHeight(string$)
```

**BESCHREIBUNG**

Dieser Befehl gibt die Höhe des Textes zurück der durch `string$` angegeben wird, als wenn er auf das Display gezeichnet worden wäre. Also nimmt sie auf die aktuell eingestellte Schriftart ebenso Rücksicht wie auf den Schriftstil.

**EINGABEN**

`string$` Quelltext

**RÜCKGABEWERTE**

`height` Höhe des Textes

**BEISPIEL**

```
height = TextHeight("Hello World")
pos = (480 - height) / 2
Locate(0, pos)
Print("Hello World")
```

Obenstehendes Beispiel zentriert den Text "Hello World" vertikal auf einem Nx480 Display.

## 52.39 TextOut

**BEZEICHNUNG**

`TextOut` – gibt Text auf dem Bildschirm aus

**ÜBERSICHT**

```
TextOut(x, y, text$, table))
```

**BESCHREIBUNG**

Dieser Befehl gibt den in `text$` angegebenen Text an der Stelle aus, die durch die Koordinaten `x` und `y` festgelegt ist. Sie hat den Vorteil, dass Sie Hollywoods spezielle Konstanten als Koordinaten benutzen (z.B. `#CENTER`, `#BOTTOM`...) was bei dem Befehl `Print()` nicht möglich ist, da `Locate()` diese Richtungskonstanten nicht verarbeiten kann. Wenn Ebenen aktiviert sind, fügt dieser Befehl eine neue Ebene vom Typ `#TEXTOUT` dem Ebenenstapel hinzu.

Ab Hollywood 2.5 können Sie die **Textformatierungen** von Hollywood in der Zeichenkette `text$` an `CreateTextObject()` übergeben. Mit diesen Formatmarkierungen können Sie die Schriftart und Farbe Ihres Textes on-the-fly ändern. Formatmarkierungen beginnen und enden immer mit einer eckigen Klammer (`'[']`). Wenn Sie nur eine eckige Klammer ausgeben möchten, müssen Sie zwei eckige Klammern (`'[[']`) verwenden. Bei nur einer eckigen Klammer erwartet Hollywood immer eine Formatmarkierung. Weitere Informationen zu diesem Thema finden Sie im Kapitel **Textformatierungen**.

Seit Hollywood 4.0 hat sich die Syntax von diesem Befehl etwas verändert. Während die alte Syntax noch wegen der Kompatibilität unterstützt wird, sollten Sie für neue Skripts die aktuelle Syntax verwenden, welche die Tabelle `table` als Argument 4 akzeptiert. Die Tabelle kann die folgenden Elemente enthalten:

**Align:** Ermöglicht die Angabe der Ausrichtung des Textes nach einem Zeilenumbruchzeichen. Folgende Ausrichtungen werden unterstützt:

`#LEFT` Linksbündig.

**#RIGHT**      Rechtsbündig.

**#CENTER**     Zentriert.

**#JUSTIFIED**

Gibt den Text in Blocksatz aus. (V7.0)

Für **Align** ist **#LEFT** voreingestellt.

#### WordWrap:

Wenn Sie diesen zusätzlichen Parameter angeben, kann **TextOut()** einen automatischen Wortumbruch für Sie durchführen. Mit diesem Tag können Sie eine maximale Breite für Ihren Text angeben. **TextOut()** wird dann den Wortumbruch verwenden, um sicherzustellen, dass kein Text über dieses Limit hinausgeht. Wenn Sie dieses Argument nicht oder auf 0 setzen (was auch voreingestellt ist), ist der Text so breit wie er benötigt wird. Ab Hollywood 9.1 können Sie auch bedingte Bindestriche oder Leerzeichen mit Nullbreite verwenden, um den Wortumbruch anzupassen. Da es sich jedoch um Unicode-Zeichen handelt, müssen Sie sicherstellen, dass Sie in diesem Fall die UTF-8-Codierung verwenden.

#### Encoding:

Dieser Parameter kann verwendet werden, um die Zeichencodierung von **text\$** anzugeben. Voreingestellt ist die Standardcodierung für die Textbibliothek, welche mit dem Befehl **SetDefaultEncoding()** festgelegt ist. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details. (V4.7)

#### Color:

Mit diesem Tag können Sie die Textfarbe angeben. Die Farbe muss als **ARGB-Wert** angegeben werden. Wenn Sie diesen Tag nicht angeben, verwendet **TextOut()** die Farbe, die mit dem Befehl **SetFontColor()** festgelegt wurde.

#### Pen:

Wenn der Palettenmodus **#PALETTEMODE\_PEN** ist, kann dieser Tag verwendet werden, um den Stift festzulegen, der zum Zeichnen des Textes verwendet werden soll. Wenn der Palettenmodus **#PALETTEMODE\_PEN** ist und **Pen** nicht angegeben ist, wird stattdessen der Stift verwendet, der mit **SetDrawPen()** gesetzt wurde. (V9.0)

#### Linespacing:

Mit diesem Tag können Sie den Abstand zwischen den Zeilen anpassen. Sie können dies auf einen positiven oder negativen Wert setzen. Ein positiver Wert vergrößert den Zeilenabstand, ein negativer Wert verringert ihn. (V9.0)

#### Charspacing:

Ermöglicht Ihnen, den Abstand zwischen den Zeichen anzupassen. Sie können diesen auf einen positiven oder negativen Wert setzen. Ein positiver Wert vergrößert den Abstand zwischen den Zeichen, ein negativer Wert verringert ihn. (V10.0)

#### Tabs:

Wenn Sie Tabulatoren verwenden möchten, müssen Sie diesen Tag auf eine Tabelle setzen, die die gewünschten Tabulatoren enthält. Die Tabulatoren müssen als Pixelwerte relativ zur x-Position übergeben werden. Wenn die

Zeichenkette, die Sie an `TextOut()` übergeben, Tabulatoren enthält und Sie diesen Tag nicht setzen, werden alle Tabulatoren in Leerzeichen umgewandelt. (V9.0)

#### **ListMode:**

Setzen Sie diesen Tag auf `True`, um `TextOut()` in den Listenmodus zu versetzen. Im Listenmodus können Sie mit `TextOut()` geordnete und ungeordnete Listen erstellen. Im Listenmodus müssen Sie in der Zeichenkette, die Sie an `TextOut()` übergeben, Tabulatoren verwenden, um die gewünschte Einrückung zu signalisieren. Standardmäßig verwenden alle Listenelemente das in `DefListBullet` angegebene Aufzählungszeichen. Es ist auch möglich, `TextOut()` anzuweisen, benutzerdefinierte Aufzählungszeichen zu verwenden, indem Sie den Tag `ListBullet` setzen. Dasselbe gilt für die Einrückung, den Versatz und Abständen der Liste, wobei die Standardwerte mit `DefListIndent`, `DefListOffset` sowie `DefListSpacing` und die benutzerdefinierten Werte mit `ListIndent` und `ListOffset` festgelegt werden können. Weitere Informationen zu all diesen Optionen finden Sie weiter unten. Beachten Sie auch, dass sich `ListMode` und `Tab` gegenseitig ausschließen. Sie können nicht beide gleichzeitig verwenden. (V9.0)

#### **DefListBullet:**

Verwenden Sie diesen Tag, um das Standardaufzählungszeichen festzulegen, das verwendet wird, wenn sich `TextOut()` im Listenmodus befindet. Dies kann auf ein Unicode-Zeichen (als numerischer Codepunkt, nicht als Zeichenkette!) oder eine der folgenden Konstanten gesetzt werden:

##### **#BULLET\_DASH:**

Strich. Das ist die Voreinstellung.

##### **#BULLET\_CROSS:**

Kreuz.

##### **#BULLET\_CIRCLE:**

Kreis.

##### **#BULLET\_HOLLOWCIRCLE:**

Hohlkreis. (V9.1)

##### **#BULLET\_BOX:**

Rechteck.

##### **#BULLET\_CHECKMARK:**

Häkchen.

##### **#BULLET\_ARROW:**

Pfeil.

##### **#BULLET\_DIAMOND:**

Diamand.

##### **#BULLET\_NUMERIC:**

Nummerierte Liste: 1. 2. 3...

**#BULLET\_NUMERICSSINGLE:**  
Nummerierte Liste: 1) 2) 3)...

**#BULLET\_NUMERICDOUBLE:**  
Nummerierte Liste: (1) (2) (3)...

**#BULLET\_LALPHA:**  
Nummerierte Liste: a. b. c...

**#BULLET\_LALPHASINGLE:**  
Nummerierte Liste: a) b) c)...

**#BULLET\_LALPHADOUBLE:**  
Nummerierte Liste: (a) (b) (c)...

**#BULLET\_UALPHA:**  
Nummerierte Liste: A. B. C...

**#BULLET\_UALPHASINGLE:**  
Nummerierte Liste: A) B) C)...

**#BULLET\_UALPHADOUBLE:**  
Nummerierte Liste: (A) (B) (C)...

**#BULLET\_LROMAN:**  
Nummerierte Liste: i. ii. iii...

**#BULLET\_LROMANSINGLE:**  
Nummerierte Liste: i) ii) iii)...

**#BULLET\_LROMANDOUBLE:**  
Nummerierte Liste: (i) (ii) (iii)...

**#BULLET\_URMAN:**  
Nummerierte Liste: I. II. III...

**#BULLET\_URMANSINGLE:**  
Nummerierte Liste: I) II) III)...

**#BULLET\_URMANDOUBLE:**  
Nummerierte Liste: (I) (II) (III)...

**#BULLET\_NONE:**  
Kein Aufzählungszeichen.

Wenn Sie in Ihrer Liste verschiedene Arten von Aufzählungszeichen verwenden müssen, müssen Sie stattdessen den Tag `ListBullet` verwenden. Weitere Informationen finden Sie unten.

Beachten Sie auch, dass der Startversatz von nummerierten Aufzählungstypen wie `#BULLET_NUMERIC`, `#BULLET_LALPHA` usw. mit den Tags `DefListOffset` und `ListOffset` festgelegt werden kann. Weitere Informationen finden Sie unten.

Standardmäßig verwenden Aufzählungszeichen die Farbe, welche mit `SetFontColor()` eingestellt wurde. Um eine andere Farbe einzustellen, verwenden Sie den Befehl `SetBulletColor()`. Siehe [Abschnitt 52.30 \[SetBulletColor\]](#), [Seite 1161](#), für Details. (V9.0)

**ListBullet:**

Wenn Sie verschiedene Arten von Aufzählungszeichen in Ihrer Liste verwenden möchten, müssen Sie diesen Tag auf eine Tabelle setzen, die die einzelnen Aufzählungszeichen enthält, die Ihre Liste verwenden soll. `TextOut()` extrahiert dann für jede neue Liste, die er beginnt, einen neuen Aufzählungstyp aus Ihrer Tabelle. Wie bei `DefListBullet` können die einzelnen Elemente in der Tabelle, die Sie an `ListItems` übergeben, entweder numerische Unicode-Codepunkte oder vordefinierte `#BULLET_XXX`-Konstanten sein, wie oben beschrieben. Wenn in `ListBullet` mehr Listen als Tabellenelemente vorhanden sind, wird das in `DefListBullet` angegebene Aufzählungszeichen verwendet. (V9.0)

**DefListIndent:**

Dieser Tag kann verwendet werden, um die Anzahl der Leerzeichen anzugeben, die zum Einrücken von Listenelementen verwendet werden sollen. Die Standardeinrückung ist 4, was bedeutet, dass Listenelemente standardmäßig mit 4 Leerzeichen eingerückt werden. Sie können auch benutzerdefinierte Stufen der Einrückung für bestimmte Listen angeben. Dies kann mit dem Tag `ListIndent` erfolgen. Siehe unten für weitere Details. (V9.0)

**ListIndent:**

Wenn Sie in Ihrer Liste verschiedene Einrückungsstufen verwenden möchten, müssen Sie diesen Tag auf eine Tabelle setzen, die die einzelnen Einrückungen enthält, die Ihre Liste verwenden soll. `TextOut()` extrahiert dann für jede neue Liste, die er beginnt, einen neuen Einzugswert aus Ihrer Tabelle. Wenn in `ListIndent` mehr Listen als Tabellenelemente vorhanden sind, wird die in `DefListIndent` angegebene Standardeinrückung verwendet. (V9.0)

**DefListOffset:**

Wenn Sie einen nummerierten Aufzählungstyp wie `#BULLET_NUMERIC` oder `#BULLET_LALPHA` verwenden, können Sie diesen Tag verwenden, um einen Startversatz für die Nummerierung anzugeben. Beachten Sie, dass Versätze ab 0 gezählt werden. Wenn Sie hier also einen Versatz von 0 angeben, beginnt die Nummerierung bei `#BULLET_NUMERIC` bei 1 und bei `#BULLET_ALPHA` bei "a". Der Standardwert für dieses Tag ist 0. Sie können auch benutzerdefinierte Versatzstufen für bestimmte Listen angeben. Dies kann mithilfe des Tags `ListOffset` erfolgen, siehe unten für weitere Details. (V9.0)

**ListOffset:**

Wenn Sie in Ihrer Liste verschiedene Versätze verwenden möchten, müssen Sie diesen Tag auf eine Tabelle setzen, die die einzelnen Versätze enthält, die Ihre Liste verwenden soll. `TextOut()` extrahiert dann für jede neue Liste, die er startet, einen neuen Listen-Versatz-Wert aus Ihrer Tabelle. Wenn in `ListOffset` mehr Listen als Tabellenelemente vorhanden sind, wird der in `DefListOffset` angegebene Standardlisten-Versatz verwendet. (V9.0)

**DefListSpacing:**

Mit diesem Tag kann der Standardabstand angegeben werden, der zwischen den Elementen in den einzelnen Listen verwendet werden soll. Dies ist standardmäßig 0. Sie können auch benutzerdefinierte Abstandswerte für



bestimmte Listen angeben. Dies kann durch die Verwendung des Tags `ListSpacing` erfolgen, siehe unten für weitere Einzelheiten. (V9.1)

**ListSpacing:**

Wenn Sie unterschiedliche Zeilenabstände für Ihre Listen verwenden möchten, müssen Sie diesen Tag auf eine Tabelle setzen, die die einzelnen Zeilenabstände enthält, die Ihre Listen verwenden sollen. `TextOut()` extrahiert dann einen neuen Abstandswert aus Ihrer Tabelle für jede neue Liste, die neu beginnt. Wenn in `ListSpacing` mehr Listen als Tabellenelemente vorhanden sind, wird der in `DefListSpacing` angegebene Standardabstandswert verwendet. (V9.1)

**FrameMode:**

Wenn `TextOut()` im Listenmodus verwendet wird und Ebenen aktiviert sind, hat die resultierende Ebene vom Typ `#TEXTOUT` mehrere Einzelbilder, die Sie mit `NextFrame()` oder allen anderen Ebenenbefehlen, die Animations Ebenen unterstützen, durchlaufen können. Dadurch ist es möglich, nacheinander durch die Listeneinträge zu blättern oder einen Eintrag nach dem anderen anzuzeigen. Das erste Einzelbild enthält immer alle Listenelemente. Der Inhalt der anderen Einzelbilder hängt davon ab, was im Tag `FrameMode` angegeben wurde. Dieser Tag kann wie folgt eingestellt werden:

**#FRAMEMODE\_SINGLE:**

Wenn Sie diesen Einzelbild-Modus verwenden, ist pro Einzelbild nur ein einziges Listenelement sichtbar, d.h. das Einzelbild 2 enthält nur das erste Listenelement, das Einzelbild 3 enthält nur das zweite Listenelement und so weiter.

**#FRAMEMODE\_FULL:**

Wenn Sie diesen Einzelbild-Modus verwenden, sind immer auch alle vorherigen Listenelemente sichtbar. Das bedeutet, dass das Einzelbild 3 den ersten und den zweiten Listeneintrag enthält, das Einzelbild 4 die ersten drei Listeneinträge und so weiter.

Wenn nicht angegeben, ist `FrameMode` standardmäßig auf `#FRAMEMODE_FULL` eingestellt. Beachten Sie, dass Sie mit dem Tag `Frame` (siehe unten) festlegen können, welches Einzelbild anfänglich sichtbar sein soll. (V9.0)

**Frame:**

Wenn `TextOut()` im Listenmodus verwendet wird und Ebenen aktiviert sind, hat die resultierende Ebene vom Typ `#TEXTOUT` mehrere Einzelbilder, die Sie mit `NextFrame()` oder allen anderen Ebenenbefehlen, die Animations Ebenen unterstützen, durchlaufen können. Dadurch ist es möglich, nacheinander durch die Listeneinträge zu blättern oder einen Eintrag nach dem anderen anzuzeigen. Das erste Einzelbild enthält immer alle Listenelemente. Der Inhalt der anderen Einzelbilder hängt davon ab, was im Tag `FrameMode` angegeben wurde (siehe oben). Der Tag `Frame` kann verwendet werden, um das Einzelbild anzugeben, das zuerst sichtbar sein soll. Einzelbilder werden ab 1 gezählt. (V9.0)

**SimpleList:**

Wenn dies auf **True** gesetzt ist, extrahiert **TextOut()** nicht nacheinander die Konfiguration der Listenauflählungszeichen aus den Tabellen **ListBullet** usw., sondern verwendet einfach statisch das Element am angegebenen Tabulatorindex. D.h. Tabulatorposition 1 verwendet immer das Aufzählungszeichen, das bei Index 1 in der Tabelle **ListBullet** angegeben ist, Tabulatorposition 2 verwendet das Aufzählungszeichen, das bei Index 2 in der Tabelle **ListBullet** angegeben ist usw. Dies schränkt Ihre Flexibilität ein, kann aber die Arbeit erleichtern, wenn Sie immer dieselbe Konfiguration für jede Tabulatorposition haben wollen. (V9.1)

Darüber hinaus kann das optionale Tabellenargument **table** auch eine oder mehrere der **Standard-Tags zum Zeichnen** enthalten. Siehe **Abschnitt 29.17 [Standard-Tags zum Zeichnen]**, **Seite 613**, für Details.

Beachten Sie, dass Hollywood nur westlichen Standard von links nach rechts basierten Text auf horizontalen Linien ausgerichtet unterstützt. Von rechts nach links und vertikalem Text wird derzeit nicht unterstützt.

Beachten Sie auch, dass beim Schreiben auf ein palettenbasiertes Ziel und wenn der Palettenmodus auf **#PALETTEMODE\_PEN** eingestellt ist, dieser Befehl mit dem Stift über **SetDrawPen()** schreibt anstelle der Schriftfarbe, welche mit dem Befehl **SetFontColor()** oder dem obigen Tag **Color** gesetzt wurde.

**EINGABEN**

<b>x</b>	x-Position für den Text
<b>y</b>	y-Position für den Text
<b>text\$</b>	Zeichenkette, welche ausgegeben wird
<b>table</b>	optional: Tabelle mit weiteren Konfigurationsmöglichkeiten (siehe oben) (V4.0)

**BEISPIEL**

```
TextOut(#CENTER, #CENTER, "Hello World!")
```

Der obige Code gibt "Hello World!" in der Mitte des Displays aus.

```
For Local k = 100 To 600 Step 100 Do
  Line(k, 0, k, 480, #RED)
TextOut(0, 0, "One\tTwo\tThree\tFour\tFive\tSix",
  {Tabs = {100, 200, 300, 400, 500}})
```

Dieser Code zeigt, wie man Tabulatoren mit **TextOut()** verwendet.

```
SetFont(#SANS, 18)
TextOut(0, 0, "Pizzas\n"..
  "\tProsciutto\n"..
  "\tFunghi\n"..
  "\tMargarita\n"..
  "Drinks\n"..
  "\tAlcoholic\n"..
```

```

"\t\tBeer\n"..
"\t\tWine\n"..
"\t\tNon-alcoholic\n"..
"\t\tCoke\n"..
"\t\tWater",
{ListMode = True,
DefListBullet = #BULLET_CIRCLE,
ListBullet = {#BULLET_DASH}})

```

Der obige Code zeigt, wie man eine Liste mit `TextOut()` erstellt.

## 52.40 TextWidth

### BEZEICHNUNG

`TextWidth` – gibt die Breite eines Textes zurück

### ÜBERSICHT

```
width = TextWidth(string$, t)
```

### FRÜHERE SYNTAX

```
width = TextWidth(string$, encoding)
```

### BESCHREIBUNG

Dieser Befehl gibt die Breite des in `string$` angegebenen Textes zurück, als wenn er auf das Display gezeichnet wäre. Deshalb berücksichtigt sie die gewählte Schriftart ebenso wie den Schriftstil.

Hinweis: Dieser Befehl gibt nur die Vorwärtsbewegung des Cursors zurück. Dies ist oft weniger als der Text tatsächlich belegt, wenn er auf das Display geschrieben wird. Wenn Sie detaillierte Informationen über den tatsächlichen Umfang eines Textes benötigen, verwenden Sie stattdessen den Befehl `TextExtent()`.

Ab Hollywood 10.0 akzeptiert dieser Befehl ein optionales Tabellenargument, mit dem Sie die folgenden zusätzlichen Optionen angeben können:

#### Encoding:

Dieser Tag kann verwendet werden, um die von `string$` verwendete Zeichencodierung anzugeben. Dies verwendet standardmäßig die Zeichencodierung, die als Standardcodierung der Textbibliothek `SetDefaultEncoding()` verwendet wird. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details. (V10.0)

#### Charspacing:

Ermöglicht Ihnen, den Abstand zwischen den Zeichen anzupassen. Sie können diesen auf einen positiven oder negativen Wert setzen. Ein positiver Wert vergrößert den Abstand zwischen den Zeichen, ein negativer Wert verringert ihn. (V10.0)

### EINGABEN

<code>string\$</code>	Quelltext
<code>t</code>	optional: Tabellenargument, das weitere Optionen enthält (siehe oben) (V10.0)

**EINGABEN**

**string\$** Quelltext

**encoding** optional: Zeichencodierung, die von **string\$** verwendet wird (standardmäßig auf die beim letzten Aufruf von **SetDefaultEncoding()** angegebene Codierung der Textbibliothek) (V4.7)

**RÜCKGABEWERTE**

**width** Breite des Texts

**BEISPIEL**

```
width = TextWidth("Hello World")
pos = (640 - width) / 2
Locate(pos, 0)
Print("Hello World")
```

Dieses Beispiel zentriert den Text "Hello World" horizontal auf einem 640 Pixel breiten Display.

## 52.41 TransformTextObject

**BEZEICHNUNG**

TransformTextObject – wendet eine affine Transformation auf ein Textobjekt an (V10.0)

**ÜBERSICHT**

```
TransformTextObject(id, sx, rx, ry, sy[, smooth])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um eine affine Transformation auf ein Textobjekt anzuwenden. Sie müssen diesem Befehl eine 2x2-Transformationsmatrix übergeben, die definiert, wie jedes Pixel im Textobjekt transformiert wird. Dieser Befehl ist nützlich, wenn Sie gleichzeitig Drehung und Skalierung anwenden möchten. Natürlich könnte man das mit Aufrufen von **ScaleTextObject()** und dann **RotateTextObject()** machen, aber das würde zu Qualitätseinbußen führen. Wenn Sie die Transformation stattdessen mit **TransformTextObject()** durchführen, wird alles in einem einzigen Durchlauf erledigt.

Die 2x2-Transformationsmatrix besteht aus vier Fließkommafaktoren:

**sx:** Gibt den Skalierungswert auf der x-Achse an. Dieser darf nicht Null sein. Wenn er negativ ist, wird das Textobjekt auf der y-Achse gespiegelt.

**rx:** Gibt den Grad der Drehung auf der x-Achse an. Dies kann 0 sein.

**ry:** Gibt den Grad der Drehung auf der y-Achse an. Dies kann 0 sein.

**sy:** Gibt den Skalierungswert auf der y-Achse an. Dieser darf nicht Null sein. Wenn er negativ ist, wird das Textobjekt auf der x-Achse gespiegelt.

Die Identitätsmatrix ist definiert als

$$\begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$$

Wenn Sie diese Matrix übergeben, wird keine Transformation angewendet, da keine Rotation und keine Skalierung definiert sind. D.h. wenn Hollywood diese Matrix auf jedes

Pixel in Ihrem Textobjekt anwenden würde, wäre das Ergebnis nur eine Kopie des Textobjekts. Aber wenn `TransformTextObject()` feststellt, dass Sie eine Identitätsmatrix übergeben haben, wird es natürlich sofort zurückkehren und nichts tun.

Das optionale Argument `smooth` kann auf `True` gesetzt werden, wenn Hollywood bei der Transformation interpolieren soll. Dies führt zu Ergebnissen, die besser aussehen, aber die Interpolation ist ziemlich langsam.

Bitte beachten Sie: Sie sollten Transformationsoperationen immer mit dem ursprünglichen Textobjekt durchführen. Wenn Sie beispielsweise Textobjekt 1 in 12 x 8 Pixel skalieren und es dann wieder in 640 x 480 umwandeln, erhalten Sie ein durcheinandergebrachtes Textobjekt. Daher sollten Sie immer das ursprüngliche Textobjekt behalten und nur Kopien davon transformieren.

Beachten Sie, dass `TransformTextObject()` bei Vektortextobjekten immer auf dem nicht transformierten Textobjekt operiert. Das bedeutet, dass alle vorherigen Transformationen, die mit `TransformTextObject()`, `ScaleTextObject()` oder `RotateTextObject()` auf das Textobjekt angewendet wurden, rückgängig gemacht werden, wenn `TransformTextObject()` aufgerufen wird.

#### EINGABEN

<code>id</code>	ID des zu transformierenden Textobjekts
<code>sx</code>	Faktor x skalieren; darf nie 0 sein
<code>rx</code>	Faktor x drehen
<code>ry</code>	Faktor y drehen
<code>sy</code>	Faktor y skalieren; darf nie 0 sein
<code>smooth</code>	optional: ob die affine Transformation Interpolation verwenden soll oder nicht

#### BEISPIEL

```
angle = Rad(45)      ; convert degrees to radians
TransformTextObject(1, Cos(angle), Sin(angle), -Sin(angle), Cos(angle))
```

Der obige Code dreht das Textobjekt Nummer 1 um 45 Grad unter Verwendung einer 2x2-Transformationsmatrix.

## 52.42 UseFont

#### BEZEICHNUNG

UseFont – wechselt die aktuelle Schriftart (V4.5)

#### ÜBERSICHT

`UseFont(id)`

#### BESCHREIBUNG

Dieser Befehl ändert die aktuelle in `id` angegebene Schriftart. Die hier angegebene Schriftarten-ID muss entweder von `OpenFont()` oder `@FONT` vorgeladen worden sein.

Die Schriftart wird beim Aufruf dieses Befehls zurückgesetzt.

**EINGABEN**

id            Identifikator der Schriftart

**BEISPIEL**

Siehe [Abschnitt 52.24 \[OpenFont\]](#), Seite 1156.

## 53 Vektorgrafikbibliothek

### 53.1 AddArcToPath

#### BEZEICHNUNG

AddArcToPath – fügt einen elliptischen Bogen dem Pfad hinzu (V5.0)

#### ÜBERSICHT

AddArcToPath(id, xc, yc, ra, rb, start, end[, clockwise])

#### BESCHREIBUNG

Dieser Befehl fügt einen elliptischen Bogen in den Pfad, welcher im Argument `id` angegeben ist. Sie müssen den Mittelpunkt des Bogens in den Argumenten `xc` und `yc` angeben. Die Radien geben Sie in `ra` sowie `rb` an und die Start- und End-Winkel tragen Sie bei `start` und `end` ein. Alle Winkel sind in Grad angegeben. Wenn Sie eine geschlossene Ellipse haben wollen, muss das Startargument 0 und das Endargument 360 sein. In diesem Fall wäre aber die Verwendung des Befehls `AddEllipseToPath()` natürlich einfacher. Das optionale Argument `clockwise` kann verwendet werden, ob der elliptische Bogen im Uhrzeigersinn gezeichnet wird oder nicht. Dieser Tag ist standardmäßig auf `True` gesetzt, womit der elliptische Bogen im Uhrzeigersinn gezeichnet wird. Wenn Sie diesen Tag hingegen auf `False` setzen, wird `AddArcToPath()` die Winkel gegen den Uhrzeigersinn miteinander verbinden.

Beachten Sie, dass `AddArcToPath()` keinen Mittelpunktsteil hinzufügt. Wenn Sie die Start- und Endwinkel mit dem Mittelpunkt des Bogens verbinden möchten, müssen Sie dies manuell erledigen, indem Sie `MoveTo()` vor `AddArcToPath()` aufrufen und danach den Befehl `LineTo()` verwenden.

Beachten Sie auch, dass `AddArcToPath()` nur einen neuen Unterpfad beginnt und kein aktiver Unterpfad ist. Sonst wird es einfach seine Eckpunkte mit dem gerade aktiven Unterpfad verbinden. Wenn Sie dies nicht wünschen, müssen Sie manuell einen neuen Unterpfad öffnen, bevor Sie `AddArcToPath()` aufrufen. Darüber hinaus schließt `AddArcToPath()` auch nicht den aktiven Unterpfad, wenn er fertig ist.

#### EINGABEN

<code>id</code>	Identifikator des Pfades, wo Sie den elliptischen Bogen hinzufügen möchten
<code>xc</code>	Mittelpunkt x des Bogens
<code>yc</code>	Mittelpunkt y des Bogens
<code>ra</code>	Bogenradius auf der x-Achse
<code>rb</code>	Bogenradius auf der y-Achse
<code>start</code>	Startwinkel in Grad (muss positiv sein)
<code>end</code>	Endwinkel in Grad (muss auch positiv sein)
<code>clockwise</code>	optional: <code>True</code> ist im Uhrzeigersinn, <code>False</code> gegen den Uhrzeigersinn (voreingestellt ist <code>True</code> )

## 53.2 AddBoxToPath

### BEZEICHNUNG

AddBoxToPath – fügt ein Rechteck dem Pfad hinzu (V5.0)

### ÜBERSICHT

AddBoxToPath(id, x, y, width, height[, table])

### BESCHREIBUNG

Dieser Befehl fügt ein Rechteck in dem in `id` angegebenen Pfad ein. Sie müssen die Position und die Größe des Rechtecks in den anderen vier Argumenten übergeben. AddBoxToPath() startet einen neuen Unterpfad für das Rechteck und schließt es wieder.

Das optionale Argument `table` ermöglicht es Ihnen, den Stil des Rechtecks zu konfigurieren. Die folgenden Tags sind derzeit möglich:

#### RoundLevel:

Sie können diesen Tag angeben, um ein Rechteck mit abgerundeten Ecken zu erstellen. Mit einer Prozentzahl geben Sie an, wie stark die Abrundung der Ecken sein wird (mögliche Werte sind 0 bis 100). Der Standardwert ist 0, womit die Ecken nicht abgerundet werden.

#### CornerA, CornerB, CornerC, CornerD:

Mit diesen vier Tags können Sie die Feinabstimmung der Eckenabrundung des Rechtecks vornehmen. Sie können eine Rundungsstufe (0 bis 100) für jede Ecke des Rechtecks angeben, so dass Sie ein Rechteck erstellen, in dem nicht alle Ecken abgerundet sind, oder wo die verschiedenen Ecken unterschiedliche Rundungen haben. Diese Tags setzen alle Einstellungen des Tags RoundLevel außer Kraft.

### EINGABEN

<code>id</code>	Identifikator des Pfades, dem Sie das Rechteck hinzufügen möchten
<code>x</code>	x-Position des Rechtecks
<code>y</code>	y-Position des Rechtecks
<code>width</code>	Breite des Rechtecks
<code>height</code>	Höhe des Rechtecks
<code>table</code>	optional: Tabelle für weitere Optionen (siehe oben)

## 53.3 AddCircleToPath

### BEZEICHNUNG

AddCircleToPath – fügt einen Kreis dem Pfad hinzu (V5.0)

### ÜBERSICHT

AddCircleToPath(id, xc, yc, radius)



**BESCHREIBUNG**

Dieser Befehl fügt einen Kreis in dem in `id` angegebenen Pfad ein. Mit den Argumenten `xc` und `yc` übergeben Sie den Mittelpunkt des Kreises. Im Argument `radius` geben Sie den Kreisradius an.

Beachten Sie, dass `AddCircleToPath()` nur einen neuen Unterpfad beginnt und kein aktiver Unterpfad ist. Sonst wird es einfach seine Eckpunkte mit dem gerade aktiven Unterpfad verbinden. Wenn Sie dies nicht wünschen, müssen Sie manuell einen neuen Unterpfad öffnen, bevor Sie `AddCircleToPath()` aufrufen. Darüber hinaus schließt `AddCircleToPath()` auch nicht den aktiven Unterpfad, wenn er fertig ist.

**EINGABEN**

<code>id</code>	Identifikator des Pfades, dem Sie den Kreis hinzufügen möchten
<code>xc</code>	x-Koordinate des Kreismittelpunktes
<code>yc</code>	y-Koordinate des Kreismittelpunktes
<code>radius</code>	Kreisradius

## 53.4 AddEllipseToPath

**BEZEICHNUNG**

`AddEllipseToPath` – fügt eine Ellipse dem Pfad hinzu (V5.0)

**ÜBERSICHT**

`AddEllipseToPath(id, xc, yc, ra, rb)`

**BESCHREIBUNG**

Dieser Befehl fügt eine Ellipse in dem in `id` angegebenen Pfad ein. Mit den Argumenten `xc` und `yc` übergeben Sie den Mittelpunkt der Ellipse. In den Argumenten `ra` und `rb` geben Sie die Radien der Ellipse an.

Beachten Sie, dass `AddEllipseToPath()` nur einen neuen Unterpfad beginnt und kein aktiver Unterpfad ist. Sonst wird es einfach seine Eckpunkte mit dem gerade aktiven Unterpfad verbinden. Wenn Sie dies nicht wünschen, müssen Sie manuell einen neuen Unterpfad öffnen, bevor Sie `AddEllipseToPath()` aufrufen. Darüber hinaus schließt `AddEllipseToPath()` auch nicht den aktiven Unterpfad, wenn er fertig ist.

**EINGABEN**

<code>id</code>	Identifikator des Pfades, dem Sie die Ellipse hinzufügen
<code>xc</code>	x-Koordinate des Ellipsenmittelpunktes
<code>yc</code>	y-Koordinate des Ellipsenmittelpunktes
<code>ra</code>	Ellipsenradius auf der x-Achse
<code>rb</code>	Ellipsenradius auf der y-Achse

## 53.5 AddTextToPath

### BEZEICHNUNG

AddTextToPath – fügt Vektortext dem Pfad hinzu (V5.0)

### ÜBERSICHT

AddTextToPath(id, t\$, table))

### BESCHREIBUNG

Dieser Befehl fügt den Text in **t\$** als Vektorgrafik dem aktuellen Pfad hinzu. Aus diesem Grund wird dieser Befehl nur dann funktionieren, wenn ein Vektorfont (zum Beispiel eine TrueType-Schriftart) gerade aktiv ist. Mit Bitmap-Schriften kann dieser Befehl nicht arbeiten. **AddTextToPath()** wird die aktive Schriftart verwenden, die von dem letzten Aufruf von **SetFont()** oder **UseFont()** gesetzt wurde.

Bitte beachten Sie, dass **AddTextToPath()** den Text über den aktuellen y-Punkt hinzufügt. Wenn also der aktuelle y-Pfadpunkt 240 ist und Sie den Text, der 36 Pixel hoch ist hinzufügen, wird der Text an einer y-Position von 204 ( $240 - 36 = 204$ ) anstelle von 240 (wie der aktuelle y-Pfadpunkt) platziert werden.

Bitte beachten Sie, dass derzeit einige Einschränkungen gelten:

- Die Schriftart, welche **AddTextToPath()** verwendet, muss mit **#FONTENGINE\_INBUILT** geöffnet worden sein. Schriften, die hingegen mit **#FONTENGINE\_NATIVE** geöffnet wurden, funktionieren zur Zeit nicht. Die eingebauten Standardschriftarten **#SANS**, **#SERIF** und **#MONOSPACE** arbeiten gut mit **AddTextToPath()**.
- Alle Stile, welche von **SetFontStyle()** festgelegt wurden, werden von **AddTextToPath()** ignoriert. Wenn Sie kursiven oder fetten Text möchten, müssen Sie eine separate Schrift öffnen, die fett und/oder kursiv gedruckten Vektorgrafiken bereits in ihren Daten hat. Der einfache Aufruf von **SetFontStyle()** funktioniert nicht mit Vektortext.
- Ebenso werden alle Formatierungs-Tags, die von **Print()**, **TextOut()** und **CreateTextObject()** erkannt werden, von **AddTextToPath()** ignoriert. Es ist derzeit nicht möglich, Formatierungs-Tags mit diesem Befehl zu verwenden.
- NeueZeile-Zeichen ('\n') werden ebenfalls von diesem Befehl ignoriert.

Mit dem optionalen Argument **table** können Sie weitere Optionen festlegen:

#### Encoding:

Dieses Argument kann verwendet werden, um die Zeichencodierung innerhalb von **t\$** anzugeben. Voreingestellt ist die Standardcodierung für die Textbibliothek, welche mit dem Befehl **SetDefaultEncoding()** festgelegt ist. Siehe [Abschnitt 52.31 \[SetDefaultEncoding\]](#), [Seite 1162](#), für Details.

### EINGABEN

<b>id</b>	Identifikator des Pfades, dem Sie den Text hinzufügen möchten
<b>t\$</b>	Text, welcher dem Pfad hinzugefügt wird
<b>table</b>	optional: Tabelle für weitere Optionen (siehe oben)

### BEISPIEL

EnableLayers

```

SetFillStyle(#FILLCOLOR)
SetFormStyle(#ANTIALIAS)
SetFont("Arial", 100, {Engine = #FONTENGINE_INBUILT})
StartPath(1)
MoveTo(1, 0, 0)
AddTextToPath(1, "Hello World")
DrawPath(1, #CENTER, #CENTER + 100, #BLUE, {AnchorX = 0.5,
      AnchorY = 0.5, Rotate = 45})

```

Der obige Code erzeugt einen Vektorpfad, der den Text "Hello World" enthält. Der Pfad wird dann um 45 Grad gedreht.

## 53.6 AppendPath

### BEZEICHNUNG

AppendPath – hängt den Pfad an einen anderen Pfad an (V5.0)

### ÜBERSICHT

```
AppendPath(id, src)
```

### BESCHREIBUNG

Dieser Befehl hängt den in `src` angegebenen Pfad ans Ende des in `id` angegebenen Pfades. Alle Pfaddaten werden so kopiert, dass Sie den Pfad `src` nach dem Anhängenvorgang aus dem Arbeitsspeicher löschen können.

### EINGABEN

<code>id</code>	Identifikator des Pfades, der modifiziert werden soll
<code>src</code>	Identifikator des Pfades, der angehängt werden soll

## 53.7 ClearPath

### BEZEICHNUNG

ClearPath – entfernt alle Punkte und Unterpfade aus einem Pfad (V5.0)

### ÜBERSICHT

```
ClearPath(id)
```

### BESCHREIBUNG

Dieser Befehl wird den angegebenen Pfad komplett leeren. Alle Pfadpunkte und alle Unterpfade werden entfernt, jedoch der Pfad selbst wird nicht aus dem Arbeitsspeicher gelöscht. So können Sie das Hinzufügen von Punkten wieder nach diesem Befehl starten. Wenn Sie einen Pfad vollständig aus dem Arbeitsspeicher löschen möchten, verwenden Sie stattdessen den Befehl `FreePath()`.

### EINGABEN

<code>id</code>	Identifikator des Pfades, den Sie leeren möchten
-----------------	--

## 53.8 ClosePath

### BEZEICHNUNG

ClosePath – schließt den aktuellen Unterpfad (V5.0)

### ÜBERSICHT

ClosePath(id)

### BESCHREIBUNG

Dieser Befehl wird den aktuellen Unterpfad des in `id` angegebenen Pfades schließen. Er wird durch das Hinzufügen einer Linie zum Startpunkt des aktuellen Unterpfades geschlossen. Wenn Sie neue Eckpunkte nach dem Aufruf von `ClosePath()` hinzufügen möchten, müssen Sie mit den Befehlen `StartSubPath()` und `MoveTo()` einen neuen Unterpfad erstellen.

### EINGABEN

`id`            Identifikator des Pfades, der geschlossen werden soll

## 53.9 CopyPath

### BEZEICHNUNG

CopyPath – kopiert einen Pfad (V5.0)

### ÜBERSICHT

[id] = CopyPath(src, dst)

### BESCHREIBUNG

Dieser Befehl kopiert exakt den `src` angegeben Pfad und weist ihm die ID `dst` zu. Das Argument `dst` ist entweder eine ID, die für den neuen Pfad verwendet wird, oder kann `Nil` sein, wobei in diesem Fall `CopyPath()` automatisch eine freie ID auswählen wird und sie an Sie zurückgibt.

