

in

COLLABORATORS

	<i>TITLE :</i> in		
<i>ACTION</i>	<i>NAME</i>	<i>DATE</i>	<i>SIGNATURE</i>
WRITTEN BY		February 12, 2023	

REVISION HISTORY

NUMBER	DATE	DESCRIPTION	NAME

Contents

1	in	1
1.1	Dust - Dokumentation	1
1.2	5. Tutorial	1
1.3	Tutorium 4 - Erzeugen von Welleninterferenzen mittels FOR-Schleifen	2
1.4	Tutorium 5 - Programmieren eines Partikel-Effektes: Explosionen	3
1.5	Tutorium 3 - Einbindung der Objekte in LIGHTWAVE	5
1.6	Tutorium 2 - Erzeugung einer 2d-Transversalwelle (180 Bilder)	5
1.7	4. DOKUMENTATION	7
1.8	4.5 Der ARexx-Port	8
1.9	4.4. Das Partikel-System	8
1.10	Das Dust-Particle-Format	10
1.11	4.3. Die Preview-Funktion	10
1.12	4.2 Programmkonzept	11
1.13	Unterstützte Objektformate	13
1.14	Datentypen	14
1.15	Mathematische Ausdrücke	14
1.16	Identifizier	15
1.17	Schleifen	15
1.18	Formatierungsbefehle	16
1.19	4.1. Installation	16
1.20	1. Programmbeschreibung	17
1.21	2. System- und Anwendervoraussetzungen	19
1.22	3. Programmstatus und Anwenderlizenz	19
1.23	7. Die Adresse des Authors	20
1.24	6. Copyrights	20
1.25	4.4 Beschreibung aller Befehle	20
1.26	1. Laden und Speichern von Objekten	21
1.27	2. Modifizieren oder Erzeugen von Einzelobjekten	22
1.28	3. Modifizieren oder Erzeugen von Objekt-Sequenzen	25
1.29	4. Modifizieren von Objektattributen	27

1.30	5. Information zu bestimmten Objekt-Komponenten	29
1.31	6. Funktionen für Programmierer	30
1.32	7. Verschiedene Funktionen (Fenster, Interface, ...)	30
1.33	4.5 Beschreibung aller Programmparameter	33
1.34	Was kann das Programm wirklich ?	35
1.35	Programm-Parameter ASPECT	39
1.36	Programm-Parameter BWLEFT	39
1.37	Programm-Parameter BWTOP	39
1.38	Programm-Parameter KEEPASPECT	39
1.39	Programm-Parameter WINDOWSTACK	39
1.40	Programm-Parameter WINDOWPRI	40
1.41	Programm-Parameter WARNINGS	40
1.42	Programm-Parameter CHECKMOUSE	40
1.43	Programm-Parameter BREAKWIN	40
1.44	Programm-Parameter RANDOM	40
1.45	Programm-Parameter BACKFACES	41
1.46	Programm-Parameter SAVESPHEREP	41
1.47	Programm-Parameter ALIGNP	41
1.48	Programm-Parameter EXFILE	41
1.49	Programm-Parameter EXFORMAT	42
1.50	Programm-Parameter LEFT	42
1.51	Programm-Parameter TOP	42
1.52	Programm-Parameter WIDTH	42
1.53	Programm-Parameter DRAWMODE	42
1.54	Programm-Parameter ROTX	43
1.55	Programm-Parameter ROTZ	43
1.56	Programm-Parameter ZOOM	43
1.57	Befehl ABOUT	43
1.58	Befehl ANIMCFUNC	43
1.59	Befehl ANIMFUNC	43
1.60	Befehl ANIMPPOSFUNC	44
1.61	Befehl ANIMPROTFUNC	44
1.62	Befehl ANIMPSCLFUNC	45
1.63	Befehl AVAIL	45
1.64	Befehl AXALIGN0	45
1.65	Befehl AXPOS	45
1.66	Befehl AXSIZE	46
1.67	Befehl BUILD	46
1.68	Befehl BUILDRND	46

1.69 Befehl CD	46
1.70 Befehl CENTERAXIS	46
1.71 Befehl CFUNC	46
1.72 Befehl CLOSEWINDOWS	47
1.73 Befehl COLOR	47
1.74 Befehl COPY	47
1.75 Befehl COPYATTS	47
1.76 Befehl COPYP	47
1.77 Befehl COPYPPOS	48
1.78 Befehl COPYPROT	48
1.79 Befehl COPYPSCL	48
1.80 Befehl DISTORT	48
1.81 Befehl DITHER	48
1.82 Befehl EXEC	49
1.83 Befehl EXPLODE	49
1.84 Befehl EXPLODEFRAME	49
1.85 Befehl ECHO	49
1.86 Befehl FUNC	49
1.87 Befehl GET	50
1.88 Befehl GETOCOUNT	50
1.89 Befehl GETPSIZE	50
1.90 Befehl GETPPOS	50
1.91 Befehl GETPROT	51
1.92 Befehl GETPSCL	51
1.93 Befehl HARDNESS	51
1.94 Befehl JOIN	51
1.95 Befehl JOINP	51
1.96 Befehl KILL	52
1.97 Befehl KILLFREEPOINTS	52
1.98 Befehl KILLEDGE	52
1.99 Befehl KILLFACE	52
1.100 Befehl KILLP	52
1.101 Befehl KILLPOINT	52
1.102 Befehl LOAD	53
1.103 Befehl MERGE	53
1.104 Befehl MORPH	53
1.105 Befehl MORPHFRAME	53
1.106 Befehl MORPHATTS	53
1.107 Befehl DEFORMMORPH	53

1.108	Befehl O2P	54
1.109	Befehl P2O	54
1.110	Befehl PEXPLODE	54
1.111	Befehl PPOSFUNC	54
1.112	Befehl PROTFUNC	55
1.113	Befehl PSCLFUNC	55
1.114	Befehl PSTATS	55
1.115	Befehl PSTATS2	55
1.116	Befehl PFALL	55
1.117	Befehl PFALL2	56
1.118	Befehl POSITIVE	56
1.119	Befehl PWAVE1D	56
1.120	Befehl PWAVE1DFRAME	56
1.121	Befehl PWAVE2D	56
1.122	Befehl PWAVE2DFRAME	57
1.123	Befehl PWAVE3D	57
1.124	Befehl PWAVE3DFRAME	57
1.125	Befehl RANDOMPPOS	58
1.126	Befehl RANDOMPROT	58
1.127	Befehl RANDOMPSCL	58
1.128	Befehl REFL	58
1.129	Befehl RENAME	58
1.130	Befehl REQUEST	58
1.131	Befehl ROTATE	59
1.132	Befehl ROUGHNESS	59
1.133	Befehl SAVE	59
1.134	Befehl SAVECONFIG	59
1.135	Befehl LOADCONFIG	60
1.136	Befehl SAVEP	60
1.137	Befehl SAVEPOBJ	60
1.138	Befehl SCALE	60
1.139	Befehl SCALEFACES	60
1.140	Befehl SCALEP	60
1.141	Befehl SET	61
1.142	Befehl SETCLST	61
1.143	Befehl SETPPOS	61
1.144	Befehl SETPOINT	61
1.145	Befehl SETPROT	61
1.146	Befehl SETPSCL	61

1.147Befehl SHININESS	62
1.148Befehl SIZE	62
1.149Befehl SPEC	62
1.150Befehl STAGING2	62
1.151Befehl STAGING3	63
1.152Befehl STATS	63
1.153Befehl STATS2	63
1.154Befehl TIME	63
1.155Befehl TRANS	64
1.156Befehl TRANSLATE	64
1.157Befehl TRIANGULATE	64
1.158Befehl WAVE1D	64
1.159Befehl WAVE1DFRAME	64
1.160Befehl WAVE2D	65
1.161Befehl WAVE2DFRAME	65
1.162Befehl WAVE3D	65
1.163Befehl WAVE3DFRAME	65
1.164Befehl WINDOW	65
1.165Befehl WRITEATTS	66
1.166Befehl WRITEAXIS	66
1.167Befehl WRITECLST	67
1.168Befehl WRITEEDGES	67
1.169Befehl WRITEFACES	67
1.170Befehl WRITEPOINTS	67
1.171Befehl WRITEPPOS	67
1.172Befehl WRITEPROT	67
1.173Befehl WRITEPSCL	68
1.174Befehl ;	68
1.175Befehl !	68
1.176Befehl MEMORYP	68
1.177Befehl MEMORY	68
1.178Befehl LOADSEQ	68
1.179Befehl SAVESEQ	69
1.180Befehl SAVEPSEQ	69
1.181Befehl WINDOWSEQ	69
1.182Befehl P2OSEQ	69
1.183Befehl CALC oder	69
1.184Programm-Parameter KEEPSCALE	70
1.185Programm-Parameter SFORMAT	70