### EINGABEN

`src`            Identifikator des Pfades, der kopiert wird

`dst`            Identifikator des neuen Pfades oder `Nil` für die automatische Auswahl der ID

### RÜCKGABEWERTE

`id`            optional: Identifikator des neuen Pfades; wird nur zurückgegeben, wenn beim Argument `src` `Nil` angegeben wurde (siehe oben)

## 53.10 CurveTo

### BEZEICHNUNG

CurveTo – fügt eine Kurve dem Pfad hinzu (V5.0)

### ÜBERSICHT

CurveTo(id, x1, y1, x2, y2, x3, y3)

**BESCHREIBUNG**

Dieser Befehl wird eine kubische Bézier-Spline-Kurve dem Pfad hinzufügen. Die Kurve wird vom aktuellen Punkt zur in `x3/y3` angegebenen Position gezeichnet werden. Die Kontrollpunkte der Spline-Kurve geben Sie in den Argumenten `x1/y1` und `x2/y2` an. Wenn `CurveTo()` fertig ist, wird der aktuelle Punkt im Pfad `x3/Y3` angegeben.

**EINGABEN**

<code>id</code>	Identifikator des Pfades, dem Sie die Kurve hinzufügen
<code>x1</code>	x-Koordinate des Kontrollpunktes #1
<code>y1</code>	y-Koordinate des Kontrollpunktes #1
<code>x2</code>	x-Koordinate des Kontrollpunktes #2
<code>y2</code>	y-Koordinate des Kontrollpunktes #2
<code>x3</code>	x-Koordinate des Endpunktes der Kurve
<code>y3</code>	y-Koordinate des Endpunktes der Kurve

## 53.11 DrawPath

**BEZEICHNUNG**

`DrawPath` – zeichnet einen Vektorpfad (V5.0)

**ÜBERSICHT**

`DrawPath(id, x, y[, color], table)`

**BESCHREIBUNG**

Dies zeichnet den in `id` angegebenen Vektorpfad an die in `x` und `y` angegebene Position mit der in Argument `color` angegebenen Farbe. Der Vektorpfad wird mit dem Formstil gezeichnet, welcher mit dem Befehl `SetFormStyle()` festgelegt wurde. Außerdem wird er mit der in `SetFillStyle()` und `SetFillRule()` gewählten Konfiguration gefüllt. Wenn Sie den Vektorpfad im Umrissmodus zeichnen (d.h. als Füllstil `#FILLNONE` eingestellt ist), dann wird `DrawPath()` auch die Einstellungen von `SetLineJoin()`, `SetLineCap()` und `SetDash()` berücksichtigen. `color` kann entweder ein **RGB-Wert** oder ein **ARGB-Wert** sein.

Das optionale Argument `table` kann verwendet werden, um eine oder mehrere der **Standard-Tags zum Zeichnen** anzugeben. Siehe **Abschnitt 29.17** [`StandardDrawTags`], **Seite 613**, für weitere Informationen über die Standard-Tags, die fast alle Zeichnungsbeefehle von Hollywood unterstützen.

Wenn Ebenen aktiviert sind, wird dieser Befehl eine neue Ebene des Typs `#VECTORPATH` dem Ebenenstapel hinzufügen.

Beachten Sie, dass `DrawPath()` nur eine einzelne Farbe pro Pfad verwenden darf. Wenn Sie mehrfarbige Pfade verwenden möchten, können Sie mit dem Befehl `PathToBrush()` mehrere Pfade innerhalb eines Vektorpinselobjekts kombinieren. Siehe **Abschnitt 53.26** [`PathToBrush`], **Seite 1199**, für Details.

Beachten Sie außerdem, dass beim Zeichnen auf ein palettenbasiertes Ziel und wenn der Palettenmodus auf `#PALETTEMODE_PEN` eingestellt ist, dieser Befehl mit dem Stift über `SetDrawPen()` zeichnet anstelle der Schriftfarbe, welche mit dem Befehl `SetFontColor()` gesetzt wurde.

**EINGABEN**

<b>id</b>	Identifikator des Pfades, der gezeichnet werden soll
<b>x</b>	Zielkoordinate x
<b>y</b>	Zielkoordinate y
<b>color</b>	optional: <b>RGB-</b> oder <b>ARGB-Farbe</b> (voreingestellt ist <b>#BLACK</b> ) Farbe ist optional, da sie nicht erforderlich ist, wenn Sie in eine Maske oder einen Alphakanal zeichnen
<b>table</b>	optional: Tabelle für weitere Optionen; kann eine oder mehrere <b>Standard-Tags zum Zeichnen</b> beinhalten

**BEISPIEL**

```
SetFillStyle(#FILLNONE)
SetFormStyle(#ANTIALIAS)
x=25.6 y=128.0
x1=102.4 y1=230.4
x2=153.6 y2=25.6
x3=230.4 y3=128.0

StartPath(1)
MoveTo(1, x, y)
CurveTo(1, x1, y1, x2, y2, x3, y3)
SetLineWidth(10)
DrawPath(1, 0, 0, #BLACK)
```

```
ClearPath(1)
MoveTo(1, x, y)
LineTo(1, x1, y1)
MoveTo(1, x2, y2)
LineTo(1, x3, y3)
SetLineWidth(6)
DrawPath(1, 0, 0, ARGB(128, #RED))
```

Der obige Code zeichnet eine Kurve und zwei Linien, welche die Kontrollpunkte der Kurve darstellen.

```
EnableLayers
SetFillStyle(#FILLCOLOR)
SetFormStyle(#ANTIALIAS)
StartPath(1)
AddBoxToPath(1, 0, 0, 100, 100)
AddBoxToPath(1, 150, 0, 100, 100)
AddBoxToPath(1, 0, 150, 100, 100)
AddBoxToPath(1, 150, 150, 100, 100)
DrawPath(1, #CENTER, #CENTER, #RED, {Border = True, bordersize = 5})
```

Der obige Code zeichnet einen Vektorpfad, der ein wenig wie die Flagge der Schweiz aussieht.

## 53.12 ForcePathUse

### BEZEICHNUNG

ForcePathUse – verwendet immer pfadbasiertes Zeichnen (V5.0)

### ÜBERSICHT

ForcePathUse(enable)

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um alle Zeichnungsbefehle von Hollywoods Bibliothek für Grafikgrundelemente in den neuen vektorpfadbasierende Zeichnungsbibliothek umzuleiten. Dies ist besonders für runde Formen wie Kreise, Bögen und Ellipsen nützlich, da die Bibliothek für Grafikgrundelemente zeilenorientiert ist, was bedeutet, dass runde Formen nie perfekt rund aussehen werden, weil ihre runde Form durch Linien angenähert werden. Die Vektorpfadbasierende Zeichnungsbibliothek kann auf der anderen Seite perfekt runde Formen zeichnen, die besser aussehen als die der Bibliothek für Grafikgrundelemente.

Um das pfadbasierte Zeichnen für Grafikgrundelemente zu aktivieren, müssen Sie das Argument **enable** auf **True** setzen. Dann werden die folgenden Grafikgrundelemente gepatcht: **Arc()**, **Box()**, **Circle()**, **Ellipse()**, und **Polygon()**.

Denken Sie daran, dass wenn Sie pfadbasiertes Zeichnen für die Grafikgrundelemente ermöglichen, werden Sie immer Ebenen vom Typ **#VECTORPATH** bei aktivierten Ebenen erstellen. Die Bibliothek für Grafikgrundelemente würde auf der anderen Seite Ebenen vom Typ **#ARC**, **#BOX**, **#CIRCLE**, **#ELLIPSE** und **#POLYGON** hinzufügen. Normalerweise macht dies keinen großen Unterschied, aber es kann ein Problem werden, wenn Sie die Attribute einer Ebene versuchen mit **SetLayerStyle()** zu ändern, da Ebenen des Typs **#VECTORPATH** nicht die gleiche Funktionalität bieten wie die anderen Ebenentypen. Siehe [Abschnitt 25.48 \[SetLayerStyle\]](#), [Seite 449](#), für weitere Informationen über die unterstützten Attribute der verschiedenen Ebenentypen.

Bitte beachten Sie auch, dass vor der Hollywoodversion 6.0 Ihr Skript ein Plugin für Vektoren benötigt, wenn Sie hier **True** übergeben. Erst ab Hollywood 6.0 ist eine Vektorenbibliothek integriert.

Schließlich sollte erwähnt werden, dass vektorpfadbasierendes Zeichnen natürlich langsamer ist als das Zeichnen mit der Bibliothek für Grafikgrundelemente. Auf modernen Systemen ist jedoch kaum ein großer Unterschied auszumachen.

### EINGABEN

<b>enable</b>	<b>True</b> , um vektorpfadbasierendes Zeichnen zu aktivieren, <b>False</b> , um die Umleitung zu deaktivieren
---------------	--

## 53.13 FreePath

### BEZEICHNUNG

FreePath – löscht ein Pfadobjekt aus dem Arbeitsspeicher (V5.0)

**ÜBERSICHT**

`FreePath(id)`

**BESCHREIBUNG**

Dieser Befehl wird den in `id` angegebenen Pfad aus dem Arbeitsspeicher löschen. Nach dem Aufruf dieses Befehls wird der angegebene Pfad nicht mehr verfügbar sein. Wenn Sie nur alle Knoten und Unterpfade von einem Pfad entfernen möchten, sollten Sie stattdessen `ClearPath()` verwenden.

**EINGABEN**

`id` ID des aus dem Arbeitsspeicher zu löschenden Pfads

## 53.14 GetCurrentPoint

**BEZEICHNUNG**

`GetCurrentPoint` – ermittelt den aktuellen Punkt des Pfades (V5.0)

**ÜBERSICHT**

`x, y = GetCurrentPoint(id)`

**BESCHREIBUNG**

Dieser Befehl gibt den aktuellen Pfad zurück, der in `id` angegeben wurde.

**EINGABEN**

`id` ID des Pfades

**RÜCKGABEWERTE**

`x` x-Position des aktuellen Punktes

`y` y-Position des aktuellen Punktes

## 53.15 GetDash

**BEZEICHNUNG**

`GetDash` – gibt das aktuelle Strichmuster zurück (V7.1)

**ÜBERSICHT**

`offset[, t] = GetDash()`

**BESCHREIBUNG**

Dieser Befehl gibt das aktuelle Strichmuster zurück, welches mit dem Befehl `SetDash()` gesetzt wurde. Der erste Rückgabewert `offset` wird auf den Versatz gesetzt, bei dem das Strichmuster beginnt, und der zweite Rückgabewert `t` ist eine Tabelle, die die Längen der einzelnen sicht- und unsichtbaren Abschnitte enthält. Siehe [Abschnitt 53.30 \[SetDash\]](#), [Seite 1202](#), für Details.

Wenn momentan kein Strichmuster aktiv ist, wird -1 in `offset` zurückgegeben und es gibt keinen zweiten Rückgabewert.

**EINGABEN**

keine



**RÜCKGABEWERTE**

<code>offset</code>	Versatz, bei dem das Strichmuster beginnt oder -1 für kein Strichmuster
<code>t</code>	optional: Tabelle mit sicht-/unsichtbaren Abschnitten (nur wenn <code>offset</code> nicht -1 ist)

**53.16 GetFillRule****BEZEICHNUNG**

`GetFillRule` – gibt die momentane Füllregel für überlappende Pfade zurück (V7.1)

**ÜBERSICHT**

```
rule = GetFillRule()
```

**BESCHREIBUNG**

Dieser Befehl gibt die aktuelle Füllregel zurück, die mit `SetFillRule()` gesetzt wurde. Dies wird entweder `#FILLRULEWINDING` oder `#FILLRULEEVENODD` sein. Siehe [Abschnitt 53.31 \[SetFillRule\]](#), [Seite 1203](#), für Details.

**EINGABEN**

keine

**RÜCKGABEWERTE**

<code>rule</code>	aktuelle Füllregel
-------------------	--------------------

**53.17 GetLineCap****BEZEICHNUNG**

`GetLineCap` – gibt den aktuellen Stil der Linienenden zurück (V7.1)

**ÜBERSICHT**

```
style = GetLineCap()
```

**BESCHREIBUNG**

Dieser Befehl gibt den aktuellen Linienendenstil zurück, der mit `SetLineCap()` gesetzt wurde. Dies ist entweder `#CAPBUTT`, `#CAPROUND` oder `#CAPSQUARE`. Siehe [Abschnitt 53.32 \[SetLineCap\]](#), [Seite 1204](#), für Details.

**EINGABEN**

keine

**RÜCKGABEWERTE**

<code>style</code>	aktueller Linienendenstil
--------------------	---------------------------

## 53.18 GetLineJoin

### BEZEICHNUNG

GetLineJoin – gibt die Art beim Verbinden von Linien zurück (V7.1)

### ÜBERSICHT

```
style = GetLineJoin()
```

### BESCHREIBUNG

Dieser Befehl gibt die aktuelle Art beim Verbinden von Linien zurück, die mit dem Befehl `SetLineJoin()` gesetzt wurde. Dies ist entweder `#JOINMITER`, `#JOINROUND` oder `#JOINBEVEL`. Siehe [Abschnitt 53.33 \[SetLineJoin\]](#), [Seite 1204](#), für Details.

### EINGABEN

keine

### RÜCKGABEWERTE

`style`      aktuelle Art zum Verbinden von Linien

## 53.19 GetMiterLimit

### BEZEICHNUNG

GetMiterLimit – gibt die aktuelle Gehrungsgrenze zurück (V7.1)

### ÜBERSICHT

```
limit = GetMiterLimit()
```

### BESCHREIBUNG

Dieser Befehl gibt die aktuelle Gehrungsgrenze zurück, die mit `SetMiterLimit()` gesetzt wurde. Siehe [Abschnitt 53.34 \[SetMiterLimit\]](#), [Seite 1205](#), für Details.

### EINGABEN

keine

### RÜCKGABEWERTE

`limit`      aktuelle Gehrungsgrenze

## 53.20 GetPathExtents

### BEZEICHNUNG

GetPathExtents – berechnet die Ausmaße des Pfades (V5.0)

### ÜBERSICHT

```
x1, y1, x2, y2 = GetPathExtents(id)
```

### BESCHREIBUNG

Dieser Befehl berechnet die Ausmaße des in `id` angegebenen Pfades. Der aktuelle Füll- und Formstil wird berücksichtigt. Die Ausdehnungen werden als Begrenzungsrechteck zurückgegeben. Beachten Sie, dass alle Rückgabewerte absolute Punktpositionen angeben. Die Breite und Höhe des Pfades erhalten Sie, wenn Sie bei diesen Positionen `x1` von `x2` und `y1` von `y2` subtrahiert haben.

**EINGABEN**

`id` ID des Pfades

**RÜCKGABEWERTE**

`x1` x-Position Anfang

`y1` y-Position Anfang

`x2` x-Position Ende

`y2` y-Position Ende

## 53.21 IsPathEmpty

**BEZEICHNUNG**

`IsPathEmpty` – überprüft, ob ein Pfad leer ist (V5.0)

**ÜBERSICHT**

```
ret = IsPathEmpty(id)
```

**BESCHREIBUNG**

Mit diesem Befehl kann überprüft werden, ob ein Pfad leer ist oder nicht. Ist er leer, gibt dieser Befehl `True` zurück, andernfalls `False`.

**EINGABEN**

`id` Identifikator des Pfades

**RÜCKGABEWERTE**

`ret` `True`, wenn der Pfad leer ist, ansonsten `False`

## 53.22 LineTo

**BEZEICHNUNG**

`LineTo` – fügt eine Linie dem Pfad hinzu (V5.0)

**ÜBERSICHT**

```
LineTo(id, x, y)
```

**BESCHREIBUNG**

Dieser Befehl wird eine Linie von dem aktuellen Punkt des in `id` angegebenen Pfads zum Punkt `x/y` hinzuzufügen. Wenn `LineTo()` fertig ist, wird der aktuelle Punkt des Pfads `x/y` sein.

**EINGABEN**

`id` ID des Pfads, dem die Linie hinzugefügt wird

`x` x-Koordinate des Zielpunkts

`y` y-Koordinate des Zielpunkts

## 53.23 MoveTo

### BEZEICHNUNG

MoveTo – setzt den aktuellen Punkt und beginnt einen neuen Unterpfad (V5.0)

### ÜBERSICHT

MoveTo(id, x, y)

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen neuen Unterpfad an der angegebenen Stelle zu beginnen. Wenn MoveTo() fertig ist, wird der aktuelle Punkt des Pfades x/y sein. Für die meisten Fälle ist dies der bevorzugte Weg, um einen neuen Unterpfad zu beginnen. Der Befehl **StartSubPath()** ist nur für den seltenen Fall zu empfehlen, wenn Sie ohne einen aktuellen Punkt einen Unterpfad erstellen wollen.

### EINGABEN

id	Identifikator des Pfads
x	x-Koordinate des Zielpunkts
y	y-Koordinate des Zielpunkts

## 53.24 NormalizePath

### BEZEICHNUNG

NormalizePath – verschiebt den Pfad zum Ursprung (V5.0)

### ÜBERSICHT

NormalizePath(id)

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen Pfad zu normalisieren. Normalisieren bedeutet, dass alle Leerzeichen an den linken und oberen Seiten des Pfades abgeschnitten werden, so dass der Pfad in die Ursprungsposition in der oberen linken Ecke bewegt wird.

### EINGABEN

id	ID des Pfads
----	--------------

## 53.25 PathItems

### BEZEICHNUNG

PathItems – durchläuft einzelne Pfadelemente (V7.0)

### ÜBERSICHT

f = PathItems(id)

### BESCHREIBUNG

Dieser Befehl kann zusammen mit der generischen **For**-Anweisung verwendet werden, um alle Elemente in einem Pfad zu durchlaufen. Er gibt eine Iterator-Funktion zurück, die eine Tabelle zurückgibt, die Informationen über das nächste Element im Pfad enthält.

Sobald alle Pfadpositionen zurückgegeben wurden, gibt die Iterator-Funktion **Nil** zurück, um die generische **For**-Anweisung zu beenden.

Siehe [Abschnitt 11.4 \[Generische For-Anweisung\]](#), [Seite 135](#), für Details.

Die von **PathItems()** zurückgegebene Tabelle enthält ein Zeichenketten-Feld **Type**, der den Pfadtyp beschreibt. Alle weiteren Tabellenfelder hängen von dem in **Type** zurückgegebenen Pfadtyp ab. Folgende Typen werden derzeit unterstützt:

**NewSubPath:**

Dieses Element startet einen neuen Teilpfad. Der aktuelle Punkt ist zu diesem Zeitpunkt undefiniert. Es gibt keine zusätzlichen Argumente.

**ClosePath:**

Schließt den aktuellen Pfad, indem er eine Linie vom aktuellen Punkt zum ersten Punkt im Teilpfad zeichnet. Es gibt keine zusätzlichen Argumente.

**MoveTo:** Es wird ein neuer Teilpfad begonnen. Der aktuelle Punkt des Unterpfades wird auf die angegebene Position gesetzt. Die Tabelle **MoveTo** enthält drei weitere Argumente:

**Rel:** Dies ist ein boolescher Wert, der angibt, ob die Koordinaten relative oder absolute Werte sind. Wenn dies **True** ist, müssen die Koordinaten als relativ zum aktuellen Punkt interpretiert werden.

**X:** Die x-Koordinate der neuen Position.

**Y:** Die y-Koordinate der neuen Position.

**LineTo:** Dies zeichnet eine Linie vom aktuellen Punkt zur angegebenen Position. Darüber hinaus wird es den aktuellen Punkt auf den Endpunkt der Linie ändern, wenn sie fertig ist. Die Tabelle **LineTo** enthält die folgenden drei weiteren Argumente:

**Rel:** Dies ist ein boolescher Wert, der angibt, ob die Koordinaten relative oder absolute Werte sind. Wenn dies **True** ist, müssen die Koordinaten als relativ zum aktuellen Punkt interpretiert werden.

**X:** Die x-Koordinate der neuen Position.

**Y:** Die y-Koordinate der neuen Position.

Wenn es keinen aktuellen Punkt gibt, wird sich **LineTo** so verhalten, als wäre es **MoveTo**, d.h. es wird einfach den aktuellen Punkt auf den angegebenen Scheitelpunkt setzen.

**CurveTo:** Hiermit wird eine Bézier-Kurve gezeichnet, die vom aktuellen Punkt bis zu der Position läuft, die in den letzten zwei Koordinaten übergeben wurde. Die anderen vier Koordinaten sind die Kontrollpunkte für die Kurve. Außerdem ändert es den aktuellen Punkt zum Endpunkt der Kurve, wenn sie fertig ist. Die Tabelle **CurveTo** enthält die folgenden sieben weitere Argumente:

**Rel:** Dies ist ein boolescher Wert, der angibt, ob die Koordinaten relative oder absolute Werte sind. Wenn dies **True** ist, müssen

die Koordinaten als relativ zum aktuellen Punkt interpretiert werden.

- X1:** Die x-Koordinate des ersten Kontrollpunktes.
- Y1:** Die y-Koordinate des ersten Kontrollpunktes.
- X2:** Die x-Koordinate des zweiten Kontrollpunktes.
- Y2:** Die y-Koordinate des zweiten Kontrollpunktes.
- X3:** Die x-Koordinate des Endpunktes.
- Y3:** Die y-Koordinate des Endpunktes.

Wenn es keinen aktuellen Punkt gibt, wird **CurveTo** den ersten Kontrollpunkt (x1, y1) als aktuellen Punkt verwenden.

**Arc:** Damit wird ein elliptischer Bogen gezeichnet. **Arc** eröffnet einen neuen Teilpfad für den neuen Bogen nur für den Fall, dass es keinen momentan aktiven Teilpfad gibt. Wenn bereits ein Teilpfad vorhanden ist, wird **Arc** seinen Anfangspunkt mit dem aktuellen Punkt des Teilpfades verbinden. **Arc** schließt den Teilpfad nicht, wenn er seine Eckpunkte hinzugefügt hat. **Arc** verbindet nicht automatisch den Start- und Endwinkel des Bogens mit seinem Mittelpunkt. Dies muss ausdrücklich angegeben werden, indem wir vor und nach **Arc** getrennte **MoveTo**- und **LineTo**-Befehle benutzen. Die Tabelle **Arc** enthält die folgenden zusätzlichen Argumente:

- XC:** Den Mittelpunkt x des Bogens.
- YC:** Den Mittelpunkt y des Bogens.
- RA:** Bogenradius auf der x-Achse.
- RB:** Bogenradius auf der y-Achse.
- Start:** Startwinkel in Grad.
- End:** Endwinkel in Grad.

**Clockwise:**

Ob die Winkel im Uhrzeigersinn gezeichnet werden oder nicht.

Wenn **Arc** fertig ist, wird der aktuelle Punkt auf die Position des Endwinkels gesetzt.

**Box:** Dieser Befehl zeichnet ein Rechteck. **Box** öffnet zuerst einen neuen Unterpfad, dann werden die Eckpunkte des Rechtecks hinzugefügt und schließt den Teilpfad, wenn er fertig ist. Optional kann das Rechteck abgerundete Ecken haben. Die **Box**-Tabelle enthält die folgenden zusätzlichen Argumente:

- X:** X-Position des Rechtecks.
- Y:** Y-Position des Rechtecks.
- Width:** Breite des Rechtecks.
- Height:** Höhe des Rechtecks.

- Round:** Dies ist eine Tabelle, die vier Ganzzahlenwerte im Bereich von 0 bis 100 enthält, mit denen die Rundung der vier Ecken des Rechtecks in Grad angegeben werden. Ein Wert von 0 bedeutet keine Rundung, 100 hingegen Vollrundung.
- Text:** Dieser Befehl zeichnet einen Vektortext relativ zum aktuellen Punkt. Die einzelnen Zeichen werden als geschlossene Unterpfade hinzugefügt. Die Tabelle **Text** enthält die folgenden zusätzlichen Argumente:
- Size:** Gewünschte Schriftgröße.
- Text:** Die Zeichenkette mit dem Text.
- Wenn **Text** fertig ist, wird der aktuelle Punkt gesetzt, wo das nächste Zeichen angezeigt wird.

**EINGABEN**

**id** Identifikator des Pfades, welcher durchlaufen wird

**RÜCKGABEWERTE**

**f** Iterator-Funktion für die generische Schleife

**BEISPIEL**

```
For t In PathItems(1) Do DebugPrint(t.type)
```

Der obige Code durchläuft alle Elemente in Pfad 1 und gibt den Typ jedes Elements auf dem Debug-Gerät aus.

## 53.26 PathToBrush

**BEZEICHNUNG**

**PathToBrush** – wandelt Pfad(e) in Vektorpinsel um (V7.0)

**ÜBERSICHT**

```
[id] = PathToBrush(id, table)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um einen oder mehrere Pfade in einen Vektorpinsel umzuwandeln. **PathToBrush()** wird einen neuen Pinsel mit dem in **id** angegebenen Identifikator erstellen oder wenn Sie statt dessen in **id Nil** übergeben, wählt **PathToBrush()** automatisch einen freien Identifikator für den neuen Pinsel und gibt ihn zurück.

Das Umwandeln von Pfaden in Vektorpinseln hat den Vorteil, dass Sie den einzelnen Pfaden, die in einem einzigen Vektorpinsel kombiniert werden, verschiedene Farben zuordnen können, so dass Sie einfach mehrfarbige Pfade innerhalb eines einzigen Pinselobjekts verwalten können. Darüber hinaus können die in dem Vektorpinsel kombinierten Pfade auch unterschiedliche Zeichnungsstile verwenden.

Sie müssen eine Tabelle im Argument **table** übergeben, welches eine Anzahl von Untertabellen enthält. In dieser sind Informationen über die einzelnen Pfade angegeben, die in den Vektorpinsel eingebettet werden sollen. Die Pfade werden in genau der gleichen Reihenfolge in den Vektorpinsel gezeichnet, wie sie in dieser Tabelle erscheinen.

Beachten Sie, dass jeder Pfad, der in den Vektorpinsel eingebettet werden soll, zuerst normalisiert wird. So werden standardmäßig alle Pfade in der oberen linken Ecke des Vektorpinsels gezeichnet. Sie können dieses Verhalten ändern, indem Sie die Argumente **X** und **Y** in den einzelnen Untertabellen für jeden Pfad angeben, der dem Vektorpinsel hinzugefügt werden soll (siehe unten).

Folgende Untertabellenfelder können angegeben werden:

- ID:** Dies muss auf den Identifikator des Pfadobjekts gesetzt werden, das in den zu erstellenden Vektorpinsel eingebettet werden soll. Dies muss immer angegeben werden. Beachten Sie, dass `PathToBrush()` eine Kopie dieses Pfades erstellt, so dass nachfolgende Änderungen des Pfades den neuen Vektorpinsel in keiner Weise beeinflussen werden. Sie können diesen Pfad auch aus dem Speicher löschen, nachdem Sie ihn dem Vektorpinsel hinzugefügt haben.
- X:** Die X-Position, wo dieser Pfad in den Vektorpinsel gezeichnet werden soll. Diese Position muss relativ zur linken Ecke des Vektorpinsels sein. Beachten Sie, dass `PathToBrush()` intern den Pfad normalisiert, bevor er ihn dem Vektorpinsel hinzufügt, so dass Sie dieses Feld normalerweise verwenden müssen, um den Pfad korrekt innerhalb des Vektorpinsels zu positionieren. Voreinstellung auf 0.
- Y:** Die Y-Position, wo dieser Pfad in den Vektorpinsel gezeichnet werden soll. Diese Position muss relativ zur linken Ecke des Vektorpinsels sein. Beachten Sie, dass `PathToBrush()` intern den Pfad normalisiert, bevor er ihn dem Vektorpinsel hinzufügt, so dass Sie dieses Feld normalerweise verwenden müssen, um den Pfad korrekt innerhalb des Vektorpinsels zu positionieren. Voreinstellung auf 0.
- Color:** Der Pfad wird in dieser **ARGB-Farbe** gezeichnet. Diese Farbe kann auch eine Transparenzeinstellung enthalten. Voreingestellt ist **#BLACK**.

Beachten Sie, dass die Form- und Füllstile, die von den einzelnen Pfaden verwendet werden sollen, diejenigen sind, die zum Zeitpunkt der Erstellung des Pfades mit `StartPath()` aktiv waren. Dies unterscheidet sich von der Art und Weise, wie Form- und Füllstile beim Zeichnen von Pfaden mit `DrawPath()` arbeiten. `DrawPath()` verwendet die Form- und Füllstile, die aktiv sind, wenn `DrawPath()` aufgerufen wird. `PathToBrush()` wird die Form- und Füllstile benutzen, die aktiv waren, wenn `StartPath()` auf den einzelnen Pfaden aufgerufen wurde. Dies ermöglicht Ihnen, verschiedene Form- und Füllstile für die einzelnen Pfade zu verwenden, die in den Vektorpinsel eingebettet werden.

## EINGABEN

- id** Identifikator für den neuen Pinsel oder **Nil** für die **automatische ID-Zuweisung**
- table** Tabelle, die Pfade enthält, um sie in den Vektorpinsel einzubinden (siehe oben)

## RÜCKGABEWERTE

- id** optional: Identifikator des neuen Pinsels; wird nur zurückgegeben, wenn **Nil** als ID übergeben wurde (siehe oben)



**BEISPIEL**

```
PathToBrush(1, {{ID=1, Color=#RED}, {ID=2, Color=#BLUE, X=100}})
```

Dieser Code bettet die Pfade 1 und 2 innerhalb in einen neuen Vektorpinsel, der die ID 1 verwendet. Der Pfad 1 wird in Rot und der Pfad 2 wird blau gezeichnet. Zusätzlich wird der Pfad 2 an der X-Position 100 gezeichnet, während der Pfad 1 bei 0.0 gezeichnet wird.

## 53.27 RelCurveTo

**BEZEICHNUNG**

RelCurveTo – fügt eine relative Kurve dem Pfad hinzu (V5.0)

**ÜBERSICHT**

```
RelCurveTo(id, dx1, dy1, dx2, dy2, dx3, dy3)
```

**BESCHREIBUNG**

Dieser Befehl macht das gleiche wie `CurveTo()`, außer dass die Koordinaten Delta-Werte anstelle von absoluten Positionen sind. Die Delta-Koordinaten werden als relative Verschiebung des aktuellen Punktes im Pfad interpretiert.

**EINGABEN**

id	Identifikator des Pfades, dem Sie die Kurve hinzufügen
dx1	delta x-Koordinate des Kontrollpunktes #1
dy1	delta y-Koordinate des Kontrollpunktes #1
dx2	delta x-Koordinate des Kontrollpunktes #2
dy2	delta y-Koordinate des Kontrollpunktes #2
dx3	delta x-Koordinate des Zielpunktes der Kurve
dy3	delta y-Koordinate des Zielpunktes der Kurve

## 53.28 RelLineTo

**BEZEICHNUNG**

RelLineTo – fügt eine relative Linie dem Pfad hinzu (V5.0)

**ÜBERSICHT**

```
RelLineTo(id, dx, dy)
```

**BESCHREIBUNG**

Dieser Befehl macht das gleiche wie `LineTo()`, außer dass die Koordinaten Delta-Werte anstelle von absoluten Positionen sind. Die Delta-Koordinaten werden als relative Verschiebung des aktuellen Punktes im Pfad interpretiert.

**EINGABEN**

id	ID des Pfads, dem die Linie hinzugefügt wird
----	--

<code>dx</code>	delta x-Koordinate des Zielpunktes der Linie
<code>dy</code>	delta y-Koordinate des Zielpunktes der Linie

## 53.29 RelMoveTo

### BEZEICHNUNG

RelMoveTo – setzt relativen aktuellen Punkt und beginnt Unterpfad (V5.0)

### ÜBERSICHT

RelMoveTo(id, dx, dy)

### BESCHREIBUNG

Dieser Befehl macht das gleiche wie `MoveTo()`, außer dass die Koordinaten Delta-Werte anstelle von absoluten Positionen sind. Die Delta-Koordinaten werden als relative Verschiebung des aktuellen Punktes im Pfad interpretiert.

### EINGABEN

<code>id</code>	ID des Pfades
<code>dx</code>	delta x-Koordinate des Zielpunktes
<code>dy</code>	delta y-Koordinate des Zielpunktes

## 53.30 SetDash

### BEZEICHNUNG

SetDash – definiert das Strichmuster eines Pfades (V5.0)

### ÜBERSICHT

SetDash(offset, on1, off1, ...)

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um ein Strichmuster für Pfade zu definieren, die mit dem Befehl `DrawPath()` gezeichnet wurden. Ein Strichmuster besteht aus einer unbegrenzten Anzahl von Abschnitten. Ein Abschnitt besteht aus einem Linienstück und einer Lücke, die beim Argument `offset` beginnt. Im Argument `on1` übergeben Sie die Länge des Linienstücks, der sichtbar sein soll gefolgt von der Länge der Lücke, die unsichtbar sein wird. Dieses Muster wiederholen Sie so oft, wie Sie möchten. Wenn einen Umrisspfad zeichnen, wird `DrawPath()` dieses Strichmuster für alle Linien zeichnen. Wenn das Strichmuster nicht die gesamte Pfadlänge abdeckt, wird es immer und immer wiederholt werden.

Um das Strichmuster zu entfernen, rufen Sie diesen Befehl mit -1 im Argument `on1` auf. Bitte beachten Sie, dass das Strichmuster nur dann verwendet wird, wenn Sie Umrissvektoren zeichnen. Es kann nicht verwendet werden, wenn Vektorpfade gefüllt sind.

Beachten Sie auch, dass die interne Bibliothek für vektorbasiertes Zeichnen, welche mit Hollywood 6.0 eingeführt wurde, derzeit keine Strichmuster unterstützt. Verwenden Sie ein voll ausgestattetes Vektorenplugin, wenn Sie Strichmusterlinien benötigen. Siehe [Abschnitt 53.39 \[Bemerkungen zum Vektorgrafikplugin\]](#), [Seite 1208](#), für Details.

**EINGABEN**

<code>offset</code>	Versatz, wo das Strichmuster anfängt
<code>on1</code>	Länge des sichtbaren Liniensegments
<code>off1</code>	Länge der unsichtbaren Lücke
<code>...</code>	optional: weitere Definitionen von Abschnitten (Liniensegmente und Lücken), so viele Sie benötigen

**BEISPIEL**

```
SetFillStyle(#FILLNONE)
SetFormStyle(#ANTIALIAS)
SetLineWidth(10)
SetDash(0, 10, 10, 20, 20, 30, 30, 40, 40)
StartPath(1)
MoveTo(1, 0, 0)
LineTo(1, 640, 480)
DrawPath(1, 0, 0, #RED)
```

Der obige Code zeichnet eine Linie mit Strichmuster, die vier verschiedene Abschnitte hat: Der erste Abschnitt ist 20 Einheiten, der zweite 40 Einheiten, der dritte 60 Einheiten und der vierte 80 Einheiten (40 Liniensegmente plus 40 Lücken) lang.

**53.31 SetFillRule****BEZEICHNUNG**

`SetFillRule` – definiert die Füllregel für überlappende Pfade (V5.0)

**ÜBERSICHT**

```
SetFillRule(rule)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um zu definieren, wie `DrawPath()` die Pfade füllen soll, die sich überlappen. Derzeit werden die folgenden Füllregeln unterstützt:

**#FILLRULEWINDING:**

Füllt alle überlappenden Pfade nur, wenn sie nicht gewunden sind. Dies ist die Standardeinstellung.

**#FILLRULEEVENODD:**

Füllt überlappende Pfade, wenn die Gesamtzahl der Kreuzungen ungerade ist.

**EINGABEN**

<code>rule</code>	gewünschte Füllregel (siehe oben für mögliche Einstellungen)
-------------------	--

## 53.32 SetLineCap

### BEZEICHNUNG

SetLineCap – definiert den aktuellen Stil der Linienenden (V5.0)

### ÜBERSICHT

SetLineCap(style)

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um zu definieren, wie `DrawPath()` die Enden der Linien zeichnen soll, die nicht mit anderen verbunden sind. Derzeit werden die folgenden Arten von Enden unterstützt:

#### #CAPBUTT:

Hiermit werden keine speziellen Linienenden gezeichnet. Sobald die Linie endet, wird einfach mit dem Zeichnen aufgehört. Dies ist der Standardmodus.

#### #CAPROUND:

Zeichnet runde Linienenden

#### #CAPSQUARE:

Zeichnet rechteckige Linienenden

Bitte beachten Sie, dass der Stil der Linienenden nur dann verwendet wird, wenn Umrisslinien gezeichnet werden. Mit gefüllten Vektorpfaden kann dieser Befehl nicht verwendet werden.

### EINGABEN

style      gewünschter Stil der Linienenden (siehe oben für mögliche Einstellungen)

## 53.33 SetLineJoin

### BEZEICHNUNG

SetLineJoin – definiert die Art beim Verbinden von Linien (V5.0)

### ÜBERSICHT

SetLineJoin(style)

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um zu definieren, wie `DrawPath()` die Linien verbinden soll, wenn ein Vektorpfad gezeichnet wird. Derzeit werden die folgenden Verbindungsstile unterstützt:

#### #JOINMITER:

Verwendet Gehrungsverbindung (eine spitzwinklige Ecke). Dies ist der Standard-Modus. Die Gehrungsgrenze kann mit dem Befehl `SetMiterLimit()` eingestellt werden. Siehe [Abschnitt 53.34 \[SetMiterLimit\]](#), [Seite 1205](#), für Details.

#### #JOINROUND:

Verbindungsline mit ihren Enden als Kreise zeichnen. Dies ergibt den Eindruck von einem dicken Stift.

**#JOINBEVEL:**

Verbindet die Linien, indem in der Hälfte der Linienbreite die Enden schräg abgeschnitten werden.

Bitte beachten Sie, dass der Stil der Linienendenverbindung nur dann verwendet wird, wenn Umrisslinien gezeichnet werden. Mit gefüllten Vektorpfaden kann dieser Befehl nicht verwendet werden.

**EINGABEN**

`style`      gewünschte Art der Verbindung (siehe oben für die verschiedenen Möglichkeiten)

### 53.34 SetMiterLimit

**BEZEICHNUNG**

SetMiterLimit – setzt die Gehrungsgrenze (V7.1)

**ÜBERSICHT**

`SetMiterLimit(limit)`

**BESCHREIBUNG**

Dieser Befehl setzt die Gehrungsgrenze auf den durch `limit` vorgegebenen Wert. Dies kann ein gebrochener Wert sein. Die Gehrungsgrenze wird verwendet, wenn der Verbindungsstil auf `#JOINMITER` gesetzt ist, um festzulegen, wann Linien mit einer Fase und wann sie mit einer Gehrung verbunden werden sollen. Beachten Sie, dass `#JOINMITER` der Standard-Stil beim Verbinden von Linien ist.

Beim Zeichnen von Linienenden wird die Länge der Gehrung durch die Linienbreite dividiert und wenn das Ergebnis dieser Teilung größer ist als die mit diesem Befehl eingestellte Gehrungsgrenze, werden Linien mit einer Fase verbunden.

Hollywoods voreingestellte Gehrungsgrenze ist 10.

**EINGABEN**

`limit`      gewünschte Gehrungsgrenze

### 53.35 SetVectorEngine

**BEZEICHNUNG**

SetVectorEngine – wählt das Vektorgrafik-Plugin (V6.0)

**ÜBERSICHT**

`SetVectorEngine(engine$)`

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um das Plugin zu wählen, welches für das Zeichnen von vektorbasierten Formen verwendet werden soll. Sie müssen einfach nur den Namen des Plugins diesem Befehl übergeben. Um die in Hollywood integrierte Vektorenbibliothek zu verwenden, übergeben Sie in `engine$ default`.

Beachten Sie, dass es durchaus erlaubt ist, interne und externe Vektorenbibliotheken in einem einzigen Skript zu verwenden. Es ist sogar möglich, mehrere Vektorenbibliotheken in einem einzigen BGPic zu verwenden. Zum Beispiel könnte die erste Ebene die interne Vektorenbibliothek zum Zeichnen verwenden, während die zweite Ebene ein Vektoren-Plugin benutzt. Dies wird alles unterstützt. Sie können sogar die Vektorenbibliotheken von einzelnen Ebenen mit dem Befehl `SetLayerStyle()` ändern, indem Sie den Tag `VectorEngine` einstellen.

#### EINGABEN

`engine$`    Name des Plugin, welches Sie zum Zeichnen von Vektorgrafiken verwenden möchten

## 53.36 StartPath

#### BEZEICHNUNG

`StartPath` – beginnt einen neuen Pfad (V5.0)

#### ÜBERSICHT

`[id] = StartPath(id)`

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um ein neues Pfadobjekt zu erstellen, das unter dem Identifikator `id` zur Verfügung gestellt wird. Alternativ können Sie `Nil` als `id` übergeben. In diesem Fall wird `StartPath()` automatisch eine freie ID auswählen und sie zurückgeben.

Sobald der neue Pfad erstellt wird, sollten Sie zunächst einen aktuellen Punkt für den Pfad definieren, indem Sie den Befehl `MoveTo()` aufrufen. Danach können Sie mit dem Hinzufügen von Unterpfaden starten.

Ein Vektorpfad ist ein Wurzelobjekt für eine unendliche Anzahl von Unterpfaden. Sie können es als eine Sammlung von einer unendlichen Anzahl von einzelnen Elementen eines Vektorpfads vorstellen. Jeder Unterpfad innerhalb eines Vektorpfads kann als separates Element angesehen werden. Beachten Sie aber, dass wenn Sie einen Vektorpfad mit `DrawPath()` zeichnen, Sie nur eine globale Farbe angeben können, die von allen Unterpfaden innerhalb des Vektorpfads verwendet wird. Somit ist es nicht möglich, verschiedene Farben innerhalb eines einzigen Pfadobjekts zu verwenden. Wenn Sie eine andere Farbe verwenden möchten, müssen Sie zunächst ein neues Pfadobjekt erstellen.

#### EINGABEN

`id`            Identifikator für den neuen Pfad oder `Nil` für die automatische ID-Auswahl

#### RÜCKGABEWERTE

`id`            optional: ID des neuen Pfades; dieser wird nur zurückgegeben, wenn Sie `Nil` in `id` übergeben haben (siehe oben)

#### BEISPIEL

Siehe [Abschnitt 53.11 \[DrawPath\]](#), Seite 1189.

## 53.37 StartSubPath

### BEZEICHNUNG

StartSubPath – beginnt einen neuen Unterpfad (V5.0)

### ÜBERSICHT

StartSubPath(id)

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen neuen Unterpfad innerhalb des in `id` angegebenen Pfadobjekts zu beginnen. Der neue Unterpfad wird keinen aktuellen Punkt erhalten, so dass Sie die meiste Zeit besser den Befehl `MoveTo()` aufrufen, um einen neuen Unterpfad zu starten. `StartSubPath()` wird nur für die seltenen Fälle bevorzugt, in denen ein aktueller Punkt nicht erwünscht ist; zum Beispiel, es kann beim Hinzufügen von einem/einer Kreis/Ellipse/Bogen auf einem Pfad mit aktuellem Punkt ärgerlich sein, weil sich dann der/die Kreis/Ellipse/Bogen mit dem Pfad verbinden würde. In den meisten Fällen sollten Sie jedoch `MoveTo()` anstelle von `StartSubPath()` verwenden.

### EINGABEN

`id`            ID des Pfades

## 53.38 TranslatePath

### BEZEICHNUNG

TranslatePath – verschiebt einen Pfad mit Deltawerten (V5.0)

### ÜBERSICHT

TranslatePath(id, dx, dy)

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen Pfad mit Deltawerten zu verschieben. Übersetzen bedeutet, dass jeder Punkt in dem Pfad durch den angegebenen Deltawert verschoben wird. Ein positiver Deltawert verschiebt nach rechts (oder unten) und ein negativer Deltawert verschiebt nach links (oder oben). Bei einem Deltawert von 0 geschieht nichts.

### EINGABEN

`id`            ID des Pfades, der verschoben wird

`dx`           Deltawert für die x-Achse

`dy`           Deltawert für die y-Achse

### BEISPIEL

TranslatePath(1, -100, 150)

Der obige Code verschiebt den Pfad 1 um 100 Pixel nach links und 150 Pixel nach unten.

### 53.39 Vektorgrafikplugin

Bevor Hollywood 6.0 alle Befehle von Hollywoods Vektorenbibliothek verwendet, muss ein separates Plugin, das vektorbasiertes Zeichnen ermöglicht, geladen werden. Ab Hollywood 6.0 gibt es grundlegende Unterstützung für vektorbasiertes Zeichnen auch ohne externes Vektorenplugin. Allerdings unterstützt die interne Bibliothek für vektorbasiertes Zeichnen nicht alle Befehle der Pluginbibliothek und vielleicht sehen die Formen auch nicht so gut aus wie von einem spezialisierten Plugin gezeichnet. Dennoch sollte es für die meisten Zwecke ausreichen.

Für die besten Ergebnisse und die Kompatibilität sollten Sie ein externes Vektorenplugin installieren, die eine dedizierte Vektorenwiedergabe verwendet. Zum Beispiel, indem Sie entweder plattformunabhängige Engine wie Cairo oder OS vektorbasierte Zeichnungstechnologien wie Apples Quartz 2D oder Microsoft GDI+ verwenden.

Das Plugin muss in demselben Verzeichnis wie das ausführbare Hollywoodprogramm sein. Falls Sie eine ausführbare Datei von Hollywood zusammengestellt haben und es verteilen möchten, müssen Sie das Plugin in das gleiche Verzeichnis wie das kompilierte Programm setzen. Auf AmigaOS und kompatible Geräte, können Sie auch das Plugin ins Verzeichnis LIBS:Hollywood setzen. Wenn das "Hollywood" Verzeichnis in LIBS: noch nicht vorhanden ist, erstellen Sie bitte selber. Unter macOS, müssen Sie das Plugin innerhalb des Anwendungspakets setzen, das heißt innerhalb des `HollywoodInterpreter.app/Contents/MacOS` Verzeichnis. Beachten Sie, dass `HollywoodInterpreter.app` innerhalb des Anwendungspakets `Hollywood.app` gespeichert ist, nämlich in `Hollywood.app/Contents/Resources`.

Nachdem das Plugin installiert wurde, verwenden Sie den Befehl `SetVectorEngine()`, um es zu aktivieren. Siehe [Abschnitt 53.35 \[SetVectorEngine\]](#), [Seite 1205](#), für Details.



## 54 Veraltete Befehle

### 54.1 ACTIVEWINDOW

#### BEZEICHNUNG

ACTIVEWINDOW – window got active / VERALTET

#### ÜBERSICHT

Label(ACTIVEWINDOW)

#### BESCHREIBUNG

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie nun den Befehl `InstallEventHandler()`.

Hollywood will `Gosub()` to this label whenever the window becomes active again. You normally do not need this event but if you want to do something when your window becomes active again, use this event.

#### EINGABEN

Keine

### 54.2 BreakWhileMouseOn

#### BEZEICHNUNG

BreakWhileMouseOn – break next WhileMouseOn() command (V1.9) / VERALTET

#### ÜBERSICHT

BreakWhileMouseOn()

#### BESCHREIBUNG

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

This function breaks the next `WhileMouseOn()` command that will be executed. You usually use this function if your ONBUTTONCLICK label shall return immediately to your event loop instead of the `WhileMouseOn()` command in your ONBUTTONOVER label.

Normally, if you have an ONBUTTONCLICK and an ONBUTTONOVER label, Hollywood will return to your ONBUTTONOVER label after the mouse button was released because the mouse is still over your button after the user clicked the mouse. If you want Hollywood to return to your main event loop after a mouse click, use this command.

Note: This command is only required in rare cases.

#### EINGABEN

Keine

#### BEISPIEL

```
While(quit = FALSE)
    WaitEvent
Wend
```

```

Label(ONBUTTONOVER1)
Print("Mouse over button 1")
WhileMouseOn
Print("Mouse no longer over button 1")
Return

```

```

Label(ONBUTTONCLICK1)
Print("Mouse click on button 1")
WhileMouseDown
Print("Mouse button released")
BreakWhileMouseOn
Return

```

Have a look at the above code. The `BreakWhileMouseOn()` causes Hollywood to return immediately to the `WaitEvent()` loop. If there was no `BreakWhileMouseOn()` in the code above, Hollywood would return to the `WhileMouseOn()` command and wait until the user moves the mouse out of the button area.

## 54.3 ClearEvents

### BEZEICHNUNG

ClearEvents – clear user events (V1.5) / VERALTET

### ÜBERSICHT

ClearEvents()

### BESCHREIBUNG

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

This function clears all events.

### EINGABEN

none

### BEISPIEL

```

CreateButton(1,10,10,110,110)
CreateButton(2,130,130,230,230)
CreateKeyDown(1,"ESC")
ClearEvents
End

```

```

Label(SIZEWINDOW)
Return

```

```

Label(ONJOYFIRE1)
Return

```

```

Label(CLOSEWINDOW)

```

**Return**

The above code defines some events and calls `ClearEvents()`. This function will now clear the following events: Button 1, Button 2 and Keydown 1. The events `SIZEWINDOW`, `ONJOYFIRE1` and `CLOSEWINDOW` will not be cleared because they are only declared as labels. If you want to get rid of them, you can disable them.

## 54.4 CLOSEWINDOW

**BEZEICHNUNG**

`CLOSEWINDOW` – user clicked the window’s close box / VERALTET

**ÜBERSICHT**

`Label(CLOSEWINDOW)`

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie nun den Befehl `InstallEventHandler()`.

Hollywood will `Gosub()` to this label when the user clicks the close box of the window. This is useful if you want to pop up a requester and ask if he really wants to quit, for example.

**EINGABEN**

none

## 54.5 CreateButton

**BEZEICHNUNG**

`CreateButton` – create a new button / VERALTET

**ÜBERSICHT**

`CreateButton(id,x1,y1,x2,y2)`

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie nun den Befehl `MakeButton()`.

Use this command to declare a button on your display. The button is defined by the coordinates `x1:y1` to `x2:y2`. If the user moves the mouse over the button, Hollywood will `Gosub()` to the label with the name `ONBUTTONOVER` and the number you specified for the button. If the user clicks the button, Hollywood will jump to the label with the name `ONBUTTONCLICK` and the number of your button.

**EINGABEN**

<code>id</code>	desired identifier for the button
<code>x1</code>	source left edge
<code>y1</code>	source top edge
<code>x2</code>	destination left edge

y2            destination top edge

### BEISPIEL

```
CreateButton(1,0,0,200,200)
CreateButton(2,201,0,400,200)
CreateKeyDown(1,"ESC")
```

```
While(quit=FALSE)
    WaitEvent
Wend
End
```

```
Label(ONBUTTONOVER1)
Print("Mouse over button 1")
WhileMouseOn
Print("Mouse out of button 1")
Return
```

```
Label(ONBUTTONCLICK1)
Print("User clicked button 1")
WhileMouseDown
Print("User released left mouse button")
Return
```

```
Label(ONBUTTONRIGHTCLICK1)        ; requires Hollywood 1.5
Print("User right-clicked button 1")
WhileRightMouseDown
Print("User released right mouse button")
Return
```

```
Label(ONBUTTONOVER2)
Print("Mouse over button 2")
WhileMouseOn
Print("Mouse out of button 2")
Return
```

```
Label(ONBUTTONCLICK2)
Print("User clicked button 2")
WhileMouseDown
Print("User released left mouse button")
Return
```

```
Label(ONBUTTONRIGHTCLICK2)
Print("User right-clicked button 2")        ; requires Hollywood 1.5
WhileRightMouseDown
Print("User released right mouse button")
Return
```

```
Label(ONKEYDOWN1)
quit=TRUE
Return
```

The above code creates two buttons on the screen and monitors the user activity. If he presses the escape key, this demo will quit. This example shows a good way of handling the user input: It is advised that you use a loop like

```
While(quit=False)
  WaitEvent
Wend
```

to handle the user's input. However you have to make sure that you always return to the `WaitEvent()` in the loop.

## 54.6 CreateKeyDown

### BEZEICHNUNG

CreateKeyDown – create a new keydown object / VERALTET

### ÜBERSICHT

```
CreateKeyDown(id,key$)
```

### BESCHREIBUNG

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

Use this command to declare a key to monitor. If the user presses this key, Hollywood will `Gosub()` to the label called ONKEYDOWN with the number as specified in this command.

Key\$ is a string representing a key on your keyboard. This can be one of the following control keys:

UP	cursor up
DOWN	cursor down
RIGHT	cursor right
LEFT	cursor left
HELP	help key
DEL	delete key
BACKSPACE	backspace key
TAB	tab key
RETURN	return/enter key
ESC	escape
SPACE	space key

F1 – F10    function keys

The other keys can be accessed by just specifying the character of the key in the string, e.g. "A", "!" or "-".

The following keys cannot be monitored: Alt keys, command keys and the control key.

#### EINGABEN

id            desired identifier for the button  
key\$        key to monitor

#### BEISPIEL

```
CreateKeydown(1,"ESC")
While(quit=FALSE)
    WaitEvent
Wend
End
```

```
Label(ONKEYDOWN1)
quit=TRUE
Return
```

The above code waits for the user to press the escape key. Then it quits. A code structure like above is recommended for your applications.

## 54.7 DisableEvent

#### BEZEICHNUNG

DisableEvent – disable an event / VERALTET

#### ÜBERSICHT

DisableEvent(type,id)

#### BESCHREIBUNG

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie nun den Befehl **DisableButton()**.

This function disables the event specified by type and id. Once an event is disabled, it will not be monitored any longer. However, you can enable it again later by using the **EnableEvent()**.

Do not forget to specify the '#' prefix for all events because you are passing constants!

#### EINGABEN

type        event type (e.g. #ONBUTTONOVER, #ONBUTTONCLICK, #CLOSEWINDOW)  
id        event number to disable

#### BEISPIEL

```
DisableEvent(#ONBUTTONOVER,1)
DisableEvent(#ONBUTTONCLICK,1)
DisableEvent(#ONBUTTONRIGHTCLICK,1)
```

The above code completely disables the monitoring of button 1 (every button has three events `#ONBUTTONOVER`, `#ONBUTTONCLICK` and `#ONBUTTONRIGHTCLICK`). Therefore if you want to disable a button completely you will have to call `DisableEvent()` thrice.

## 54.8 DisableEventHandler

### BEZEICHNUNG

`DisableEventHandler` – disable Hollywood’s event handler / VERALTET

### ÜBERSICHT

`DisableEventHandler()`

### BESCHREIBUNG

Achtung: Dieser Befehl wurde nur bis Hollywood Version 1.9 unterstützt. Verwenden Sie nun stattdessen den Befehl `CheckEvent()`.

This function disables the internal event handler of Hollywood. Siehe [Abschnitt 54.9 \[EnableEventHandler\]](#), [Seite 1215](#), for more information about the internal event handler.

### EINGABEN

none

### BEISPIEL

Siehe [Abschnitt 54.9 \[EnableEventHandler\]](#), [Seite 1215](#).

## 54.9 EnableEventHandler

### BEZEICHNUNG

`EnableEventHandler` – enable Hollywood’s event handler / VERALTET

### ÜBERSICHT

`EnableEventHandler()`

### BESCHREIBUNG

Achtung: Dieser Befehl wurde nur bis Hollywood Version 1.9 unterstützt. Verwenden Sie nun stattdessen den Befehl `CheckEvent()`.

This function enables the internal event handler of Hollywood. This means, that you do not have to call `WaitEvent()` any longer because Hollywood will always check if there are any events that occurred. Using the internal event handler is useful if you want to call some functions when there is no input but also monitor user input, e.g. if you are doing a slide show with some effects you cannot call `WaitEvent()` every second but you still want that the user can quit the show by pressing some button on your screen. Then it would be wise to call `EnableEventHandler()`. Once it is enabled, you can do what you want but all events are still monitored.

Please note: Use this function only when you really need it. It has major disadvantages compared to an input loop together with `WaitEvent()` because you will never know when an event was raised and from where. If Hollywood’s event handler is enabled, events can be raised always. It could even happen, that an event breaks commands that are still busy,

e.g. `DisplayTransitionFX()`. It is not a good idea to use `EnableEventHandler()` in your projects because you will lose the control of your application. `EnableEventHandler()` is also very likely to be removed from Hollywood in future versions. So you should stay on the safe side, which means: Use an input loop with `WaitEvent()`.

#### EINGABEN

none

#### BEISPIEL

```
EnableEventHandler
DisplayBGPic(1)
Wait(200)
DisplayBGPic(2)
...
Label(ONBUTTONCLICK1)
End
```

The above code enables the event handler and then starts a slide show but the user will always be able to press a button although you do not call `WaitEvent()`.

## 54.10 EnableEvent

#### BEZEICHNUNG

`EnableEvent` – enable an event / VERALTET

#### ÜBERSICHT

```
EnableEvent(type,id)
```

#### BESCHREIBUNG

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie nun den Befehl `EnableButton()`.

This function enables the event specified by type and id. Once an event is enabled, it will be monitored by Hollywood. You will only have to call this function if you disabled the event before because by default, all events are enabled.

Do not forget to specify the '#' prefix for all events because you are passing constants!

#### EINGABEN

type	event type (e.g. <code>#ONBUTTONOVER</code> , <code>#ONBUTTONCLICK</code> , <code>#CLOSEWINDOW</code> )
id	event number to enable

#### BEISPIEL

```
EnableEvent(#ONBUTTONOVER,1)
EnableEvent(#ONBUTTONCLICK,1)
EnableEvent(#ONBUTTONRIGHTCLICK,1)
```

The above code completely enables the monitoring of button 1 (every button has three events `#ONBUTTONOVER`, `#ONBUTTONCLICK` and `#ONBUTTONRIGHTCLICK`). Therefore if you want to enable a button completely you will have to call `EnableEvent()` thrice.



## 54.11 GetEventCode

### BEZEICHNUNG

GetEventCode – get event specific code (V1.5) / VERALTET

### ÜBERSICHT

```
code = GetEventCode()
```

### BESCHREIBUNG

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

This function is used right after Hollywood jumped to an event label. If an event has to tell you additional information, you can get this information through this function.

For example, **ONBUTTONCLICKALL** returns the identifier of the button that caused Hollywood to jump to the label.

### EINGABEN

none

### RÜCKGABEWERTE

code            event specific code

### BEISPIEL

Siehe [Abschnitt 54.20 \[ONBUTTONCLICKALL\]](#), Seite 1221.

## 54.12 Gosub

### BEZEICHNUNG

Gosub – call a subroutine / VERALTET

### ÜBERSICHT

```
Gosub(label)
```

### BESCHREIBUNG

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie nun **Funktionen** anstelle von Labels.

This function jumps to the subroutine specified by label. The subroutine can call **Return()** to return to the point from where it was called.

### EINGABEN

label            identifier for a label (defined with **Label()**)

### BEISPIEL

```
a$="Hello World"
Gosub(PRINTTEXT)
WaitLeftMouse
End
```

```
Label(PRINTTEXT)
Print(a$)
```

**Return**

The above code prints the text "Hello World" on the screen and waits for the left mouse. Then quits.

## 54.13 Goto

**BEZEICHNUNG**

Goto – jump to a new location / VERALTET

**ÜBERSICHT**

Goto(label)

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie nun **Funktionen** anstelle von Labels.

This function jumps to the location specified by label. Execution will continue there and it is not possible to get back to the point from where the label was called. If you need this, use the **Gosub()** command.

**EINGABEN**

label        identifier for a label (defined with **Label()**)

**BEISPIEL**

```
a$="Hello World"
Goto(PRINTTEXT)
WaitLeftMouse      ; this code will never be reached
End                 ; !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```

```
Label(PRINTTEXT)
Print(a$)
WaitLeftMouse
End
```

The above code prints the text "Hello World" on the screen and waits for the left mouse. Then quits.

## 54.14 INACTIVEWINDOW

**BEZEICHNUNG**

INACTIVEWINDOW – window got inactive / VERALTET

**ÜBERSICHT**

Label(INACTIVEWINDOW)

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie nun den Befehl **InstallEventHandler()**.

Hollywood will **Gosub()** to this label whenever the window becomes inactive. You normally do not need this event but if you want to do something when your window becomes inactive, use this event.

**EINGABEN**

none

## 54.15 Label

**BEZEICHNUNG**

Label – declare a new label / VERALTET

**ÜBERSICHT**

Label(name)

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie nun **Funktionen** anstelle von Labels.

This function declares a new label with the specified name. You can jump to this label with the **Gosub()** and **Goto()** commands then. Please note that name is not a string! You need to specify a variable name that will be used as the reference for this label.

**EINGABEN**

name            identifier to use

**BEISPIEL**

Siehe [Abschnitt 54.12 \[Gosub\]](#), Seite 1217.

Siehe [Abschnitt 54.13 \[Goto\]](#), Seite 1218.

## 54.16 ModifyButton

**BEZEICHNUNG**

ModifyButton – modify button data / VERALTET

**ÜBERSICHT**

ModifyButton(id,x1,y1,x2,y2)

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

This function changes the configuration of the button specified by **id**.

**EINGABEN**

id            identifier of the button  
x1            desired new left edge position  
y1            desired new top edge position  
x2            desired new destination left edge position

y2                      desired new destination top edge position

## 54.17 ModifyKeyDown

### BEZEICHNUNG

ModifyKeyDown – modify keydown object data / VERALTET

### ÜBERSICHT

ModifyKeyDown(id,key\$)

### BESCHREIBUNG

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

This function changes the configuration of the keydown specified by id.

Siehe [Abschnitt 54.6 \[CreateKeyDown\]](#), [Seite 1213](#), for the keys you can specify in key\$.

### EINGABEN

id                      identifier of the button  
key\$                    desired new key to monitor

## 54.18 MOVEWINDOW

### BEZEICHNUNG

MOVEWINDOW – user moved the window / VERALTET

### ÜBERSICHT

Label(MOVEWINDOW)

### BESCHREIBUNG

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie nun den Befehl [InstallEventHandler\(\)](#).

Hollywood will [Gosub\(\)](#) to this label whenever the window is moved. This is useful to know when your window is transparent. Those windows will be closed and reopened when they are moved, therefore you may need to redraw somethings after the window was moved.

### EINGABEN

none

## 54.19 ONBUTTONCLICK

### BEZEICHNUNG

ONBUTTONCLICK – user clicked a button / VERALTET

### ÜBERSICHT

Label(ONBUTTONCLICKx)

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

This event is raised when the user clicks on a declared button. Hollywood will **Gosub()** then to the label with the name **ONBUTTONCLICKx** where x is the identifier of the button. For example, Hollywood will **Gosub()** to the label called **ONBUTTONCLICK15** when the user clicks on the button with the number 15.

**EINGABEN**

x                    button number

**BEISPIEL**

Siehe [Abschnitt 54.5 \[CreateButton\]](#), Seite 1211.

**54.20 ONBUTTONCLICKALL****BEZEICHNUNG**

**ONBUTTONCLICKALL** – user clicked any button (V1.5) / VERALTET

**ÜBERSICHT**

Label(**ONBUTTONCLICKALL**)

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

If you specify this event and it is enabled, Hollywood will always jump to this label when it gets an **ONBUTTONCLICK** event. If you want to know, which button caused the label jump, just call **GetEventCode()**. It will return which button caused the event.

This event is very useful if you have e.g. 100 or more buttons and every button just does a **Gosub()** to a sub-routine which receives an id which button was pressed. Instead of defining 100 labels now, you can just use **ONBUTTONCLICKALL** together with **GetEventCode()**.

For example, if you have 3 buttons with id's from 1 to 3, Hollywood would normally jump to **ONBUTTONCLICK1**, **ONBUTTONCLICK2** or **ONBUTTONCLICK3** if one of those buttons was clicked. But if you define an **ONBUTTONCLICKALL** event, Hollywood will always jump to this label.

**EINGABEN**

none

**BEISPIEL**

```
CreateButton(1,10,10,100,100)
CreateButton(2,130,130,230,230)
CreateButton(3,260,260,360,360)
CreateKeyDown(1,"F1")
CreateKeyDown(2,"F2")
CreateKeyDown(3,"F3")
While(quit = FALSE)
```

```

    WaitEvent
Wend

Label(ONBUTTONCLICKALL)
WhileMouseDown
id = GetEventCode()
Print("User left-clicked button # ")
Print(id)
Return

Label(ONBUTTONRIGHTCLICKALL)
WhileRightMouseDown
id = GetEventCode()
Print("User right-clicked button # ")
Print(id)
Return

Label(ONBUTTONOVERALL)
WhileMouseOn
id = GetEventCode()
Print("User moved mouse over button # ")
Print(id)
Return

Label(ONKEYDOWNALL)
WhileKeyDown
id = GetEventCode()
Print("User invoked keydown # ")
Print(id)
Return

```

The above code uses the ALL special events together with GetEventCode() to find out button events that occurred.

## 54.21 ONBUTTONOVER

### BEZEICHNUNG

ONBUTTONOVER – user moved the mouse over a button / VERALTET

### ÜBERSICHT

Label(ONBUTTONOVERx)

### BESCHREIBUNG

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

This event is raised when the user moves the mouse over a declared button. Hollywood will **Gosub()** then to the label with the name ONBUTTONOVERx where x is the identifier of the button. For example, Hollywood will **Gosub()** to the label called

ONBUTTONOVER15 when the user moves the mouse over the button with the number 15.

**EINGABEN**

x                    button number

**BEISPIEL**

Siehe [Abschnitt 54.5 \[CreateButton\]](#), Seite 1211.

## 54.22 ONBUTTONOVERALL

**BEZEICHNUNG**

ONBUTTONOVERALL – user moved mouse over any button (V1.5) / VERALTET

**ÜBERSICHT**

Label(ONBUTTONOVERALL)

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

If you specify this event and it is enabled, Hollywood will always jump to this label when it gets an **ONBUTTONOVER** event. If you want to know, which button caused the label jump, just call **GetEventCode()**. It will return which button caused the event.

This event is very useful if you have e.g. 100 or more buttons and every button just does a **Gosub()** to a sub-routine which receives an id which button was pressed. Instead of defining 100 labels now, you can just use ONBUTTONOVERALL together with **GetEventCode()**.

**EINGABEN**

none

**BEISPIEL**

Siehe [Abschnitt 54.20 \[ONBUTTONCLICKALL\]](#), Seite 1221.

## 54.23 ONBUTTONRIGHTCLICK

**BEZEICHNUNG**

ONBUTTONRIGHTCLICK – user right-clicked a button (V1.5) / VERALTET

**ÜBERSICHT**

Label(ONBUTTONRIGHTCLICKx)

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

This event is raised when the user right-clicks on a declared button. Hollywood will **Gosub()** then to the label with the name ONBUTTONRIGHTCLICKx where x is the identifier of the button. For example, Hollywood will **Gosub()** to the label called ONBUTTONRIGHTCLICK15 when the user clicks on the button with the number 15.

**EINGABEN**

x                    button number

**BEISPIEL**

Siehe [Abschnitt 54.5 \[CreateButton\]](#), Seite 1211.

## 54.24 ONBUTTONRIGHTCLICKALL

**BEZEICHNUNG**

ONBUTTONRIGHTCLICKALL – user right-clicked any button (V1.5) / VERALTET

**ÜBERSICHT**

Label (ONBUTTONRIGHTCLICKALL)

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

If you specify this event and it is enabled, Hollywood will always jump to this label when it gets an **ONBUTTONRIGHTCLICK** event. If you want to know, which button caused the label jump, just call **GetEventCode()**. It will return which button caused the event.

This event is very useful if you have e.g. 100 or more buttons and every button just does a **Gosub()** to a sub-routine which receives an id which button was pressed. Instead of defining 100 labels now, you can just use ONBUTTONRIGHTCLICKALL together with **GetEventCode()**.

**EINGABEN**

none

**BEISPIEL**

Siehe [Abschnitt 54.20 \[ONBUTTONCLICKALL\]](#), Seite 1221.

## 54.25 ONJOYDOWN

**BEZEICHNUNG**

ONJOYDOWN – user moved Joystick down (V1.5) / GELÖSCHT

**ÜBERSICHT**

Label (ONJOYDOWNx)

**BESCHREIBUNG**

Achtung: Dieses Label wird ab Hollywood 2.0 nicht mehr unterstützt.

This event is raised when the user moves the Joystick plugged in port x down.

**EINGABEN**

x                    port number (usually 1 for the standard Joystick port)

**BEISPIEL**

Siehe [Abschnitt 54.31 \[ONJOYUP\]](#), Seite 1226.



## 54.26 ONJOYDOWNLEFT

### BEZEICHNUNG

ONJOYDOWNLEFT – user moved Joystick down left (V1.5) / GELÖSCHT

### ÜBERSICHT

Label(ONJOYDOWNLEFTx)

### BESCHREIBUNG

Achtung: Dieses Label wird ab Hollywood 2.0 nicht mehr unterstützt.

This event is raised when the user moves the Joystick plugged in port x down left.

### EINGABEN

x                    port number (usually 1 for the standard Joystick port)

### BEISPIEL

Siehe [Abschnitt 54.31 \[ONJOYUP\]](#), Seite 1226.

## 54.27 ONJOYDOWNRIGHT

### BEZEICHNUNG

ONJOYDOWNRIGHT – user moved Joystick down right (V1.5) / GELÖSCHT

### ÜBERSICHT

Label(ONJOYDOWNRIGHTx)

### BESCHREIBUNG

Achtung: Dieses Label wird ab Hollywood 2.0 nicht mehr unterstützt.

This event is raised when the user moves the Joystick plugged in port x down right.

### EINGABEN

x                    port number (usually 1 for the standard Joystick port)

### BEISPIEL

Siehe [Abschnitt 54.31 \[ONJOYUP\]](#), Seite 1226.

## 54.28 ONJOYFIRE

### BEZEICHNUNG

ONJOYFIRE – user pressed Joystick button (V1.5) / GELÖSCHT

### ÜBERSICHT

Label(ONJOYFIREx)

### BESCHREIBUNG

Achtung: Dieses Label wird ab Hollywood 2.0 nicht mehr unterstützt.

This event is raised when the user presses the fire button of the Joystick that is plugged in port x.

### EINGABEN

x                    port number (usually 1 for the standard Joystick port)

**BEISPIEL**

Siehe [Abschnitt 54.31 \[ONJOYUP\]](#), Seite 1226.

**54.29 ONJOYLEFT****BEZEICHNUNG**

ONJOYLEFT – user moved Joystick up right (V1.5) / GELÖSCHT

**ÜBERSICHT**

Label(ONJOYLEFTx)

**BESCHREIBUNG**

Achtung: Dieses Label wird ab Hollywood 2.0 nicht mehr unterstützt.

This event is raised when the user moves the Joystick plugged in port x up left.

**EINGABEN**

x                    port number (usually 1 for the standard Joystick port)

**BEISPIEL**

Siehe [Abschnitt 54.31 \[ONJOYUP\]](#), Seite 1226.

**54.30 ONJOYRIGHT****BEZEICHNUNG**

ONJOYRIGHT – user moved Joystick right (V1.5) / GELÖSCHT

**ÜBERSICHT**

Label(ONJOYRIGHTx)

**BESCHREIBUNG**

Achtung: Dieses Label wird ab Hollywood 2.0 nicht mehr unterstützt.

This event is raised when the user moves the Joystick plugged in port x right.

**EINGABEN**

x                    port number (usually 1 for the standard Joystick port)

**BEISPIEL**

Siehe [Abschnitt 54.31 \[ONJOYUP\]](#), Seite 1226.

**54.31 ONJOYUP****BEZEICHNUNG**

ONJOYUP – user moved Joystick up (V1.5) / GELÖSCHT

**ÜBERSICHT**

Label(ONJOYUPx)

**BESCHREIBUNG**

Achtung: Dieses Label wird ab Hollywood 2.0 nicht mehr unterstützt.

This event is raised when the user moves the Joystick plugged in port x up.

**EINGABEN**

x            port number (usually 1 for the standard Joystick port)

**BEISPIEL**

```
While(quit = FALSE)
WaitEvent
Wend
```

```
Label(ONJOYUP1)
NPrint("Joy up")
Return
```

```
Label(ONJOYUPRIGHT1)
NPrint("Joy up right")
Return
```

```
Label(ONJOYRIGHT1)
NPrint("Joy right")
Return
```

```
Label(ONJOYDOWNRIGHT1)
NPrint("Joy down right")
Return
```

```
Label(ONJOYDOWN1)
NPrint("Joy down")
Return
```

```
Label(ONJOYDOWNLEFT1)
NPrint("Joy down left")
Return
```

```
Label(ONJOYLEFT1)
NPrint("Joy left")
Return
```

```
Label(ONJOYUPLEFT1)
NPrint("Joy up left")
Return
```

```
Label(ONJOYFIRE1)
NPrint("Joy fire")
Return
```

The above code shows how to query input from the Joystick.

## 54.32 ONJOYUPLEFT

### BEZEICHNUNG

ONJOYUPLEFT – user moved Joystick up right (V1.5) / GELÖSCHT

### ÜBERSICHT

Label(ONJOYUPLEFTx)

### BESCHREIBUNG

Achtung: Dieses Label wird ab Hollywood 2.0 nicht mehr unterstützt.

This event is raised when the user moves the Joystick plugged in port x up left.

### EINGABEN

x            port number (usually 1 for the standard Joystick port)

### BEISPIEL

Siehe [Abschnitt 54.31 \[ONJOYUP\]](#), Seite 1226.

## 54.33 ONJOYUPRIGHT

### BEZEICHNUNG

ONJOYUPRIGHT – user moved Joystick up right (V1.5) / GELÖSCHT

### ÜBERSICHT

Label(ONJOYUPRIGHTx)

### BESCHREIBUNG

Achtung: Dieses Label wird ab Hollywood 2.0 nicht mehr unterstützt.

This event is raised when the user moves the Joystick plugged in port x up right.

### EINGABEN

x            port number (usually 1 for the standard Joystick port)

### BEISPIEL

Siehe [Abschnitt 54.31 \[ONJOYUP\]](#), Seite 1226.

## 54.34 ONKEYDOWN

### BEZEICHNUNG

ONKEYDOWN – user pressed a key / VERALTET

### ÜBERSICHT

Label(ONKEYDOWNx)

### BESCHREIBUNG

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

This event is raised when the user presses a monitored key. Hollywood will `Gosub()` then to the label with the name `ONKEYDOWNx` where `x` is the identifier of the keydown. For example, Hollywood will `Gosub()` to the label called `ONKEYDOWN15` when the user presses the key with the number 15.

**EINGABEN**

`x`                      key number

**BEISPIEL**

Siehe [Abschnitt 54.6 \[CreateKeyDown\]](#), Seite 1213.

## 54.35 ONKEYDOWNALL

**BEZEICHNUNG**

`ONKEYDOWNALL` – user invoked any keydown (V1.5) / VERALTET

**ÜBERSICHT**

`Label(ONKEYDOWNALL)`

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

If you specify this event and it is enabled, Hollywood will always jump to this label when it gets an `ONKEYDOWN` event. If you want to know, which keydown caused the label jump, just call `GetEventCode()`. It will return which keydown caused the event.

This event is very useful if you have e.g. 100 or more keydown's and every keydown just does a `Gosub()` to a sub-routine which receives an id which key was pressed. Instead of defining 100 labels now, you can just use `ONKEYDOWNALL` together with `GetEventCode()`.

**EINGABEN**

none

**BEISPIEL**

Siehe [Abschnitt 54.20 \[ONBUTTONCLICKALL\]](#), Seite 1221.

## 54.36 RemoveButton

**BEZEICHNUNG**

`RemoveButton` – remove a button / VERALTET

**ÜBERSICHT**

`RemoveButton(id)`

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie nun den Befehl `DeleteButton()`.

This function will remove the button specified by `id` from Hollywood's button database. Events of this button will no longer be received.

**EINGABEN**

id            button to delete

**BEISPIEL**

```
RemoveButton(1)
```

Deletes button 1.

## 54.37 RemoveKeyDown

**BEZEICHNUNG**

RemoveKeyDown – remove a keydown object / VERALTET

**ÜBERSICHT**

```
RemoveKeyDown(id)
```

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

This function will remove the keydown object specified by `id` from Hollywood's keydown database. Events of this keydown will no longer be received.

**EINGABEN**

id            keydown to delete

**BEISPIEL**

```
RemoveKeydown(1)
```

Deletes keydown object 1.

## 54.38 Return

**BEZEICHNUNG**

Return – return to last Gosub() / VERALTET

**ÜBERSICHT**

```
Return()
```

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger im Zusammenhang mit `Gosub()` verwenden. Bitte benutzen Sie nun **Funktionen** anstelle von Labels. Zusammen mit **Funktionen** darf selbstverständlich die **Return-Anweisung** benutzt werden.

This function will return to the point where the last routine was gosub'ed from.

**EINGABEN**

none

**BEISPIEL**

Siehe [Abschnitt 54.12 \[Gosub\]](#), Seite 1217.

## 54.39 SIZEWINDOW

### BEZEICHNUNG

SIZEWINDOW – user changed the window size / VERALTET

### ÜBERSICHT

Label(SIZEWINDOW)

### BESCHREIBUNG

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie nun den Befehl **InstallEventHandler()**.

Hollywood will **Gosub()** to this label when the user changes the window size. If your application shall support resizable windows, you must place some code after this label which redraws/repositions your objects.

### EINGABEN

none

## 54.40 WhileKeyDown

### BEZEICHNUNG

WhileKeyDown – halt until key is up (V1.5) / VERALTET

### ÜBERSICHT

WhileKeyDown()

### BESCHREIBUNG

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

This function can be used after an **ONKEYDOWN** event occurred. If you call this function, Hollywood will wait until the user releases the key which caused the event.

If you do not call this function after your Label(ONKEYDOWNx) you may receive several events if the user holds down the key a bit longer.

### EINGABEN

none

## 54.41 WhileMouseDown

### BEZEICHNUNG

WhileMouseDown – halt until the mouse button is up / VERALTET

### ÜBERSICHT

WhileMouseDown()

### BESCHREIBUNG

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

This function must be called after a `ONBUTTONCLICK` event occurred. If you call this function, Hollywood will wait until the user has finished his mouse button click. It is absolutely necessary to call this function right after the `ONBUTTONCLICK` event occurred because even if the user clicks the mouse only once, several events may be generated. To prevent this, call this function right after the `ONBUTTONCLICK` occurred.

**EINGABEN**

none

**BEISPIEL**

Siehe [Abschnitt 54.5 \[CreateButton\]](#), Seite 1211.

## 54.42 WhileMouseOn

**BEZEICHNUNG**

WhileMouseOn – halt until the user moves mouse out of a button / VERALTET

**ÜBERSICHT**

`WhileMouseOn()`

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

This function can be called after a `ONBUTTONOVER` event occurred. If you call this function, Hollywood will wait until the user moves the mouse out of the button it is currently over. This is useful when you want to display a brush when the mouse is over a button (hover effect).

This function is optional after an `ONBUTTONOVER` event (other than the [WhileMouseDown\(\)](#) function which is required after a `ONBUTTONCLICK` event!)

**EINGABEN**

none

**BEISPIEL**

Siehe [Abschnitt 54.5 \[CreateButton\]](#), Seite 1211.

## 54.43 WhileRightMouseDown

**BEZEICHNUNG**

WhileRightMouseDown – halt until the right mouse button is up (V1.5) / VERALTET

**ÜBERSICHT**

`WhileRightMouseDown()`

**BESCHREIBUNG**

Achtung: Dieser Befehl ist Teil der Hollywood 1.x Ereignisbibliothek. Sie sollten ihn nicht länger verwenden. Bitte benutzen Sie die Befehle der neuen Bibliothek ab Hollywood 2.0.

This function must be called after a `ONBUTTONRIGHTCLICK` event occurred. If you call this function, Hollywood will wait until the user has finished his mouse button click.



It is absolutely necessary to call this function right after the `ONBUTTONRIGHTCLICK` event occurred because even if the user clicks the mouse only once, several events may be generated. To prevent this, call this function right after the `ONBUTTONRIGHTCLICK` occurred.

**EINGABEN**

none

**BEISPIEL**

Siehe [Abschnitt 54.5 \[CreateButton\]](#), Seite 1211.



## 55 Videobibliothek

### 55.1 Übersicht

Die Videobibliothek von Hollywood enthält Befehle zum Laden und Abspielen von Video-Objekten. Video-Objekte sind Hollywood-Objekte, die einen Videodatenstrom enthalten, der mit einem Audiodatenstrom gebündelt werden kann. Bei der Wiedergabe eines Video-Objekts sorgt Hollywood dafür, dass Video- und Audiodatenströme perfekt aufeinander abgestimmt sind.

Sie können eine Videodatei mit der Präprozessor-Anweisung `@VIDEO` oder dem Befehl `OpenVideo()` von der Festplatte öffnen. Um ein Video abzuspielen, verwenden Sie den Befehl `PlayVideo()`.

Die Videobibliothek von Hollywood unterstützt zwei verschiedene Systeme:

1. **Integriertes Videosystem:** Dies ist ein plattformunabhängiges Videosystem, das auf allen Plattformen unterstützt wird. Es ist der flexibelste Video-Abspieler und unterstützt erweiterte Funktionen wie Videoebenen und Plugins. Der Nachteil ist, dass die Dekodierung vollständig in Software erfolgt, weshalb große Videos (oder Videos mit 50 Bildern pro Sekunde oder mehr) möglicherweise stottern. In diesem Fall können Sie stattdessen den nativen Video-Abspieler verwenden (siehe unten). Der native Video-Abspieler ist oft hardwarebeschleunigt, weshalb die Videowiedergabe auch in sehr hohen Auflösungen oder mit vielen Bildern pro Sekunde noch flüssig ist. Beachten Sie, dass das einzige Videoformat, das vom eingebauten Video-Abspieler unterstützt wird, das CDXL-Videoformat ist, das Anfang der 90er Jahre von Commodore entwickelt wurde. CDXL ist für die heutigen Videoanforderungen nicht sehr nützlich, aber der große Vorteil des eingebauten Video-Abspielers besteht darin, dass er Videos über Hollywood-Plugins laden kann. Die Installation von Video-Plugins kann die Funktionalität des integrierten Video-Abspielers erheblich verbessern und es Hollywood ermöglichen, viele verschiedene Videoformate abzuspielen.
2. **Natives Videosystem:** Dies wird nur unter Windows, macOS und iOS unterstützt. Dieses System lädt und spielt Videos über die Videoschnittstelle des Betriebssystems ab. Unter Windows erfolgt die Videowiedergabe über Media Foundation und DirectShow, während macOS und iOS AV Foundation oder QuickTime (auf älteren Systemen) verwenden. Dieser Abspieler ist nicht so flexibel wie der integrierte Video-Abspieler. Er unterstützt keine Video-Ebenen und es unterstützt auch keine Videoformat-Lader, die von Hollywood-Plugins zur Verfügung gestellt werden. Aber er kann viel schneller sein, da native Video-Abspieler oft hardwarebeschleunigt sind. Die Anzahl der Videoformate, die vom nativen Video-Abspieler wiedergegeben werden können, ist ebenfalls begrenzt. Das beste Format für die Verwendung mit dem nativen Video-Abspieler ist MPEG4, da dies auf allen Plattformen außer auf sehr alten macOS- oder Windows-Versionen unterstützt wird.

Standardmäßig öffnet Hollywood zuerst die Videodatei mit dem eingebauten Videosystem. Sie können dieses Verhalten ändern, indem Sie den Tag `Loader` beim Aufruf der Befehle `@VIDEO` oder `OpenVideo()` verwenden. Siehe [Abschnitt 55.17 \[VIDEO\]](#), [Seite 1247](#), für Details.

## 55.2 CloseVideo

### BEZEICHNUNG

CloseVideo – schließt ein Video (V5.0)

### ÜBERSICHT

CloseVideo(id)

### BESCHREIBUNG

Dieser Befehl löscht das in `id` angegebene Video aus dem Arbeitsspeicher und schließt das Video. Wenn das Video noch läuft, wird es zunächst gestoppt werden. Obwohl Hollywood alle Ressourcen automatisch frei gibt, wenn es verlassen wird, sollten Sie `CloseVideo()` benutzen, wenn Sie eine Videodatei nicht mehr brauchen, weil sie so den Speicherverbrauch reduzieren.

### EINGABEN

`id` ID des Videos, das geschlossen werden soll

## 55.3 DisplayVideoFrame

### BEZEICHNUNG

DisplayVideoFrame – zeigt ein Einzelbild eines Videos an (V6.0)

### ÜBERSICHT

DisplayVideoFrame(id, x, y, pos[, table])

### BESCHREIBUNG

Dieser Befehl zeigt an den angegebenen Koordinaten ein einzelnes Bild eines Videos an und fügt eine neue Ebene des Typs `#VIDEO` dem Ebenenstapel hinzu. Das Einzelbild ist nicht als absolute Indexposition angegeben, sondern als ein Zeitstempel in Millisekunden. Um das erste Bild anzuzeigen, würden Sie 0 im Argument `pos` übergeben.

Bitte beachten Sie, dass dieser Befehl derzeit nicht mit deaktivierten Ebenen funktioniert. Die Ebenen müssen aktiviert sein, um mit diesem Befehl arbeiten zu können.

`DisplayVideoFrame()` erkennt auch das optionale Argument `table`, wo Sie eine oder mehrere der [Standard-Tags zum Zeichnen](#) angeben können. Siehe [Abschnitt 29.17 \[Standard-Tags zum Zeichnen\]](#), [Seite 613](#), für weitere Informationen über die Standard-Tags, die fast alle Zeichnungsbefehle von Hollywood unterstützen.

### EINGABEN

<code>id</code>	ID des Videos
<code>x</code>	gewünschte x-Koordinate
<code>y</code>	gewünschte y-Koordinate
<code>pos</code>	Zeitstempel des Einzelbildes in Millisekunden, welches angezeigt werden soll
<code>table</code>	optional: Tabelle für weitere Optionen

### BEISPIEL