1.186	Programm-Parameter OUTLINED	70
1.187	Programm-Parameter PAGER	70
1.188	Programm-Parameter HELPDIR	71
1.189	Programm-Parameter HELPDIR2	71
1.190	Befehl PMORPH	71
1.191	Befehl WINDOWCLOSE	71
1.192	Befehl WINDOWDRAWMODE	71
1.193	Befehl WINDOWOUTLINED	71
1.194	Befehl WINDOWPERSPECTIVE	72
1.195	Befehl WINDOWREDRAW	72
1.196	Befehl WINDOWRESCALE	72
1.197	Befehl WINDOWROTX	72
1.198	Befehl WINDOWROTZ	72
1.199	Befehl WINDOWSAVE	72
1.200	Befehl WINDOWZOOM	72
1.201	Befehl WINDOWFRONT	73
1.202	Befehl WINDOWPOS	73
1.203	Befehl WINDOWSIZE	73
1.204	Befehl LOADVS	73
1.205	Befehl LOADGROUPOBJ	73
1.206	Befehl SHOWTDDD	73
1.207	Befehl BUILDMORPH	74
1.208	Befehl BUILDMORPHRND	74
1.209	Befehl O2S	74
1.210	Befehl SAVESPHERES	74
1.211	Befehl LWSTAGING	74
1.212	Programm-Parameter QUIET	74
1.213	Programm-Parameter LOG	75
1.214	Programm-Parameter LOGFILE	75
1.215	Befehl FILETYPE	75
1.216	Programm-Parameter LWCMD1, LWCMD2, LWCMD3	75
1.217	Befehl SAVEVS	75
1.218	Befehl SAVETDDD	76
1.219	Befehl SAVELW	76
1.220	Befehl SAVEMC4D	76
1.221	Befehl REXX	76
1.222	Befehl ADDFACE	77
1.223	Befehl BRSAXALIGN0	77
1.224	Befehl BRSAXPOS	77

1.225	Befehl BRSAXSIZE	77
1.226	Befehl BRSDIR	77
1.227	Befehl BRSNAME	77
1.228	Befehl CENTERBRSAXIS	78
1.229	Befehl CENTERTXTAXIS	78
1.230	Befehl COPYBRS	78
1.231	Befehl COPYTXT	78
1.232	Befehl KILLBRS	78
1.233	Befehl KILLTXT	78
1.234	Befehl TXTAXALIGN0	78
1.235	Befehl TXTAXPOS	79
1.236	Befehl TXTAXSIZE	79
1.237	Befehl TXTDIR	79
1.238	Befehl TXTNAME	79
1.239	Befehl TXTPARAM	79
1.240	Befehl SHOWBRS	79
1.241	Befehl SHOWTXT	79
1.242	Befehl ROTATEAXIS	80
1.243	Befehl ROTATEBRSAXIS	80
1.244	Befehl ROTATETXTAXIS	80
1.245	Programm-Parameter OPTEDGES	80
1.246	Programm-Parameter COMPLETE	80
1.247	Programm-Parameter ACTVAL	80
1.248	Befehl WATER	81
1.249	Befehl WATERFRAME	81
1.250	Befehl WATERZ	81
1.251	Befehl WATERZFRAME	82
1.252	Befehl SETCOLSGROUP	82
1.253	Befehl GETCOLSGROUP	82
1.254	Befehl WRITESGROUP	82
1.255	Befehl ADDSGROUP	82
1.256	Befehl SUBSGROUP	82
1.257	Tutorium 1 - MORPH und Imagine-States	83
1.258	Befehl COPYCLST	83
1.259	Befehl MORPHSGROUP	83
1.260	Befehl RENAMESGROUP	83
1.261	Befehl SHOWVALUES	84
1.262	Befehl OCOUNT	84
1.263	Befehl PCOUNT	84

1.264	Befehl	ECOUNT	84
1.265	Befehl	FCOUNT	84
1.266	Befehl	GETPOINT	84
1.267	Befehl	LATTICE	85
1.268	Befehl	INSERTPOINT	85
1.269	Befehl	IF	85
1.270	Programm-Parameter	ECHO	86
1.271	Programm-Parameter	SPLINETYPE	86
1.272	Programm-Parameter	SPLINEENDS	86
1.273	Programm-Parameter	SPLINESUBDIV	86
1.274	Befehl	INTERPOLATEDATA	86
1.275	Befehl	CUTSG	87
1.276	Befehl	EXTRACTSG	87
1.277	Befehl	SMOOTH	87
1.278	Befehl	SMOOTHINNER	87
1.279	Befehl	SMOOTHSG	88
1.280	Programm-Parameter	MAXANGLE	88
1.281	Befehl	COPYAXIS	89
1.282	Programm-Parameter	STARTPCORR	89
1.283	Programm-Parameter	FORCESWAP	89
1.284	Programm-Parameter	INTERPMODE	89
1.285	Befehl	CHECKOBJECT	89
1.286	Befehl	EXPANDSG	90
1.287	Befehl	SHRINKSG	90
1.288	Befehl	SAMEPOS	90
1.289	Befehl	CDEFORM	90
1.290	Befehl	CDEFORMINTERP	92

Chapter 1

in

1.1 Dust - Dokumentation

```
#####
#
#           Dust V2.32 - Copyright ©1994 by A.Maschke           #
#           All rights reserved.                               #
#-----#
#
#           Dokumentation V1.2                                 #
#
#####
```

1. Programmbeschreibung
2. System- und Anwendervoraussetzungen
3. Programmstatus und Garantie
4. DOKUMENTATION
5. Tutorial
6. Copyrights
7. Die Adresse des Authors
(Letzte Änderungen: 1 October 1995)

1.2 5. Tutorial

Imagine bietet ab Version 2.9 Objekte mit verschiedenen Zuständen, ↔
sogenannte
States, an.

Tutorium 1 - MORPH und Imagine-States

demonstriert,
wie man aus ZWEI Objekten mit unterschiedlicher Flächen- und Punktzahl EIN States-Objekt macht.

Um Animationen zu berechnen, müssen die Objekte in den Renderer geladen werden. Bei den meisten Programmen außer Imagine und Lightwave ist das sehr aufwendig.

Für Imagine gibt es ein Programm names ISL, dessen Verwendung im

Tutorium 2 - Erzeugung einer 2d-Transversalwelle (180 Bilder)
demonstriert wird.

Tutorium 3 - Einbindung der Objekte in LIGHTWAVE
beschreibt die Erzeugung einer Animation mit Lightwave.

Die Linearkombination von Spezialeffekten wird anhand der Erzeugung von Welleninterferenzen im

Tutorium 4 - Erzeugen von Welleninterferenzen mittels FOR- ←
Schleifen

erläutert.

Das letzte Tutorium widmet sich der Programmierung eines Particle-Effektes:

Tutorium 5 - Programmieren eines Partikel-Effektes: Explosionen

1.3 Tutorium 4 - Erzeugen von Welleninterferenzen mittels FOR-Schleifen

Um Interferenzen zu erzeugen, muß man mehrere Wellen nacheinander über ein Objekt laufen lassen. Per Hand ist dies bei vielen Frames nicht durchführbar. Hier eignet sich der Einsatz einer FOR-Schleife.

Wir wollen nun 3 Wellen über eine Plane laufen lassen und dabei 180 Objekte erzeugen. Dazu sind folgende Schritte notwendig (für jedes Einzelobjekt):

- Laden der Plane
- Erzeugen der ersten Welle
- Erzeugen der zweiten Welle
- Erzeugen der dritten Welle
- Speichern des Objektes

Wir legen nun ein Script an, daß dies für das erste Bild tut:

```
"load(1,plane)
wave2dframe(1,180,1,2,t,12.0,24,20,30,1.0,60.0)
wave2dframe(2,180,1,3,t,14.0,36,-30,10,0.9,-30.0)
wave2dframe(3,180,1,2,t,11.0,22,5,-30,1.1,45.0)
save(2,hdl:objects/obj.0001)"
```

Nun ändern wir die Zahl 1 in eine Schleifenvariable, z.B. i, und

führen die gesamte Prozedur 180 mal aus:

```
"load(1,plane)
for(i,1,180)
  wave2dframe(1,180,i,2,t,12.0,24,20,30,1.0,60.0)
  wave2dframe(2,180,i,3,t,14.0,36,-30,10,0.9,-30.0)
  wave2dframe(3,180,i,2,t,11.0,22,5,-30,1.1,45.0)
  save(2,hdl:objects/obj.%)
end"
```

Dieses Script kann durch "exec(script)" ausgeführt werden, "staging3(hdl:objects/obj,1,180,1,180,ram:staging.a)" erzeugt die zugehörige staging-Datei.