```
DisplayVideoFrame(1, #CENTER, #CENTER, 0)
```

Dieser Code zeigt das erste Einzelbild des Videos 1 in der Mitte des Bildschirms an.

## 55.4 ForceVideoDriver

### BEZEICHNUNG

ForceVideoDriver – erzwingt die Verwendung eines bestimmten Grafiktreibers / VERALTET (V5.1)

### ÜBERSICHT

ForceVideoDriver(driver)

### BESCHREIBUNG

Beachten Sie, dass dieser Befehl seit Hollywood 6.0 veraltet ist, weil Sie jetzt einfach den neuen Loader-Tag mit **OpenVideo()** und **@VIDEO** verwenden können. Das Loader-Äquivalent für #VIDDRV\_OS ist **native** und das Loader-Äquivalent für #VIDDRV\_HOLLYWOOD ist **inbuilt|plugin**.

Dieser Befehl kann verwendet werden, um den Grafiktreiber anzugeben, welcher den Befehl **OpenVideo()** verwenden soll. Hollywood unterstützt derzeit die folgenden zwei Grafiktreiber:

#### #VIDDRV\_HOLLYWOOD:

Der Hollywoodplattform unabhängige Videospieler ist der Standardtreiber. Er unterstützt die Wiedergabe des CDXL Format sowie alle Formate, für die Sie ein Plugin haben.

#### #VIDDRV\_OS:

Dieser Treiber verwendet das native Videosystem des OS. Dies wird derzeit nur unter Windows, macOS und iOS unterstützt. Unter Windows verwendet dieser Treiber die Media Foundation oder DirectShow-Technologie, während er auf macOS sowie iOS AV Foundation oder Quicktime (auf älteren Systemen) verwendet.

In der Standardeinstellung wird dem Treiber #VIDDRV\_HOLLYWOOD Vorrang vor #VIDDRV\_OS gegeben, was bedeutet, dass Hollywood zuerst versucht, das Video mit dem eingebauten Videospieler anzuzeigen. Nur wenn das nicht funktioniert, wird Hollywood zu dem nativen Abspieler des OS wechseln.

### EINGABEN

driver      gewünschter Videotreiber

## 55.5 GetVideoFrame

### BEZEICHNUNG

GetVideoFrame – konvertiert ein Einzelbild in einen Pinsel (V5.0)

### ÜBERSICHT

[id] = GetVideoFrame(brushid, frame, videoid[, unit])

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um ein Einzelbild eines Videos in einen Pinsel zu konvertieren. Das Video muss zuvor mit dem Befehl **OpenVideo()** oder der Präprozessor-Anweisung **@VIDEO** geöffnet worden sein. Im Argument **brushid** müssen Sie eine ID für

den Pinsel übergeben (alternativ können Sie auch **Nil** für die automatische ID-Auswahl angeben). Im Argument **frame** müssen Sie festlegen, welches Einzelbild des Videos konvertiert werden soll und im Argument **videoid** geben sie die ID des Videos ein.

Das optionale Argument **unit** wird verwendet, um festzulegen, wie der Wert in **frame** interpretiert werden sollte. Wenn **unit** auf 0 gesetzt ist, dann wird der Wert in **frame** als absoluter Einzelbildindex interpretiert. Dies ist auch die Standardeinstellung. Wenn **unit** auf 1 gesetzt ist, dann wird der Wert in **frame** als Zeitstempel in Millisekunden angesehen und **GetVideoFrame()** wird das Einzelbild an diesem Zeitstempel konvertieren. Es wird empfohlen, **unit** mit dem Wert 1 für den Zugriff auf Einzelbilder zu verwenden, da dies in der Regel viel schneller ist. Wenn Sie trotzdem Einzelbilder durch ihren absoluten Index bestimmen müssen, lesen Sie bitte die untenstehende Warnung.

Bitte beachten Sie, dass der Zugriff auf Einzelbilder durch den absoluten Index ist in der Regel eine sehr aufwendige Operation ist, da Hollywood für die meisten Videoformate den ganzen Weg vom Video durchlaufen muss, bis das angeforderte Einzelbild erreicht ist. Dies erfordert viel Zeit und ist daher nur von begrenztem praktischen Nutzen. Jedoch gibt es einen Sonderfall, wo der Zugriff über den absoluten Index sehr effizient ist, und das wäre beim sequentiellen Auslesen der Einzelbilder eines Videos. "Sequentielles Auslesen" bedeutet, dass Sie ein Bild nach dem anderen aus dem Video lesen (zuerst Einzelbild 1, dann Einzelbild 2, dann Einzelbild 3, etc.). Dies kann sehr schnell erfolgen. Was hingegen wieder viel Zeit in Anspruch nimmt, ist das Auslesen der Einzelbilder in Rückwärtsrichtung (zuerst Einzelbild 10, dann Einzelbild 9, nachher Einzelbild 8, etc.), oder bei großen Sprüngen zwischen den Einzelbilder (Nummer 1, dann Nummer 1000, nachher Nummer 5000 etc.). Sequentielles lesen wird in diesen Fällen effizienter sein.

Um die genaue Anzahl der Einzelbilder eines Videos herauszufinden, können Sie den Befehl **GetAttribute()** mit dem Objekttyp **#VIDEO** und dem Attribut **#ATTRNUMFRAMES** verwenden.

## EINGABEN

<b>id</b>	ID für den Pinsel oder <b>Nil</b> für die automatische ID-Auswahl
<b>frame</b>	Einzelbild; das Format von diesem Argument wird weiter unten in <b>unit</b> festgelegt
<b>videoid</b>	ID des Videos
<b>unit</b>	optional: Format für das Argument <b>frame</b> ; dies kann entweder 0 sein, womit das Argument <b>frame</b> einen absoluten Einzelbildindex beinhaltet oder 1, wodurch das Argument <b>frame</b> einen Zeitstempel in Millisekunden verwendet (Standard ist 0).

## RÜCKGABEWERTE

<b>id</b>	optional: ID des Pinsels; wird nur zurückgegeben, wenn <b>Nil</b> im Argument <b>brushid</b> übergeben wurde (siehe oben)
-----------	---

## BEISPIEL

```
my_frame = GetVideoFrame(Nil, 1, 2)
DisplayBrush(my_frame, #CENTER, #CENTER)
```

Der obige Code speichert das Einzelbild 1 vom Video 2 in einem neuen Pinsel. Der Pinsel wird dann in der Mitte des Displays angezeigt.

## 55.6 IsVideo

### BEZEICHNUNG

IsVideo – überprüft, ob eine Datei in einem unterstützten Videoformat ist (V5.0)

### ÜBERSICHT

```
ret = IsVideo(file$, table)
```

### BESCHREIBUNG

Dieser Befehl überprüft, ob die Datei im Argument `file$` in einem unterstützten Videoformat ist. Wenn ja, wird `True` zurück gegeben, andernfalls `False`. Wenn dieser Befehl `True` zurückgibt, können Sie das Video mit `OpenVideo()` öffnen.

Ab Hollywood 6.0 hat dieser Befehl ein optionales Argument `table`, mit dem Sie weitere Optionen konfigurieren können:

**Loader:** Mit diesem Tag können Sie einen oder mehrere Formatlademodule angeben, die die Datei laden soll. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Lademodulen beinhaltet. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, der die angegebene Datei öffnet. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Adaptermodulen beinhaltet. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), [Seite 96](#), für Details. (V6.0)

**UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), [Seite 100](#), für Details. (V10.0)

Siehe [Abschnitt 55.8 \[OpenVideo\]](#), [Seite 1240](#), für eine Liste der unterstützten Videoformate.

### EINGABEN

`file$`      Datei, die überprüft wird

`table`      optional: konfiguriert weitere Optionen (V6.0)

### RÜCKGABEWERTE

`ret`      `True`, wenn das Video in einem unterstützten Format vorliegt, andernfalls `False`

## 55.7 IsVideoPlaying

### BEZEICHNUNG

IsVideoPlaying – überprüft, ob ein Video gerade abgespielt wird (V5.0)

### ÜBERSICHT

```
playing = IsVideoPlaying(id)
```

### BESCHREIBUNG

Dieser Befehl überprüft, ob das in `id` angegebene Video gerade abgespielt wird. Falls dem so ist, wird **True** zurückgegeben, andernfalls **False**.

### EINGABEN

`id` ID des zu überprüfenden Videos

### RÜCKGABEWERTE

`playing` **True**, falls es gerade abgespielt wird, andernfalls **False**

## 55.8 OpenVideo

### BEZEICHNUNG

OpenVideo – öffnet eine Videodatei (V5.0)

### ÜBERSICHT

```
[id] = OpenVideo(id, filename$[, table])
```

### BESCHREIBUNG

Dieser Befehl öffnet die in `filename$` angegebene Videodatei und weist ihr die in `id` eingetragene ID zu. Wenn Sie **Nil** in `id` übergeben haben, wird `OpenVideo()` automatisch eine freie ID auswählen und zurückgeben. Die Videodatei wird für die Wiedergabe geöffnet und vorbereitet. Die Videowiedergabe wird immer direkt von der Festplatte durchgeführt, was bedeutet, dass `OpenVideo()` keine Daten vorpuffert. Es wird nur alle erforderlichen Parameter für die Videowiedergabe initialisieren.

Videoformate, die auf allen Plattformen unterstützt werden, sind CDXL und Formate, für die Sie ein Plugin haben. Je nach Plattform, auf der Hollywood ausgeführt wird, können mehr Videoformate unterstützt werden. Auf Windows ist Hollywood in der Lage, alle Videoformate zu öffnen, für die Sie ein Media Foundation- oder Directshow-Codec installiert haben. Unter macOS kann Hollywood alle Videoformate öffnen, die durch AV Foundation oder Quicktime (auf älteren Macs) unterstützt werden.

Ab Hollywood 6.0 akzeptiert dieser Befehl ein optionales Argument `table`, das die folgenden Optionen kennt:

**Loader:** Mit diesem Tag können Sie einen oder mehrere Formatlademodule angeben, die die Datei laden soll. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Lademodulen beinhaltet. Standardmäßig wird der mit `SetDefaultLoader()` eingestellte Lader verwendet. Dies bedeutet, dass Hollywood zuerst alle Plugins durchgeht, ob sie die Videodatei laden können, dann wird das integrierte Lademodul überprüft



(derzeit nur CDXL) und schlussendlich wird Hollywood die Videoschnittstelle des OS abfragen, um dieses Video abzuspielen. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), Seite 96, für Details. (V6.0)

**Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, der die angegebene Datei öffnet. Dies muss mit einer Zeichenkette festgelegt werden, die den/die Namen eines oder mehreren Adaptermodulen beinhaltet. Standardmäßig wird der mit `SetDefaultAdapter()` eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), Seite 96, für Details. (V6.0)

**UserTags:**

Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), Seite 100, für Details. (V10.0)

Dieser Befehl ist auch als Präprozessor vorhanden: Mit `@VIDEO` laden Sie die Videodateien vor!

## EINGABEN

**id**            Identifikator des Videos oder `Nil` für die [automatische ID-Auswahl](#)

**filename\$**  
die zu ladende Videodatei

**table**        optional: Tabelle mit weiteren Parametern (V6.0)

## RÜCKGABEWERTE

**id**            optional: ID des Videos; wird nur zurückgegeben, wenn Sie beim Argument `id Nil` angegeben haben (siehe oben)

## BEISPIEL

```
OpenVideo(1, "intro.avi")
PlayVideo(1)
```

Der obige Code lädt das Video "intro.avi" und spielt es ab.

# 55.9 PauseVideo

## BEZEICHNUNG

`PauseVideo` – hält ein Video an (V5.0)

## ÜBERSICHT

`PauseVideo(id)`

## BESCHREIBUNG

Dieser Befehl hält das Video mit dem Identifikator `id` an. Dieses Video muss laufen, wenn Sie diesen Befehl aufrufen. Sie können die Wiedergabe später wieder durch den Befehl `ResumeVideo()` fortsetzen.

Bitte beachten Sie, dass wenn ein Video anhält, es nicht vom Display entfernt wird. Stattdessen wird das aktuell angezeigte Bild eingefroren, bis Sie `ResumeVideo()` aufrufen. Wenn Sie ein Video vollständig vom Display entfernen möchten, müssen Sie `StopVideo()` aufrufen.

## EINGABEN

`id` ID des Videos, welches Sie anhalten möchten

## 55.10 PlayVideo

### BEZEICHNUNG

`PlayVideo` – startet die Wiedergabe eines Videos (V5.0)

### ÜBERSICHT

`PlayVideo(id[, x, y, table])`

### BESCHREIBUNG

Dieser Befehl startet die Wiedergabe des in `id` angegebenen Videos. Dieses Video muss entweder mit der Präprozessor-Anweisung `@VIDEO` oder dem Befehl `OpenVideo()` geöffnet worden sein. Die optionalen Argumente `x` und `y` legen fest, wo auf dem Display das Video erscheinen soll. Wenn Sie diese Argumente nicht angeben, wird `PlayVideo()` die Koordinaten vom Befehl `SetVideoPosition()` verwenden. Wenn Sie `SetVideoPosition()` für dieses Video nicht aufgerufen haben, wird standardmäßig die Position auf 0:0 (linke obere Ecke des Displays) benutzt.

`PlayVideo()` arbeitet in einer asynchronen Weise. Nachdem Sie ein Video gestartet haben, können Sie die Wiedergabe mit den Befehlen `StopVideo()` oder `StopLayer()` steuern, je nachdem ob Sie Normale- oder Ebenenwiedergabemodus verwenden.

Hollywood unterstützt zwei verschiedene Videowiedergabemodi: Normale und Ebenenwiedergabe. Normale Wiedergabe ist der schnellste und optimierteste Modus, da er beispielsweise die Hardwarebeschleunigung durch die Verwendung Video-Overlays nutzen kann. Der Nachteil des Normalmodus ist, dass er einige Einschränkungen hat (siehe unten für weitere Informationen). Ebenenwiedergabemodus ist sehr flexibel. Da Ihr Video in einer Hollywoodebene geschrieben wurde, können Sie alle Ebenenbefehle mit Ihrem Video verwenden, wie z.B. drehen, Transparenz und Übergangseffekte oder spezielle Bildfilter anwenden. Der Nachteil des Ebenenwiedergabemodus ist, dass er ziemlich langsam ist, weil er nicht die Hardwarebeschleunigung verwenden kann. Somit benötigen Sie viel rohe CPU-Leistung, um in diesem Modus anständige Einzelbildraten zu erhalten. In den meisten Fällen sollte der normale Wiedergabemodus ausreichend sein. Ebenenwiedergabemodus ist nur dann erforderlich, wenn Sie während der Videowiedergabe erweiterte Funktionalität anwenden wollen.

Es ist wichtig zu beachten, dass `PlayVideo()` als Standard den normalen Wiedergabemodus verwendet, auch wenn Ebenen für das aktuellen BGPic aktiviert ist. Wenn `PlayVideo()` Ebenenwiedergabe verwenden soll, werden Sie dies explizit angeben müssen, indem Sie den Tag `UseLayer` im optionalen Argument `table` auf `True` setzen.

Normaler Wiedergabemodus kommt mit der Einschränkung, dass Videos immer über allem anderen angezeigt wird. Das bedeutet, dass es für Sie unmöglich ist, auf einem

Video zu zeichnen. Stattdessen werden alle Grafikbefehle immer unter dem Videobereich gezeichnet. Auch Sprites werden nie über der Videografik erscheinen. Außerdem wird das Video solange sichtbar bleiben, bis Sie `StopVideo()` aufrufen. Ein Anhalten des Videos wird es nicht aus dem Display entfernen. Stattdessen wird das aktuell angezeigte Bild eingefroren werden, bis Sie `ResumeVideo()` aufrufen. Wenn Sie ein Video vollständig aus dem Display entfernen möchten, müssen Sie immer den Befehl `StopVideo()` benutzen. Wenn Sie ein Video in der normalen Wiedergabe bewegen oder die Größe ändern möchten, müssen Sie die Befehle `SetVideoPosition()` und `SetVideoSize()` für diese Aufgabe verwenden.

Im Ebenenwiedergabemodus können Sie alle Befehle der Ebenenbibliothek von Hollywood verwenden, um die Videowiedergabe zu steuern, d.h. Sie können die Videogröße und Ausrichtung mit den Befehlen `ScaleLayer()` und `RotateLayer()` neu definieren sowie die z-Reihenfolge mit dem Befehl `SetLayerZPos()` ändern. Sie können mit den Befehlen `ShowLayer()` und `HideLayer()` Videos ein- und ausblenden sowie Transparenz oder Filter mit `SetLayerTransparency()` und `SetLayerFilter()` anwenden. Alle Befehle der Ebenenbibliothek von Hollywood können mit Video-Ebenen verwendet werden. Zum Stoppen oder Anhalten einer Video-Ebene, verwenden Sie die Befehle `StopLayer()` und `PauseLayer()`. Um an eine neue Position innerhalb des Videos zu springen, verwenden Sie `SeekLayer()`. Mit `SetLayerVolume()` ändern Sie die Lautstärke einer Video-Ebene.

Bitte beachten Sie, dass Ebenenwiedergabe nur dann möglich ist, wenn das Video mit dem eingebauten Videotreiber oder einem Plugin geöffnet wurde. Ebenenwiedergabe wird nicht unterstützt, wenn die Videowiedergabe über den Treiber des Host-Betriebssystems läuft. Sie können den Videotreiber mit dem Tag `Loader` des Befehls `OpenVideo()` oder der Präprozessor-Anweisung `@VIDEO` ändern.

Es gibt keine Limits, wie viele Videos gleichzeitig abgespielt werden können. Hardwarebeschleunigung kann jedoch oft nur für ein einzelnes Video verwendet werden. Wenn mehrere Videos gleichzeitig abgespielt werden, wechselt oft Hollywood auf Softwarewiedergabe, die langsamer ist. Bitte beachten Sie auch, dass die Videowiedergabe im Allgemeinen eine starke CPU benötigt. 68k-Prozessoren sind für diese Aufgabe viel zu langsam (außer auf WinUAE).

Beachten Sie, dass wenn ein neues BGPic mit dem Befehl `DisplayBGPic()` angezeigt wird, Hollywood automatisch alle Videos im normalen Modus stoppt. Videos im Ebenenwiedergabemodus werden jedoch weiterhin abgespielt, wenn das BGPic geändert wurde. So müssen Sie explizit die Video-Ebenen mit dem Befehl `StopLayer()` stoppen, bevor Sie das neue BGPic anzeigen.

Ab Hollywood 6.0 akzeptiert `PlayVideo()` das optionale Argument `table`, um die folgenden Optionen zu konfigurieren:

#### UseLayer:

Wenn Sie diesen Tag auf `True` setzen, wird `PlayVideo()` den Ebenenwiedergabemodus verwenden. Sie müssen **Ebenen aktivieren**, bevor Sie diesen Tag nutzen können. Siehe **Abschnitt 25.1 [Einführung in die Ebenentechnik]**, **Seite 405**, für Details. Wenn der Ebenenwiedergabemodus verwendet wird, wird dieser Befehl eine neue Ebene des Typs `#VIDEO` zum Ebenenstapel hinzuzufügen. Siehe oben für weitere Informationen über den Unterschied zwi-

schen normalen und Ebenenwiedergabemodus. Der Standardwert ist **False**. (V6.0)

**Channel:** Bestimmt den Kanal für die Wiedergabe der Audiospur des Videos. In der Standardeinstellung wählt **PlayVideo()** automatisch einen freien Kanal und liefert eine Fehlermeldung, wenn es keinen freien Kanal hat. Um dieses Verhalten zu ändern, können Sie dieses Feld verwenden. Wenn angegeben, wird es immer die Audiowiedergabe auf diesem Kanal erzwingen. Wenn der Kanal bereits abgespielt wird, wird er zunächst gestoppt werden. (V6.1)

Wenn der Ebenenwiedergabemodus verwendet wird, können Sie auch eine oder mehrere der **Standard-Tags zum Zeichnen** im optionalen Argument **table** angeben. Siehe **Abschnitt 29.17 [Standard-Tags zum Zeichnen], Seite 613**, für weitere Informationen über die Standard-Tags, die fast alle Zeichnungsbefehle von Hollywood unterstützen.

#### EINGABEN

<b>id</b>	Identifikator des Videos, welches abgespielt werden soll
<b>x</b>	optional: gewünschte x-Position für das Video (voreingestellt ist die Position, welche Sie mit dem Befehl <b>SetVideoPosition()</b> definiert haben oder 0)
<b>y</b>	optional: gewünschte y-Position für das Video (voreingestellt ist die Position, welche Sie mit dem Befehl <b>SetVideoPosition()</b> definiert haben oder 0)
<b>table</b>	optional: Tabelle für weitere Optionen (V6.0)

#### BEISPIEL

Siehe **Abschnitt 55.8 [OpenVideo], Seite 1240**.

## 55.11 ResumeVideo

#### BEZEICHNUNG

ResumeVideo – setzt ein angehaltenes Video fort (V5.0)

#### ÜBERSICHT

ResumeVideo(id)

#### BESCHREIBUNG

Dieser Befehl setzt die Wiedergabe des in **id** angegebenen Videos fort, welches vorher mit dem Befehl **PauseVideo()** angehalten wurde.

#### EINGABEN

<b>id</b>	Identifikator des Videos, welches fortgesetzt werden soll
-----------	---

## 55.12 SeekVideo

#### BEZEICHNUNG

SeekVideo – springt an eine bestimmte Position im Video (V5.0)

#### ÜBERSICHT

SeekVideo(id, pos)

**BESCHREIBUNG**

Sie können diesen Befehl verwenden, um die in `pos` angegebene Position im Video `id` zu springen. Das Video muss nicht laufen. Wenn das Video abgespielt wird und Sie `SeekVideo()` aufrufen, wird es sofort an die angegebene Position springen. Die Position wird in Millisekunden angegeben. Wenn Sie also an die Position 03.24 springen möchten, würden Sie den Wert 204000 in `pos` übergeben, weil  $3 * 60 * 1000 + 24 * 1000 = 204000$  ist.

Bitte beachten Sie, dass das Springen an eine bestimmte Position im Video ein komplexer Vorgang ist. Es gibt Videoformate, die keine Positionstabellen haben, so dass Hollywood sich zunächst an die Position annähert und dann einige Feinabstimmungen und Einzelbildersuche durchführt, so dass die endgültige Position ein wenig abseits von der in `SeekVideo()` angegebenen Position liegen kann. Es kann auch vorkommen, dass Hollywood nicht direkt zu einem Einzelbild springt, so könnte es am linken Rand des Bildschirms (Screen) Artefakte vom vorherigen Einzelbild haben.

**EINGABEN**

<code>id</code>	Identifikator des Videos
<code>pos</code>	neue Position im Video (in Millisekunden)

**55.13 SetVideoPosition****BEZEICHNUNG**

`SetVideoPosition` – verschiebt die Ausgabeposition eines Videos (V5.0)

**ÜBERSICHT**

`SetVideoPosition(id, x, y)`

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um die Ausgabeposition eines Videos zu verschieben. Wenn das Video gerade läuft, wird es sofort an die neue Position bewegt werden. Wenn es nicht läuft, wird die angegebene Position gespeichert, bis Sie das nächste Mal `PlayVideo()` aufrufen.

Bitte beachten Sie, dass dieser Befehl nicht für Videos verwendet werden kann, die im Ebenenmodus abgespielt werden. Sie können die Position von Video-Ebenen mit den Befehlen aus der Ebenenbibliothek ändern, z.B. mit `ShowLayer()`.

**EINGABEN**

<code>id</code>	ID des Videos, dessen Ausgabeposition verschoben werden soll
<code>x</code>	gewünschte x-Position des Videos
<code>y</code>	gewünschte y-Position des Videos

**BEISPIEL**

`SetVideoPosition(1, #RIGHT, #BOTTOM)`

Der obige Code verschiebt das Video 1 an den rechten unteren Rand des aktuellen Displays.

## 55.14 SetVideoSize

### BEZEICHNUNG

SetVideoSize – ändert die Ausgabegröße des Videos (V5.0)

### ÜBERSICHT

```
SetVideoSize(id, width, height[, smooth])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die Abmessungen eines Videos zu ändern. Wenn das Video gerade läuft, wird es sofort auf die neuen Dimensionen skaliert. Wenn es nicht läuft, werden die angegebenen Maße gespeichert, bis Sie das nächste Mal `PlayVideo()` aufrufen.

Sie können die spezielle Konstante `#KEEPASPRAT` entweder in `width` oder `height` (Breite oder Höhe) übergeben. Hollywood wird dann das Seitenverhältnis des Videos berücksichtigen und die Größe der anderen Seite automatisch berechnen. Als Alternative können Sie bei `width` und `height` auch eine prozentuale Zeichenfolge angeben, z.B. "50%".

Ab Hollywood 5.1 können Sie das optionale Argument `smooth` auf `True` setzen, um Antialiasing beim skalieren zu benutzen, welches zu einem glatteren Aussehen führt, aber dessen Berechnung länger dauert. Bitte beachten Sie, dass interpolierte Skalierung für Videos nur verfügbar ist, wenn auf der Hollywood-Plattform ein unabhängiger Videotreiber ohne Hardwarebeschleunigung benutzt wird.

Bitte beachten Sie, dass dieser Befehl nicht für Videos verwendet werden kann, die im Ebenenmodus abgespielt werden. Sie können die Größe von Video-Ebenen mit den Befehlen aus der Ebenenbibliothek ändern, z.B. mit `ShowLayer()`.

### EINGABEN

<code>id</code>	ID des Videos, dessen Größe Sie ändern wollen
<code>width</code>	gewünschte neue Breite des Videos
<code>height</code>	gewünschte neue Höhe des Videos
<code>smooth</code>	optional: Skalierung mit oder ohne Antialiasing; voreingestellt ist <code>False</code> (V5.1)

### BEISPIEL

```
SetVideoSize(1, 640, 480)
```

Der obige Code skaliert das Video 1 auf die Größe von 640x480 Pixel.

```
SetVideoSize(2, "50%", "50%")
```

Der obige Code schrumpft das Video Nummer 2 auf die Hälfte seiner Größe.

## 55.15 SetVideoVolume

### BEZEICHNUNG

SetVideoVolume – ändert die Lautstärke eines Videos (V5.0)

**ÜBERSICHT**

```
SetVideoVolume(id, volume)
```

**BESCHREIBUNG**

Dieser Befehl ändert die Lautstärke des in `id` angegebenen Videos. Wenn das Video gerade abgespielt wird, wird die Lautstärke sofort geändert. Dem Argument `volume` kann auch eine Zeichenfolge übergeben werden, die eine prozentuale Angabe beinhaltet, z.B. "50%".

**EINGABEN**

<code>id</code>	Identifikator des Videos
<code>volume</code>	neue Lautstärke für das Video (Bereich geht von 0=still bis 64=volle Lautstärke oder prozentuale Angabe)

**55.16 StopVideo****BEZEICHNUNG**

StopVideo – stoppt ein laufendes Video (V5.0)

**ÜBERSICHT**

```
StopVideo(id)
```

**BESCHREIBUNG**

Dieser Befehl stoppt das in `id` angegebene Video und entfernt es vom Display. Das Video muss entweder laufen oder gerade pausieren.

**EINGABEN**

<code>id</code>	ID des Videos, welches gestoppt wird
-----------------	--------------------------------------

**55.17 VIDEO****BEZEICHNUNG**

VIDEO – lädt ein Video vor (V5.0)

**ÜBERSICHT**

```
@VIDEO id, filename$[, table]
```

**BESCHREIBUNG**

Benutzen Sie diese Präprozessor-Anweisung, um ein Video vorzuladen, um es später mit dem Befehl `PlayVideo()` abzuspielen.

Videoformate, die auf allen Plattformen unterstützt werden, sind CDXL und Formate, für die Sie ein Plugin haben. Je nach Plattform, auf der Hollywood ausgeführt wird, können mehr Videoformate unterstützt werden. Auf Windows ist Hollywood in der Lage, alle Videoformate zu öffnen, für die ein Media Foundation- oder Directshow-Codec installiert ist. Unter macOS kann Hollywood alle Videoformate öffnen, die durch AV Foundation oder Quicktime (auf älteren Macs) unterstützt werden.

Das Argument **table** ist optional. Es ist eine Tabelle, die weitere Möglichkeiten für den Öffnungsvorgang verwendet werden kann. Die folgenden Felder der Tabelle können gesetzt werden:

- Link:** Setzen Sie dieses Feld auf **False**, wenn Sie dieses Video nicht in die ausführbare Datei/das Applet einbinden wollen, wenn Sie Ihr Skript kompilieren. Dieses Feld ist standardmäßig auf **True** gesetzt, was bedeutet, dass das Video mit der ausführbaren Datei/dem Applet beim Kompilieren verknüpft wird.
- Loader:** Mit diesem Tag können Sie ein oder mehrere Formatlademodule angeben, die dieses Video laden sollen. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehrerer Lademodulen beinhaltet. Standardmäßig wird der mit **SetDefaultLoader()** eingestellte Lader verwendet. Dies bedeutet, dass Hollywood zuerst mit allen installierten Plugins versucht, die Videodatei zu laden, dann versucht das integrierte Lademodul (derzeit nur CDXL) mit dem Video umzugehen und schließlich wird Hollywood die Videoschnittstelle des Host-OS benutzen, um dieses Video abzuspielen. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), Seite 96, für Details. (V6.0)
- Adapter:** Mit diesem Tag können Sie ein oder mehrere Dateiadapter angeben, die die angegebene Datei öffnen sollen. Dies muss als eine Zeichenfolge festgelegt werden, die den/die Namen eines oder mehrerer Adaptermodulen enthält. Standardmäßig wird der mit **SetDefaultAdapter()** eingestellte Adapter verwendet. Siehe [Abschnitt 7.9 \[Lade- und Adaptermodule\]](#), Seite 96, für Details. (V6.0)
- UserTags:** Dieser Tag kann verwendet werden, um zusätzliche Daten anzugeben, die an Lader und Adapter übergeben werden sollen. Wenn Sie diesen Tag verwenden, müssen Sie ihn auf eine Tabelle mit Schlüssel-Wert-Paaren setzen, die die zusätzlichen Daten enthalten, die an Plugins übergeben werden sollen. Siehe [Abschnitt 7.10 \[Benutzer-Tags\]](#), Seite 100, für Details. (V10.0)

Wenn Sie das Video manuell laden wollen, verwenden Sie den Befehl **OpenVideo()**.

## EINGABEN

- id** einen Wert, um das Video später im Code zu identifizieren
- filename\$** die Videodatei, die sie laden möchten
- table** optional: eine Tabelle mit weiteren Optionen

## BEISPIEL

```
@VIDEO 1, "intro.avi"
```

Der obige Code lädt das Video "intro.avi", so dass Sie es später mit dem Befehl **PlayVideo()** abspielen können.



## 56 Windows-Bibliothek

### 56.1 CreateShortcut

#### BEZEICHNUNG

CreateShortcut – erstellt eine Verknüpfung zu einer Datei (V4.7)

#### ÜBERSICHT

CreateShortcut(src\$, dest\$, desc\$)

#### PLATTFORMEN

Nur Microsoft Windows

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um eine \*.lnk-Verknüpfung zu der Datei **src\$** in der Datei **dest\$** zu erstellen. Die Verknüpfung wird die in **desc\$** übergebene Beschreibung verwenden. Die in **src\$** übergebene Quelldatei kann entweder eine ausführbare Datei oder eine Dokumentdatei sein.

#### EINGABEN

**src\$**            Quelldatei, auf die die Verknüpfung verweisen soll  
**dest\$**           Verknüpfungsdatei, die erstellt werden soll  
**desc\$**           Beschreibungs-Zeichenfolge der Verknüpfung

#### BEISPIEL

```
CreateShortcut("test.exe", "test.lnk", "Test shortcut")
```

Der obige Code erstellt einen Link zur Datei "test.exe" als "test.lnk" mit der Beschreibung "Test-shortcut".

### 56.2 GetShortcutPath

#### BEZEICHNUNG

GetShortcutPath – gibt den Pfad der Verknüpfung zurück (V5.2)

#### ÜBERSICHT

p\$, desc\$ = GetShortcutPath(f\$)

#### PLATTFORMEN

Nur Microsoft Windows

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um den vollständigen Pfad aus der in **f\$** angegebenen \*.lnk-Verknüpfungsdatei abzurufen. Der Pfad, auf den diese Verknüpfung zeigt, wird dann im ersten Rückgabewert **p\$** zurückgegeben. Wenn die Verknüpfungsdatei eine Beschreibung enthält, wird sie im zweiten Wert **desc\$** zurückgegeben.

#### EINGABEN

**f\$**                Verknüpfungsdatei

**RÜCKGABEWERTE**

`p$`            Vollständiger Pfad, auf den die Verknüpfung verweist  
`desc$`        Beschreibung der Verknüpfung (falls vorhanden)

**BEISPIEL**

```
p$ = GetShortcutPath("test.lnk")
```

Der obige Code gibt den vollständigen Pfad der Verknüpfung "test.lnk" zurück.

**56.3 ReadRegistryKey****BEZEICHNUNG**

`ReadRegistryKey` – liest einen Schlüssel aus der Registry (V4.5)

**ÜBERSICHT**

```
value = ReadRegistryKey(base, key$)
```

**PLATTFORMEN**

Nur Microsoft Windows

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um einen Schlüssel aus der Windows-Registry zu lesen. Sie müssen sowohl den Basisbaum als auch den zu lesenden Schlüssel angeben. Der Basisbaum kann eine der folgenden Konstanten sein:

```
#HKEY_CLASSES_ROOT
#HKEY_CURRENT_CONFIG
#HKEY_LOCAL_MACHINE
#HKEY_USERS
#HKEY_CURRENT_USER
```

Der Rückgabewert ist eine Zahl, falls der Registry-Schlüssel eine Zahl enthält. Wenn der Registry-Schlüssel eine Zeichenfolge oder binäre Daten enthält, werden Sie eine Zeichenkette als Rückgabewert erhalten. Hollywood-Zeichenketten sind in der Lage, binäre Daten zu enthalten, weil sie Nullzeichen in ihnen erlauben.

**EINGABEN**

`base`            Eine der Basisbaumkonstanten von oben  
`key$`            abfragen des Registry-Schlüssels

**RÜCKGABEWERTE**

`value`           Wert des angegebenen Registry-Schlüssels; wird entweder eine Zahl sein oder eine Zeichenfolge

**BEISPIEL**

```
program_files$ = ReadRegistryKey(#HKEY_LOCAL_MACHINE,
    "Software/Microsoft/Windows/CurrentVersion/ProgramFilesDir")
```

Der obige Code liest den Standardspeicherort der Programme unter Windows aus der Registry. Auf einem deutschen Windows-System wird dies in der Regel "C:/Programme" sein.

## 56.4 WriteRegistryKey

### BEZEICHNUNG

WriteRegistryKey – schreibt einen Schlüssel in die Registry (V4.5)

### ÜBERSICHT

WriteRegistryKey(base, key\$, value)

### PLATTFORMEN

Nur Microsoft Windows

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um einen Schlüssel in die Windows-Registry zu schreiben. Sie müssen den Basisbaum **base**, den Schlüssel **key** und den Wert **value** angeben, der in den angegebenen Schlüssel geschrieben werden soll. Wenn dieser Schlüssel nicht vorhanden ist, wird er durch diesen Befehl erzeugt. Der Basisbaum kann eine der folgenden Konstanten sein:

```
#HKEY_CLASSES_ROOT  
#HKEY_CURRENT_CONFIG  
#HKEY_LOCAL_MACHINE  
#HKEY_USERS  
#HKEY_CURRENT_USER
```

Der Wert für den Schlüssel kann entweder eine Zahl oder eine Zeichenfolge sein. Sie können auch binäre Daten in die Registrierung schreiben, indem Sie eine Zeichenfolge übergeben. Hollywoods Zeichenfolgen sind in der Lage, beliebige binäre Daten zu enthalten, weil Sie Nullzeichen in ihnen erlauben. Unter normalen Umständen sollte jedoch das Schreiben von Zahlen oder normale Zeichenfolgen in die Registry ausreichend sein.

### EINGABEN

<b>base</b>	eine der Basisbaumkonstanten von oben
<b>key\$</b>	den Registry-Schlüssel, der erstellt/geändert wird
<b>value</b>	Wert, der gesetzt wird; kann entweder eine Zahl oder eine Zeichenfolge sein



## 57 Zeichenkettenbibliothek

### 57.1 AddStr

#### BEZEICHNUNG

AddStr – hängt eine Teilzeichenkette an eine Zeichenkette an / VERALTET

#### ÜBERSICHT

```
var$ = AddStr(string1$, string2$)
```

#### BESCHREIBUNG

Dieser Befehl ist veraltet und nur noch aus Kompatibilitätsgründen hier aufgeführt. Ab Hollywood 2.0 sollten Sie den **Verkettungsoperator** `..` verwenden, um zwei Zeichenketten zu verknüpfen.

Hängt `string2$` an das Ende von `string1$` an und gibt die neue Zeichenkette in der Variable `var$` zurück.

Siehe auch `InsertStr()` und `ReplaceStr()`.