Nun kann die Animation, die übrigens phantastisch aussieht (bei hohen Specular-Werten wie geschmolzenes Plastic), berechnet werden.

1.4 Tutorium 5 - Programmieren eines Partikel-Effektes: Explosionen

1. Allgemeiner Ablauf

1. Skalierung

Bei einer Explosion bleibt die Größe der fliegenden Partikel wohl fast konstant ←

hierum brauchen wir uns nicht kümmern

2. Position

Dazu muß eine Differentialgleichung aufgestellt und gelöst werden, die Lösung in diesem Falle könnte lauten:

```
x=f(x0,vx0,eta,t)
y=f(y0,vy0,eta,t)
z=g(y0,vz0,eta,g,t)
```

(vx0,vy0,vz0:Anfangsgeschwindigkeit, x0,y0,z0:Anfangspositionen, g: ←
Fallbeschleunigung,
eta:Zähigkeit der Luft)

Die Anfangsgeschwindigkeit muß zuerst für jedes Teilchen bestimmt werden. Dazu ermitteln wir zuerst den Mittelpunkt aller Teilchen.

Nun berechnen wir für jedes Teilchen den Differenzvektor zwischen seiner Position und dem Mittelpunkt und normieren diesen Vektor.

Wird dieser Vektor mit dem Betrag der Anfangsgeschwindigkeit, der von Teilchen zu Teilchen etwas variieren sollte, multipliziert, so erhalten wir den gesuchten Anfangsgeschwindigkeitsvektor.

Eta sollte etwa im Bereich von -0.1 bis -0.00001 liegen, g von -10 bis 0.

3. Rotation

Die Rotationswinkel sollten so bestimmt werden, daß sich größere Teilchen langsamer drehen als kleinere, das sieht realistischer aus. Dazu müssen wir die tatsächliche maximale Ausdehnung jedes Objekts bestimmen und davon die kleinste auswählen. Dieses Objekt mit der kleinsten Ausdehnung soll nun n Umdrehungen ausführen. Dann müssen wir für alle anderen Objekte das ←
Verhältnis

ihrer Ausdehnung zu der des kleinsten Objektes bestimmen und daraus dessen Umdrehungszahl ermitteln.

(Die tatsächliche Größe eines Partikels ist gleich
 (psize.x*scl.x,psize.y*scl.y,psize.z*scl.z))

In diesem Beispiel lassen wir die Teilchen aber NICHT rotieren.

4.Animation

Die Zeit dient als Parameter, d.h., wir erhöhen in jedem Frame die aktuelle Zeit um Zeitdauer/(frames-1), berechnen die Koordinaten, und erhöhen die Rotationswinkel um angle/(frames-1).

2. Umsetzung

 Alle hier besprochenen Dateien werden dem Programm im Verzeichnis Tutorial3 beiliegen.

Benutzer einer anderen Programmiersprache als OBERON müssen das Programm erst in ihre Sprache übersetzen. Dieses Programm kann aber als Basis für alle anderen eigenen Programme verwendet werden; es enthält alle Dust-spezifischen Prozeduren, die je benötigt werden.

Das Skript, das vom dem hier besprochenen Programm erzeugt werden soll, ist als "Tutorial3/PExample.script" beigelegt, damit Sie Ihr Programm auf Richtigkeit überprüfen können.

1. Erzeugen des Partikel-Objects

Wir verwenden eine Kugel als Struktur- und einen Würfel als Shape-Objekt, als Daten verwenden wir nur OCOUNT und das Feld PPOS. Das Script sollte also etwa wie folgt aussehen:

```
"load(1,objects/sl)      (Kugel laden)
load(2,objects/cl)      (Würfel laden)
o2p(1,2,1,p)            (Partikel-Objekt erzeugen)
savep(1,PExample.obj)   (abspeichern)
getocount(1)            (OCOUNT als Binärdatei "PExample.oCount" ausgeben
!copy T:Dust.output PExample.oCount
getppos(1)              (PPOS als Binärdatei "PExample.PPOS" ausgeben
!copy T:Dust.output PExample.PPOS
!delete T:Dust.output"
```

2. Das Programm

Das Programm muß zuerst die Binärdaten einladen. Es wird dann ein Dust-Script am Bildschirm erzeugen, daß das Partikel-Objekt modifiziert (Frame für Frame) und abspeichert. Die Bildschirmausgabe leiten wir natürlich in eine Datei um. (Dies ist der einfachste Weg, eine gut formatierte ASCII-Datei zu erhalten.)

Wir berechnen also n Objekte nach folgendem Schema:

- für jedes Partikel die neue Position nach den o.g. Bewegungsgleichungen bestimmen
- diese Positionen als SETPPOS-Kommandos auf dem Bildschirm ausgeben
- danach ein Save-Kommando (Format je nach Bedarf) ausgeben
- nächstes Objekt

3. Endgültige Objekterzeugung

Das Programm, es heie "PExample" starten wir nun durch
 "PExample >PExample.script".
 Danach (ca. 10 Sekunden) starten wir Dust und laden das Partikel-Objekt:
 "load(1,PExample.obj)".
 Nun knnen wir das von unserem Programm erzeugte Skript durch
 "exec(PExample.script)"
 ausfhren lassen und erhalten nach wenigen Sekunden 12 Imagine-Objekte.
 Fertig !

1.5 Tutorium 3 - Einbindung der Objekte in LIGHTWAVE

Vorbemerkung: Ich besitze Lightwave nicht und habe also die folgenden Funktionen nicht getestet.

Im Gegensatz zu Imagine ldt Lightwave beim Einlesen der Scene-Datei alle darin erwhnten Objekte sofort in den Speicher. Da dies bei z.B. einer Welleninterferenz aus 480 Objekten auf den meisten Rechnern unmglich ist, mu man dies irgendwie umgehen. Folgendes Verfahren mte gut funktionieren:

Beispiel einer Welle aus 480 Objekten:

 Im allgemeinen wird man eine groe Szene aufbauen, in der irgendwo ein von Dust transformiertes Objekt auftaucht. So ist der erste Schritt die Modellierung dieser Szene (Umgebung). Danach ldt man eines der von Dust transformierten Objekte hinzu, z.B. "hd0:tobj.0020". Die dann z.B. als "hdl:scene" gespeicherte Scene-Datei wird von Dust 480 mal reproduziert ("hdl:scene.0001" bis "hdl:scene.0480"), wobei immer fr "hd0:tobj.0020" ein anderes Objekt eingesetzt wird.

ACHTUNG: Die Scene darf nur aus einem Objekt bestehen:
 firstframe=lastframe=1

Der verwendete Befehl heit "lwstaging", in diesem Fall mu er wie folgt aufgerufen werden:

```
"lwstaging(hd0:tobj,1,480,1,480,hdl:scene) "
```

Zustzlich erzeugt Dust ein ARexx-Script "hdl:scene.rexx", das Lightwave alle 480 Bilder hintereinander berechnen lt. Dazu startet man Lightwave und gibt in einer Shell ein:

```
"rx hdl:scene.rexx".
```

Bemerkung: 1. Mchte man spter doch nderungen an der Umgebung vornehmen, so macht das keine Probleme, nur mu der lwstaging-Befehl danach noch einmal ausgefhrt werden.
 2. Es ist auch eine absteigende Reihenfolg der Objekte mglich
 3. Verbesserungsvorschlge sind willkommen

1.6 Tutorium 2 - Erzeugung einer 2d-Transversalwelle (180 Bilder)

-Zuerst erzeugen wir in Imagine eine Plane (20*20 Punkte), drehen sie um die x-Achse um 90 Grad und setzen die Ausrichtung der Axis wieder auf 0 0 0.

-Nach dem Einladen des Objekts in Dust erzeugen wir durch Eingabe von
`"wave2d(1,180,hd2:obj,t)"`
 die 180 Objekte.

-Zurück in Imagine erzeugen wir ein neues Projekt "wave1" und laden Objekt 1 ("obj.0001") in den Stage Editor, stellen die Kamera richtig ein und erzeugen eine Lichtquelle.