#### EINGABEN

`string1$` bestehende Zeichenkette

`string2$` anzuhängende Zeichenkette

#### RÜCKGABEWERTE

`var$` aus den beiden Quellen gebildete Zeichenkette

#### BEISPIEL

```
test$ = AddStr("Hello", " World!")
```

```
Print(test$)
```

Das gibt "Hello World!" aus.

### 57.2 ArrayToStr

#### BEZEICHNUNG

ArrayToStr – wandelt ein Array mit Codepunktwerten in eine Zeichenkette um (V6.0)

#### ÜBERSICHT

```
s$ = ArrayToStr(t[, encoding])
```

#### BESCHREIBUNG

Dieser Befehl liest alle Codepunktwerte in der Tabelle `t`, fügt sie in einer Zeichenkette zusammen und gibt diese Zeichenfolge zurück. `ArrayToStr()` wird das Lesen von Werten aus `t` stoppen, wenn es einen Codepunktwert von 0 (Zeichenketten-Terminator) oder das Ende der Tabelle antrifft.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` gesetzt wurde. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Um Zeichenketten in Arrays zu konvertieren, können Sie den Befehl `StrToArray()` nutzen. Siehe [Abschnitt 57.58 \[StrToArray\]](#), [Seite 1297](#), für Details.

#### EINGABEN

`t` eine Tabelle mit den ASCII-Werten einer beliebigen Anzahl von Codepunkt-werten

`encoding` optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

#### RÜCKGABEWERTE

`s$` Zeichenkette aus den Codepunktwerten in der Tabelle

#### BEISPIEL

```
s$ = ArrayToStr({'H', 'e', 'l', 'l', 'o'})
DebugPrint(s$)
Gibt "Hello" aus.
```

## 57.3 Asc

#### BEZEICHNUNG

Asc – gibt den Codepunkt des Zeichens in der Zeichenkette zurück

#### ÜBERSICHT

```
var = Asc(string$, pos, encoding)
```

#### BESCHREIBUNG

Gibt den Codepunktwert des Zeichens am Index `pos` in der Zeichenkette `string$` zurück. Der `pos`-Index muss in Zeichen und nicht in Bytes angegeben werden. Wenn Sie das Argument `pos` weglassen, wird der Codepunkt des ersten Zeichens zurückgegeben.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Wenn Sie mit reinen Bytes anstelle von Codepunkten arbeiten möchten, können Sie entweder den Parameter `#ENCODING_RAW` eingeben oder den Befehl `ByteAsc()` verwenden. Siehe [Abschnitt 57.6 \[ByteAsc\]](#), [Seite 1256](#), für Details.

Siehe auch `Chr()` und `ByteChr()`.

#### EINGABEN

`string$` Eingabezeichenfolge

`pos` optional: Index des Zeichens, dessen Codepunkt ermittelt werden soll (voreingestellt ist 0) (V7.0)

`encoding` optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

#### RÜCKGABEWERTE

`var` Codepunktwert

**BEISPIEL**

```
result = Asc("A")
Print(result)
```

Dies wird "65" ausgegeben, da es der Codepunktwert von "A" ist.

## 57.4 Base64Str

**BEZEICHNUNG**

Base64Str – kodiert oder dekodiert Base64-Daten (V6.0)

**ÜBERSICHT**

```
daten$ = Base64Str(s$, decode)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um beliebige Daten in das Base64-Format zu kodieren oder eine Base64 formatierte Zeichenkette zurück in seine ursprünglich binäre Daten zu dekodieren. Das zweite Argument gibt an, ob dieser Befehl die Daten kodieren oder entschlüsseln soll.

**EINGABEN**

s\$	Daten zum Kodieren oder Dekodieren
decode	True um Base64-Daten zu entschlüsseln, False um Daten in das Base64-Format zu kodieren

**RÜCKGABEWERTE**

daten\$	kodierte oder dekodierte Daten
---------	--------------------------------

## 57.5 BinStr

**BEZEICHNUNG**

BinStr – konvertiert einen Wert in eine binäre formatierte Zeichenfolge (V2.0)

**ÜBERSICHT**

```
bin$ = BinStr(val[, length])
```

**BESCHREIBUNG**

Dieser Befehl wandelt den angegebenen Wert von `val` ins Binär-Format (Basis 2) um und gibt ihn als Zeichenkette zurück. Das optionale Argument `length` ermöglicht es Ihnen, wie viele Bits in die Zeichenkette gesetzt werden. Dies kann `#INTEGER` für 32 Bits, `#SHORT` für 16 Bits und `#BYTE` für 8 Bit betragen. Standardmäßig wird `#INTEGER` verwendet.

**EINGABEN**

val	Wert zum konvertieren
length	optional: wie viele Bits werden konvertiert (muss <code>#INTEGER</code> , <code>#SHORT</code> oder <code>#BYTE</code> sein) (V3.0)

**RÜCKGABEWERTE**

bin\$	Binär-Format von <code>val</code>
-------	-----------------------------------

**BEISPIEL**

```
a$ = BinStr(255, #BYTE)
```

Dies wird die Zeichenkette "11111111" zurückgeben.

## 57.6 ByteAsc

**BEZEICHNUNG**

ByteAsc – gibt ein einzelnes Byte aus einer Zeichenkette zurück (V7.1)

**ÜBERSICHT**

```
v = ByteAsc(string$, pos)
```

**BESCHREIBUNG**

Gibt den Wert des Bytes an dem in `pos` angegebenen Zeichenkettenindex zurück. Der Rückgabewert `v` liegt im Bereich von 0 bis 255.

Da Zeichenketten von Hollywood nicht nur Text sondern auch Binärdaten enthalten können, eignet sich dieser Befehl für den Zugriff auf Binärdaten bei bestimmten Indizes. Der normale Befehl `Asc()` hingegen ist besser geeignet, um mit Textzeichenketten umzugehen, da er voreingestellt im Unicode-Modus arbeitet. Das bedeutet, dass die Eingabezeichenkette in gültigem UTF-8 sein muss.

Siehe auch `Chr()` und `ByteChr()`.

**EINGABEN**

`string$`     Zeichenkette

`pos`             optional: Index des zurückzugebenden Bytes (voreingestellt ist 0)

**RÜCKGABEWERTE**

`v`                Zeichenketten-Byte am angegebenen Index

## 57.7 ByteChr

**BEZEICHNUNG**

ByteChr – konvertiert ein einzelnes Byte in eine Zeichenkette (V7.1)

**ÜBERSICHT**

```
s$ = ByteChr(v)
```

**BESCHREIBUNG**

Konvertiert den in `v` angegebenen Bytewert in eine Zeichenfolge. `v` muss im Bereich von 0 bis 255 liegen.

Da Zeichenketten von Hollywood nicht nur Text sondern auch rohe Binärdaten enthalten können, eignet sich dieser Befehl zum Zusammensetzen von Zeichenketten mit nicht codierten Byte-Daten. Der normale Befehl `Chr()` hingegen ist besser geeignet, um mit Textzeichenfolgen umzugehen, da er voreingestellt im Unicode-Modus arbeitet. Das bedeutet, dass er voreingestellt Zeichenketten im Format UTF-8 erstellt. Wird z.B. 255 übergeben, führt dies aufgrund der UTF-8-Codierungsregeln zu einer Zeichenfolge mit zwei Bytes.



Siehe auch `Asc()` und `ByteAsc()`.

#### EINGABEN

`v` Byte, das in eine Zeichenkette konvertiert werden soll (im Bereich von 0 bis 255).

#### RÜCKGABEWERTE

`s$` Zeichenkette, die zurückgeben wird

## 57.8 ByteLen

#### BEZEICHNUNG

`ByteLen` – gibt die Länge der Zeichenfolge in Bytes zurück (V7.0)

#### ÜBERSICHT

```
len = ByteLen(s$)
```

#### BESCHREIBUNG

Dieser Befehl gibt die Länge der Zeichenkette `s$` in Bytes zurück. Wenn Sie die Zeichenfolgenlänge in Zeichen ermitteln wollen, verwenden Sie stattdessen `StrLen()`. Siehe [Abschnitt 57.56 \[StrLen\]](#), [Seite 1296](#), für Details.

In der UTF-8-Zeichencodierung kann ein einzelnes Zeichen einen Speicherplatz von bis zu 4 Bytes benötigen. In der Zeichencodierung ISO 8859-1 gibt es keinen Unterschied zwischen Byte und Zeichengrößen.

#### EINGABEN

`s$` Eingabezeichenfolge

#### RÜCKGABEWERTE

`len` Länge der Eingabezeichenfolge in Bytes

#### BEISPIEL

```
len = ByteLen("äöü")
Print(len)
```

Wenn Hollywood im Unicode-Modus ist, wird `ByteLen` 6 zurückgeben, da jedes der Zeichen zwei Bytes in der UTF-8-Zeichencodierung benötigt. Im Modus ISO 8859-1 gibt es keinen Unterschied zwischen Zeichen und Bytes, was bedeutet, dass der obige Code 3 zurückgibt.

## 57.9 ByteOffset

#### BEZEICHNUNG

`ByteOffset` – konvertiert Zeichen nach Byte-Versatz (V7.0)

#### ÜBERSICHT

```
boff = ByteOffset(s$, coff[, encoding])
```

**BESCHREIBUNG**

Dieser Befehl gibt den Byte-Versatz des Zeichens zurück, welcher innerhalb der Zeichenkette `s$` bei dem in `coff` angegebenen Versatz liegt. Dieser Versatz ist in Zeichen, welcher bei 0 beginnt.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

In der Zeichencodierung UTF-8 kann ein einzelnes Zeichen einen Speicherplatz von bis zu 4 Bytes benötigen. In der Zeichencodierung ISO 8859-1 hingegen gibt es keinen Unterschied zwischen Byte und Zeichengrößen. Daher ist es nicht sinnvoll, diesen Befehl mit der Zeichencodierung `#ENCODING_ISO8859_1` aufzurufen.

Um einen Byte-Versatz in einen Zeichen-Versatz umzuwandeln, verwenden Sie den Befehl `CharOffset()`. Siehe [Abschnitt 57.12 \[CharOffset\]](#), [Seite 1260](#), für Details.

**EINGABEN**

<code>s\$</code>	Eingabezeichenfolge
<code>coff</code>	Zeichen-Versatz, der einem Byte-Versatz zugeordnet wird (beginnt bei 0)
<code>encoding</code>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

**RÜCKGABEWERTE**

<code>boff</code>	Byte-Versatz des angegebenen Zeichens
-------------------	---------------------------------------

**BEISPIEL**

```
boff = ByteOffset("äöü", 2)
Print(boff)
```

Wenn Hollywood im Unicode-Modus ist, wird dies 4 zurückgeben, da die beiden Zeichen vor dem "ü" jeweils 2 Byte im UTF-8-Code besitzen. In ISO 8859-1 gibt es keinen Unterschied zwischen Zeichen und Bytes, also wird in diesem Fall 2 zurückgegeben.

## 57.10 ByteStrStr

**BEZEICHNUNG**

ByteStrStr – konvertiert den Wert in rohe binäre Bytes (V8.0)

**ÜBERSICHT**

```
r$ = ByteStrStr(v[, type, le])
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um einen numerischen Wert im Argument `v` in rohe binäre Bytes umzuwandeln, die als Zeichenkette in `r$` zurückgegeben werden. Die Anzahl der Bytes, die in die zurückgegebene Zeichenkette geschrieben werden, hängt von dem Typ ab, den Sie im Argument `type` übergeben. Folgende Typen werden derzeit unterstützt:

<code>#BYTE:</code>	Speichert einen 8-Bit-Wert (1 Byte) in der Rückgabezeichenkette.
---------------------	--

- #SHORT:** Speichert einen 16-Bit-Wert (2 Bytes) in der Rückgabezeichenkette.
- #INTEGER:** Speichert einen 32-Bit-Wert (4 Bytes) in der Rückgabezeichenkette. Dies ist die Standardeinstellung.
- #FLOAT:** Speichert einen 32-Bit-Fließkommawert (4 Bytes) in der Rückgabezeichenkette.
- #DOUBLE:** Speichert einen 64-Bit-Fließkommawert (8 Bytes) in der Rückgabezeichenkette.

Für alle Multibyte-Typen, d.h. alle Typen außer **#BYTE**, können Sie das zusätzliche Argument **le** verwenden, um die Reihenfolge anzugeben, in der die Bytes in der Rückgabezeichenkette **r\$** gespeichert werden sollen. Wenn Sie **le** auf **True** setzen, werden die Bytes in der Little-Endian-Reihenfolge (LSB zuerst) gespeichert. Andernfalls werden die Bytes in der Big-Endian-Reihenfolge (MSB zuerst) gespeichert. Big Endian ist auch die Standardeinstellung.

Wenn Sie rohe Bytes in einen Wert konvertieren müssen, können Sie den Befehl **ByteVal()** verwenden. Siehe [Abschnitt 57.11 \[ByteVal\]](#), [Seite 1259](#), für Details.

#### EINGABEN

- v** numerischer Wert zum Konvertieren in Binärdaten
- type** optional: Typ des in Zeichenkette zu speichernden Werts (Standardeinstellung ist **#INTEGER**)
- le** optional: ob die Little-Endian-Bytereihenfolge verwendet werden soll (**True**) oder nicht (**False**) (Standardeinstellung ist **False**)

#### RÜCKGABEWERTE

- r\$** resultierende Zeichenkette

## 57.11 ByteVal

#### BEZEICHNUNG

ByteVal – konvertiert rohe binäre Bytes als numerischen Wert (V8.0)

#### ÜBERSICHT

```
v = ByteVal(s$, type, le)
```

#### BESCHREIBUNG

Mit diesem Befehl können rohe binäre Bytes aus der in **s\$** übergebenen Zeichenkette in einen numerischen Wert umgewandelt werden. Die Anzahl der Bytes, die aus der Zeichenkette **s\$** gelesen werden, hängt von dem Typ ab, den Sie im Argument **type** übergeben. Folgende Typen werden derzeit unterstützt:

- #BYTE:** Liest einen 8-Bit-Wert (1 Byte) aus der Zeichenkette und gibt ihn zurück.
- #SHORT:** Liest einen 16-Bit-Wert (2 Bytes) aus der Zeichenkette und gibt ihn zurück.
- #INTEGER:** Liest einen 32-Bit-Wert (4 Bytes) aus der Zeichenkette und gibt ihn zurück. Dies ist die Voreinstellung.

**#FLOAT:** Liest einen 32-Bit-Fließkommawert (4 Bytes) aus der Zeichenkette und gibt ihn zurück.

**#DOUBLE:** Liest einen 64-Bit-Fließkommawert (8 Bytes) aus der Zeichenkette und gibt ihn zurück.

Für alle Multibyte-Typen, d.h. alle Typen außer **#BYTE**, können Sie das zusätzliche Argument **le** verwenden, um die Reihenfolge anzugeben, in der die Bytes aus **s\$** gelesen werden sollen. Wenn Sie **le** auf **True** setzen, werden die Bytes in der Little-Endian-Reihenfolge (LSB zuerst) gelesen. Andernfalls werden die Bytes in der Big-Endian-Reihenfolge (MSB zuerst) gelesen. Big Endian ist auch Voreingestellt.

Beachten Sie, dass das Ergebnis für alle Ganzzahlen-Typen immer vorzeichenlos ist. Sie können den Befehl **Cast()** verwenden, wenn Sie das Ergebnis in einen signierten Typ (mit Vorzeichen) umwandeln müssen. Siehe [Abschnitt 35.12 \[Cast\]](#), [Seite 720](#), für Details.

Siehe auch [ByteStrStr\(\)](#).

## EINGABEN

**s\$** Zeichenkette zum Lesen von Daten

**type** optional: Typ des zu lesenden Wertes (Standardeinstellung **#INTEGER**)

**le** optional: ob die Little-Endian-Bytereihenfolge verwendet werden soll (**True**) oder nicht (**False**) (Standardeinstellung ist **False**)

## RÜCKGABEWERTE

**v** resultierender Wert

# 57.12 CharOffset

## BEZEICHNUNG

**CharOffset** – konvertiert Byte nach Zeichen-Versatz (V7.0)

## ÜBERSICHT

**coff** = **CharOffset(s\$, boff[, encoding])**

## BESCHREIBUNG

Dieser Befehl gibt den Versatz des Zeichens zurück, welcher innerhalb der Zeichenkette **s\$** bei dem in **boff** angegebenen Versatz liegt. Dieser Versatz ist in Byte, welcher bei 0 beginnt.

Mit dem optionalen Parameter **encoding** kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit **SetDefaultEncoding()** eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

In der Zeichencodierung UTF-8 kann ein einzelnes Zeichen einen Speicherplatz von bis zu 4 Bytes benötigen. In der Zeichencodierung ISO 8859-1 hingegen gibt es keinen Unterschied zwischen Byte und Zeichengrößen. Daher ist es nicht sinnvoll, diesen Befehl mit der Zeichencodierung **#ENCODING\_ISO8859\_1** aufzurufen.

Um einen Zeichen-Versatz in einen Byte-Versatz umzuwandeln, verwenden Sie den Befehl **ByteOffset()**. Siehe [Abschnitt 57.9 \[ByteOffset\]](#), [Seite 1257](#), für Details.

**EINGABEN**

<code>s\$</code>	Eingabezeichenfolge
<code>boff</code>	Byte-Versatz, der einem Zeichen-Versatz zugeordnet wird (beginnt bei 0)
<code>encoding</code>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

**RÜCKGABEWERTE**

<code>coff</code>	Zeichen-Versatz des angegebenen Zeichens
-------------------	--

**BEISPIEL**

```
coff = CharOffset("äöü", 2)
Print(coff)
```

Wenn Hollywood im Unicode-Modus ist, wird dies 1 zurückgeben, da das Zeichen "ä" 2 Bytes im UTF-8-Code besitzt. In ISO 8859-1 gibt es keinen Unterschied zwischen Zeichen und Bytes, also wird in diesem Fall 1 zurückgegeben.

## 57.13 CharWidth

**BEZEICHNUNG**

`CharWidth` – ermittelt die Byte-Breite eines Zeichens (V7.0)

**ÜBERSICHT**

```
w = CharWidth(s$[, pos, encoding])
```

**BESCHREIBUNG**

Berechnet die Byte-Breite des Zeichens an der Position `pos` innerhalb der Zeichenkette `s$`. Die Position muss in Zeichen und nicht in Bytes angegeben werden. Das Argument `pos` ist optional und ist standardmäßig auf 0 gesetzt (der Anfang der Zeichenkette), wenn es weggelassen wird.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

In der Zeichencodierung UTF-8 kann ein einzelnes Zeichen einen Speicherplatz von bis zu 4 Bytes benötigen. In der Zeichencodierung ISO 8859-1 hingegen gibt es keinen Unterschied zwischen Byte und Zeichengrößen. Daher ist es nicht sinnvoll, diesen Befehl mit der Zeichencodierung `#ENCODING_ISO8859_1` aufzurufen.

**EINGABEN**

<code>s\$</code>	Eingabezeichenfolge
<code>pos</code>	optional: Index des Zeichens in Zeichen, dessen Breite berechnet werden soll
<code>encoding</code>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

**RÜCKGABEWERTE**

<code>w</code>	Byte-Breite des Zeichens an der angegebenen Position
----------------	--

**BEISPIEL**

```
w = CharWidth("ä")
Print(w)
```

Wenn Hollywood im Unicode-Modus ist, wird dies 2 zurückgeben, da das Zeichen "ä" 2 Byte im UTF-8-Code besitzt. In ISO 8859-1 gibt es keinen Unterschied zwischen Zeichen und Bytes, also wird in diesem Fall 1 zurückgegeben.

**57.14 Chr****BEZEICHNUNG**

Chr – konvertiert einen Codepunktwert in eine Zeichenkette

**ÜBERSICHT**

```
var$ = Chr(value[, encoding])
```

**BESCHREIBUNG**

Konvertiert den in `value` enthaltenen Codepunktwert in die Zeichenkette `var$`.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` gesetzt wurde. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Wenn Sie mit reinen Bytes anstelle von Codepunkten arbeiten möchten, können Sie entweder den Parameter `#ENCODING_RAW` eingeben oder den Befehl `ByteChr()` verwenden. Siehe [Abschnitt 57.7 \[ByteChr\]](#), [Seite 1256](#), für Details.

Siehe auch `Asc()`, `ByteAsc()` und `ByteChr()`.

**EINGABEN**

<code>value</code>	Codepunktwert
<code>encoding</code>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

**RÜCKGABEWERTE**

<code>var\$</code>	Zeichenkette, welches das Ergebnis enthält
--------------------	--

**BEISPIEL**

```
test$=Chr(65)
Print(test$)
```

Das wird "A" auf den Bildschirm ausgeben, da 65 sowohl in ASCII- als auch in Unicode-Zeichencodierungen der Codepunktwert für "A" ist.

**57.15 CompareStr****BEZEICHNUNG**

CompareStr – vergleicht zwei Zeichenketten lexikalisch (V7.0)

**ÜBERSICHT**

```
r = CompareStr(s1$, s2$[, casesen, encoding])
```

**BESCHREIBUNG**

Dieser Befehl vergleicht `s1$` mit `s2$` und gibt zurück, wie die beiden Zeichenketten zueinander stehen. Wenn `s1$` lexikalisch kleiner ist als `s2$`, wird -1 zurückgegeben. Wenn `s1$` lexikalisch größer ist als `s2$`, wird 1 zurückgegeben. Sind diese beiden Zeichenketten gleich, ist der Rückgabewert 0.

Das optionale Argument `casesen` kann verwendet werden, um festzulegen, ob bei den Zeichenketten die Groß-/Kleinschreibung beachtet werden soll oder nicht. Dies ist standardmäßig der globale Standardmodus, bei dem die Groß-/Kleinschreibung beachtet wird, der mit `IgnoreCase()` festgelegt wurde. Siehe [Abschnitt 57.26 \[IgnoreCase\]](#), [Seite 1271](#), für Details.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

**EINGABEN**

<code>s1\$</code>	erste Zeichenkette zum Vergleichen
<code>s2\$</code>	zweite Zeichenkette zum Vergleichen
<code>casesen</code>	optional: <code>True</code> für einen Vergleich mit Groß-/Kleinschreibung, sonst <code>FALSE</code> ; voreingestellt ist <code>True</code> oder welcher Standardwert auch immer mit dem Befehl <code>IgnoreCase()</code> eingestellt wurde
<code>encoding</code>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

**RÜCKGABEWERTE**

<code>r</code>	wie die beiden Zeichenketten lexikalisch zueinander stehen
----------------	--

**BEISPIEL**

```
DebugPrint(CompareStr("z", "a"))
```

Der obige Code gibt 1 aus, da "z" lexikalisch größer ist als "a".

## 57.16 ConvertStr

**BEZEICHNUNG**

`ConvertStr` – wandelt die Zeichencodierungen um (V7.0)

**ÜBERSICHT**

```
c$ = ConvertStr(s$, inencoding, outencoding)
```

**BESCHREIBUNG**

Konvertiert `s$` aus der Zeichencodierung, die durch `inencoding` angegeben ist, in die Zeichencodierung, die durch `outencoding` angegeben ist, und gibt sie zurück. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für eine Liste gültiger Codierungen.

**EINGABEN**

<code>s\$</code>	Eingabezeichenfolge
------------------	---------------------

`inencoding`  
     Quellzeichencodierung

`outencoding`  
     Zielzeichencodierung

## RÜCKGABEWERTE

`c$`           konvertierte Zeichenfolge in die neue Zeichencodierung

## 57.17 CountStr

### BEZEICHNUNG

`CountStr` – zählt, wie oft die Teilzeichenkette in der Zeichenkette vorkommt (V4.5)

### ÜBERSICHT

```
n = CountStr(s$, sub$[, casesen, startpos, encoding])
```

### BESCHREIBUNG

Dieser Befehl zählt, wie oft `sub$` innerhalb von `s$` vorkommt. Das optionale Argument `casesen` gibt an, ob die Großkleinschreibung unterschieden wird oder nicht. Außerdem können Sie das Argument `startpos` verwenden, um einen Versatz in `s$` anzugeben, an dem `CountStr()` mit dem Zählen beginnen soll. Dieser Versatz ist in Zeichen, nicht in Bytes. Position 0 bedeutet den Anfang der Zeichenkette.

Der Parameter `casesen` verwendet standardmäßig den globalen Standardmodus, bei dem die Groß-/Kleinschreibung beachtet wird, der mit dem Befehl `IgnoreCase()` festgelegt wurde. Siehe [Abschnitt 57.26 \[IgnoreCase\]](#), [Seite 1271](#), für Details.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch [StrLen\(\)](#).

### EINGABEN

<code>s\$</code>	Quellenzeichenkette
<code>sub\$</code>	die Teilzeichenkette, welche in <code>s\$</code> gezählt wird
<code>casesen</code>	optional: wenn Sie hier <code>True</code> setzen, dann wird die Groß- und Kleinschreibung beim Suchen/Zählen berücksichtigt; voreingestellt ist <code>True</code> oder welcher Standardwert auch immer mit dem Befehl <code>IgnoreCase()</code> eingestellt wurde
<code>startpos</code>	optional: Zeichenversatz innerhalb von <code>s\$</code> , um die Suche zu starten (standardmäßig auf 0)
<code>encoding</code>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

## RÜCKGABEWERTE

`n`           Anzahl Vorkommen von `sub$` in `s$`



**BEISPIEL**

```
ret = CountStr("What is that on your head? Is that a new hat? " ..
    "You have not had that on our last chat!", "hat")
```

Dieser Code gibt 6 zurück, weil "hat" sechs Mal in der Quellenzeichenkette vorkommt.

**57.18 CRC32Str****BEZEICHNUNG**

CRC32Str – berechnet die CRC32-Prüfsumme von einer Zeichenkette (V5.0)

**ÜBERSICHT**

```
sum = CRC32Str(s$)
```

**BESCHREIBUNG**

Dieser Befehl berechnet die CRC32-Prüfsumme der Zeichenfolge in `s$` und gibt sie zurück. Beachten Sie, dass in Hollywood Zeichenketten auch binäre Daten enthalten können, so dass Sie diesen Befehl auch mit Nicht-Text-Zeichenfolgen verwenden können.

Wenn Sie die CRC32-Prüfsumme einer Datei berechnen, verwenden Sie stattdessen den `CRC32()` Befehl.

Siehe auch `MD5Str()` und `MD5()`.

**EINGABEN**

`s$`            Zeichenkette, dessen Prüfsumme Sie berechnen wollen

**RÜCKGABEWERTE**

`sum`            CRC32-Prüfsumme der Zeichenkette

**57.19 EmptyStr****BEZEICHNUNG**

EmptyStr – überprüft, ob eine Zeichenkette leer ist (V7.1)

**ÜBERSICHT**

```
bool = EmptyStr(s$)
```

**BESCHREIBUNG**

Dieser Befehl gibt `True` zurück, wenn die Zeichenkette leer ist, d.h. wenn er nur leere Zeichen enthält. Die folgenden Zeichen sind leere Zeichen: Leerzeichen, Seitenvorschub ("`\f`"), neue Zeile ("`\n`"), Wagenrücklauf ("`\r`"), horizontaler Tabulator ("`\t`") und vertikaler Tabulator ("`\v`").

**EINGABEN**

`s$`            die zu überprüfende Zeichenkette

**RÜCKGABEWERTE**

`bool`            `True`, wenn die Eingabezeichenkette nur leere Zeichen enthält, andernfalls `False`.

## 57.20 EndsWith

### BEZEICHNUNG

EndsWith – überprüft, ob die Zeichenkette mit der Teilzeichenkette endet (V7.1)

### ÜBERSICHT

```
bool = EndsWith(s$, substr$[, casesen, encoding])
```

### BESCHREIBUNG

Mit diesem Befehl kann überprüft werden, ob **s\$** mit der in **substr\$** angegebenen Teilzeichenkette endet. Ist dies der Fall, wird **True** zurückgegeben, ansonsten **False**. Wenn das optionale Argument **casesen** auf **False** gesetzt ist, müssen die Zeichen in Groß-/Kleinschreibung nicht übereinstimmen. **casesen** verwendet standardmäßig den globalen Standardmodus, bei dem die Groß-/Kleinschreibung beachtet wird, der mit **IgnoreCase()** festgelegt wurde. Siehe [Abschnitt 57.26 \[IgnoreCase\]](#), [Seite 1271](#), für Details.

Mit dem optionalen Parameter **encoding** kann die zu verwendende Zeichencodierung eingestellt werden. Voreingestellt ist die Standardeinstellung für die Zeichenkettenkodierung, die mit **SetDefaultEncoding()** festgelegt wurde. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch **StartsWith()**, **LeftStr()**, **MidStr()**, **RightStr()**, **FindStr()**, **ReverseFindStr()**, **UnleftStr()**, **UnmidStr()** und **UnrightStr()**.

### EINGABEN

<b>s\$</b>	Eingabezeichenkette
<b>substr\$</b>	Zeichenkette, die mit <b>s\$</b> verglichen wird
<b>casesen</b>	optional: ob die Groß-/Kleinschreibung beim Vergleichen aktiviert werden soll ( <b>True</b> ) oder nicht ( <b>False</b> ) (voreingestellt ist <b>True</b> , d.h. Groß-/Kleinschreibung wird berücksichtigt oder welcher Standardwert auch immer mit dem Befehl <b>IgnoreCase()</b> festgelegt wurde)
<b>encoding</b>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

### RÜCKGABEWERTE

<b>bool</b>	boolescher Wert, der Erfolg ( <b>True</b> ) oder Misserfolg ( <b>False</b> ) anzeigt
-------------	--

## 57.21 Eval

### BEZEICHNUNG

Eval – wertet den Zeichenkettenausdruck aus (V5.0)

### ÜBERSICHT

```
val = Eval(expr$[, table])
```

### BESCHREIBUNG

**Eval()** wertet den numerischen Ausdruck in **expr\$** aus und gibt das Ergebnis als Zahl zurück. Die Zeichenfolge in **expr\$** kann alle Operatoren beinhalten, die von Hollywood unterstützt wird. Ausnahme ist der Verkettungsoperator für Zeichenfolgen, weil dieser Operator eine Zeichenfolge erfordert, während der Befehl **Eval()** nur mit Zahlen

funktioniert. Sie können auch bestimmte Teilausdrücke priorisieren, in dem Sie Klammern verwenden. `Eval()` verwendet die gleichen Prioritäten wie Hollywood selbst. Siehe [Abschnitt 9.7 \[Prioritäten der Operatoren\]](#), [Seite 116](#), für eine Liste aller Hollywood-Operatoren und ihre Prioritäten.

Zahlen innerhalb `expr$` können dezimal oder hexadezimal angegeben werden. Wenn Sie das Hexadezimalformat verwenden, müssen Sie der Nummer als Präfix das Dollarzeichen (\$) verwenden.

Es ist auch möglich, Variablen im Ausdruck zu verwenden. Ihre Skript-Variablen stehen jedoch `Eval()` nicht automatisch zur Verfügung. Um Ihre Skriptvariablen für `Eval()` verfügbar zu machen, müssen Sie den Tag `MapVariables` im optionalen Tabellenargument auf `True` setzen. Alternativ können Sie auch private Variablen für `Eval()` definieren, indem Sie den Tag `Variables` im optionalen Tabellenargument setzen. Wenn sowohl Skriptvariablen als auch private Variablen verwendet werden, haben die privaten Vorrang.

Hier ist eine Liste aller Tags, die derzeit von der optionalen Tabelle unterstützt wird:

#### **Variables:**

Dieser Tag wird verwendet, um `Eval()` eine Reihe von Variablen zu übergeben, auf die Sie in Ihrem Ausdruck zugreifen können. Sie müssen hier ein Array übergeben, die aus einer Anzahl von **Name** und **Value**-Paaren besteht. In **Name** muss der gewünschte Namen für die Variable und in **Value** muss der enthalten Wert sein, mit der die Variable initialisiert werden soll. Die Semantik des Variablennamen muss die gleiche wie in Hollywood sein, das heißt es dürfen nur alphanumerische Zeichen und nur die Sonderzeichen "\$", "!" und "-" verwendet werden. Das erste Zeichen darf keine Zahl sein. Siehe unten für ein Beispiel.

#### **MapVariables:**

Wenn dieser Tag auf `True` gesetzt ist, berücksichtigt `Eval()` bei der Auswertung des Ausdrucks auch normale Hollywood-Variablen. Beachten Sie, dass die Variablen nur numerische Variablen sein dürfen und die im Tag `Variables` (siehe oben) deklarierten Vorrang haben. Die Voreinstellung ist `False`. (V9.0)

#### **NoDeclare:**

Wenn dieser Tag auf `True` gesetzt ist, werden alle nicht deklarierten Variablen im Ausdruck als 0 behandelt. Standardmäßig gibt `Eval()` einen Fehler aus, wenn versucht wird, auf eine nicht deklarierte Variable zuzugreifen. Die Voreinstellung ist `False`. (V9.0)

### **EINGABEN**

`expr$`      der zu berechnende Ausdruck  
`table`      optional: Tabelle mit weiteren Optionen

### **RÜCKGABEWERTE**

`val`          Resultat

### **BEISPIEL**

```
v = Eval("5*(6+1)")
```

Gibt 35 zurück.

```
v = Eval("var1*(var2+1)", {Variables = {{Name = "var1", Value = 5},
                                     {Name = "var2", Value = 6}}})
```

Der obige Code macht das gleiche wie das erste Beispiel, verwendet aber Variablen anstelle von direkten Zahlen.

```
var1 = 5
var2 = 6
v = Eval("var1*(var2+1)", {MapVariables = True})
```

Der obige Code macht dasselbe wie die ersten beiden Beispiele, ordnet jetzt jedoch die Hollywood-Variablen `var1` und `var2` dem Variablenraum von `Eval()` zu.

## 57.22 FindStr

### BEZEICHNUNG

FindStr – findet eine Teilzeichenkette in einer Zeichenkette

### ÜBERSICHT

```
pos = FindStr(string$, substring$[, casesensitive, startpos, encoding])
```

### BESCHREIBUNG

Sucht in `string$` nach `substring$` und gibt die Position zurück. Diese Position wird in Zeichen und nicht in Bytes zurückgegeben, beginnend bei Position 0 für das erste Zeichen. Wenn `substring$` nicht gefunden werden kann, wird -1 zurückgegeben. Das optionale Argument `casesensitive` erlaubt Ihnen anzugeben, ob Groß- und Kleinschreibung beachtet werden soll (`True`) oder nicht (`False`). Dies ist standardmäßig der globale Standardmodus, bei dem die Groß-/Kleinschreibung beachtet wird, der mit `IgnoreCase()` festgelegt wurde. Siehe [Abschnitt 57.26 \[IgnoreCase\]](#), [Seite 1271](#), für Details.

Ab Hollywood 4.5 können Sie auch eine Startposition für die Suche in dem optionalen Argument `startpos` angeben. Dieser Versatz ist in Zeichen und nicht in Bytes. Position 0 bedeutet den Anfang der Zeichenkette. Dies ist auch die Voreinstellung.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `ReverseFindStr()`, `LeftStr()`, `MidStr()`, `RightStr()`, `EndsWith()`, `StartsWith()`, `ReverseFindStr()`, `UnleftStr()`, `UnmidStr()` und `UnrightStr()`.

### EINGABEN

`string$`    Zeichenfolge, in der gesucht wird

`substring$`  
die zu suchende Teilzeichenkette

**casesensitive** wenn Sie hier **True** setzen, dann wird die Groß- und Kleinschreibung beim Suchen berücksichtigt, bei **False** nicht; die Voreinstellung ist **True** oder welcher Standardwert auch immer mit dem Befehl **IgnoreCase()** festgelegt wurde

**startpos** optional: Startposition in Zeichen innerhalb von **string\$** (voreingestellt ist 0) (V4.5)

**encoding** optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

## RÜCKGABEWERTE

**pos** die Position in Zeichen von **substring\$** in **string\$** oder falls nicht gefunden -1.

## BEISPIEL

```
result = FindStr("Hello World!", "World")
Print(result)
```

Gibt "6" aus, weil dies die Position vom ersten Zeichen "W" ist.

## 57.23 FormatNumber

### BEZEICHNUNG

**FormatNumber** – konvertiert eine Zahl in eine Zeichenkette mit Zifferntrennung (V10.0)

### ÜBERSICHT

```
s$ = FormatNumber(n[, decimals, point$, thousands$])
```

### BESCHREIBUNG

Dieser Befehl wandelt die durch **n** angegebene Zahl in eine Zeichenkette um, trennt die Ziffern durch Tausender und schneidet die Dezimalstellen auf die im Argument **decimals** übergebene Zahl ab. Sie können auch das Zeichen, das als Dezimaltrennzeichen verwendet werden soll, in **point\$** und das Zeichen, das als Tausendertrennzeichen verwendet werden soll, in **thousands\$** übergeben. Dadurch eignet sich **FormatNumber()** für die gebietsschemaabhängige Formatierung von Zahlen. Sie können den Befehl **GetLocaleInfo()** verwenden, um den Dezimalpunkt und das Tausendertrennzeichen für das aktuelle Gebietsschema abzurufen. Siehe [Abschnitt 34.8 \[GetLocaleInfo\]](#), [Seite 701](#), für Details.

### EINGABEN

**n** Zahl, die in eine Zeichenkette umgewandelt werden soll

**decimals** optional: Anzahl der zu verwendenden Dezimalstellen (voreingestellt ist 0)

**point\$** optional: als Dezimalpunkt zu verwendendes Zeichen (voreingestellt ist ".")

**thousands\$** optional: als Tausendertrennzeichen zu verwendendes Zeichen (voreingestellt ist ",")

## RÜCKGABEWERTE

**s\$** formatierte Zeichenkette

**BEISPIEL**

```
t = GetLocaleInfo()
s$ = FormatNumber(1234567.89, 2, t.DecimalPoint, t.ThousandSeparator)
DebugPrint(s$)
```

Der obige Code wandelt 1234567,89 unter Verwendung des Dezimalzeichens und des Tausendertrennzeichens des aktuellen Gebietsschemas in eine Zeichenkette um.

**57.24 FormatStr****BEZEICHNUNG**

FormatStr – schreibt eine Zeichenkette im C-Stil (V5.0)

**ÜBERSICHT**

```
s$ = FormatStr(fmt$, ...)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um eine C-Stil formatierte Zeichenkette mit Hollywood zu erstellen. Sie übergeben die Formatierungsvorlage im ersten Argument und Sie müssen für jedes Zeichen in der Vorlage ein zusätzliches Argument angeben. Hollywood unterstützt die meisten der Zeichen der C `printf` Deklaration. Hier ist eine Liste aller Token, die derzeit unterstützt werden:

<code>%c</code>	Einzelnes ASCII-Zeichen
<code>%d</code>	Dezimale Ganzzahl mit Vorzeichen
<code>%i</code>	Gleich wie <code>%d</code>
<code>%o</code>	Ganzzahl als Oktalzahl
<code>%u</code>	Dezimale Ganzzahl ohne Vorzeichen
<code>%x</code>	Ganzzahl als Hexadezimalzahl (ohne Vorzeichen, mit Kleinbuchstaben)
<code>%X</code>	Ganzzahl als Hexadezimalzahl (ohne Vorzeichen, mit Großbuchstaben)
<code>%e</code>	Fließkommazahl in Exponentialnotation ( <code>[-]d.ddd e±dd</code> )
<code>%E</code>	Gleich wie <code>%e</code> aber mit Großbuchstabennotation ( <code>[-]d.ddd E±dd</code> )
<code>%f</code>	Fließkommazahl in normaler Notation ( <code>[-]ddd.ddd</code> )
<code>%g</code>	Fließkommazahl in <code>%e</code> oder <code>%f</code> Format (welche kompakter ist)
<code>%G</code>	Gleich wie <code>%g</code> aber mit Großbuchstabennotation
<code>%s</code>	Zeichenkette

Sie können auch eine Zahl in dem Token angeben, welche die Anzahl der Zeichen begrenzt. Zum Beispiel, wenn Sie das Token `"%.6x"` verwenden, wird die durch diesen Befehl erzeugte Hexadezimalzahl immer 6 Ziffern haben.

Da das Prozentzeichen für Token verwendet wird, müssen Sie zwei Prozentzeichen (`%%`) setzen, falls Sie eines in einer Zeichenkette verwenden wollen.

**EINGABEN**

`fmt$`           Formatierungsvorlage, die ein oder mehrere Tokens verwendet (siehe oben für die unterstützten Token)

`...`           zusätzliche Argumente (eine für jedes Token in `fmt$`)

**RÜCKGABEWERTE**

`s$`           resultierende Zeichenkette

**BEISPIEL**

```
a = 128
```

```
s$ = FormatStr("The number " .. a .. " is $%x in hexadecimal notation", a)
```

Der obige Code konvertiert die Zahl 128 in die hexadezimale Notation.