-Nach dem Abspeichern kopieren wir die Datei "wave1.imp/staging" nach T:

-Durch Aufruf des ISL-Programms 'Destage' durch
`"destage t:staging t:staging.a"`
 erzeugen wir eine ASCII-Datei, welche wir in einen Editor laden.

-Irgendwo befindet sich die Zeile
`"ACTOR FRAMES 1 180 NAME "hd2:obj.0001" CYCLE 0. 0. TRANSITION 0"`

Diese muß durch die Objekt-Sequenz ersetzt werden, die nötigen Zeilen erzeugen wir in Dust durch
`"staging3(hd2:obj,1,180,1,180,ram:tt)"`.

Die Datei "ram:tt" fügen wir nun an der betreffenden Stelle im Text ein und löschen dann die Zeile:

```
"ACTOR FRAMES 1 180 NAME "hd2:obj.0001" CYCLE 0. 0. TRANSITION 0"
```

An der Stelle dieser sollte jetzt stehen:

```
"ACTOR FRAMES 1 1 NAME "hd2:obj.0001" CYCLE 0. 0. TRANSITION 0"
```

```
"ACTOR FRAMES 2 2 NAME "hd2:obj.0002" CYCLE 0. 0. TRANSITION 0"
```

```
"ACTOR FRAMES 3 3 NAME "hd2:obj.0003" CYCLE 0. 0. TRANSITION 0"
```

```
.
```

```
.
```

```
.
```

```
"ACTOR FRAMES 180 180 NAME "hd2:obj.0180" CYCLE 0. 0. TRANSITION 0"
```

-Nach dem Abspeichern des Staging-Files müssen wird das ASCII-File wieder in das Imagine-Format umwandeln:

```
"restage t:staging.a t:staging".
```

-Die Datei "t:staging" kopieren wir nun an den alten Platz (wave1.imp/staging) und die Animation kann berechnet werden.

Bemerkung: Alle diese Schritte kann man auch durch ein Shell-Script automatisieren ↔
 :

```
.key project,obj,from,to
```

```
.bra {
```

```
.ket }
```

```
;this script creates an Imagine3.1-staging file for an object-sequence
```

```
;Parameters:
```

```
; project: full path of the Imagine-project, e.g. "hd1:imagine/wave.imp"
```

```
; obj: object-base-name with path, e.g. "hd0:obj"
```

```
; from: first object, e.g. 1
```



```
; last: last object

;Example: stageit hd1:Grafix/Imagine3.1/test.imp shit:obj 1 12

;Required software: Dust, ISL3.x, CEd and Ed (quickstarter) in your C:-drawer,
;the Rexx-Master must be active, too.

;init
clear
set echo=off

echo "Creating a backup of the old staging-file"
copy {project}/staging {project}/staging.o

echo "Creating a-Dust-script"
echo >T:Dust.tmp "staging3({obj},{from},{to},{from},{to},T:staging.tmp) "

echo "Running Dust"
Dust T:Dust.tmp

echo "Running Destage"
destage {project}/staging T:staging.a

echo "Running CEd"
Ed T:staging.a
rx "address 'rexx_ced' 'search for {obj}.'"
rx "address 'rexx_ced' 'Beg of line'"
rx "address 'rexx_ced' 'Delete line'"
rx "address 'rexx_ced' Include file 'T:staging.tmp'"
rx "address 'rexx_ced' 'Save'"
rx "address 'rexx_ced' 'Quit'"

echo "Running Restage"
restage T:staging.a {project}/staging

;clean up
delete >NIL: T:Dust.tmp T:staging.a T:staging.tmp

echo "Bye !"
```

1.7 4. DOKUMENTATION

- 4.1 Installation
 - 4.2 Programmkonzept
 - 4.3 Die Preview-Funktion
 - 4.4 Das Particle-System
 - 4.5 Der ARexx-Port
 - 4.6 BESCHREIBUNG ALLER BEFEHLE
-

4.7 BESCHREIBUNG ALLER PROGRAMMPARAMETER

1.8 4.5 Der ARexx-Port

Wenn Sie Dust durch andere Applikationen oder eine grafische Benutzeroberfläche kontrollieren möchten, so benötigen Sie einen ARexx-Port.

```

NAME: "Dust"
COMMANDS: PARSE <cmd> - läßt Dust die Zeichenkette <cmd> ausführen,
                    dabei sind alle Befehle außer "EXIT" erlaubt ↔
                    erlaubt
EXIT          - verläßt den ARexx-Modus (der durch das
                    "REXX"-Kommando aktiviert wurde)

```

Um in den ARexx-Modus zu gelangen, muß der "REXX"-Befehl aufgerufen werden.

Beispiele mit dem "rx"-Befehl:

```

rx "address 'Dust' 'EXIT'"
rx "address 'Dust' 'PARSE ?'" (ruft die Hilfefunktion auf)
rx "address 'Dust' 'PARSE load(1,s1)'" (lädt ein Objekt)

```

Ein anderes Beispiel:

Sie haben ein GUI geschrieben und möchten Dust im Hintergrund starten und automatisch beenden.

Schreiben Sie eine Batch-Datei "rexx.dust" mit dem Inhalt:

```

rexx
exit

```

und starten Sie Dust durch "Dust rexx.bat".

Nun führt Dust solange PARSE-Anweisungen aus, bis Sie den ARexx-EXIT-Befehl aufrufen. Danach wird die Abarbeitung der Batchdatei fortgesetzt, das Programm also beendet.

Zu diesem Zweck wurde auch der QUIET-Parameter eingeführt, der sämtliche Textausgaben von Dust unterdrückt.

Natürlich können im ARexx-Modus auch Batch-Dateien ausgeführt werden.

1.9 4.4. Das Partikel-System

Das Partikel-System ist eine der innovativsten Funktionen des Programmes. ↔

Die dabei von mir entwickelte Datenstruktur ist so einfach zu handhaben, daß jeder Programmierer jetzt seine eigenen Partikeleffekte kreieren kann, wie z.B. einen Wasserfall aus 3000 Kugeln.

Das einzige, was dabei getan werden muß, ist die Berechnung der Position, der Rotationswinkel und der Skalierungsfaktoren jedes Partikels nach den entsprechenden Bewegungsgleichungen.
Die Objekt-Kreation übernimmt Dust vollständig.

Natürlich existiert auch schon eine Reihe von Particle-Effekten.

Um ein Partikel-Objekt zu erzeugen benötigt Dust im Normalfall zwei Objekte:
-eines, welches für die räumliche Struktur des späteren Gesamt-Objektes verantwortlich ist: das Struktur-Objekt und
-eines welches das Aussehen der Partikel bestimmt: das Shape-Objekt.

Nun gibt es zwei Methoden, die Position und Größe der Partikel festzulegen:
-die Flächenmethode (FACE): in den Mittelpunkt jeder Fläche des Struktur-Objekts wird ein Shape-Objekt "gesetzt", dessen Größe etwa der größten Ausdehnung dieser Fläche entspricht
-die Punktmethode (POINT): jeder Punkt des Struktur-Objekts "bekommt" ein Shape-Objekt, hierbei haben alle Partikel die gleiche Größe, die sich so ergibt, daß alle Partikel zusammen das gleiche Volumen wie das Struktur-Objekt einnehmen.

(Natürlich wird nur das Volumen der Bounding-Boxes berechnet.)

Die Objekt-Attribute werden jeweils vom Shape-Objekt übernommen.

Ist der Programm-Parameter ALIGNP gesetzt, so werden die Partikel (Shape-Objekte) so ausgerichtet, daß ihre x-Achse mit den Flächennormalen des Strukturobjektes zusammenfällt. Die funktioniert allerdings im POINT-Modus, wo das Programm "künstliche Flächennormalen" erzeugen muß, nicht immer.

Intern erzeugt Dust dann eine Daten-Struktur, die die Positionen, Drehwinkel und Skalierungsfaktoren der Partikel sowie das Shape-Objekt enthält.

Diese

Struktur

ist sehr speichersparend und kann sowohl geladen als auch gespeichert werden.

Die Funktion P20 erzeugt daraus dann ein "richtiges" 3D-Objekt.

Partikel-Objekte können wie normale Objekte in Dust kopiert, zusammengefügt oder gelöscht werden; die Skalierungs-, Rotations- oder Translations-Funktionen wirken sich hier auf die einzelnen Partikel aus.

Die Farben der einzelnen Flächen des Shape-Objektes werden später reproduziert, was unerreichte Effekte ermöglicht.