```
a = 255
```

```
s$ = FormatStr("The number " .. a .. " is $%.6x in RGB notation", a)
```

Der obige Code wandelt die Nummer 255 in einen 6-stelligen hexadezimalen Wert, der oft verwendet wird, um RGB-Farben anzugeben.

## 57.25 HexStr

**BEZEICHNUNG**

HexStr – konvertiert einen Wert in eine hexadezimale Zeichenfolge (V1.5)

**ÜBERSICHT**

```
hex$ = HexStr(val)
```

**BESCHREIBUNG**

Dieser Befehl wandelt den durch `val` angegebenen Wert in hexadezimale Ziffern und gibt sie als Zeichenfolge zurück. Die zurückgegebene Zeichenkette wird mit einem Dollarzeichen (\$) vorangestellt werden und alle alphabetischen hexadezimalen Ziffern werden in Großbuchstaben geschrieben.

**EINGABEN**

`val`           umzuwandelnder Wert

**RÜCKGABEWERTE**

`hex$`          hexadezimale Schreibweise von `val`

**BEISPIEL**

```
a$ = HexStr(255)
```

Dies gibt die Zeichenkette "\$FF" zurück.

## 57.26 IgnoreCase

**BEZEICHNUNG**

IgnoreCase – definiert die Standardeinstellung für Groß-/Kleinschreibung (V9.0)

**ÜBERSICHT**

`IgnoreCase(casesen)`

**BESCHREIBUNG**

Mit diesem Befehl können Sie festlegen, ob bei Befehlen wie z.B. `FindStr()` und `ReplaceStr()` standardmäßig zwischen Groß- und Kleinschreibung unterschieden werden soll oder nicht. Wenn Sie das Argument `casesen` auf `True` setzen, wird bei diesen Befehlen zwischen Groß- und Kleinschreibung unterschieden. Wenn Sie es auf `False` setzen, wird die Groß-/Kleinschreibung nicht beachtet. Standardmäßig wird zwischen Groß- und Kleinschreibung unterschieden.

`IgnoreCase()` wirkt sich auf die folgenden Befehle von Hollywood aus:

- `CompareStr()`
- `CountStr()`
- `EndsWith()`
- `FindStr()`
- `ReplaceStr()`
- `ReverseFindStr()`
- `StartsWith()`

**EINGABEN**

`casesen`    `True`, wenn bei Befehlen standardmäßig zwischen Groß- und Kleinschreibung unterschieden werden soll, andernfalls `False`

**57.27 InsertStr****BEZEICHNUNG**

`InsertStr` – fügt eine Teil- in eine Zeichenkette mit/ohne Überschreibung ein (V4.5)

**ÜBERSICHT**

`var$ = InsertStr(s$, sub$, pos[, overwrite, encoding])`

**BESCHREIBUNG**

Dieser Befehl fügt `sub$` in `s$` an der Position von `pos` ein (0 ist der Anfang der Zeichenkette). Diese Position wird in Zeichen und nicht in Bytes zurückgegeben. Wenn das optionale Argument `overwrite` auf `True` gesetzt ist, wird `sub$` alle Zeichen ab der eingefügten Position überschreiben.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `ReplaceStr()` und ...

**EINGABEN**

`s$`            In diese Zeichenkette wird eingefügt

`sub$`        Teilzeichenkette, die eingefügt wird



**pos** an dieser Position wird die Teilzeichenkette in Zeichen eingefügt (0 ist der Anfang)

**overwrite** optional: mit **True** werden die Zeichen nach rechts überschrieben, bei **False** nach rechts verschoben (voreingestellt ist **False**, d.h. ohne Überschreibung)

**encoding** optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

**RÜCKGABEWERTE**

**var\$** resultierende Zeichenkette

**BEISPIEL**

```
Print(InsertStr("Hollywood is a very cool program!", " very", 19))
```

Dieser Code gibt "Hollywood is a very very cool program!" aus

```
Print(InsertStr("Hollywood is a very cool program!", "good", 20, True))
```

Der obenstehnde Code gibt "Hollywood is a very good program!" aus

**57.28 IsAlNum****BEZEICHNUNG**

IsAlNum – überprüft, ob das Zeichen alphanumerisch ist (V7.0)

**ÜBERSICHT**

```
bool = IsAlNum(s$, pos, encoding)
```

**BESCHREIBUNG**

Überprüft, ob das Zeichen beim Index **pos** innerhalb der Zeichenkette **s\$** entweder eine Dezimalzahl oder ein Groß-/Kleinbuchstabe ist. Bei einem Buchstaben oder Dezimalzahl wird **True** zurückgegeben, andernfalls **False**. Der optionale Parameter **pos** muss in Zeichen und nicht in Bytes angegeben werden. Er ist auf 0 voreingestellt, womit das erste Zeichen in **s\$** getestet wird.

Mit dem optionalen Parameter **encoding** kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit **SetDefaultEncoding()** eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch [IsAlpha\(\)](#), [IsDigit\(\)](#), [IsLower\(\)](#), [IsUpper\(\)](#), [IsGraph\(\)](#), [IsPrint\(\)](#), [IsPunct\(\)](#), [IsSpace\(\)](#), [IsXDigit\(\)](#), [IsCntrl\(\)](#) und [ValidateStr\(\)](#).

**EINGABEN**

**s\$** Quellzeichenkette

**pos** optional: Index des Zeichens, welches überprüft wird (voreingestellt ist 0)

**encoding** optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

**RÜCKGABEWERTE**

`bool`      `True` bei einem Buchstaben oder Dezimalzahl, andernfalls `False`

**57.29 IsAlpha****BEZEICHNUNG**

`IsAlpha` – prüft, ob das Zeichen alphabetisch ist (V7.0)

**ÜBERSICHT**

```
bool = IsAlpha(s$, pos, encoding)
```

**BESCHREIBUNG**

Prüft, ob das Zeichen beim Index `pos` innerhalb der Zeichenfolge `s$` ein alphabetischer Buchstabe ist und gibt `True` zurück, wenn es so ist, andernfalls `False`. Der optionale Parameter `pos` muss in Zeichen und nicht in Bytes angegeben werden. Er ist voreingestellt auf 0, womit das erste Zeichen in `s$` getestet wird.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `IsAlNum()`, `IsDigit()`, `IsLower()`, `IsUpper()`, `IsGraph()`, `IsPrint()`, `IsPunct()`, `IsSpace()`, `IsXDigit()`, `IsCntrl()` und `ValidateStr()`.

**EINGABEN**

`s$`      Quellzeichenkette

`pos`      optional: Index des Zeichens, welches überprüft wird (voreingestellt ist 0)

`encoding` optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

**RÜCKGABEWERTE**

`bool`      `True` bei einem Buchstaben, andernfalls `False`

**57.30 IsCntrl****BEZEICHNUNG**

`IsCntrl` – prüft, ob das Zeichen ein Steuerzeichen ist (V7.0)

**ÜBERSICHT**

```
bool = IsCntrl(s$, pos, encoding)
```

**BESCHREIBUNG**

Überprüft, ob das Zeichen beim Index `pos` innerhalb der Zeichenkette `s$` ein Steuerzeichen ist und gibt `True` zurück, wenn dem so ist, andernfalls `False`. Ein Steuerzeichen ist ein Zeichen, welches nicht druckbar ist, z.B. Backspace, Tab, Zeilenvorschub, Escape, etc. Der optionale Parameter `pos` muss in Zeichen und nicht in Bytes angegeben werden. Er ist voreingestellt auf 0, womit das erste Zeichen in `s$` getestet wird.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `IsAlNum()`, `IsAlpha()`, `IsDigit()`, `IsLower()`, `IsUpper()`, `IsGraph()`, `IsPrint()`, `IsPunct()`, `IsSpace()`, `IsXDigit()` und `ValidateStr()`.

#### EINGABEN

`s$` Quellzeichenkette

`pos` optional: Index des Zeichens, welches überprüft wird (voreingestellt ist 0)

`encoding` optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

#### RÜCKGABEWERTE

`bool` `True` bei einem Steuerzeichen, andernfalls `False`

### 57.31 IsDigit

#### BEZEICHNUNG

`IsDigit` – prüft, ob das Zeichen eine dezimale Ziffer ist (V7.0)

#### ÜBERSICHT

```
bool = IsDigit(s$[, pos, encoding])
```

#### BESCHREIBUNG

Überprüft, ob das Zeichen beim Index `pos` innerhalb der Zeichenkette `s$` eine dezimale Ziffer ist. Falls dem so ist, wird `True` zurückgegeben, ansonsten `False`. Der optionale Parameter `pos` muss in Zeichen und nicht in Bytes angegeben werden. Er ist voreingestellt auf 0, womit das erste Zeichen in `s$` getestet wird.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `IsAlNum()`, `IsAlpha()`, `IsLower()`, `IsUpper()`, `IsGraph()`, `IsPrint()`, `IsPunct()`, `IsSpace()`, `IsXDigit()`, `IsCntrl()` und `ValidateStr()`.

#### EINGABEN

`s$` Quellzeichenkette

`pos` optional: Index des Zeichens, welches überprüft wird (voreingestellt ist 0)

`encoding` optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

#### RÜCKGABEWERTE

`bool` `True` bei einer dezimalen Ziffer, sonst `False`

## 57.32 IsGraph

### BEZEICHNUNG

IsGraph – prüft, ob das Zeichen eine grafische Darstellung ist (V7.0)

### ÜBERSICHT

```
bool = IsGraph(s$, pos, encoding)
```

### BESCHREIBUNG

Dieser Befehl überprüft, ob das Zeichen beim Index `pos` innerhalb der Zeichenkette `s$` eine grafische Darstellung ist und gibt `True` zurück, falls dem so ist, ansonsten `False`. Dieser Befehl ist grundsätzlich gleich wie `IsPrint()`, außer dass er nicht `True` für ein Leerzeichen zurückgibt. Der optionale Parameter `pos` muss in Zeichen und nicht in Bytes angegeben werden. Er ist voreingestellt auf 0, womit das erste Zeichen in `s$` getestet wird.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `IsAlNum()`, `IsAlpha()`, `IsDigit()`, `IsLower()`, `IsUpper()`, `IsPrint()`, `IsPunct()`, `IsSpace()`, `IsXDigit()`, `IsCntrl()` und `ValidateStr()`.

### EINGABEN

<code>s\$</code>	Quellzeichenkette
<code>pos</code>	optional: Index des Zeichens, welches überprüft wird (voreingestellt ist 0)
<code>encoding</code>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

### RÜCKGABEWERTE

<code>bool</code>	<code>True</code> oder <code>False</code> , je nach dem Testergebnis
-------------------	--

## 57.33 IsLower

### BEZEICHNUNG

IsLower – prüft, ob das Zeichen ein kleiner Buchstabe ist (V7.0)

### ÜBERSICHT

```
bool = IsLower(s$, pos, encoding)
```

### BESCHREIBUNG

Überprüft, ob das Zeichen beim Index `pos` innerhalb der Zeichenkette `s$` ein kleiner Buchstabe ist. Falls dem so ist, wird `True` zurückgegeben, ansonsten `False`. Der optionale Parameter `pos` muss in Zeichen und nicht in Bytes angegeben werden. Er ist voreingestellt auf 0, womit das erste Zeichen in `s$` getestet wird.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `IsUpper()`, `IsAlNum()`, `IsAlpha()`, `IsDigit()`, `IsGraph()`, `IsPrint()`, `IsPunct()`, `IsSpace()`, `IsXDigit()`, `IsCntrl()` und `ValidateStr()`.

#### EINGABEN

`s$` Quellzeichenkette

`pos` optional: Index des Zeichens, welches überprüft wird (voreingestellt ist 0)

`encoding` optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

#### RÜCKGABEWERTE

`bool` `True` bei einem kleinen Buchstaben, sonst `False`

## 57.34 IsPrint

#### BEZEICHNUNG

`IsPrint` – prüft, ob das Zeichen ausgegeben werden kann (V7.0)

#### ÜBERSICHT

`bool = IsPrint(s$[, pos, encoding])`

#### BESCHREIBUNG

Überprüft, ob das Zeichen beim Index `pos` innerhalb der Zeichenkette `s$` ausgegeben werden kann. Falls dem so ist, wird `True` zurückgegeben, ansonsten `False`. Zeichen, die ausgegeben werden können, sind all jene, die eine grafische Darstellung haben inklusive dem Leerzeichen. Diese Zeichen sind das Gegenteil von Steuerzeichen. `IsPrint()` ist grundsätzlich das gleiche wie `IsGraph()`. Der einzige Unterschied ist, dass `IsPrint()` auch für das Leerzeichen `True` zurückgibt, `IsGraph()` hingegen `False`. Der optionale Parameter `pos` muss in Zeichen und nicht in Bytes angegeben werden. Er ist voreingestellt auf 0, womit das erste Zeichen in `s$` getestet wird.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `IsAlNum()`, `IsAlpha()`, `IsDigit()`, `IsLower()`, `IsUpper()`, `IsGraph()`, `IsPunct()`, `IsSpace()`, `IsXDigit()`, `IsCntrl()` und `ValidateStr()`.

#### EINGABEN

`s$` Quellzeichenkette

`pos` optional: Index des Zeichens, welches überprüft wird (voreingestellt ist 0)

`encoding` optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

#### RÜCKGABEWERTE

`bool` `True` oder `False`, je nach dem Testergebnis

## 57.35 IsPunct

### BEZEICHNUNG

IsPunct – prüft, ob das Zeichen ein Satzzeichen ist (V7.0)

### ÜBERSICHT

```
bool = IsPunct(s$, pos, encoding)
```

### BESCHREIBUNG

Überprüft, ob das Zeichen beim Index `pos` innerhalb der Zeichenkette `s$` ein Satzzeichen ist. Ist dies der Fall, wird `True` zurückgegeben, ansonsten `False`. Der optionale Parameter `pos` muss in Zeichen und nicht in Bytes angegeben werden. Er ist voreingestellt auf 0, womit das erste Zeichen in `s$` getestet wird.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `IsAlNum()`, `IsAlpha()`, `IsDigit()`, `IsLower()`, `IsUpper()`, `IsGraph()`, `IsPrint()`, `IsSpace()`, `IsXDigit()`, `IsCntrl()` und `ValidateStr()`.

### EINGABEN

<code>s\$</code>	Quellzeichenkette
<code>pos</code>	optional: Index des Zeichens, welches überprüft wird (voreingestellt ist 0)
<code>encoding</code>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

### RÜCKGABEWERTE

<code>bool</code>	<code>True</code> bei einem Satzzeichen, sonst <code>False</code>
-------------------	---

## 57.36 IsSpace

### BEZEICHNUNG

IsSpace – prüft, ob das Zeichen ein unsichtbares Zeichen ist (V7.0)

### ÜBERSICHT

```
bool = IsSpace(s$, pos, encoding)
```

### BESCHREIBUNG

Dieser Befehl überprüft, ob das Zeichen beim Index `pos` innerhalb der Zeichenkette `s$` ein unsichtbares Zeichen ist. Ist dies der Fall, wird `True` zurückgegeben, ansonsten `False`. Unsichtbare Zeichen sind Zeichen wie Leerzeichen, Tabulator, neue Zeile, Wagenrücklauf usw. Der optionale Parameter `pos` muss in Zeichen und nicht in Bytes angegeben werden. Er ist voreingestellt auf 0, womit das erste Zeichen in `s$` getestet wird.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `IsAlNum()`, `IsAlpha()`, `IsDigit()`, `IsLower()`, `IsUpper()`, `IsGraph()`, `IsPrint()`, `IsPunct()`, `IsXDigit()`, `IsCntrl()` und `ValidateStr()`.

#### EINGABEN

`s$` Quellzeichenkette

`pos` optional: Index des Zeichens, welches überprüft wird (voreingestellt ist 0)

`encoding` optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

#### RÜCKGABEWERTE

`bool` `True` bei einem unsichtbaren Zeichen, sonst `False`

## 57.37 IsUpper

#### BEZEICHNUNG

`IsUpper` – prüft, ob das Zeichen ein großer Buchstabe ist (V7.0)

#### ÜBERSICHT

```
bool = IsUpper(s$[, pos, encoding])
```

#### BESCHREIBUNG

Überprüft, ob das Zeichen beim Index `pos` innerhalb der Zeichenkette `s$` ein großer Buchstabe ist. Falls dem so ist, wird `True` zurückgegeben, ansonsten `False`. Der optionale Parameter `pos` muss in Zeichen und nicht in Bytes angegeben werden. Er ist voreingestellt auf 0, womit das erste Zeichen in `s$` getestet wird.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `IsLower()`, `IsAlNum()`, `IsAlpha()`, `IsDigit()`, `IsGraph()`, `IsPrint()`, `IsPunct()`, `IsSpace()`, `IsXDigit()`, `IsCntrl()` und `ValidateStr()`.

#### EINGABEN

`s$` Quellzeichenkette

`pos` optional: Index des Zeichens, welches überprüft wird (voreingestellt ist 0)

`encoding` optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

#### RÜCKGABEWERTE

`bool` `True` bei einem großen Buchstaben, sonst `False`

## 57.38 IsXDigit

### BEZEICHNUNG

IsXDigit – prüft, ob das Zeichen eine hexadezimale Ziffer ist (V7.0)

### ÜBERSICHT

```
bool = IsXDigit(s$, pos, encoding)
```

### BESCHREIBUNG

Überprüft, ob das Zeichen beim Index `pos` innerhalb der Zeichenkette `s$` eine hexadezimale Ziffer ist. Falls dem so ist, wird `True` zurückgegeben, ansonsten `False`. Der optionale Parameter `pos` muss in Zeichen und nicht in Bytes angegeben werden. Er ist voreingestellt auf 0, womit das erste Zeichen in `s$` getestet wird.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `IsAlNum()`, `IsAlpha()`, `IsDigit()`, `IsLower()`, `IsUpper()`, `IsGraph()`, `IsPrint()`, `IsPunct()`, `IsSpace()`, `IsCntrl()` und `ValidateStr()`.

### EINGABEN

<code>s\$</code>	Quellzeichenkette
<code>pos</code>	optional: Index des Zeichens, welches überprüft wird (voreingestellt ist 0)
<code>encoding</code>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

### RÜCKGABEWERTE

`bool`      `True` bei einer hexadezimalen Ziffer, sonst `False`

## 57.39 LeftStr

### BEZEICHNUNG

LeftStr – kopiert die linken `n` Zeichen einer Zeichenkette in eine neue Variable

### ÜBERSICHT

```
var$ = LeftStr(string$, len[, encoding])
```

### BESCHREIBUNG

Kopiert, beginnend von links, die Anzahl `len` Zeichen aus `string$` in die Variable `var$`.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `MidStr()`, `RightStr()`, `EndsWith()`, `StartsWith()`, `FindStr()`, `ReverseFindStr()`, `UnleftStr()`, `UnmidStr()` und `UnrightStr()`.

### EINGABEN

<code>string\$</code>	Quellenzeichenkette
-----------------------	---------------------



**len**           Anzahl kopierende Zeichen

**encoding**   optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

**RÜCKGABEWERTE**

**var\$**           resultierende Zeichenkette

**BEISPIEL**

```
test$=LeftStr("Hello World!",5)
Print(test$)
```

Dies wird "Hello" ausgegeben.

**57.40 LowerStr****BEZEICHNUNG**

LowerStr – konvertiert alle Zeichen einer Zeichenkette in Kleinbuchstaben

**ÜBERSICHT**

```
var$ = LowerStr(string$[, encoding])
```

**BESCHREIBUNG**

Konvertiert alle Zeichen der Variable **string\$** in Kleinbuchstaben und übergibt die neue Zeichenkette an **var\$**.

Mit dem optionalen Parameter **encoding** kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit **SetDefaultEncoding()** eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch [UpperStr\(\)](#).

**EINGABEN**

**string\$**   zu konvertierende Zeichenkette

**encoding**   optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

**RÜCKGABEWERTE**

**var\$**           umgewandelte Zeichenkette

**BEISPIEL**

```
test$ = LowerStr("Hello World!")
Print(test$)
```

Das gibt "hello world!" aus.

**57.41 MD5Str****BEZEICHNUNG**

MD5Str – berechnet die MD5-Prüfsumme einer Zeichenfolge (V5.0)

**ÜBERSICHT**

```
sum$ = MD5Str(s$)
```

**BESCHREIBUNG**

Dieser Befehl berechnet die MD5-Prüfsumme der Zeichenfolge von `s$`. Die 128-Bit-Prüfsumme wird als Zeichenkette mit 16 hexadezimalen Ziffern in `sum$` zurückgegeben. Beachten Sie, dass Hollywood-Zeichenketten auch binäre Daten enthalten können, so dass Sie diesen Befehl auch mit Nichttext-Zeichenfolgen verwenden können.

Wenn Sie die MD5-Prüfsumme einer Datei berechnen, verwenden Sie den Befehl `MD5()`. Siehe auch `CRC32Str()` und `CRC32()`.

**EINGABEN**

`s$`            Zeichenkette, dessen Prüfsumme sie berechnen wollen

**RÜCKGABEWERTE**

`sum$`            MD5-Prüfsumme der Zeichenkette

**57.42 MidStr****BEZEICHNUNG**

MidStr – extrahiert Zeichen aus einer Zeichenkette

**ÜBERSICHT**

```
var$ = MidStr(string$, startpos[, len, encoding])
```

**BESCHREIBUNG**

Kopiert die Anzahl `len` Zeichen aus `string$` (von Position `startpos` an) in die Variable `var$`. Der optionale Parameter `startpos` muss in Zeichen und nicht in Bytes angegeben werden.

Ab Hollywood 6.1 kann `len` weggelassen oder auf -1 gesetzt werden. In diesem Fall werden die restlichen Zeichen von `string$` in die Variable `var$` geschrieben werden.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `LeftStr()`, `RightStr()`, `EndsWith()`, `StartsWith()`, `FindStr()`, `ReverseFindStr()`, `UnleftStr()`, `UnmidStr()` und `UnrightStr()`.

**EINGABEN**

`string$`    Quellvariable

`startpos`   hier soll das Kopieren beginnen (erstes Zeichen ist an Position 0)

`len`            optional: Anzahl der zu kopierenden Zeichen. Wird es weggelassen oder auf -1 gesetzt, werden die restlichen Zeichen bis zum Ende der Zeichenkette kopiert (voreingestellt ist -1, also bis zum Ende der Zeichenkette)

`encoding`    optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

**RÜCKGABEWERTE**

`var$`        Variable, die die extrahierten Zeichen empfängt

**BEISPIEL**

```
test$=MidStr("Hello World!",4,3)
Print(test$)
```

Das wird "o W" auf den Schirm schreiben.

## 57.43 PadNum

**BEZEICHNUNG**

`PadNum` – Konvertiert eine Zahl in eine Zeichenkette mit führenden Nullen (V2.0)

**ÜBERSICHT**

```
s$ = PadNum(num, len)
```

**BESCHREIBUNG**

Dieser Befehl wandelt die Ganzzahl in `num` in eine Zeichenkette um. Darüber hinaus fügt er führende Nullen ein, bis die Zeichenfolge die Länge in `len` erreicht hat. Die Zahl darf nicht negativ sein und keine Dezimalstellen enthalten. Dies erreichen Sie mit den Befehlen `Abs()` und `Int()`.

**EINGABEN**

`num`        Zahl, die in eine Zeichenkette konvertiert wird  
`len`        die Zeichenkettelänge

**RÜCKGABEWERTE**

`s$`        Zeichenkette mit führenden Nullen

**BEISPIEL**

```
DebugPrint(PadNum(9, 2))
```

Gibt "09" aus.

## 57.44 PatternFindStr

**BEZEICHNUNG**

`PatternFindStr` – analysiert die Zeichenkette mit passendem Muster und Iterator-Funktion (V5.0)

**ÜBERSICHT**

```
func, state, val = PatternFindStr(s$, pat$[, encoding])
```

**BESCHREIBUNG**

Dieser Befehl kann in Verbindung mit der generischen `For`-Anweisung verwendet werden, um die Zeichenfolge in `s$` nach dem in `pat$` angegebenen Muster zu analysieren. Wegen der generischen `For`-Anweisung gibt `PatternFindStr()` drei Werte zurück: Eine Iterator-Funktion `func`, einen privaten Zustandswert `state` und einen Anfangswert `val`.

Bei jedem Aufruf wird die Iterator-Funktion gemäß dem Muster `pat$` die Zeichenkette `s$` analysieren und die nächste Übereinstimmung zurückgeben. Wenn in `pat$` nichts angegeben wird, dann wird bei jedem Aufruf eine Übereinstimmung gefunden.

Das in `pat$` angegeben Muster muss dem Mustersyntax entsprechen, wie sie in der Dokumentation des Befehls `PatternReplaceStr()` beschrieben wird. Siehe [Abschnitt 57.47 \[PatternReplaceStr\]](#), Seite 1286, für Details.

Siehe [Abschnitt 11.4 \[Generische Version der For-Anweisung\]](#), Seite 135, für Details.

Ab Hollywood 6.0 ist dieser Befehl auch in einer Version erhältlich, die ohne generische `For`-Anweisung verwendet werden kann. Also, wenn Sie nur das erste Resultat brauchen, welches Ihnen `pat$` aus `s$` liefert, dann möchten Sie vielleicht stattdessen `PatternFindStrDirect()` verwenden. Siehe [Abschnitt 57.45 \[PatternFindStrDirect\]](#), Seite 1285, für Details.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), Seite 158, für Details.

Siehe auch `PatternFindStrShort()`.

## EINGABEN

<code>s\$</code>	die zu analysierende Zeichenkette
<code>pat\$</code>	Muster, mit dem die Zeichenkette analysiert wird
<code>encoding</code>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

## RÜCKGABEWERTE

<code>func</code>	Iterator-Funktion
<code>state</code>	privater Zustandswert
<code>val</code>	Anfangswert

## BEISPIEL

```
s$ = "Hello World from Hollywood"
For w$ In PatternFindStr(s$, "%a+") Do DebugPrint(w$)
```

Der obige Code analysiert alle Wörter in der Zeichenkette `s$` und gibt eines pro Zeile aus.

```
t = {}
s$ = "Name=Andreas, Sex=Male, Nationality=German"
For k, v in PatternFindStr(s$, "(%w+)=(%w+)") Do t[k] = v
```

Das obige Beispiel sammelt alle Paare Schlüssel=Wert aus der angegebenen Zeichenfolge in einer Tabelle.

## 57.45 PatternFindStrDirect

### BEZEICHNUNG

PatternFindStrDirect – analysiert eine Zeichenkette anhand der Musterübereinstimmung (V6.0)

### ÜBERSICHT

```
start, end, ... = PatternFindStrDirect(s$, pat$[, start, encoding])
```

### BESCHREIBUNG

Dieser Befehl analysiert die in `s$` angegebene Zeichenkette gemäß dem in `pat$` angegebenen Muster. Wenn es eine Übereinstimmung gibt, werden die Indizes, von denen die Übereinstimmung in der Quellzeichenkette beginnt und endet, zusammen mit allen aufgezeichneten Zeichenfolgen zurückgegeben. Wenn keine Übereinstimmung vorhanden ist, gibt `PatternFindStrDirect()` -1 zurück. Das optionale Argument `start` kann verwendet werden, um ein Zeichenindex innerhalb von `s$` anzugeben, an dem die Suche beginnen soll. Die Voreinstellung ist 0, was bedeutet, dass `PatternFindStrDirect()` am Anfang von `s$` beginnen wird.

Dieser Befehl funktioniert genauso wie `PatternFindStr()`, erfordert aber keine generische `For`-Anweisung. Stattdessen werden alle Übereinstimmungen zusammen mit den Start- und Endindizes zurückgegeben. Beachten Sie jedoch, dass `PatternFindStrDirect()` nicht innerhalb einer generischen `For`-Anweisung funktioniert, so dass nur das erste Vorkommen von `pat$` innerhalb `s$` behandelt wird.

Wenn Sie die Start- und Endindizes nicht benötigen, können Sie stattdessen den Befehl `PatternFindStrShort()` verwenden. Siehe [Abschnitt 57.46 \[PatternFindStrShort\]](#), [Seite 1286](#), für Details.

Das Muster in `pat$` muss dem Muster-Syntax entsprechen, wie sie in der Dokumentation des Befehls `PatternReplaceStr()` beschrieben wird. Siehe [Abschnitt 57.47 \[PatternReplaceStr\]](#), [Seite 1286](#), für Details.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

### EINGABEN

<code>s\$</code>	Zeichenkette, die analysiert wird
<code>pat\$</code>	Muster, nach dem die Zeichenfolge analysiert werden soll
<code>start</code>	optional: Position, an der die Suche beginnen soll (voreingestellt ist 0)
<code>encoding</code>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

### RÜCKGABEWERTE

<code>start</code>	Position der ersten Übereinstimmung innerhalb von <code>s\$</code> oder -1 für keine Übereinstimmung
<code>end</code>	Position der letzten Übereinstimmung innerhalb von <code>s\$</code>
<code>...</code>	einzelne Zeichenketten mit allen Übereinstimmungen

**BEISPIEL**

```
DebugPrint(PatternFindStrDirect("Name=Andreas", "(%w+)=(%w+)"))
```

Das obige Beispiel gibt die Zeichenfolgen neben dem Gleichheitszeichen und den Bereich 0 bis 11 zurück, die die gesamte Quellzeichenfolge beschreibt.

**57.46 PatternFindStrShort****BEZEICHNUNG**

PatternFindStrShort – analysiert eine Zeichenkette anhand der Musterübereinstimmung (V6.0)

**ÜBERSICHT**

```
... = PatternFindStrShort(s$, pat$[, start, encoding])
```

**BESCHREIBUNG**

Dieser Befehl funktioniert genauso wie `PatternFindStrDirect()`, aber die Anfangs- und Endindizes werden nicht zurückgegeben. Siehe [Abschnitt 57.45 \[PatternFindStrDirect\]](#), [Seite 1285](#), für Details.

Das in `pat$` angegebene Muster muss der Mustersyntax entsprechen, wie in der Dokumentation des Befehls `PatternReplaceStr()` beschrieben ist. Siehe [Abschnitt 57.47 \[PatternReplaceStr\]](#), [Seite 1286](#), für Details.

Siehe auch `PatternFindStr()`.

**EINGABEN**

<code>s\$</code>	Zeichenkette, die analysiert wird
<code>pat\$</code>	Muster, nach dem die Zeichenfolge analysiert werden soll
<code>start</code>	optional: Position, an der die Suche beginnen soll (voreingestellt ist 0)
<code>encoding</code>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

**RÜCKGABEWERTE**

... einzelne Zeichenketten mit allen Übereinstimmungen

**BEISPIEL**

```
DebugPrint(PatternFindStrShort("Name=Andreas", "(%w+)=(%w+)"))
```

Das obige Beispiel gibt die Zeichenfolgen neben dem Gleichheitszeichen zurück.

**57.47 PatternReplaceStr****BEZEICHNUNG**

PatternReplaceStr – ändert die Zeichenfolge unter Verwendung von Musterübereinstimmungen (V5.0)

**ÜBERSICHT**

```
r$, n = PatternReplaceStr(s$, pat$, repl[, n, encoding])
```

**BESCHREIBUNG**

`PatternReplaceStr()` kann verwendet werden, um den Inhalt einer Zeichenkette mit Hilfe eines Musters zu ändern. Dieser Befehl ist leistungsfähig und kann für alle Arten von Zeichenkettenoperationen verwendet werden. Sie gibt eine Kopie von `s$` zurück, in der alle Vorkommen des Musters `pat$` durch eine in `repl` angegebene Zeichenfolge ersetzt wurden. `PatternReplaceStr()` gibt im zweiten Wert `n` die Gesamtzahl der Ersetzungen zurück.

Das dritte Argument `repl` kann entweder eine Zeichenfolge oder eine Callback-Funktion sein. Wenn `repl` eine Zeichenkette ist, wird sein Wert als Ersatz verwendet. Jede Sequenz in `repl` der Form `n%`, mit `n` zwischen 1 und 9, steht für den Wert der `n`-ten Übereinstimmung der Teilzeichenkette (siehe unten). Wenn `repl` eine Funktion ist, wird diese Funktion jedes Mal aufgerufen, wenn eine Übereinstimmung stattfindet, wobei alle übereinstimmenden Teilzeichenketten als Argumente übergeben werden; Wenn das Muster keine Teilmuster enthält, wird die gesamte Übereinstimmung als einziges Argument übergeben. Wenn der von diesem Befehl zurückgegebene Wert eine Zeichenkette ist, wird er als Ersatzzeichenfolge verwendet; Andernfalls ist die Ersatzzeichenfolge die leere Zeichenkette.

Das optionale vierte Argument `n` begrenzt die maximale Anzahl der auftretenden Ersetzungen. Zum Beispiel wird, wenn `n` 1 ist, nur das erste Auftreten von `pat$` ersetzt.

Das in `pat$` angegebene Muster besteht aus einer Folge von Musterelementen. Ein Musterelement ist normalerweise eine Zeichenklasse, die wiederum einen Satz von Zeichen darstellt. Folgende Kombinationen sind bei der Beschreibung einer Zeichenklasse erlaubt:

`x` (wobei `x` keines der magischen Steuerzeichen `^$()%.*+-?` ist)

Repräsentiert das Zeichen `x` selbst.

`.` (ein Punkt)

Stellt alle Zeichen dar.

`%a` Stellt alle Buchstaben dar.

`%c` Stellt alle Steuerzeichen dar.

`%d` Stellt alle Ziffern dar.

`%g` Steht für alle Zeichen, die eine grafische Darstellung haben. (V7.0)

`%l` Stellt alle Kleinbuchstaben dar.

`%p` Stellt alle Satzzeichen dar.

`%s` Stellt alle Leerzeichen dar.

`%u` Stellt alle Großbuchstaben dar.

`%w` Stellt alle alphanumerischen Zeichen dar.

`%x` Stellt alle hexadezimalen Ziffern dar.

`%z` Repräsentiert das Zeichen mit Darstellung 0.

`%x` (wobei `x` ein nicht-alphanumerisches Zeichen ist)

Stellt das Zeichen `x` dar. Dies ist der Standardweg, um die magischen Steuerzeichen zu umgehen. Jedes Interpunktionszeichen (auch die

Nicht-Steuerzeichen) kann mit einem % vorangestellt werden, um sich selbst in einem Muster darzustellen.

**[set]** Stellt die Klasse dar, die die Vereinigung aller Zeichen in **set** ist. Eine Reihe von Zeichen kann durch Trennung der Endzeichen des Bereichs mit einem - angegeben werden. Alle oben beschriebenen %x Klassen können auch als Komponenten in **set** verwendet werden. Alle anderen Zeichen in **set** repräsentieren sich selber. Beispielsweise repräsentiert [%w\_] (oder [\_%w]) alle alphanumerischen Zeichen plus Unterstrich, [0-7] die oktalen Ziffern und [0-7%1%-] die Oktalziffern plus Kleinbuchstaben und das Zeichen -. Die Interaktion zwischen Bereichen und Klassen ist nicht definiert. Daher haben Muster wie [%a-z] oder [a-%] keine Bedeutung.

**[^set]** Stellt das Gegenteil von **set** dar, wobei **set** wie oben interpretiert wird.

Für alle Klassen, die durch einzelne Buchstaben (%a, %c, etc.) dargestellt werden, stellt der entsprechende Großbuchstabe das Gegenteil der Klasse dar. Beispielsweise stellt %S alle Nicht-Leerzeichen dar.

Die folgenden Elemente sind gültige Musterelemente:

- eine einzelne Zeichenklasse, die jedem einzelnen Zeichen in der Klasse entspricht.
- eine einzelne Zeichenklasse gefolgt von \*, die 0 oder mehr Wiederholungen von Zeichen in der Klasse entspricht. Dieses Wiederholungselement entspricht immer der längsten möglichen Reihenfolge.
- eine einzelne Zeichenklasse gefolgt von +, die 1 oder mehr Wiederholungen von Zeichen in der Klasse entspricht. Dieses Wiederholungselement entspricht immer der längsten möglichen Reihenfolge
- eine einzelne Zeichenklasse gefolgt von -, die auch 0 oder mehr Wiederholungen von Zeichen in der Klasse entspricht. Anders als \* entsprechen diese Wiederholungselemente immer der kürzesten Reihenfolge
- eine einzelne Zeichenklasse gefolgt von ?, die 0 oder 1 Vorkommen eines Zeichens in der Klasse entspricht
- %n, n zwischen 1 und 9; Dieses Element entspricht einer Teilzeichenfolge gleich der n-ten übereinstimmenden Zeichenfolge (siehe unten)
- %bxy, wobei x und y zwei verschiedene Zeichen sind; Dieses Element stimmt mit Zeichenfolgen überein, die mit x beginnen, mit y enden und wo x und y ausgeglichen sind. Dies bedeutet, dass wenn man die Zeichenfolge von links nach rechts liest und +1 für ein x und -1 für ein y zählt, das letzte y das erste y ist, in dem die Zählung 0 erreicht. Beispielsweise stimmt das Element %b() mit Ausdrücken mit ausgeglichenen Klammern überein.

Ein Muster ist eine Folge von Musterelementen. Ein ^ am Anfang eines Musters verankert die Übereinstimmung am Anfang der Objektzeichenfolge. Ein \$ am Ende eines Musters verankert die Übereinstimmung am Ende der Objektzeichenfolge. An anderen Stellen haben ^ und \$ keine besondere Bedeutung und stellen sich selber dar.

Ein Muster kann Teilmuster enthalten, die in Klammern eingeschlossen sind; sie beschreiben die Teilübereinstimmung. Wenn eine Übereinstimmung erfolgreich ist, werden die Teilzeichenfolgen der Objektzeichenfolge, die mit den Aufzeichnungen übereinstimmen,



für zukünftige Verwendung gespeichert (erfasst). Übereinstimmungen werden nach ihrer linken Klammern numeriert. Zum Beispiel wird in dem Muster "(a\*(.)%w(%s\*))" der Teil der Zeichenfolge, der mit a\*(.)%w(%s\*) übereinstimmt, die Nummer 1 erhalten; Die Zeichenübereinstimmung mit . erhält die Zahl 2 und die Teilübereinstimmung %s\* hat die Nummer 3.

Als Sonderfall erfasst das leere () die aktuelle Zeichenkettenposition (eine Zahl). Wenn wir zum Beispiel das Muster ()aa() auf die Zeichenfolge "flaaap" anwenden, gibt es zwei Positionszahlen: 3 und 5.

Ein Muster darf keine eingebetteten Nullen enthalten. Verwenden Sie stattdessen %z.

Mit dem optionalen Parameter **encoding** kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit **SetDefaultEncoding()** eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch **PatternFindStr()**, **PatternFindStrDirect()** und **PatternFindStrShort()**.

## EINGABEN

<b>s\$</b>	Zeichenkette
<b>pat\$</b>	Muster, nach dem die Zeichenkette <b>s\$</b> geändert werden soll
<b>repl</b>	Ersetzungszeichenkette oder Callback-Funktion, um das Ersetzen zu behandeln (siehe oben)
<b>n</b>	optional: Höchstzahl der Ersetzungen (Voreingestellt ist die Länge von <b>s\$</b> plus 1)
<b>encoding</b>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

## RÜCKGABEWERTE

<b>r\$</b>	resultierende Zeichenfolge
<b>n</b>	Anzahl der Ersetzungen

## BEISPIEL

```
s$ = PatternReplaceStr("Hello World", "(%w+)", "%1 %1")
```

Der obige Code gibt "Hello Hello World World" zurück.

```
s$ = PatternReplaceStr("Hello World from Hollywood", "(%w+)%s*(%w+)",
"%2 %1")
```

Dieser Code gibt "World Hello Hollywood from" zurück.

```
s$ = PatternReplaceStr("home = $HOME, user = $USER", "%$(%w+)", GetEnv)
```

Dieses Beispiel gibt "home = /home/andreas, user = andreas" zurück (auf Linux).