SPHERE-OBJECTS

Desweiteren kann Dust aus einem Struktur-Objekt sog. sphere-objects generieren. Dabei bestehen die Partikel aus mathematischen Kugeln, die auch ausgerichtet werden können (z.B. wegen Texturen)

Die sphere-objects verhalten sich ansonsten genau wie die Particle-Objekte, man kann sie explodieren lassen, laden, speichern etc.

Einziger Unterschied ist, daß man sie nicht in ein normales Objekt umwandeln kann (P20), somit werden sie als TDDD-Group-Objekt gespeichert, was den Export z.B. nach Videoscape unmöglich macht.

Ist der Programm-Parameter SAVESPHEREP gesetzt, so wird beim Speichern des Imagine-Objekts automatisch ein Dust-Particle-Object mit der

Endung ".dpo" erzeugt, weil Group-Objekte nicht wieder in Dust eingeladen werden können.

1.10 Das Dust-Particle-Format

Dust bietet jedem Programmierer die Möglichkeit, Partikel-Effekte ohne Kenntnis irgendeiner Objekt-Struktur (oder des TDDD-Formats) auf einfachste Weise zu programmieren.

Die Anforderungen an die Programmiersprache sind minimal, das Programm muß binäre Daten lesen und ASCII-Zeichen auf den Bildschirm ausgeben können (und natürlich die Berechnungen ausführen).

Partikel-Objekt-Aufbau:

Ein Partikel-Objekt besteht aus dem Shape-Objekt und den Informationen

- Position (PPOS)
- Rotationswinkel (PROT)
- Skalierungsfaktor (PSCL)

für jedes Partikel.

Um einen Partikel-Effekt zu programmieren benötigt man nur die Felder PPOS, PROT, PSCL, die Partikel-Anzahl (OCOUNT) und vielleicht die Größe (PSIZE) des Shape-Objects.

Und genau diese Werte kann Dust jeweils als Binär-Datei ausgeben.

Dafür stehen die Befehle GETPPOS, GETPPROT, GETPSCL, GOCOUNT und GETPSIZE zur Verfügung.

Um die Position, ... der einzelnen Partikel zu modifizieren, werden dann später die Befehle SETPPOS, SETPROT und SETPSCL verwendet.

Somit läßt sich folgendes (endgültiges) Schema für die externe Programmierung eines Partikel-Effektes angeben:

1. Erzeugen eines Scriptes, daß das Partikel-Objekt aus zwei normalen Objekten erzeugt und je nach Bedarf die Werte PPOS, PROT, PSCL, OCOUNT und PSIZE als Binärdatei ausgibt.
2. Eigentliches Programm:
Das Programm muß die Binärdaten lesen, die Informationen verarbeiten (Effekt) und die veränderten Daten in Dust-Syntax (Verwendung der Befehle ← SETP*) am Bildschirm ausgeben. Diese Ausgabe wird dann in eine Datei umgeleitet.
3. Die erzeugte Datei wird von Dust abgearbeitet.

1.11 4.3. Die Preview-Funktion

Für jedes der im Speicher befindlichen Objekte können beliebig viele Fenster im Multitasking-Betrieb geöffnet werden. Wird ein Objekt gelöscht oder verändert, so werden alle dazugehörigen Fenster geschlossen bzw. die Inhalte neu gezeichnet. Die Aktivität der Fenster (Ändern des

Darstellungsmodus, vergrößern, verschieben, speichern,...) kann interaktiv oder durch Dust-Befehle geschehen. Dazu besitzt jedes Fenster einen Identifier.

Die Fenster können sowohl auf dem aktuellen Public-Screen als auch auf einem eigenen Screen geöffnet werden, wobei bis zu 256 Farben genutzt werden.

Es stehen folgende Darstellungsmodi zur Verfügung:

- Bounding Box : zum schnellen rotieren, zoomen (-Taste)
- Wireframe (<w>-Taste)
- Solid (<s>-Taste)
- Color : Solidmodus, der Flächenfarben und eine Lichtquelle berücksichtigt (<c>-Taste)

Der Modus Color benötigt mindestens OS3.0.

Jedes Fenster besitzt einige zusätzliche Shortcuts:

- Cursor-Tasten : zum Rotieren
- /<Help>-Taste : zum Zoomen
- <Esc>-Taste : zum Schließen
- <o>-Taste : zum Umschalten des Outlined-Modus
(Flächen werden wie im Solid-Modus umrandet dargestellt)
- <a>-Taste : zum Anzeigen der aktuellen Rotationswinkel um die X- und die
die Z-Achse und einiger anderer Werte
- <p>-Taste : zum An-/Abschalten der Perspektive
- <r>-Taste : zum Neuskalieren (bei KEEPSCALE=true)

Für die Partikel-Objekte gibt es keine Preview-Funktion: diese müssen erst in normale Objekte umgewandelt werden (P20).

1.12 4.2 Programmkonzept

Das Programm kann beliebig viele Objekte gleichzeitig im Speicher halten, jedes Objekt erhält einen Speicherplatz, der als objectID bezeichnet wird. Alle Befehle, die der Objektmanipulation gewidmet sind, verlangen mindestens die Angabe dieses objectID. Alle Strukturen, von denen es in Dust mehrere geben kann, besitzen einen

ID (identifier), dies ist immer eine Ganzzahl größer oder gleich Null

Dust ist eine Art Programmiersprache, so arbeitet es Befehle ab und kann Schleifen ausführen. Die Gestalt aller Befehle ist einheitlich, so bestehen sie aus einem Bezeichner und den in Klammern angeben Argumenten, die durch Kommata getrennt werden. In den Helptexten werden wie üblich in spitzen Klammern die Typen

der Argumente, in eckigen Klammern
optionale Argumente angeben, z.B.:

```
distort(<objectID>,[<percent of objSize>])
```

Hier muß das erste Argument vom Typ objectID sein, das zweite kann
weggelassen werden.

Im allgemeinen werden die Argumente Zahlen sein, es lassen sich aber
fast beliebig viele globale Variablen vom Typ float definieren, die
als Argumente angegeben werden können, das Programm wandelt sie bei
Bedarf selbst in Ganzzahlen um, angenommen die Variable "a2" sei
mit dem Wert 3.345 belegt, dann versucht der Befehl

```
load(4/a+0.75,obj)
```

das Objekt "obj" an Speicherstelle 2 zu laden.

Es ist nun üblich, Objekt- oder Bild-sequenzen mit speziell formatierten
Dateinamen zu speichern, wie z.B. "obj.0001", "obj.0002",...
Dazu gibt es in Dust den

Formatbefehl

"%", der in eine Zeichenkette

einen so formatierten Zahlenwert einfügt, z.B. wird durch

```
load(2,obj.%)
```

das Objekt "obj.0001" auf Speicherstelle 2 geladen, wenn der

Schleifen

zähler auf 1 gesetzt ist.

Im allgemeinen wird nun ein Objekt in Dust eingeladen, das daraus eine
transformierte Objektsequenz erzeugt. Der dazu benutzte Befehl habe die
Syntax:

```
XYZ(<objectID>,<n>,<Dateiname>).
```

Möchte man einige Effekte linearkombinieren, so ist es vorteilhaft,
die Objekte einzeln zu erzeugen und nicht auf Festplatte zu speichern.
So gibt es zu den meisten Befehlen ein Komplement der Struktur

```
XYZFRAME(<objectID>,<n>,<i>,<dest>),
```

welcher von den n Objekten nur das i-te erzeugt und auf Speicherstelle dest
ablegt.

Die Objekte selbst können in verschiedenen

Objectformaten

geladen und gespeichert werden, wobei beim Laden der Typ von Dust ↔
erkannt

wird, beim Speichern kann das gewünschte Format dauerhaft durch den

SET

-Befehl eingestellt werden.

Bei der Programmierung von Batch-files (siehe "Starfields.bat") benötigt man manchmal die Möglichkeit, die Ausführung von bestimmten Operation vom Status irgendwelcher Variablen abhängig zu machen. Dies wird durch das

```
IF
-Construct geleistet.
```

1.13 Unterstützte Objektformate

Die unterstützten Datei-Formate beschränken sich auf die der zur Zeit am verbreitetsten 3D-Programme.

TDDD-Format: Dies ist das Standard-Format für Dust, da ich Imagine benutze

Features:

- alle Attribute werden unterstützt
- Imagine3.0-Texturen und -Brushes
- every face has its own color (CLST)
- FLST (Farbe jeder einzelnen Fläche)
- hard/soft edges
- Subgroups werden unterstützt

Einschränkungen

- keine FLST und RLST (Filter und Reflektionsvermögen jeder einzelnen Fläche)

Lightwave-Format

Features:

- Polygone werden als konvex angenommen und in Dreiecke konvertiert
- optionale Kantenoptimierung
- Surfaces werden im Imagine Subgroups konvertiert
- das DOUBLESIDED-flag kann gesetzt werden
- die Specular-Wertewerden konvertiert
- alle Texturen/Brushes/... werden berücksichtigt

Einschränkungen:

- Linien (Polygone mit zwei Punkten) und Punkt-Polygone werden ignoriert

MaxonCinema4D-Format

Features:

- Vierecke werden in Dreiecke umgewandelt
- Axis-Attribute werden konvertiert
- Chunks mit der Kennung 8 werden intern verwaltet, da sie bei allen von mir getesteten Objekte auftraten

Einschränkungen:

- Materialien werden nicht erkannt (keine Dokumentation)
- Die Chunks mit den Kennungen 5,6,7 werden ignoriert

Videoscape-Format

Features:

- Farbcodes werden unterstützt
-

-Erzeugung zweiseitiger Flächen optional

Sphere-Format (TDDD-Group-Objects, die aus echten Kugeln bestehen)

In diesem Format speichert Dust Sphere-Objects, spezielle Particle-Objekte, die aus mathematische Kugeln bestehen

Particle-Format (Dust-eigenes Format)

Dieses Format wurde geschaffen, um Particle-Object schnell und platzsparend zu speichern.

Bemerkung: Dust convertiert LW-Surfaces in TDDD-Subgroups und TDDD-Subgroups in LW-Surfaces – die meisten Objekt-Konverter (wie z.B. Vertex, Pixpro oder Castillian) können dies nicht.

1.14 Datentypen

Dust kennt folgende Datentypen:

Identifizier:

Ganzzahl größer oder gleich Null

Real-Zahl: beliebiger

mathematischer Ausdruck
, der globale Variablen
enthalten darf

Ganz-Zahl: beliebiger mathematischer Ausdruck, dessen Ergebnis gerundet wird

String: Zeichenkette mit Formatierungsbefehlen

Dateiname: String oder Leerstring, im Falle eines Leerstrings wird bei der Auswertung ein File-Requester geöffnet

1.15 Mathematische Ausdrücke

Alle Dust-Befehle akzeptieren mathematische Ausdrücke anstatt von gewöhnlichen Zahlen, andere Befehle erwarten Ausdrücke in speziellen Variablen:

1. Variablen der FUNC-Befehle

X0 - Ausgangs-x-Koordinate (z.B. eines Punktes oder Flächenmittelpunktes)
Y0 - Ausgangs-y-Koordinate
Z0 - Ausgangs-z-Koordinate
T0 - zunächst willkürlicher Parameter, wichtig für Animationen (↔
Zeitparameter)

Beispiel: Um ein Modell der Funktion $\sin(x^2+y^2)$ zu erhalten ist nur \leftrightarrow folgendes

nötig:

-man benötigt eine Plane als Ausgangsobjekt (x-y-Ebene)
 -der Aufruf "func(2,"30*sin(x0*x0/30+y0*y0/30)\",0,z)" erzeugt besagtes Objekt, wobei die Faktoren je nach Größe der Plane (bei mir von -50..50 in x- und y-Richtung) gewählt werden müssen.

2. Vordefinierte Konstanten

"pi", "e"

3. Operatoren

"+", "-", "*", "/", "^"

4. Funktionen

"entier", "int", "abs", "sqr", "sqrt",
 "exp", "ln", "log", "log10", "log2", "tentox", "twotox",
 "sin", "arcsin", "cos", "arccos", "tan", "arctan",
 "sinh", "cosh", "tanh", "artanh", "degtorad", "radtodeg"
 "rnd", "fac", "ceil", "floor", "round"

5. Hinweise

Enthält ein Dust-Befehl ein Argument, das ein mathematischer Ausdruck ist, der eine Funktion enthält, so muß dieses Argument in Anführungszeichen gesetzt werden, z.B. calc("a=3*sin(34)").

Alle Winkelangaben werden in Grad erwartet.

Im Gegensatz zu allen anderen Dust-Funktionen arbeiten die CALC-Befehle mit doppelter Genauigkeit, hier lohnt sich der Einsatz einer FPU.

1.16 Identifizier

Dust kennt folgende identifizier:

objectID : Speicherplatz eines Objekts

particleID : Speicherplatz eines Particle-Objekts

windowID : Nummer eines Fensters

brushID : Nummer eines Brushes eines Objektes

textureID : Nummer einer Textur eines Objektes

1.17 Schleifen

Dust kann beliebig tief geschachtelte FOR-Schleifen ausführen, \leftrightarrow wobei

der Schleifenzähler immer gleich der Laufvariablen der innersten Schleife gesetzt wird. Der Schleifenzähler gibt die Zahl an, die in eine Zeichenkette eingefügt wird, wenn sie Formatierungsbefehle enthält.

Seit Version 2.3 kann den Formatierungsbefehlen auch ein numerisches Argument in Klammern folgen.

Syntax:

```
FOR(<Laufvariable>,<von>,<bis>[,<SchrittWeite>]
.
.
.
END
```

Beispiel:

```
for(i,1,10)
  echo("unformatted:$, formatted:%")
end
```

oder

```
for(i,1,10)
  echo("unformatted:$(3*i+1), formatted:%(4*i*sin(12))")
end
```

1.18 Formatierungsbefehle

Enthält eine Zeichenkette Formatierungsbefehle, so werden sie durch den aktuellen Wert des Schleifenzählers oder durch die angegebene Zahl ersetzt.

\$ - fügt den Wert ohne Formatierung ein,
% - fügt den Wert auf vier Zeichen formatiert ein,

So wird aus dem String "\$. Objekt: obj.%", wenn der Schleifenzähler auf 23 gesetzt ist: "23. Objekt: obj.0023".

Aus dem String "\$ (3*i+1). Objekt: obj.%(i*12)" wird der String "7. Objekt: obj.0024", wenn die globale Variable i auf 2 gesetzt ist.

1.19 4.1. Installation

Zuerst müssen das Keyfile "Dust.key" und die Konfigurationsdatei ".dustrc" nach S: kopiert werden.

Das Programm benötigt ein Verzeichnis, in dem sich die Online-Help-Texte befinden. Normalerweise ist das das Verzeichnis "DustHelp" im aktuellen

Verzeichnis. Wollen Sie dies so belassen, so ist die Installation abgeschlossen.

Möchten Sie die Help-Texte z.B. im Verzeichnis "HELP:Dust" einrichten, so muß dies dem Programm mitgeteilt werden. Starten Sie Dust und geben Sie ein:

```
"set(helpdir,help:dust)"
"saveconfig"
"exit".
```

Nun sollte nach erneutem Programmstart nach der Eingabe von "help" die erste Textseite erscheinen.

Bemerkungen: 1. Das Programm benötigt folgende Libraries:

```
asl.library oder arp.library
mathieeedoubbas.library
mathieeedoubtrans.library
mathtrans.library
xpkmaster.library
xpkiDEA.library
rexxsyslib.library
```

und folgende Befehle im logischen C:-Verzeichnis:

```
delete
rename
execute.
```

2. Das Programm sollte von der Shell aus gestartet werden und der Stack etwa auf 30000 bytes gesetzt werden. Desweiteren ist die Verwendung der Programme

```
KingCON (History, Scroll-Balken, Filename-Completion),
Powersnap (Ausschneiden von Beispielen aus der Online-Help) und
XSize (Preview-Fenster wie unter UNIX vergrößern)
```

sehr zu empfehlen.

1.20 1. Programmbeschreibung

Dust ist eine Spezialeffekt-Software, die sich mit der Manipulation von 3D-Objekten befaßt. Es ist somit eine Ergänzung zu den gängigen Objekt-Editoren und bietet fast ausschließlich Features an, die diesen fehlen.

Zielgruppe sind hauptsächlich semiprofessionelle Anwender, die über elementare Programmierkenntnisse und mathematisches Verständnis verfügen. Außerdem sollte sich der Benutzer in der Bedienung eines 3D-Modelers und eines 3D-Renderers gut auskennen. Optimal ist hier der Einsatz des Programmes "Imagine3.1" oder "LightWave3D". Seit Version 2.32 wird auch das Objektformat des deutschen Programms "MaxonCinema4D" unterstützt.