```
Local t = {name = "Hollywood", version="5.0"}
s$ = PatternReplaceStr("$name_$version.jpg", "%$(%w+)", Function(v)
Return(t[v]) EndFunction)
```

Diesmal wird "Hollywood\_5.0.jpg" zurückgegeben.

## 57.48 RepeatStr

### BEZEICHNUNG

RepeatStr – wiederholt mehrere Male die Zeichenkette (V5.0)

### ÜBERSICHT

```
var$ = RepeatStr(s$, n)
```

### BESCHREIBUNG

Dieser Befehl wiederholt `s$` `n`-mal und gibt die resultierende Zeichenkette zurück.

### EINGABEN

`s$`            die Zeichenkette, welche wiederholt wird  
`n`            Anzahl Wiederholungen der Zeichenkette; muss >0 sein

### RÜCKGABEWERTE

`var$`            die resultierende Zeichenkette

### BEISPIEL

```
Print(RepeatStr("Hollywood!", 5))
```

Dieser Code gibt "Hollywood!Hollywood!Hollywood!Hollywood!Hollywood!" aus

## 57.49 ReplaceStr

### BEZEICHNUNG

ReplaceStr – ersetzt einen Teil einer Zeichenkette durch eine andere

### ÜBERSICHT

```
var$ = ReplaceStr(s$, search$, replace$[, cs, startpos, encoding])
```

### BESCHREIBUNG

Sucht in der Zeichenkette `s$` nach der Teilzeichenkette `search$` und ersetzt sie (falls sie gefunden wird) durch `replace$`. `cs` ist ein optionales Argument, das veranlasst, bei der Suche Groß- und Kleinschreibung zu beachten (`True`) oder nicht (`False`). Dies ist standardmäßig der globale Standardmodus, bei dem die Groß-/Kleinschreibung beachtet wird, der mit `IgnoreCase()` festgelegt wurde. Siehe [Abschnitt 57.26 \[IgnoreCase\]](#), [Seite 1271](#), für Details.

Seit Hollywood 4.5 können Sie auch eine Startposition für die Suche in dem optionalen Argument `startpos` angeben. Diese Position muss in Zeichen und nicht in Bytes angegeben werden. Mit 0 als Startposition bedeutet, Suchen & Ersetzen vom Anfang der Zeichenfolge. Dies ist auch die Standardeinstellung.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch [InsertStr\(\)](#) und ...

### EINGABEN

`s$`            in dieser Zeichenkette wird gesucht

**search\$** nach dieser Teilzeichenkette wird gesucht

**replace\$** durch diese Teilzeichenkette wird ersetzt

**cs** wenn Sie hier **True** setzen, dann wird die Groß- und Kleinschreibung beim Suchen berücksichtigt, bei **False** nicht; die Voreinstellung ist **True** oder welcher Standardwert auch immer mit dem Befehl **IgnoreCase()** festgelegt wurde

**startpos** optional: Startposition von Suchen & Ersetzen (voreingestellt ist 0) (V4.5)

**encoding** optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

## RÜCKGABEWERTE

**var\$** die neue Zeichenkette

## BEISPIEL

```
test$ = "Hello World!"
test$ = ReplaceStr(test$, "World", "People")
Print(test$)
```

Das wird "Hello People!" ausgegeben.

## 57.50 ReverseFindStr

### BEZEICHNUNG

**ReverseFindStr** – sucht eine Teilzeichenkette in einer Zeichenkette in umgekehrter Richtung (V9.0)

### ÜBERSICHT

```
pos = ReverseFindStr(string$, substring$[, casesen, startpos, encoding])
```

### BESCHREIBUNG

Sucht in **string\$** nach **substring\$** und gibt die Position der Teilzeichenkette zurück. Im Gegensatz zu **FindStr()** erfolgt die Suche in umgekehrter Richtung, d.h. vom Ende der Zeichenkette zum Anfang.

Beachten Sie, dass zwar die Suchrichtung vom Ende zum Anfang geht, die Zählung der Zeichen z.B. für die Position aber weiterhin am Anfang beginnt.

Die Position wird in Zeichen und nicht in Bytes zurückgegeben, beginnend an Position 0 für das erste Zeichen. Wenn **substring\$** nicht gefunden werden kann, wird -1 zurückgegeben. Mit dem optionalen Argument **casesen** können Sie angeben, ob bei der Suche zwischen Groß- und Kleinschreibung unterschieden werden soll. Standardmäßig wird die globale Groß-/Kleinschreibung berücksichtigt, die mit **IgnoreCase()** festgelegt wird. Siehe [Abschnitt 57.26 \[IgnoreCase\]](#), [Seite 1271](#), für Details.

Sie können auch eine Startposition für die Suche im optionalen Argument **startpos** angeben. Diese Position muss in Zeichen und nicht in Bytes angegeben werden. Standardmäßig ist **startpos** auf die Länge von **string\$** in Zeichen minus 1 eingestellt.

Mit dem optionalen Parameter **encoding** kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig die Standard-Zeichenkettencodierung, die mit

`SetDefaultEncoding()` festgelegt wurde. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `FindStr()`, `LeftStr()`, `MidStr()`, `RightStr()`, `EndsWith()`, `StartsWith()`, `UnleftStr()`, `UnmidStr()` und `UnrightStr()`.

#### EINGABEN

`string$` Zeichenkette, in der gesucht werden soll

`substring$`  
zu suchende Zeichenkette in `string$`

`casesen` `True` für eine Suche, bei der die Groß-/Kleinschreibung beachtet wird, oder `False` für eine Suche, bei der die Groß-/Kleinschreibung nicht beachtet wird; die Voreinstellung ist `True` oder die Voreinstellung, die mit dem Befehl `IgnoreCase()` eingestellt wurde

`startpos` optional: Startposition des Suchvorgangs in Zeichen (Voreingestellt ist die Länge von `string$` minus 1)

`encoding` optional: Zu verwendende Zeichencodierung (Voreingestellt ist die Standard-Zeichencodierung)

#### RÜCKGABEWERTE

`pos` Position von `substring$` in `string$` in Zeichen oder -1, wenn nichts gefunden wurde

#### BEISPIEL

```
result = ReverseFindStr("Hello, Hello!", "Hello")
Print(result)
```

Dies gibt "7" aus, da die Suche in umgekehrter Richtung erfolgt, weshalb die Position des zweiten "Hello" zurückgegeben wird.

## 57.51 ReverseStr

#### BEZEICHNUNG

`ReverseStr` – kehrt die Reihenfolge der Zeichen in der Zeichenkette um (V7.0)

#### ÜBERSICHT

```
r$ = ReverseStr(s[, encoding])
```

#### BESCHREIBUNG

Dieser Befehl kehrt die Reihenfolge der Zeichen in der Zeichenkette `s$` um und gibt die neue Zeichenfolge zurück.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

#### EINGABEN

`s$` Quellzeichenkette

**encoding** optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

## RÜCKGABEWERTE

**r\$** Zeichenkette mit umgekehrter Reihenfolge der Zeichen

## BEISPIEL

```
r$ = ReverseStr("Hello")
Print(r$)
```

Dieser Code gibt "olleH" aus.

## 57.52 RightStr

### BEZEICHNUNG

**RightStr** – kopiert die rechten **n** Zeichen einer Zeichenkette in eine neue Zeichenkette

### ÜBERSICHT

```
var$ = RightStr(string$, len[, encoding])
```

### BESCHREIBUNG

Gibt beginnend von rechts **len** Zeichen aus **string\$** in **var\$** zurück.

Mit dem optionalen Parameter **encoding** kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit **SetDefaultEncoding()** eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch [LeftStr\(\)](#), [MidStr\(\)](#), [EndsWith\(\)](#), [StartsWith\(\)](#), [FindStr\(\)](#), [ReverseFindStr\(\)](#), [UnleftStr\(\)](#), [UnmidStr\(\)](#) und [UnrightStr\(\)](#).

### EINGABEN

**string\$** Zeichenkette, aus der Zeichen extrahiert werden sollen

**len** Anzahl zu kopierender Zeichen

**encoding** optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

## RÜCKGABEWERTE

**var\$** Zeichenkette, die die extrahierten Zeichen enthält

## BEISPIEL

```
test$=RightStr("Hello World!",6)
Print(test$)
```

Gibt "World!" aus.

## 57.53 SplitStr

### BEZEICHNUNG

**SplitStr** – zerlegt eine Zeichenkette in mehrere Teile (V2.0)

**ÜBERSICHT**

```
table, count = SplitStr(src$, token$[, multiple])
```

**BESCHREIBUNG**

Dieser Befehl teilt die Zeichenkette **src\$** in mehrere Teile, welche durch das Trennzeichen in **token\$** angegeben wurde. **token\$** muss eine Zeichenfolge mit mindestens einem Zeichen sein, das als Trennzeichen in **src\$** vorkommt. **SplitStr()** wird eine Tabelle **table** mit allen Teilen und deren Anzahl in **count** zurückgeben.

Wenn das angegebene Trennzeichen in der Quellzeichenfolge nicht erscheint, wird **src\$** zurückgegeben.

Ab Hollywood 7.1 gibt es das neue optionale Argument **multiple**. Wenn dies auf **True** gesetzt ist, werden mehrere Vorkommen von **token\$** nebeneinander als ein einziges Vorkommen betrachtet. Dies kann nützlich sein, wenn Sie das Leerzeichen als **token\$** verwenden und dieser Befehl mit einer beliebigen Anzahl von Leerzeichen zwischen den verschiedenen Teilen arbeiten soll.

Beachten Sie, dass **token\$** vor Hollywood 8.0 auf eine Zeichenkette mit nur einem Zeichen beschränkt war. Dieses Limit wurde für Hollywood 8.0 aufgehoben und die Zeichenkette kann jetzt beliebig lang sein.

**EINGABEN**

<b>src\$</b>	Quellzeichenfolge
<b>token\$</b>	ein Zeichen als Trennzeichen
<b>multiple</b>	optional: ob mehrere Vorkommnisse von <b>token\$</b> nebeneinander als ein einzelnes Vorkommen behandelt werden sollen oder nicht (voreingestellt ist <b>False</b> ) (V7.1)

**RÜCKGABEWERTE**

<b>table</b>	Tabelle, in dem die neuen Teile gespeichert werden
<b>count</b>	Anzahl Teile, welche der Befehl erstellt hat

**BEISPIEL**

```
array, c = SplitStr("AmigaOS3|MorphOS|AmigaOS4|WarpOS|AROS", "|")
For k = 1 To c Do NPrint(array[k - 1])
```

Dieser Code wird folgendes ausgeben:

```
AmigaOS3
MorphOS
AmigaOS4
WarpOS
AROS
```

Die Variable **c** erhält den Wert 5, weil **SplitStr()** fünf Teile findet und speichert sie in der Tabelle.

**57.54 StartsWith****BEZEICHNUNG**

**StartsWith** – überprüft, ob die Zeichenkette mit der Teilzeichenkette beginnt (V7.1)

**ÜBERSICHT**

```
bool = StartsWith(s$, substr$[, casesen, encoding])
```

**BESCHREIBUNG**

Mit diesem Befehl kann überprüft werden, ob `s$` mit der in `substr$` angegebenen Teilzeichenkette beginnt. Ist dies der Fall, wird `True` zurückgegeben, ansonsten `False`. Wenn das optionale Argument `casesen` auf `False` gesetzt ist, müssen die Zeichen in Groß-/Kleinschreibung nicht übereinstimmen. `casesen` verwendet standardmäßig den globalen Standardmodus, bei dem die Groß-/Kleinschreibung beachtet wird, der mit `IgnoreCase()` festgelegt wurde. Siehe [Abschnitt 57.26 \[IgnoreCase\]](#), [Seite 1271](#), für Details.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung eingestellt werden. Voreingestellt ist die Standardeinstellung für die Zeichenkettenkodierung, die mit `SetDefaultEncoding()` festgelegt wurde. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `EndsWith()`, `LeftStr()`, `MidStr()`, `RightStr()`, `FindStr()`, `ReverseFindStr()`, `UnleftStr()`, `UnmidStr()` und `UnrightStr()`.

**EINGABEN**

<code>s\$</code>	Eingabezeichenkette
<code>substr\$</code>	Zeichenkette, die mit <code>s\$</code> verglichen wird
<code>casesen</code>	optional: ob die Groß-/Kleinschreibung beim Vergleichen aktiviert werden soll ( <code>True</code> ) oder nicht ( <code>False</code> ) (voreingestellt ist <code>True</code> , d.h. Groß-/Kleinschreibung wird berücksichtigt); der Standardwert ist <code>True</code> oder welcher Standardwert auch immer mit dem Befehl <code>IgnoreCase()</code> festgelegt wurde
<code>encoding</code>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

**RÜCKGABEWERTE**

<code>bool</code>	boolescher Wert, der Erfolg ( <code>True</code> ) oder Misserfolg ( <code>False</code> ) anzeigt
-------------------	--

**57.55 StripStr****BEZEICHNUNG**

`StripStr` – entfernt Leerzeichen vor/hinter einer Zeichenkette (V7.1)

**ÜBERSICHT**

```
r$ = StripStr(s$)
```

**BESCHREIBUNG**

Dieser Befehl entfernt alle leere Zeichen vor und hinter einer Zeichenkette und gibt die gekürzte Zeichenkette zurück. Die folgenden Zeichen sind leere Zeichen: Leerzeichen, Seitenvorschub ("`\f`"), neue Zeile ("`\n`"), Wagenrücklauf ("`\r`"), horizontaler Tabulator ("`\t`") und vertikaler Tabulator ("`\v`").

Siehe auch `TrimStr()`

**EINGABEN**

**s\$**            Eingabezeichenkette

**RÜCKGABEWERTE**

**r\$**            gekürzte Zeichenkette

**BEISPIEL**

```
s$ = StripStr("    Hello World    ")
```

Dies gibt "Hello World" zurück.

## 57.56 StrLen

**BEZEICHNUNG**

StrLen – gibt die Zeichenlänge einer Zeichenkette zurück

**ÜBERSICHT**

```
len = StrLen(str$[, encoding])
```

**BESCHREIBUNG**

Dieser Befehl gibt die Anzahl der Zeichen in einer Zeichenkette zurück.

Dieser Befehl gibt die Zeichenlänge von **str\$** zurück. Beachten Sie, dass dies bei Zeichenfolgen mit Unicode nicht notwendigerweise mit der Byte-Länge von **str\$** übereinstimmt. Um die Byte-Länge einer Zeichenkette herauszufinden, verwenden Sie den Befehl **ByteLen()** oder übergeben Sie eine Nicht-Unicode-Codierung in **encoding**.

Mit dem optionalen Parameter **encoding** kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit **SetDefaultEncoding()** eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch **ByteLen()** und **CountStr()**.

**EINGABEN**

**str\$**            Eingabezeichenkette

**encoding**    optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

**RÜCKGABEWERTE**

**len**            Anzahl der Zeichen in **str\$**

**BEISPIEL**

```
len = StrLen("Hello")
```

```
Print(len)
```

Gibt "5" auf den Bildschirm aus.



## 57.57 StrStr

### BEZEICHNUNG

StrStr – konvertiert eine Zahl in eine Zeichenkette

### ÜBERSICHT

```
var$ = StrStr(value[, digits])
```

### BESCHREIBUNG

Konvertiert den numerischen Wert `value` in eine Zeichenfolge und gibt diese in `var$` zurück. Das optionale Argument `digits` ermöglicht es Ihnen, die Anzahl Dezimalstellen zu definieren, wenn der Wert `value` eine reelle Zahl ist. Der Standardwert ist 2.

Siehe auch [ToString\(\)](#), [ToNumber\(\)](#) und [Val\(\)](#).

### EINGABEN

<code>value</code>	umzuwandelnde Zahl
<code>digits</code>	optional: die Anzahl Dezimalstellen bei einer reellen Zahl (voreingestellt ist 2)

### RÜCKGABEWERTE

`var$`      Zeichenkette, die den numerischen Wert enthält

### BEISPIEL

```
test$=StrStr(256)
Print(test$)
```

Schreibt "256" auf den Bildschirm.

## 57.58 StrToArray

### BEZEICHNUNG

StrToArray – konvertiert eine Zeichenkette in eine Tabelle/Array von Codepunkten

### ÜBERSICHT

```
t = StrToArray(s$)
```

### BESCHREIBUNG

Dieser Befehl wandelt jedes einzelne Zeichen in der Zeichenkette `s$` in Codepunktwerte um, setzt diese in eine Tabelle und gibt sie zurück. Die Tabelle wird so viele Elemente enthalten, wie die Zeichenfolge Zeichen hat und dazu noch eine abschließende Null.

Um die Tabelle wieder in eine Zeichenkette zu wandeln, können Sie den Befehl [ArrayToStr\(\)](#) verwenden. Siehe [Abschnitt 57.2 \[ArrayToStr\]](#), [Seite 1253](#), für Details.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit [SetDefaultEncoding\(\)](#) eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

### EINGABEN

`s$`      Quellzeichenkette

**encoding** optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

## RÜCKGABEWERTE

**t** Tabelle, welche die Codepunktwerte jedes Zeichen beinhaltet

## BEISPIEL

```
t = StrToArray("Hello World")
DebugPrint(Chr(t[6]))
Gibt "W" aus.
```

# 57.59 ToNumber

## BEZEICHNUNG

ToNumber – konvertiert eine Zeichenkette in eine Zahl (V2.0)

## ÜBERSICHT

```
n = ToNumber(s$, base)
```

## BESCHREIBUNG

Dieser Befehl konvertiert die Zeichenfolge in **s\$** in eine Zahl. Das optionale Argument **base** kann verwendet werden, um z.B. Binär-, Oktal- und Hexzahlen zu konvertieren oder eine andere Basis im Bereich von 2 bis 36. Der Standardwert ist 10 (dezimal). Bei den Basis von 11 bis 36 werden die Buchstaben des englischen Alphabets als zusätzliche Ziffern (10 = A, 35 = Z) verwendet. Groß- und Kleinschreibung wird nicht benötigt.

Ab Hollywood 6.0 kann dieser Befehl auch eine Variable vom Typ **#LIGHTUSERDATA** in eine Zahl umwandeln. Da dieser Variabletyp Zeiger speichert, ist er nur für erfahrene Anwender oder Debug-Zwecke interessant.

Siehe auch **Val()**, **ToString()** und **StrStr()**.

## EINGABEN

**s\$** Zeichenkette zum Konvertieren

**base** optional: Basis des Zahlensystems (voreingestellt ist 10)

## RÜCKGABEWERTE

**number** konvertierte Zahl

## BEISPIEL

```
r = ToNumber("10000")           ; gibt 10000 zurück
r = ToNumber("10110111", 2)     ; gibt 183 zurück
r = ToNumber("523", 8)          ; gibt 339 zurück
r = ToNumber("FFFF", 16)        ; gibt 65535 zurück
```

# 57.60 ToString

## BEZEICHNUNG

ToString – konvertiert jeden Typ in eine Zeichenkette (2.0)

**ÜBERSICHT**

```
s$ = ToString(data)
```

**BESCHREIBUNG**

Dieser Befehl kann jeden Variablentyp in eine Zeichenfolge konvertieren. Sie können Tabellen, Funktionen, Zeichenketten, Zahlen und Nil benutzen. `ToString()` wird auch für die Befehle `Print()` und `DebugPrint()` verwendet, so dass sie alle Variablentypen auch ausgeben können.

Wenn Sie eine Tabelle mit der Metatabelle `__toString` übergeben, wird diese Metamethode aufgerufen.

Siehe auch `StrStr()`, `ToNumber()`, und `Val()`.

**EINGABEN**

`data`            Wert, der in eine Zeichenkette konvertiert wird

**RÜCKGABEWERTE**

`s$`            Zeichenkette

**BEISPIEL**

```
s$ = ToString(DisplayBrush) ; gibt "Function: 74cd2456" zurück
s$ = ToString({1,2,3,4,5})  ; gibt "Table: 74ab1344" zurück
s$ = ToString(Nil)          ; gibt "Nil" zurück
s$ = ToString(5)            ; gibt "5" zurück
s$ = ToString("Hello")      ; gibt "Hello" zurück
```

**57.61 ToUserData****BEZEICHNUNG**

`ToUserData` – konvertiert eine Zahl auf einen Benutzerdaten-Zeiger (V6.0)

**ÜBERSICHT**

```
ptr = ToUserData(val)
```

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um eine beliebige Zahl in eine Variable vom Typ `#LIGHTUSERDATA` zu konvertieren, die verwendet wird, um Speicherzeiger zu speichern. Sie könnten dann diese Variable an einen Befehl übergeben, die einen `#LIGHTUSERDATA` als Parameter erwartet.

Beachten Sie, dass dieser Befehl gefährlich ist und sollte nur von Personen verwendet werden, die wissen, was sie tun. Mit Zeiger, die nicht auf den zugewiesenen Adressraum verweisen, kann Ihr Skript leicht zum Absturz bringen.

Um einen Zeiger vom Typ `#LIGHTUSERDATA` zurück in eine Zahl umzuwandeln, verwenden Sie den Befehl `ToNumber()`. Siehe [Abschnitt 57.59 \[ToNumber\]](#), [Seite 1298](#), für Details.

**EINGABEN**

`val`            Zahlenwert, der in einen Zeiger vom Typ `#LIGHTUSERDATA` konvertiert wird

**RÜCKGABEWERTE**

`ptr`            Zeiger vom Typ `#LIGHTUSERDATA`

## 57.62 TrimStr

### BEZEICHNUNG

TrimStr – löscht Zeichen am Anfang oder Ende einer Zeichenkette (V2.0)

### ÜBERSICHT

```
s$ = TrimStr(src$, chr$, tail[, encoding])
```

### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um alle Zeichen am Anfang oder Ende von **src\$** zu löschen, die **chr\$** entsprechen. Die Zeichenfolge **chr\$** darf nur ein Zeichen enthalten. Um von rechts (hinten) zu starten, setzen Sie **tail** auf **True**, bei **False** wird von der linken Seite (vorne) begonnen. Die getrimmte Zeichenfolge wird zurückgegeben.

Mit dem optionalen Parameter **encoding** kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit **SetDefaultEncoding()** eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch [StripStr\(\)](#).

### EINGABEN

<b>src\$</b>	die zu trimmende Zeichenkette
<b>chr\$</b>	ein einzelnes Zeichen
<b>tail</b>	<b>True</b> beginnt von rechts/hinten oder <b>False</b> von links/vorne
<b>encoding</b>	optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

### RÜCKGABEWERTE

<b>s\$</b>	die getrimmte Zeichenkette
------------	----------------------------

### BEISPIEL

```
a$ = TrimStr("aaaaHello World", "a", False)
DebugPrint(a$)
Gibt "Hello World" aus.
```

```
a$ = TrimStr("aaaaHello Worldaaaa", "a", True)
DebugPrint(a$)
Gibt "aaaaHello World" aus.
```

## 57.63 UnleftStr

### BEZEICHNUNG

UnleftStr – löscht die rechten n Zeichen aus einer Zeichenkette

### ÜBERSICHT

```
var$ = UnleftStr(string$, len[, encoding])
```

**BESCHREIBUNG**

Löscht die rechten `len` Zeichen aus der Zeichenkette `string$` und gibt in `var$` die neue Zeichenkette zurück.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `UnmidStr()`, `UnrightStr()`, `LeftStr()`, `MidStr()`, `RightStr()`, `EndsWith()`, `StartsWith()`, `FindStr()` und `ReverseFindStr()`.

**EINGABEN**

`string$`    Zeichenkette, in der Zeichen gelöscht werden sollen

`len`        Anzahl zu löschender Zeichen

`encoding`   optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

**RÜCKGABEWERTE**

`var$`        nicht gelöschte Zeichenfolge

**BEISPIEL**

```
test$ = UnleftStr("Hello World!", 7)
```

```
Print(test$)
```

Das wird "Hello" ausgegeben.

## 57.64 UnmidStr

**BEZEICHNUNG**

`UnmidStr` – löscht die Zeichen aus der Mitte einer Zeichenkette (V4.5)

**ÜBERSICHT**

```
var$ = UnmidStr(s$, pos, len[, encoding])
```

**BESCHREIBUNG**

Dieser Befehl löscht `len` Zeichen in der Zeichenfolge `s$` beginnend an der Position `pos`. Diese Position muss in Zeichen und nicht in Bytes angegeben werden. Position 0 ist der Anfang der Zeichenkette. Die neue Zeichenkette wird in `var$` zurückgegeben.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch `UnleftStr()`, `UnrightStr()`, `LeftStr()`, `MidStr()`, `RightStr()`, `EndsWith()`, `StartsWith()`, `FindStr()` und `ReverseFindStr()`.

**EINGABEN**

`s$`            Zeichenkette, in der Zeichen gelöscht werden sollen

`pos`         Position (in Zeichen), ab der Zeichen entfernt werden (0 ist der Anfang)

**len**           Anzahl zu löschender Zeichen

**encoding**   optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

**RÜCKGABEWERTE**

**var\$**           nicht gelöschte Zeichenfolge

**BEISPIEL**

```
Print(UnmidStr("This is definitely not a funny example", 19, 4))
```

Herauszufinden, was der mysteriöse Aufruf oben tun könnte, liegt beim Leser.

**57.65 UnrightStr****BEZEICHNUNG**

UnrightStr – löscht die linken *n* Zeichen aus einer Zeichenkette

**ÜBERSICHT**

```
var$ = UnrightStr(string$, len[, encoding])
```

**BESCHREIBUNG**

Löscht die linken **len** Zeichen aus der Zeichenkette **string\$** und gibt in **var\$** die neue Zeichenkette zurück.

Mit dem optionalen Parameter **encoding** kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit **SetDefaultEncoding()** eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch [UnleftStr\(\)](#), [UnmidStr\(\)](#), [LeftStr\(\)](#), [MidStr\(\)](#), [RightStr\(\)](#), [EndsWith\(\)](#), [StartsWith\(\)](#), [FindStr\(\)](#) und [ReverseFindStr\(\)](#).

**EINGABEN**

**string\$**    Zeichenkette, in der Zeichen gelöscht werden sollen

**len**           Anzahl zu löschender Zeichen

**encoding**   optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

**RÜCKGABEWERTE**

**var\$**           nicht gelöschte Zeichen der Zeichenfolge **string\$**

**BEISPIEL**

```
test$ = UnrightStr("Hello World!", 6)
Print(test$)
```

Das wird "World!" ausgegeben.

## 57.66 UpperStr

### BEZEICHNUNG

UpperStr – konvertiert eine Zeichenkette in Großbuchstaben

### ÜBERSICHT

```
var$ = UpperStr(string$[, encoding])
```

### BESCHREIBUNG

Konvertiert alle Zeichen in `string$` in Großbuchstaben.

Mit dem optionalen Parameter `encoding` kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit `SetDefaultEncoding()` eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Siehe auch [LowerStr\(\)](#).

### EINGABEN

`string$`    umzuwandelnde Zeichenkette

`encoding`   optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung) (V7.0)

### RÜCKGABEWERTE

`var$`        umgewandelte Zeichenkette

### BEISPIEL

```
Print(UpperStr("Hello World!"))
```

Schreibt "HELLO WORLD!" auf den Bildschirm.

## 57.67 Val

### BEZEICHNUNG

Val – konvertiert eine Zeichenkette in eine Zahl

### ÜBERSICHT

```
var, chrs = Val(string$)
```

### BESCHREIBUNG

Konvertiert die Zahl in `string$` in die Variable `var`. Falls es keine Zahl in der Zeichenkette gibt, wird `var` auf 0 gesetzt. Die Zeichenkette kann auch hexadezimale Zahlen enthalten, sofern sie mit einem "\$" anfangen.

Ab Hollywood 2.0 können Sie auch eine Zeichenfolge übergeben, die nun eine binäre Zahl enthält. Sie müssen einfach ein Präfix mit dem "%" -Zeichen verwenden. Zusätzlich wird nun ein zweiter Wert zurückgegeben, der angibt, wie viele Zeichen `Val()` aus der Zeichenkette gelesen hat. Auf diese Weise können Sie die Länge der Zahl bestimmen.

Siehe auch [ToNumber\(\)](#), [ToString\(\)](#) und [StrStr\(\)](#).

### EINGABEN

`string$`    umzuwandelnde Zeichenkette

**RÜCKGABEWERTE**

**var**            Variable, die die umgewandelte Zahl enthält

**chrs**          Anzahl der Zeichen, die konvertiert wurden

**BEISPIEL**

```
result, chrs = Val("500 people were on the train.")
Print(result, "-", chrs)
```

Das gibt "500-3" aus.

**57.68 ValidateStr****BEZEICHNUNG**

ValidateStr – prüft, ob die Zeichenkette nur gültige Zeichen enthält (V7.0)

**ÜBERSICHT**

```
ok, n = ValidateStr(s$, encoding)
```

**BESCHREIBUNG**

Überprüft die von **s\$** angegebene Zeichenfolge und gibt **True** zurück, wenn die Zeichenkette nur gültige Zeichen enthält, andernfalls **False**. Der zweite Rückgabewert enthält die Anzahl der gültigen Zeichen in der Zeichenfolge. Wenn die Validierung erfolgreich ist, entspricht dies dem Ergebnis von **StrLen()**. Andernfalls wird Ihnen der Versatz des ersten ungültigen Zeichens in der Zeichenfolge angezeigt.

Mit dem optionalen Parameter **encoding** kann die zu verwendende Zeichencodierung festgelegt werden. Dies ist standardmäßig auf die Standardcodierung eingestellt, die mit **SetDefaultEncoding()** eingestellt ist. Siehe [Abschnitt 13.2 \[Zeichencodierungen\]](#), [Seite 158](#), für Details.

Dieser Befehl ist nur sinnvoll, wenn **#ENCODING\_UTF8** verwendet wird. Wenn die Codierung auf **#ENCODING\_ISO8859\_1** gesetzt ist, gibt dieser Befehl immer **True** zurück.

Siehe auch **IsAlNum()**, **IsAlpha()**, **IsDigit()**, **IsLower()**, **IsUpper()**, **IsGraph()**, **IsPrint()**, **IsPunct()**, **IsSpace()**, **IsXDigit()** und **IsCntrl()**.

**EINGABEN**

**s\$**            Quellzeichenkette

**encoding**    optional: Zeichencodierung, welche verwendet wird (voreingestellt ist die Standardcodierung)

**RÜCKGABEWERTE**

**ok**            boolescher Wert für Erfolg (**True**) oder Misserfolg (**False**)

**n**            Anzahl der gültigen Zeichen in der Zeichenfolge (V7.1)



## 58 Zwischenablagebibliothek

### 58.1 ClearClipboard

#### BEZEICHNUNG

ClearClipboard – löscht den Inhalt der Zwischenablage (V4.5)

#### ÜBERSICHT

ClearClipboard()

#### BESCHREIBUNG

Dieser Befehl kann verwendet werden, um die Zwischenablage zu leeren. Alle Daten, die in der Zwischenablage sind, werden gelöscht.

#### EINGABEN

keine

### 58.2 GetClipboard

#### BEZEICHNUNG

GetClipboard – liest den Inhalt der Zwischenablage (V4.5)

#### ÜBERSICHT

type[, data] = GetClipboard()

#### BESCHREIBUNG

Dieser Befehl ruft die Daten ab, die momentan in der Zwischenablage sind. GetClipboard() gibt zwei Werte zurück: Der erste Rückgabewert **type** gibt das Format der Daten in der Zwischenablage an und der zweite **data** enthält dann die formatspezifischen Daten. Derzeit unterstützt Hollywood zwei verschiedene Arten von Daten in der Zwischenablage: Text und Bilder.

Wenn momentan Text in der Zwischenablage gespeichert ist, gibt GetClipboard() **#CLIPBOARD\_TEXT** in **type** zurück und eine Zeichenfolge enthält den Text in der Zwischenablage als zweiten Rückgabewert **data**.

Wenn zur Zeit ein Bild in der Zwischenablage gespeichert ist, gibt GetClipboard() **#CLIPBOARD\_IMAGE** in **type** zurück und der zweite Rückgabewert **data** wird ein Pinsel sein, welcher das Bild aus der Zwischenablage enthält. Sobald Sie die Arbeit mit diesem Pinsel erledigt haben, sollten Sie **FreeBrush()** für diesen Pinsel aufrufen, um die Speicherzuweisung frei zu geben.

Wenn weder Text noch ein Bild in der Zwischenablage ist, gibt GetClipboard() **#CLIPBOARD\_UNKNOWN** zurück. Der zweite Rückgabewert wird in diesem Fall nicht benutzt. Wenn die Zwischenablage leer ist, dann wird **#CLIPBOARD\_EMPTY** zurückgegeben.

Um benachrichtigt zu werden, wenn sich der Inhalt der Zwischenablage ändert, können Sie den **ClipboardChange** Ereignis-Handler unter Verwendung von **InstallEventHandler()** installieren.

Wenn Sie nur das Format der Daten, die sich gegenwärtig in der Zwischenablage sind, herausfinden wollen, ohne tatsächlich eine Kopie dieser Daten zu empfangen, können Sie den Befehl `PeekClipboard()` benutzen. Aber bedenken Sie, dass die Daten in der Zwischenablage sich jederzeit ändern können. Es gibt also keine Garantie, dass die Daten in der Zwischenablage beim Aufruf von `PeekClipboard()` noch vorhanden sind, wenn Sie später/anschließend `GetClipboard()` aufrufen.

#### EINGABEN

keine

#### RÜCKGABEWERTE

<b>type</b>	Format der Daten in der Zwischenablage, <code>#CLIPBOARD_EMPTY</code> oder <code>#CLIPBOARD_UNKNOWN</code>
<b>data</b>	optional: Wenn der erste Rückgabewert nicht <code>#CLIPBOARD_EMPTY</code> oder <code>#CLIPBOARD_UNKNOWN</code> ist, dann enthält dieser Rückgabewert die tatsächlichen Daten, die aus der Zwischenablage abgerufen werden; Die hier zurückgegebenen Daten hängen von dem Format ab (siehe oben).

#### BEISPIEL

```
SetClipboard(#CLIPBOARD_TEXT, "Hello clipboard!")
type, data = GetClipboard()
If type = #CLIPBOARD_TEXT
    NPrint(data)
Else
    NPrint("No text on the clipboard!")
EndIf
```

Der obige Code setzt den Text "Hello clipboard!" in die Zwischenablage und ist dann der aktuelle Inhalt, der aus der Zwischenablage abgerufen werden kann. Wenn sich kein anderes Programm mit der Zwischenablage zwischen `SetClipboard()` und `GetClipboard()` vermischt, sollte dieser Code "Hello clipboard!" auf dem Bildschirm erscheinen.

## 58.3 PeekClipboard

#### BEZEICHNUNG

PeekClipboard – prüft den Inhalt der Zwischenablage (V4.5)

#### ÜBERSICHT