Diese herausragenden Features des Programmes sind:

- lokale Metamorphose, die Metamorphosen in höchster Qualität

erlaubt

- Smooth-Funktion für Objekte, die unglaubliche Ergebnisse liefert
- direkte Unterstützung der Programme "Imagine" und "Lightwave"
- Laden, Speichern und Anzeigen von Objekt-Sequenzen
- einfache Metamorphose beliebiger Objekte (incl. der Farben der einzelnen Flächen), ←
- Partikel-System: extrem einfach zu handhaben, ermöglicht auch bei der Objekt-Modellierung unerreichte Effekte,
- realistische Explosionen (Gravitation, Stokes'sche Reibung, Drehimpulse),
- realistische Wasser-Wellen (dreidimensionale nicht-harmonische Wellen)
- ein- bis dreidimensionale Wellen verschiedener Arten incl. Partikel-Wellen
- Punkte, Flächenfarben, Partikelposition, -Drehwinkel und -Skalierungsfaktoren können algorithmisch modifiziert werden, somit können Sie Dust auch als hochwertigen Funktionsplotter verwenden
- jeder Programmierer kann mit Dust auf einfachste Weise die verschiedensten Partikel-Effekte verwirklichen, dabei werden keinerlei Kenntnisse irgendeiner Objekt-Struktur (oder gar des TDDD-Formats) vorausgesetzt
- Jede Fläche kann (wie in Imagine) eine eigene Farbe erhalten, diese werden bei der Umwandlung in Partikel-Objekte reproduziert
- Verwaltung und Veränderung von Imagine3.0-Texturen und -Brushes
- Sphere-Objects: Particle-Objekte aus mathematischen Kugeln
- Erzeugung von Objekten von externen Programmen aus möglich
- benutzerdefinierte Variablen, mathematische Ausdrücke anstatt einfacher Zahlen als Argumente, Schleifen
- Einbindung in andere Programme durch ARexx-Port möglich
- sehr leistungsfähige Preview-Funktion
- Kommandozeilen-Vervollständigung
- Online-Help

Andere Programme, wie z.B. Imagine, das von Dust direkt unterstützt wird, bieten auch einige dieser Effekte an. Der Nachteil dabei besteht darin, daß diese Effekte erst während des Render-Prozesses berechnet werden; man kann die Objekte, auf die diese Effekte einwirken, also nicht nachbearbeiten, weil es sie gar nicht gibt - mit Dust können unendlich viele Spezialeffekte linearkombiniert werden.

Was kan das Programm wirklich ?

1.21 2. System- und Anwendervoraussetzungen

Dust ist für den mindestens semiprofessionellen Anwender konzipiert, der sich mit 3D-Grafik beschäftigt und an die Grenzen der gängigen Software gelangt ist, Programmierkenntnisse in einer Hochsprache (auch ARexx) sind empfehlenswert.

Profis werden das Programm als flexibel empfinden, Einsteiger, die schon Probleme mit dem Umgang eines 3D-Renderers oder -Modelers haben, werden keine Freude daran finden.

Dust eignet sich vor allem in Verbindung mit "Imagine", "Lightwave" oder "MaxonCinema4D", andere Programme werden auch in Zukunft nicht direkt unterstützt.

So stimmen die Systemvoraussetzungen auch mit denen von Raytracern überein: möglichst schneller Rechner mit viel Speicher und schneller Festplatte - die Zeiten eines A2000 oder eines A500, auf dem das Programm übrigens auch läuft, sind endgültig vorbei.

Darüberhinaus sind das Betriebssystem OS3.1 und der AGA-Chipsatz oder eine Grafikkarte nötig, um die Preview-Funktion effizient nutzen zu können.

1.22 3. Programmstatus und Anwenderlizenz

Dust ist SHAREWARE, die Gebühr beträgt \$25 oder 25 DM in bar.

Registrierte Benutzer erhalten ein Keyfile, in dem gewisse persönliche Daten in verschlüsselter Form gespeichert sind.

Alle zum Programm mitgelieferten Dateien einschließlich des Programmes selbst mit Ausnahme des registrierten Keyfiles sind frei kopierbar.

Zuwiderhandlungen werden zivilrechtlich verfolgt, anhand des Keyfiles wird der Erstbesitzer ermittelt und dieser bestraft.

Registrierte Benutzer erhalten die Garantie, daß Programmfehler schnellstmöglich behoben werden, allerdings versende ich keine Disketten, Programm-Updates erscheinen ausschließlich im Aminet oder werden von mir per EMail verschickt.

Die Benutzung des Programmes geschieht ausschließlich auf eigenes Risiko, so übernehme ich keinerlei Haftung für irgendwelche Schäden, die bei der Arbeit mit Dust auftreten.

Wichtige Hinweise

Bei der nichtregistrierten Version sind die Befehle, die in der Datei README2 angeben sind, nicht aktivierbar.

Benutzer ältere Versionen von Dust sollten nach Erhalt einen neuen zuerst die README-Dateien und dann das HISTORY-File durchlesen, in der

Dokumentation sind die jeweils neuen Features nicht einfach zu finden.

1.23 7. Die Adresse des Authors

Andreas Maschke
Zenkerstraße 5
06108 Halle/Saale
Germany

Phone: ++49 (0)345/5170331
EMail: epghc@cluster1.urz.Uni-Halle.DE

1.24 6. Copyrights

Imagine - Copyright ©1993 Impulse Inc.
VideoScape - Copyright ©198? Aegis
LightWave - Copyright ©1990 NewTek Inc.
ISL - Copyright ©1993 Grizzly Bear Labs
Dust - Copyright ©1994 A.Maschke
XPK - Copyright ©1992 Urban Dominik Mueller, Bryan Ford and many others
XFH-Handler - Copyright ©1991 Kristian Nielsen.
IDEA - Copyright ©1992 Andre Beck (XPK-Implementation)
RTPatch - Copyright ©1994 Nico François
PowerSnap - Copyright ©1994 Nico François
Most - Copyright ©1994 Uwe Röhm
Pixel3D - Copyright ©1993 Axiom Software
ARexx - Copyright ©1987 by William S. Hawes
XSize - Copyright ©1994 by C. Melberg and G. Rehm

Alle in Dust verwendeten Algorithmen und Prozeduren habe ich selbst entwickelt, einzige Ausnahmen sind:

IFF-Saver : original by Friedtjof Siebert ("IFFSupport.mod")
Math-Parser: original by Stefan Salewski ("Formula.mod")

Somit besitzt außer mir und o.g. Autoren niemand irgendwelche Rechte an meinem Programm, solange ich sie ihm nicht schriftlich erteile !

1.25 4.4 Beschreibung aller Befehle

Die Befehle sind nach folgenden Themen geordnet:

1. Laden und Speichern von Objekten
2. Modifizieren oder Erzeugen von Einzelobjekten
3. Modifizieren oder Erzeugen von Objekt-Sequenzen
4. Modifizieren von Objektattributen

5. Information zu bestimmten Objekt-Komponenten
 6. Funktionen für Programmierer
 7. Verschiedene Funktionen (Fenster, Interface, ...)
- Bemerkung: Befehlsbezeichner werden hier zur Hervorhebung groß geschrieben,
in Dust müssen sie allerdings KLEIN angegeben werden. ↔

1.26 1. Laden und Speichern von Objekten

FILETYPE

Objekt-Format einer Datei ausgeben

LOAD

ein Object (TDDD/LW/VS/Particle-Format) laden

LOADGROUPOBJ

Laden eines Objektes aus einer Objekt-Gruppe (TDDD-Group-Object)

LOADSEQ

eine Objekt-Sequenz laden (TDDD/LW/VS/Particle-Format)

SAVE

Objekte speichern (im SFORMAT)

SAVELW

Objekte im Lightwave-Format speichern

SAVEMC4D

Objekte im MaxonCinema4D-Format speichern

SAVEP

Partikel-Objekte speichern (DUST-Format)

SAVEPSEQ

eine Partikel-Objekt-Sequenz speichern

SAVEPOBJ

Partikel-Objekte speichern (im SFORMAT)

SAVESEQ

eine Objekt-Sequenz speichern (im SFORMAT)

SAVESPHERES

ein sphere-object als TDDD-Group abspeichern

SAVETDDD

Objekte als Imagine-Objekte speichern

SAVEVS

Objekte als Videoscape-Objekte speichern

SHOWTDDD

Zeigt die Hunks (Objekte) einer TDDD-Datei an

1.27 2. Modifizieren oder Erzeugen von Einzelobjekten

ADDFACE

eine Fläche oder ein neues Object erzeugen

ADDSGROUP

eine Fläche einer Subgroup zufügen/Subgroup erzeugen

AXALIGN0

setzen der Ausrichtung des lok. Objektkoord.-sys. auf 0,0,0

AXPOS

Ändern der Position des lok. Objektkoord.-sys.