```
type = PeekClipboard()
```

#### BESCHREIBUNG

Dieser Befehl schaut in die Zwischenablage und gibt das Format der Daten in **type** zurück, die derzeit in der Zwischenablage gespeichert sind. Zur Zeit erkennt Hollywood nur Text- und Bilddaten in der Zwischenablage. Somit gibt dieser Befehl die folgenden Typen zurück:

**#CLIPBOARD\_TEXT:**

Dieser Text ist zur Zeit in der Zwischenablage.

**#CLIPBOARD\_IMAGE:**

Dieses Bild ist zur Zeit in der Zwischenablage.

**#CLIPBOARD\_EMPTY:**

Die Zwischenablage ist derzeit leer.

**#CLIPBOARD\_UNKNOWN:**

Hollywood erkennt nicht, welche Art von Daten aktuell in der Zwischenablage gespeichert sind.

Um die Daten aus der Zwischenablage abzurufen, müssen Sie den Befehl `GetClipboard()` verwenden. Beachten Sie aber, dass sich die Daten in der Zwischenablage jederzeit ändern können. So gibt es keine Garantie, dass die Daten in der Zwischenablage beim Aufruf von `PeekClipboard()` noch vorhanden sind, wenn Sie später/anschließend `GetClipboard()` aufrufen.

**EINGABEN**

keine

**RÜCKGABEWERTE**

**type**      Wert, der das Format der gegenwärtigen Daten in der Zwischenablage beschreibt

## 58.4 SetClipboard

**BEZEICHNUNG**

SetClipboard – kopiert Daten in die Zwischenablage (V4.5)

**ÜBERSICHT**

SetClipboard(**type**, **data**)

**BESCHREIBUNG**

Dieser Befehl kann verwendet werden, um neue Daten in die Zwischenablage zu kopieren. Der vorherige Inhalt der Zwischenablage wird gelöscht. Zur Zeit können Sie mit diesem Befehl entweder Text oder ein Bild in der Zwischenablage ablegen.

Um Text in die Zwischenablage zu kopieren, geben Sie in **type** `#CLIPBOARD_TEXT` und eine bestehende Zeichenkette in **data** an.

Um ein Bild in die Zwischenablage zu kopieren, geben Sie `#CLIPBOARD_IMAGE` als **type** und übergeben Sie eine ID des Pinsels mit dem Bild in **data** an.

**EINGABEN**

**type**      Format, welches die Daten für das zweite Argument angibt

**data**      hängt vom in **type** angegebenen Format ab; siehe oben für mehr Informationen

**BEISPIEL**

Siehe [Abschnitt 58.2 \[GetClipboard\]](#), Seite 1305.



## Anhang A Lizenzen

### A.1 Lua license

Lua 5.0 license

Copyright © 1994-2004 Tecgraf, PUC-Rio.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

### A.2 OpenCV license

Copyright © 2000, Intel Corporation, all rights reserved. Third party copyrights are property of their respective owners.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistribution's of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistribution's in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- The name of Intel Corporation may not be used to endorse or promote products derived from this software without specific prior written permission.

This software is provided by the copyright holders and contributors "as is" and any express or implied warranties, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose are disclaimed. In no event shall Intel or contributors be liable for any direct, indirect, incidental, special, exemplary, or consequential damages (including, but not limited to, procurement of substitute goods or services; loss of use, data, or profits; or business interruption) however caused and on any theory of liability, whether in contract, strict liability, or tort (including negligence or otherwise) arising in any way out of the use of this software, even if advised of the possibility of such damage.

### A.3 ImageMagick license

The authoritative ImageMagick license can be found at <http://www.imagemagick.org/script/license.php> and ImageMagick notices at <http://www.imagemagick.org/script/notice.php>.

Before we get to the text of the license lets just review what the license says in simple terms:

It allows you to:

- freely download and use ImageMagick software, in whole or in part, for personal, company internal, or commercial purposes;

- use ImageMagick software in packages or distributions that you create.

It forbids you to:

- redistribute any piece of ImageMagick-originated software without proper attribution;
- use any marks owned by ImageMagick Studio LLC in any way that might state or imply that ImageMagick Studio LLC endorses your distribution;
- use any marks owned by ImageMagick Studio LLC in any way that might state or imply that you created the ImageMagick software in question.

It requires you to:

- include a copy of the license in any redistribution you may make that includes ImageMagick software;
- provide clear attribution to ImageMagick Studio LLC for any distributions that include ImageMagick software.

It does not require you to:

- include the source of the ImageMagick software itself, or of any modifications you may have made to it, in any redistribution you may assemble that includes it;
- submit changes that you make to the software back to the ImageMagick Studio LLC (though such feedback is encouraged).

A few other clarifications include:

- ImageMagick is freely available without charge;
- you may include ImageMagick on a CD-ROM as long as you comply with the terms of the license;
- you can give modified code away for free or sell it under the terms of the ImageMagick license or distribute the result under a different license, but you need to acknowledge the use of the ImageMagick software;
- the license is compatible with the GPL.

The legally binding and authoritative terms and conditions for use, reproduction, and distribution of ImageMagick follow:

Copyright 1999-2009 ImageMagick Studio LLC, a non-profit organization dedicated to making software imaging solutions freely available.

#### 1. Definitions.

"License" shall mean the terms and conditions for use, reproduction, and distribution as defined by Sections 1 through 10 of this document.

"Licensor" shall mean the copyright owner or entity authorized by the copyright owner that is granting the License.

"Legal Entity" shall mean the union of the acting entity and all other entities that control, are controlled by, or are under common control with that entity. For the purposes of this definition, "control" means (i) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (ii) ownership of fifty percent (50%) or more of the outstanding shares, or (iii) beneficial ownership of such entity.

"You" (or "Your") shall mean an individual or Legal Entity exercising permissions granted by this License.

"Source" form shall mean the preferred form for making modifications, including but not limited to software source code, documentation source, and configuration files.

"Object" form shall mean any form resulting from mechanical transformation or translation of a Source form, including but not limited to compiled object code, generated documentation, and conversions to other media types.

"Work" shall mean the work of authorship, whether in Source or Object form, made available under the License, as indicated by a copyright notice that is included in or attached to the work (an example is provided in the Appendix below).

"Derivative Works" shall mean any work, whether in Source or Object form, that is based on (or derived from) the Work and for which the editorial revisions, annotations, elaborations, or other modifications represent, as a whole, an original work of authorship. For the purposes of this License, Derivative Works shall not include works that remain separable from, or merely link (or bind by name) to the interfaces of, the Work and Derivative Works thereof.

"Contribution" shall mean any work of authorship, including the original version of the Work and any modifications or additions to that Work or Derivative Works thereof, that is intentionally submitted to Licensor for inclusion in the Work by the copyright owner or by an individual or Legal Entity authorized to submit on behalf of the copyright owner. For the purposes of this definition, "submitted" means any form of electronic, verbal, or written communication intentionally sent to the Licensor by its copyright holder or its representatives, including but not limited to communication on electronic mailing lists, source code control systems, and issue tracking systems that are managed by, or on behalf of, the Licensor for the purpose of discussing and improving the Work, but excluding communication that is conspicuously marked or otherwise designated in writing by the copyright owner as "Not a Contribution."

"Contributor" shall mean Licensor and any individual or Legal Entity on behalf of whom a Contribution has been received by Licensor and subsequently incorporated within the Work.

2. Grant of Copyright License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable copyright license to reproduce, prepare Derivative Works of, publicly display, publicly perform, sublicense, and distribute the Work and such Derivative Works in Source or Object form.

3. Grant of Patent License. Subject to the terms and conditions of this License, each Contributor hereby grants to You a perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work, where such license applies only to those patent claims licensable by such Contributor that are necessarily infringed by their Contribution(s) alone or by combination of their Contribution(s) with the Work to which such Contribution(s) was submitted.

4. Redistribution. You may reproduce and distribute copies of the Work or Derivative Works thereof in any medium, with or without modifications, and in Source or Object form, provided that You meet the following conditions:

- a. You must give any other recipients of the Work or Derivative Works a copy of this License; and
- b. You must cause any modified files to carry prominent notices stating that You changed the files; and

c. You must retain, in the Source form of any Derivative Works that You distribute, all copyright, patent, trademark, and attribution notices from the Source form of the Work, excluding those notices that do not pertain to any part of the Derivative Works; and

d. If the Work includes a "NOTICE" text file as part of its distribution, then any Derivative Works that You distribute must include a readable copy of the attribution notices contained within such NOTICE file, excluding those notices that do not pertain to any part of the Derivative Works, in at least one of the following places: within a NOTICE text file distributed as part of the Derivative Works; within the Source form or documentation, if provided along with the Derivative Works; or, within a display generated by the Derivative Works, if and wherever such third-party notices normally appear. You may add Your own attribution notices within Derivative Works that You distribute, alongside or as an addendum to the NOTICE text from the Work.

You may add Your own copyright statement to Your modifications and may provide additional or different license terms and conditions for use, reproduction, or distribution of Your modifications, or for any such Derivative Works as a whole, provided Your use, reproduction, and distribution of the Work otherwise complies with the conditions stated in this License.

5. Submission of Contributions. Unless You explicitly state otherwise, any Contribution intentionally submitted for inclusion in the Work by You to the Licensor shall be under the terms and conditions of this License, without any additional terms or conditions. Notwithstanding the above, nothing herein shall supersede or modify the terms of any separate license agreement you may have executed with Licensor regarding such Contributions.

6. Trademarks. This License does not grant permission to use the trade names, trademarks, service marks, or product names of the Licensor, except as required for reasonable and customary use in describing the origin of the Work and reproducing the content of the NOTICE file.

7. Disclaimer of Warranty. Unless required by applicable law or agreed to in writing, Licensor provides the Work (and each Contributor provides its Contributions) on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied, including, without limitation, any warranties or conditions of TITLE, NON-INFRINGEMENT, MERCHANTABILITY, or FITNESS FOR A PARTICULAR PURPOSE. You are solely responsible for determining the appropriateness of using or redistributing the Work and assume any risks associated with Your exercise of permissions under this License.

8. Limitation of Liability. In no event and under no legal theory, whether in tort (including negligence), contract, or otherwise, unless required by applicable law (such as deliberate and grossly negligent acts) or agreed to in writing, shall any Contributor be liable to You for damages, including any direct, indirect, special, incidental, or consequential damages of any character arising as a result of this License or out of the use or inability to use the Work (including but not limited to damages for loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses), even if such Contributor has been advised of the possibility of such damages.

9. Accepting Warranty or Additional Liability. While redistributing the Work or Derivative Works thereof, You may choose to offer, and charge a fee for, acceptance of support, warranty, indemnity, or other liability obligations and/or rights consistent with this License.



## A.4 GD Graphics Library license

Portions copyright 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002 by Cold Spring Harbor Laboratory. Funded under Grant P41-RR02188 by the National Institutes of Health.

Portions copyright 1996, 1997, 1998, 1999, 2000, 2001, 2002 by Boutell.Com, Inc.

Portions relating to GD2 format copyright 1999, 2000, 2001, 2002 Philip Warner.

Portions relating to PNG copyright 1999, 2000, 2001, 2002 Greg Roelofs.

Portions relating to gdttf.c copyright 1999, 2000, 2001, 2002 John Ellson ([ellson@lucent.com](mailto:ellson@lucent.com)).

Portions relating to gdft.c copyright 2001, 2002 John Ellson ([ellson@lucent.com](mailto:ellson@lucent.com)).

Portions copyright 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007 Pierre-Alain Joye ([pierre@libgd.org](mailto:pierre@libgd.org)).

Portions relating to JPEG and to color quantization copyright 2000, 2001, 2002, Doug Becker and copyright (C) 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002, Thomas G. Lane. This software is based in part on the work of the Independent JPEG Group. See the file README-JPEG.TXT for more information.

Portions relating to WBMP copyright 2000, 2001, 2002 Maurice Szmurlo and Johan Van den Brande.

Permission has been granted to copy, distribute and modify gd in any context without fee, including a commercial application, provided that this notice is present in user-accessible supporting documentation.

This does not affect your ownership of the derived work itself, and the intent is to assure proper credit for the authors of gd, not to interfere with your productive use of gd. If you have questions, ask. "Derived works" includes all programs that utilize the library. Credit must be given in user-accessible documentation.

This software is provided "AS IS." The copyright holders disclaim all warranties, either express or implied, including but not limited to implied warranties of merchantability and fitness for a particular purpose, with respect to this code and accompanying documentation.

Although their code does not appear in gd, the authors wish to thank David Koblas, David Rowley, and Hutchison Avenue Software Corporation for their prior contributions.

## A.5 Bitstream Vera fonts license

The fonts have a generous copyright, allowing derivative works (as long as "Bitstream" or "Vera" are not in the names), and full redistribution (so long as they are not \*sold\* by themselves). They can be bundled, redistributed and sold with any software.

The fonts are distributed under the following copyright:

Copyright © 2003 by Bitstream, Inc. All Rights Reserved. Bitstream Vera is a trademark of Bitstream, Inc.

Permission is hereby granted, free of charge, to any person obtaining a copy of the fonts accompanying this license ("Fonts") and associated documentation files (the "Font Software"), to reproduce and distribute the Font Software, including without limitation the rights to use, copy, merge, publish, distribute, and/or sell copies of the Font Software, and to permit persons to whom the Font Software is furnished to do so, subject to the following conditions:

The above copyright and trademark notices and this permission notice shall be included in all copies of one or more of the Font Software typefaces.

The Font Software may be modified, altered, or added to, and in particular the designs of glyphs or characters in the Fonts may be modified and additional glyphs or characters may be added to the Fonts, only if the fonts are renamed to names not containing either the words "Bitstream" or the word "Vera".

This License becomes null and void to the extent applicable to Fonts or Font Software that has been modified and is distributed under the "Bitstream Vera" names.

The Font Software may be sold as part of a larger software package but no copy of one or more of the Font Software typefaces may be sold by itself.

THE FONT SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OF COPYRIGHT, PATENT, TRADEMARK, OR OTHER RIGHT. IN NO EVENT SHALL BITSTREAM OR THE GNOME FOUNDATION BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, INCLUDING ANY GENERAL, SPECIAL, INDIRECT, INCIDENTAL, OR CONSEQUENTIAL DAMAGES, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF THE USE OR INABILITY TO USE THE FONT SOFTWARE OR FROM OTHER DEALINGS IN THE FONT SOFTWARE.

Except as contained in this notice, the names of Gnome, the Gnome Foundation, and Bitstream Inc., shall not be used in advertising or otherwise to promote the sale, use or other dealings in this Font Software without prior written authorization from the Gnome Foundation or Bitstream Inc., respectively. For further information, contact: fonts at gnome dot org.

## A.6 Pixman license

The following is the MIT license, agreed upon by most contributors. Copyright holders of new code should use this license statement where possible. They may also add themselves to the list below.

Copyright 1987, 1988, 1989, 1998 The Open Group

Copyright 1987, 1988, 1989 Digital Equipment Corporation

Copyright 1999, 2004, 2008 Keith Packard

Copyright 2000 SuSE, Inc.

Copyright 2000 Keith Packard, member of The XFree86 Project, Inc.

Copyright 2004, 2005, 2007, 2008, 2009, 2010 Red Hat, Inc.

Copyright 2004 Nicholas Miell

Copyright 2005 Lars Knoll & Zack Rusin, Trolltech

Copyright 2005 Trolltech AS

Copyright 2007 Luca Barbato

Copyright 2008 Aaron Plattner, NVIDIA Corporation

Copyright 2008 Rodrigo Kumpera

Copyright 2008 Andrea Tupinambai

Copyright 2008 Mozilla Corporation

Copyright 2008 Frederic Plourde

Copyright 2009, Oracle and/or its affiliates. All rights reserved.  
Copyright 2009, 2010 Nokia Corporation

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice (including the next paragraph) shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## A.7 LuaSocket license

LuaSocket 3.0. Copyright © 2004-2013 Diego Nehab

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## A.8 librs232 license

Copyright (c) 2011 Petr Stetiar <ynezz@true.cz>, Gaben Ltd.

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## A.9 UsbSerial license

Copyright (c) 2014 Felipe Herranz

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

## A.10 SDL license

Simple DirectMedia Layer

Copyright (C) 1997-2019 Sam Lantinga <slouken@libsdl.org>

This software is provided 'as-is', without any express or implied warranty. In no event will the authors be held liable for any damages arising from the use of this software.

Permission is granted to anyone to use this software for any purpose, including commercial applications, and to alter it and redistribute it freely, subject to the following restrictions:

1. The origin of this software must not be misrepresented; you must not claim that you wrote the original software. If you use this software in a product, an acknowledgment in the product documentation would be appreciated but is not required.
2. Altered source versions must be plainly marked as such, and must not be misrepresented as being the original software.
3. This notice may not be removed or altered from any source distribution.

## A.11 LGPL license

GNU LESSER GENERAL PUBLIC LICENSE Version 3, 29 June 2007

Copyright (C) 2007 Free Software Foundation, Inc. <<http://fsf.org/>> Everyone is permitted to copy and distribute verbatim copies of this license document, but changing it is not allowed.

This version of the GNU Lesser General Public License incorporates the terms and conditions of version 3 of the GNU General Public License, supplemented by the additional permissions listed below.

### 0. Additional Definitions.

As used herein, "this License" refers to version 3 of the GNU Lesser General Public License, and the "GNU GPL" refers to version 3 of the GNU General Public License.

"The Library" refers to a covered work governed by this License, other than an Application or a Combined Work as defined below.

An "Application" is any work that makes use of an interface provided by the Library, but which is not otherwise based on the Library. Defining a subclass of a class defined by the Library is deemed a mode of using an interface provided by the Library.

A "Combined Work" is a work produced by combining or linking an Application with the Library. The particular version of the Library with which the Combined Work was made is also called the "Linked Version".

The "Minimal Corresponding Source" for a Combined Work means the Corresponding Source for the Combined Work, excluding any source code for portions of the Combined Work that, considered in isolation, are based on the Application, and not on the Linked Version.

The "Corresponding Application Code" for a Combined Work means the object code and/or source code for the Application, including any data and utility programs needed for reproducing the Combined Work from the Application, but excluding the System Libraries of the Combined Work.

### 1. Exception to Section 3 of the GNU GPL.

You may convey a covered work under sections 3 and 4 of this License without being bound by section 3 of the GNU GPL.

### 2. Conveying Modified Versions.

If you modify a copy of the Library, and, in your modifications, a facility refers to a function or data to be supplied by an Application that uses the facility (other than as an argument passed when the facility is invoked), then you may convey a copy of the modified version:

- a) under this License, provided that you make a good faith effort to ensure that, in the event an Application does not supply the function or data, the facility still operates, and performs whatever part of its purpose remains meaningful, or
- b) under the GNU GPL, with none of the additional permissions of this License applicable to that copy.

### 3. Object Code Incorporating Material from Library Header Files.

The object code form of an Application may incorporate material from a header file that is part of the Library. You may convey such object code under terms of your choice, provided that, if the incorporated material is not limited to numerical parameters, data structure

layouts and accessors, or small macros, inline functions and templates (ten or fewer lines in length), you do both of the following:

- a) Give prominent notice with each copy of the object code that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the object code with a copy of the GNU GPL and this license document.

#### 4. Combined Works.

You may convey a Combined Work under terms of your choice that, taken together, effectively do not restrict modification of the portions of the Library contained in the Combined Work and reverse engineering for debugging such modifications, if you also do each of the following:

- a) Give prominent notice with each copy of the Combined Work that the Library is used in it and that the Library and its use are covered by this License.
- b) Accompany the Combined Work with a copy of the GNU GPL and this license document.
- c) For a Combined Work that displays copyright notices during execution, include the copyright notice for the Library among these notices, as well as a reference directing the user to the copies of the GNU GPL and this license document.

d) Do one of the following:

0) Convey the Minimal Corresponding Source under the terms of this License, and the Corresponding Application Code in a form suitable for, and under terms that permit, the user to recombine or relink the Application with a modified version of the Linked Version to produce a modified Combined Work, in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.

1) Use a suitable shared library mechanism for linking with the Library. A suitable mechanism is one that (a) uses at run time a copy of the Library already present on the user's computer system, and (b) will operate properly with a modified version of the Library that is interface-compatible with the Linked Version.

e) Provide Installation Information, but only if you would otherwise be required to provide such information under section 6 of the GNU GPL, and only to the extent that such information is necessary to install and execute a modified version of the Combined Work produced by recombining or relinking the Application with a modified version of the Linked Version. (If you use option 4d0, the Installation Information must accompany the Minimal Corresponding Source and Corresponding Application Code. If you use option 4d1, you must provide the Installation Information in the manner specified by section 6 of the GNU GPL for conveying Corresponding Source.)

#### 5. Combined Libraries.

You may place library facilities that are a work based on the Library side by side in a single library together with other library facilities that are not Applications and are not covered by this License, and convey such a combined library under terms of your choice, if you do both of the following:

- a) Accompany the combined library with a copy of the same work based on the Library, uncombined with any other library facilities, conveyed under the terms of this License.
- b) Give prominent notice with the combined library that part of it is a work based on the Library, and explaining where to find the accompanying uncombined form of the same work.

#### 6. Revised Versions of the GNU Lesser General Public License.

The Free Software Foundation may publish revised and/or new versions of the GNU Lesser General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Library as you received it specifies that a certain numbered version of the GNU Lesser General Public License "or any later version" applies to it, you have the option of following the terms and conditions either of that published version or of any later version published by the Free Software Foundation. If the Library as you received it does not specify a version number of the GNU Lesser General Public License, you may choose any version of the GNU Lesser General Public License ever published by the Free Software Foundation.

If the Library as you received it specifies that a proxy can decide whether future versions of the GNU Lesser General Public License shall apply, that proxy's public statement of acceptance of any version is permanent authorization for you to choose that version for the Library.





# Index

## A

Abs	715
ACos	715
ActivateDisplay	360
ACTIVEWINDOW	1209
Add	715
AddArcToPath	1183
AddBoxToPath	1183
AddCircleToPath	1184
AddEllipseToPath	1185
AddFontPath	1137
AddIconImage	879
AddMove	407
AddStr	1253
AddTab	1138
AddTextToPath	1185
AllocConsoleColor	653
AllocMem	1061
AllocMemFromPointer	1061
AllocMemFromVirtualFile	1062
ANIM	185
APPAUTHOR	217
APPCOPYRIGHT	217
APPDESCRIPTION	217
AppendPath	1187
APPENTRY	218
APPICON	219
APPIDENTIFIER	222
APPTITLE	222
APPVERSION	223
Arc	597
ArcDistortBrush	905
ARGB	567
ArrayToStr	1253
Asc	1254
ASin	716
Assert	337
AsyncDrawFrame	231
ATan	716
ATan2	717

## B

BACKFILL	360
BarrelDistortBrush	906
Base64Str	1255
Beep	1089
BeepConsole	653
BeginAnimStream	189
BeginDoubleBuffer	567
BeginRefresh	569
BGPIC	619
BGPicToBrush	907
BinStr	1255

BitClear	717
BitComplement	718
BitSet	718
BitTest	719
BitXor	719
Blue	571
BlurBrush	907
Box	598
BreakEventHandler	477
BreakWhileMouseOn	1209
BRUSH	908
BrushToBGPic	622
BrushToGray	911
BrushToMonochrome	912
BrushToPenArray	912
BrushToRGBArray	913
ByteAsc	1256
ByteChr	1256
ByteLen	1257
ByteOffset	1257
ByteStrStr	1258
ByteVal	1259

## C

CallJavaMethod	761
CancelAsyncDraw	233
CancelAsyncOperation	233
CanonizePath	237
Cast	720
CATALOG	696
Ceil	720
ChangeApplicationIcon	880
ChangeBrushTransparency	914
ChangeDirectory	237
ChangeDisplayMode	362
ChangeDisplaySize	364
ChangeInterval	477
CharcoalBrush	915
CharOffset	1260
CharWidth	1261
CheckEvent	477
CheckEvents	478
Chr	1262
Circle	599
ClearClipboard	1305
ClearConsole	654
ClearConsoleStyle	654
ClearEvents	1210
ClearInterval	479
ClearMove	409
ClearObjectData	807
ClearPath	1187
ClearScreen	571
ClearSerialQueue	1005



DirectoryItems .....	256
DisableAdvancedConsole .....	660
DisableButton .....	481
DisableEvent .....	1214
DisableEventHandler .....	1215
DisableLayers .....	412
DisableLineHook .....	1090
DisableMenuItem .....	751
DisablePlugin .....	985
DisablePrecalculation .....	575
DisableVWait .....	576
DISPLAY .....	370
DisplayAnimFrame .....	193
DisplayBGPic .....	627
DisplayBGPicPart .....	628
DisplayBGPicPartFX .....	629
DisplayBrush .....	930
DisplayBrushFX .....	931
DisplayBrushPart .....	932
DisplaySprite .....	1078
DisplayTextObject .....	1145
DisplayTextObjectFX .....	1146
DisplayTransitionFX .....	630
DisplayVideoFrame .....	1236
Div .....	722
DoMove .....	412
DownloadFile .....	773
DrawConsoleBorder .....	661
DrawConsoleBox .....	662
DrawConsoleHLine .....	663
DrawConsoleVLine .....	663
DrawPath .....	1189
DumpLayers .....	413
DumpMem .....	1064

## E

EdgeBrush .....	933
Ellipse .....	601
ELSE .....	1090
ELSEIF .....	1091
EmbossBrush .....	933
EmptyStr .....	1265
EnableAdvancedConsole .....	664
EnableButton .....	481
EnableEvent .....	1216
EnableEventHandler .....	1215
EnableLayers .....	414
EnableLineHook .....	1091
EnableMenuItem .....	751
EnablePlugin .....	985
EnablePrecalculation .....	576
EnableVWait .....	576
End .....	1092
EndDoubleBuffer .....	577
EndianSwap .....	722
ENDIF .....	1092
EndRefresh .....	577

EndSelect .....	934
EndsWith .....	1265
Eof .....	258
EraseConsole .....	665
Error .....	523
ERROR .....	523
EscapeQuit .....	481
Eval .....	1266
Execute .....	258
Exists .....	261
ExitOnError .....	523
Exp .....	723
ExtendBrush .....	934
ExtractPalette .....	847

## F

FILE .....	262
FileAttributes .....	263
FileLength .....	264
FileLines .....	265
FilePart .....	266
FilePos .....	266
FileRequest .....	344
FileSize .....	267
FileToString .....	267
FillMem .....	1064
FillMusicBuffer .....	1026
FindStr .....	1268
FinishAnimStream .....	194
FinishAsyncDraw .....	235
FlashConsole .....	665
Flip .....	578
FlipBrush .....	935
FlipSprite .....	1078
FloodFill .....	936
Floor .....	723
FlushFile .....	268
FlushMusicBuffer .....	1028
FlushSerialPort .....	1005
FONT .....	1147
FontRequest .....	346
ForcePathUse .....	1191
ForceSound .....	1029
ForceVideoDriver .....	1236
ForEach .....	1121
ForEachI .....	1122
FormatConsoleLine .....	666
FormatDate .....	697
FormatNumber .....	1269
FormatStr .....	1270
Frac .....	723
FreeAnim .....	194
FreeBGPic .....	634
FreeBrush .....	937
FreeClipRegion .....	578
FreeConsoleColor .....	667
FreeConsoleWindow .....	667

FreeDisplay .....	387
FreeGlyphCache .....	1148
FreeIcon .....	884
FreeLayers .....	415
FreeMem .....	1065
FreeMenu .....	752
FreeModule .....	1030
FreePalette .....	848
FreePath .....	1191
FreePointer .....	746
FreeSample .....	1030
FreeSprite .....	1079
FreeTextObject .....	1149
FrExp .....	724
FullPath .....	269

## G

GammaBrush .....	938
GammaPalette .....	848
GCInfo .....	1093
GetAllocConsoleColor .....	668
GetAnimFrame .....	194
GetApplicationInfo .....	224
GetApplicationList .....	175
GetAsset .....	763
GetAttribute .....	808
GetAvailableFonts .....	1149
GetBaudRate .....	1006
GetBestPen .....	849
GetBrushLink .....	938
GetBrushPen .....	940
GetBulletColor .....	1151
GetCatalogString .....	699
GetChannels .....	1030
GetCharMaps .....	1151
GetClipboard .....	1305
GetCommandLine .....	225
GetConnectionIP .....	779
GetConnectionPort .....	780
GetConnectionProtocol .....	781
GetConsoleBackground .....	668
GetConsoleChr .....	669
GetConsoleColor .....	669
GetConsoleControlChr .....	670
GetConsoleCursor .....	671
GetConsoleOrigin .....	671
GetConsoleSize .....	672
GetConsoleStr .....	672
GetConsoleStyle .....	673
GetConsoleWindow .....	673
GetConstant .....	1093
GetCountryInfo .....	699
GetCurrentDirectory .....	269
GetCurrentPoint .....	1192
GetDash .....	1192
GetDataBits .....	1007
GetDate .....	323

GetDateNum .....	324
GetDefaultAdapter .....	1094
GetDefaultEncoding .....	1152
GetDefaultLoader .....	1094
GetDirectoryEntry .....	270
GetDisplayModes .....	387
GetDTR .....	1007
GetEnv .....	271
GetErrorName .....	562
GetEventCode .....	1216
GetFileArgument .....	226
GetFileAttributes .....	271
GetFillRule .....	1193
GetFillStyle .....	602
GetFlowControl .....	1008
GetFontColor .....	1152
GetFontStyle .....	1153
GetFormStyle .....	603
GetFPSLimit .....	579
GetFreePen .....	849
GetFrontScreen .....	176
GetHostName .....	782
GetIconProperties .....	884
GetItem .....	1123
GetKerningPair .....	1154
GetLanguageInfo .....	700
GetLastError .....	563
GetLayerAtPos .....	415
GetLayerGroupMembers .....	416
GetLayerGroups .....	416
GetLayerPen .....	417
GetLayerStyle .....	417
GetLineCap .....	1193
GetLineJoin .....	1193
GetLineWidth .....	604
GetLocaleInfo .....	701
GetLocalInterfaces .....	782
GetLocalIP .....	783
GetLocalPort .....	784
GetLocalProtocol .....	785
GetMACAddress .....	786
GetMemoryInfo .....	1094
GetMemPointer .....	1066
GetMemString .....	1066
GetMetaTable .....	1123
GetMiterLimit .....	1194
GetMonitors .....	388
GetObjectData .....	836
GetObjects .....	836
GetObjectType .....	837
GetPalettePen .....	850
GetParity .....	1008
GetPathExtents .....	1194
GetPatternPosition .....	1031
GetPen .....	851
GetPlugins .....	985
GetProgramDirectory .....	273
GetProgramInfo .....	226

GetPubScreens .....	177
GetRandomColor .....	579
GetRandomFX .....	580
GetRawArguments .....	227
GetRealColor .....	581
GetRTS .....	1009
GetSampleData .....	1031
GetSerializeMode .....	993
GetShortcutPath .....	1249
GetSongPosition .....	1032
GetStartDirectory .....	274
GetStopBits .....	1009
GetSystemCountry .....	702
GetSystemInfo .....	1095
GetSystemLanguage .....	707
GetTempFileName .....	274
GetTime .....	325
GetTimer .....	325
GetTimestamp .....	326
GetTimeZone .....	327
GetType .....	1097
GetVersion .....	1097
GetVideoFrame .....	1237
GetVolumeInfo .....	275
GetVolumeName .....	276
GetWeekday .....	327
Gosub .....	1217
Goto .....	1218
GrabDesktop .....	581
Green .....	582
GroupLayer .....	418

## H

HaveConsole .....	674
HaveFreeChannel .....	1033
HaveItem .....	1124
HaveObject .....	837
HaveObjectData .....	838
HavePlugin .....	988
HaveVolume .....	276
HexStr .....	1271
HideConsoleCursor .....	674
HideDisplay .....	388
HideKeyboard .....	764
HideLayer .....	420
HideLayerFX .....	420
HidePointer .....	746
HideScreen .....	178
Hypot .....	724

## I

ICON .....	888
IF .....	1099
IgnoreCase .....	1271
IIf .....	1101
ImageRequest .....	349
INACTIVEWINDOW .....	1218
INCLUDE .....	1102
IncreasePointer .....	1067
InitConsoleColor .....	675
InKeyStr .....	482
InsertConsoleChr .....	676
InsertConsoleLine .....	677
InsertConsoleStr .....	677
InsertItem .....	1124
InsertLayer .....	422
InsertSample .....	1033
InsertStr .....	1272
InstallEventHandler .....	483
Int .....	725
Intersection .....	582
InvertAlphaChannel .....	941
InvertBrush .....	942
InvertMask .....	942
InvertPalette .....	852
IPairs .....	1125
IsAbsolutePath .....	277
IsA1Num .....	1273
IsAlpha .....	1274
IsAnim .....	195
IsAnimPlaying .....	196
IsBrushEmpty .....	943
IsChannelPlaying .....	1034
IsCntrl .....	1274
IsDigit .....	1275
IsDirectory .....	277
IsFinite .....	725
IsGraph .....	1275
IsInf .....	726
IsKeyDown .....	503
IsLeftMouse .....	505
IsLower .....	1276
IsMenuItemDisabled .....	752
IsMenuItemSelected .....	753
IsMidMouse .....	505
IsModule .....	1035
IsMusic .....	1036
IsMusicPlaying .....	1035
IsNan .....	726
IsNil .....	1103
IsOnline .....	786
IsPathEmpty .....	1195
IsPicture .....	583
IsPrint .....	1277
IsPunct .....	1277
IsRightMouse .....	506
IsSample .....	1037
IsSamplePlaying .....	1037

IsSound .....	1038
IsSpace .....	1278
IsTableEmpty .....	1126
IsUnicode .....	1103
IsUpper .....	1279
IsVideo .....	1238
IsVideoPlaying .....	1239
IsXDigit .....	1279

## J

JoyAxisX .....	647
JoyAxisY .....	648
JoyAxisZ .....	649
JoyButton .....	649
JoyDir .....	650
JoyHat .....	651

## L

Label .....	1219
LayerExists .....	423
LayerGroupExists .....	423
LayerToBack .....	424
LayerToFront .....	424
Ld .....	727
LdExp .....	728
LeftMouseQuit .....	506
LeftStr .....	1280
LegacyControl .....	1104
Limit .....	728
Line .....	605
LineTo .....	1195
LINKER .....	1105
ListItems .....	1126
ListRequest .....	350
Ln .....	729
LoadAnim .....	197
LoadAnimFrame .....	200
LoadBGPic .....	634
LoadBrush .....	943
LoadIcon .....	892
LoadModule .....	1039
LoadPalette .....	852
LoadPlugin .....	989
LoadPrefs .....	228
LoadSample .....	1039
LoadSprite .....	1079
Locate .....	1154
Log .....	729
LowerStr .....	1281

## M

MakeButton .....	507
MakeConsoleChr .....	678
MakeDate .....	328
MakeDirectory .....	278
MakeHostPath .....	279
MatchPattern .....	280
Matrix2D .....	586
Max .....	729
MD5 .....	281
MD5Str .....	1281
MemToTable .....	1067
MENU .....	754
MergeLayers .....	425
MidStr .....	1282
Min .....	730
MixBrush .....	947
MixRGB .....	586
MixSample .....	1041
Mod .....	731
ModifyAnimFrames .....	202
ModifyButton .....	1219
ModifyKeyDown .....	1220
ModifyLayerFrames .....	427
ModulateBrush .....	948
ModulatePalette .....	854
MonitorDirectory .....	281
MouseX .....	511
MouseY .....	511
MoveAnim .....	202
MoveBrush .....	948
MoveConsoleWindow .....	680
MoveDisplay .....	390
MoveFile .....	283
MoveLayer .....	428
MovePointer .....	747
MoveSprite .....	1082
MoveTextObject .....	1155
MoveTo .....	1195
MOVEWINDOW .....	1220
Mul .....	731
MUSIC .....	1043

## N

NearlyEqual .....	732
NextDirectoryEntry .....	288
NextFrame .....	429
NextItem .....	1127
NormalizePath .....	1196
NPrint .....	1156

## O

OilPaintBrush .....	950
ONBUTTONCLICK .....	1220
ONBUTTONCLICKALL .....	1221
ONBUTTONOVER .....	1222
ONBUTTONOVERALL .....	1223
ONBUTTONRIGHTCLICK .....	1223
ONBUTTONRIGHTCLICKALL .....	1224
ONJOYDOWN .....	1224
ONJOYDOWNLEFT .....	1224
ONJOYDOWNRIGHT .....	1225
ONJOYFIRE .....	1225
ONJOYLEFT .....	1226
ONJOYRIGHT .....	1226
ONJOYUP .....	1226
ONJOYUPLIFT .....	1228
ONJOYUPRIGHT .....	1228
ONKEYDOWN .....	1228
ONKEYDOWNALL .....	1229
OpenAmigaGuide .....	178
OpenAnim .....	203
OpenAudio .....	1044
OpenCatalog .....	712
OpenConnection .....	787
OpenConsole .....	681
OpenDirectory .....	289
OpenDisplay .....	390
OpenFile .....	290
OpenFont .....	1156
OpenMusic .....	1045
OpenResourceMonitor .....	340
OpenSerialPort .....	1010
OpenURL .....	1106
OpenVideo .....	1240
OPTIONS .....	1106

## P

Pack .....	1128
PadNum .....	1283
Pairs .....	1129
PALETTE .....	854
PaletteToGray .....	856
ParseDate .....	329
PathItems .....	1196
PathPart .....	292
PathRequest .....	351
PathToBrush .....	1199
PatternFindStr .....	1283
PatternFindStrDirect .....	1284
PatternFindStrShort .....	1286
PatternReplaceStr .....	1286
PauseLayer .....	429
PauseModule .....	1046
PauseMusic .....	1046
PauseTimer .....	329
PauseVideo .....	1241
Peek .....	1068

PeekClipboard .....	1306
PenArrayToBrush .....	950
PerformSelector .....	764
PermissionRequest .....	353
PerspectiveDistortBrush .....	952
Pi .....	732
PixelateBrush .....	953
PlayAnim .....	206
PlayAnimDisk .....	207
PlayLayer .....	430
PlayModule .....	1047
PlayMusic .....	1047
PlaySample .....	1048
PlaySubsong .....	1050
PlayVideo .....	1242
Plot .....	606
Poke .....	1070
PolarDistortBrush .....	953
PollSerialQueue .....	1013
Polygon .....	607
PopupMenu .....	758
Pow .....	733
Print .....	1157

## Q

QuantizeBrush .....	954
---------------------	-----

## R

Rad .....	733
RaiseOnError .....	564
RasterizeBrush .....	955
RawDiv .....	733
RawEqual .....	1129
RawGet .....	1130
RawSet .....	1131
ReadBrushPixel .....	956
ReadByte .....	292
ReadBytes .....	293
ReadChr .....	294
ReadConsoleKey .....	681
ReadConsoleStr .....	683
ReadDirectory .....	294
ReadFloat .....	295
ReadFunction .....	296
ReadInt .....	297
ReadLine .....	298
ReadMem .....	1070
ReadPen .....	857
ReadPixel .....	608
ReadRegistryKey .....	1250
ReadSerialData .....	1013
ReadShort .....	298
ReadString .....	299
ReadTable .....	994
ReceiveData .....	789
ReceiveUDPData .....	791

Red.....	587
ReduceAlphaChannel.....	957
RefreshConsole.....	684
RefreshDisplay.....	394
RefreshLayer.....	430
RelCurveTo.....	1201
RelLineTo.....	1201
RelMoveTo.....	1202
RemapBrush.....	957
RemoveBrushPalette.....	958
RemoveButton.....	1229
RemoveIconImage.....	894
RemoveItem.....	1132
RemoveKeyDown.....	1230
RemoveLayer.....	431
RemoveLayerFX.....	431
RemoveLayers.....	433
RemoveSprite.....	1082
RemoveSprites.....	1083
Rename.....	300
RenderLayer.....	433
RepeatStr.....	1289
ReplaceColors.....	959
ReplaceStr.....	1290
REQUIRE.....	990
ResetKeyStates.....	512
ResetTabs.....	1158
ResetTimer.....	330
ResolveHostName.....	792
ResumeLayer.....	433
ResumeModule.....	1051
ResumeMusic.....	1051
ResumeTimer.....	330
ResumeVideo.....	1244
Return.....	1230
ReverseFindStr.....	1291
ReverseStr.....	1292
RewindDirectory.....	301
RGB.....	587
RGBArrayToBrush.....	959
RightStr.....	1293
Rnd.....	734
RndF.....	734
RndStrong.....	735
Rol.....	736
Ror.....	737
RotateBrush.....	961
RotateLayer.....	434
RotateTextObject.....	1158
Round.....	737
Rt.....	738
Run.....	301
RunCallback.....	513
RunRexxScript.....	179

## S

SAMPLE.....	1051
Sar.....	738
SaveAnim.....	208
SaveBrush.....	962
SaveIcon.....	894
SavePalette.....	858
SavePrefs.....	229
SaveSample.....	1053
SaveSnapshot.....	589
ScaleAnim.....	211
ScaleBGPic.....	638
ScaleBrush.....	964
ScaleLayer.....	434
ScaleSprite.....	1083
ScaleTextObject.....	1159
SCREEN.....	394
ScrollConsole.....	684
Seek.....	306
SeekLayer.....	435
SeekMusic.....	1053
SeekVideo.....	1244
SelectAlphaChannel.....	965
SelectAnim.....	211
SelectBGPic.....	639
SelectBrush.....	967
SelectConsoleWindow.....	685
SelectDisplay.....	396
SelectLayer.....	436
SelectMask.....	969
SelectMenuItem.....	758
SelectPalette.....	859
SendApplicationMessage.....	180
SendData.....	793
SendMessage.....	644
SendRexxCommand.....	181
SendUDPData.....	794
SepiaToneBrush.....	971
SerializeTable.....	995
SetAllocConsoleColor.....	686
SetAlphaIntensity.....	971
SetAnimFrameDelay.....	213
SetBaudRate.....	1014
SetBorderPen.....	860
SetBrushDepth.....	972
SetBrushPalette.....	973
SetBrushPen.....	974
SetBrushTransparency.....	975
SetBrushTransparentPen.....	976
SetBulletColor.....	1161
SetBulletPen.....	860
SetChannelVolume.....	1054
SetClipboard.....	1307
SetClipRegion.....	591
SetConsoleBackground.....	686
SetConsoleColor.....	687
SetConsoleCursor.....	688
SetConsoleOptions.....	689



SetConsoleStyle.....	690	SetPaletteDepth.....	867
SetConsoleTitle.....	691	SetPaletteMode.....	868
SetCycleTable.....	861	SetPalettePen.....	869
SetDash.....	1202	SetPaletteTransparentPen.....	870
SetDataBits.....	1015	SetPanning.....	1055
SetDefaultAdapter.....	1110	SetParity.....	1017
SetDefaultEncoding.....	1162	SetPen.....	870
SetDefaultLoader.....	1111	SetPitch.....	1056
SetDepth.....	862	SetPointer.....	747
SetDisplayAttributes.....	397	SetRTS.....	1017
SetDitherMode.....	863	SetScreenTitle.....	181
SetDrawPen.....	864	SetSerializeMode.....	997
SetDrawTagsDefault.....	593	SetSerializeOptions.....	1000
SetDTR.....	1016	SetShadowPen.....	872
SetEnv.....	307	SetSpriteZPos.....	1084
SetEventTimeout.....	514	SetStandardIconImage.....	899
SetFileAttributes.....	307	SetStandardPalette.....	872
SetFileEncoding.....	308	SetStopBits.....	1018
SetFillRule.....	1203	SetSubtitle.....	400
SetFillStyle.....	609	SetTimeout.....	517
SetFlowControl.....	1016	SetTimerElapse.....	331
SetFont.....	1162	SetTitle.....	401
SetFontColor.....	1165	SetTransparentPen.....	874
SetFontStyle.....	1166	SetTransparentThreshold.....	875
SetFormStyle.....	611	SetTrayIcon.....	900
SetFPSLimit.....	593	SetVarType.....	1113
SetGradientPalette.....	865	SetVectorEngine.....	1205
SetIconProperties.....	896	SetVideoPosition.....	1245
SetInterval.....	515	SetVideoSize.....	1245
SetIOMode.....	309	SetVideoVolume.....	1246
SetLayerAnchor.....	438	SetVolume.....	1056
SetLayerBorder.....	439	SetWBIcon.....	901
SetLayerDepth.....	440	Sgn.....	739
SetLayerFilter.....	440	SharpenBrush.....	977
SetLayerName.....	446	Shl.....	739
SetLayerPalette.....	447	ShowConsoleCursor.....	692
SetLayerPen.....	448	ShowDisplay.....	401
SetLayerShadow.....	448	ShowKeyboard.....	766
SetLayerStyle.....	449	ShowLayer.....	466
SetLayerTint.....	463	ShowLayerFX.....	467
SetLayerTransparency.....	464	ShowNotification.....	1113
SetLayerTransparentPen.....	465	ShowPointer.....	747
SetLayerVolume.....	465	ShowRinghioMessage.....	182
SetLayerZPos.....	466	ShowScreen.....	183
SetLineCap.....	1203	ShowToast.....	766
SetLineJoin.....	1204	Shr.....	740
SetLineWidth.....	612	Sin.....	741
SetListItems.....	1132	SIZEWINDOW.....	1230
SetMargins.....	1168	Sleep.....	1115
SetMaskMode.....	976	SolarizeBrush.....	978
SetMasterVolume.....	1054	SolarizePalette.....	875
SetMetaTable.....	1133	Sort.....	1133
SetMiterLimit.....	1205	SplitStr.....	1293
SetMusicVolume.....	1055	SPRITE.....	1084
SetNetworkProtocol.....	795	Sqrt.....	741
SetNetworkTimeout.....	796	StartConsoleColorMode.....	692
SetObjectData.....	838	StartPath.....	1206
SetPalette.....	866	StartSubPath.....	1206

StartsWith.....	1294
StartTimer.....	332
StopAnim.....	213
StopChannel.....	1057
StopLayer.....	468
StopModule.....	1057
StopMusic.....	1058
StopSample.....	1058
StopTimer.....	332
StopVideo.....	1247
StringRequest.....	354
StringToFile.....	310
StripStr.....	1295
StrLen.....	1296
StrStr.....	1296
StrToArray.....	1297
Sub.....	742
SwapLayers.....	469
SwirlBrush.....	978
SystemRequest.....	355

## T

TableItems.....	1134
TableToMem.....	1071
Tan.....	742
TextExtent.....	1168
TextHeight.....	1171
TextOut.....	1172
TextWidth.....	1179
TimerElapsed.....	333
TimestampToDate.....	333
TintBrush.....	979
TintPalette.....	876
ToHostName.....	796
ToIP.....	796
ToNumber.....	1298
ToString.....	1298
TouchConsoleWindow.....	693
ToUserData.....	1299
TransformBox.....	594
TransformBrush.....	980
TransformLayer.....	469
TransformPoint.....	595
TransformTextObject.....	1180
TranslateLayer.....	470
TranslatePath.....	1207
TrimBrush.....	981
TrimStr.....	1299

## U

UndefinedVirtualStringFile.....	310
Undo.....	471
UndoFX.....	473
UngroupLayer.....	474
UnleftStr.....	1300
UnmidStr.....	1301
Unpack.....	1135
UnrightStr.....	1302
UnsetEnv.....	311
UploadFile.....	797
UpperStr.....	1302
UseCarriageReturn.....	311
UseFont.....	1181
UTCToDate.....	334

## V

Val.....	1303
ValidateDate.....	335
ValidateStr.....	1304
VERSION.....	1115
Vibrate.....	767
VIDEO.....	1247
VWait.....	596

## W

Wait.....	1116
WaitAnimEnd.....	214
WaitEvent.....	518
WaitKeyDown.....	520
WaitLeftMouse.....	521
WaitMidMouse.....	521
WaitMusicEnd.....	1058
WaitPatternPosition.....	1059
WaitRightMouse.....	521
WaitSampleEnd.....	1059
WaitSongPosition.....	1059
WaitTimer.....	335
WARNING.....	341
WaterRippleBrush.....	982
WhileKeyDown.....	1231
WhileMouseDown.....	1231
WhileMouseOn.....	1232
WhileRightMouseDown.....	1232
Wrap.....	743
WriteAnimFrame.....	215
WriteBrushPixel.....	983
WriteByte.....	312
WriteBytes.....	312
WriteChr.....	313
WriteFloat.....	314
WriteFunction.....	315
WriteInt.....	316
WriteLine.....	317
WriteMem.....	1072
WritePen.....	876

WriteRegistryKey.....	1250	WriteString.....	318
WriteSerialData.....	1018		
WriteShort.....	317	WriteTable.....	1000