AXSIZE

Ändern der Größe des lokalen Objektkoordinatensystems

CDEFORM

locale kontrollierte Metamorphose

CDEFORMINTERP

locale kontrollierte Metamorphose mit Interpolation des
Zielobjektes ←

CENTERAXIS

lokales Koordinatensystem zentrieren

CHECKOBJECT

Objekt überprüfen und fehlerhafte Kanten, Flächen, Punkte löschen

COPY

Objekte kopieren

COPYP

Partikel-Objekte kopieren

COPYPPOS

Partikel-Positionen kopieren (um ANIMP*FUNC zu kombinieren)

COPYPROT

Partikel-Drehwinkel kopieren (um ANIMP*FUNC zu kombinieren)

COPYPSCL

Partikel-Skalierungsfaktoren kopieren (um ANIMP*FUNC zu
kombinieren) ←

CUTSG

Subgroup einschließlich Punkten, Flächen und Kanten löschen

DISTORT

Objekte verbeulen

EXPANDSG
Subgroup um ihre Randflächen erweitern

EXPLODE
realistische Explosionen

EXPLODEFRAME
realistische Explosionen

EXTRACTSG
neues Objekt nach Vorgabe einer Subgroup erzeugen

FUNC
Punkte eines Objektes algorithmisch verändern

INSERTPOINT
einen Punkt in eine Fläche einfügen (wie Imagine's "fracture")

JOIN
zwei Objekte verbinden

JOINP
zwei Partikel-Objekte verbinden

KILL
Objekte löschen

KILLFREEPOINTS
unbenutzte Punkte löschen

KILLEGE
Kanten löschen

KILLFACE
Flächen löschen

KILLP
Partikelobjekte löschen

KILLPOINT
Punkte löschen

LATTICE
Flächen extrudieren, um gitterähnliches Object zu erzeugen

MERGE
unnötige Punkte löschen (schnell)

MORPHFRAME
Dreiecks-Metamorphose

O2P
Zwei Objekte in in Partikel-Objekt konvertieren

O2S
Erzeugen eines sphere-objects

P20
ein Partikel-Objekt in ein normales Objekt konvertieren

PMORPH
Morph-Preprocessor

POSITIVE
Bewegen eines Objektes in den positiven Halbraum

PPOSFUNC
Positionen der Partikel algorithmisch verändern

PROTFUNC
Drehwinkel der Partikel algorithmisch verändern

PSCLFUNC
Skalierungsfaktoren der Partikel algorithmisch verändern

PWAVE1DFRAME
Transversal-/Longitudinal- Partikel-Welle entlang der x-Achse

PWAVE2DFRAME
Transversal-/Longitudinal- Partikel-Welle entlang der x-y-Ebene

PWAVE3DFRAME
3D-Partikel-Welle

RANDOMPPOS
Partikel-Position verändern

RANDOMPROT
Partikel-Drehwinkel verändern

RANDOMPSCL
Partikel-Größe verändern

ROTATE
Objekte rotieren

ROTATEAXIS
Rotieren des lokalen Koordinatensystems eines Objekts

SAMEPOS
eine Koordinate aller Punkte einer Subgroup auf einen Wert setzen

SCALE
Objekte skalieren

SCALEFACES
Objekt-Flächen bezüglich der Flächenmittelpunkte skalieren

SCALEP
Partikel skalieren

SETPPOS
Setzen der Position eines Partikels

SETPOINT
Setzen der Position eines Punktes

SETPROT
Setzen der Drehwinkel eines Partikels

SETPSCL
Setzen der Skalierungsfaktoren eines Partikels

SHRINKSG
die Randflächen aus einer Subgroup entfernen

SMOOTH
Objekte mit Hilfe einer Spline-Interpolation weicher machen

SMOOTHINNER
Smoothing, ohne die Randzone des Objektes zu verändern

SMOOTHSG
Smoothing einer Subgroup

SUBSGROUP
eine Fläche aus einer Subgroup entfernen/Subgroup entfernen

TRANSLATE
Objekte umbewegen

TRIANGULATE
ungebundene Flächen erzeugen

WATERFRAME
Wasser-Wellen

WATERZFRAME
Wasser-Wellen wie WATERZ

WAVE1DFRAME
eindimensionale Wellen

WAVE2DFRAME
zweidimensionale Wellen

WAVE3DFRAME
dreidimensionale Wellen

1.28 3. Modifizieren oder Erzeugen von Objekt-Sequenzen

ANIMFUNC
animierte algorithmische Veränderungen der Objektpunkte

ANIMCFUNC
animierte algorithmische Veränderungen der Flächenfarben

ANIMPOSFUNC
animierte algorithmische Veränderungen der Partikel-Positionen

ANIMPROTFUNC
animierte algorithmische Veränderungen der Partikel-Drehwinkel

ANIMPSCLFUNC
animierte algorithmische Veränderungen der Partikel- ←
Skalierungsfaktoren

BUILD
Objekte durch Löschen von Flächen verschwinden lassen

BUILDRND
zufallsgesteuert

BUILDMORPH
Build-Morph (linear)

BUILDMORPHRND
Build-Morph (zufallsgesteuert)

DEFORMMORPH
Deformations-Metamorphose

EXPLODE
realistische Explosionen

MORPH
Dreiecks-Metamorphose

MORPHATTS
Attribute einer Objekt-Sequenz interpolieren

P2OSEQ
eine Partikel-Objekt-Sequenz in eine Objekt-Sequenz konvertieren

PEXPLODE
eine realistische Partikel-Explosion durchführen

PFALL
Gravitation

PFALL2
Gravitation, für jeden Punkt unterschiedlich

PWAVE1D
Transversal-/Longitudinal- Partikel-Welle entlang der x-Achse

PWAVE2D
Transversal-/Longitudinal- Partikel-Welle entlang der x-y-Ebene

PWAVE3D
3D-Partikel-Welle

RENAME
Objekt-Sequenzen umbenennen

WATER

Wasser-Wellen

WATERZ
Wasser-Wellen, nur z-Koordinate verändern

WAVE1D
eindimensionale Wellen

WAVE2D
zweidimensionale Wellen

WAVE3D
dreidimensionale Wellen

1.29 4. Modifizieren von Objektattributen

BR SAXALIGN0
Ausrichten einer Brush-Axis auf 0,0,0

BR SAXPOS
Verändern der Brush-Position

BR SAXSIZE
Verändern der Brush-Größe

BR SDIR
Verändern des Pfades aller Brushes eines Objekts

BR SNAME
Verändern des Namens eines Brushes

CENTERBR SAXIS
Koordinatensystem eines Brushes zentrieren

CENTERTXTAXIS
Koordinatensystem einer Textur zentrieren

CFUNC
Farben der Objektflächen algorithmisch verändern

COLOR
Objektattribut ändern

COPYATTS
Attribute kopieren

COPYAXIS
Axis-Attribute kopieren

COPYBRS
Kopieren/Anhängen der Brushes von einem Objekt zum anderen

COPYCLST
Kopieren der Flächenfarben von einem Objekt zum anderen

COPYTXT
Kopieren/Anhängen der Texturen von einem Objekt zum anderen

DITHER
Objektattribut ändern

HARDNESS
Objektattribut ändern

KILLBRS
einen oder alle Brushes eines Objektes löschen

KILLTXT
eine oder alle Texturen eines Objektes löschen

REFL
Objektattribut ändern

ROTATEBRSAxis
Rotieren des lokalen Koordinatensystems eines Brushe

ROTATETXTAxis
Rotieren des lokalen Koordinatensystems einer Textur

ROUGHNESS
Objektattribut ändern

SETCLST
Farbe einer einzelnen Fläche ändern

SETCOLSGROUP
Farbe einer Subgroup setzen

SHININESS
Objektattribut ändern

SPEC
Objektattribut ändern

TRANS
Objektattribut ändern

TXTAXALIGN0
Ausrichten einer Textur-Axis auf 0,0,0

TXTAXPOS
Verändern der Textur-Position

TXTAXSIZE
Verändern der Textur-Größe

TXTDIR
Verändern des Pfades aller Texturen eines Objekts

TXTNAME
Verändern des Namens eine Textur

TXTPARAM
Verändern eines Textur-Parameters

1.30 5. Information zu bestimmten Objekt-Komponenten

GETCOLSGROUP
Farbe einer Subgroup anzeigen

PSTATS
Information über Partikel-Objekte

PSTATS2
Information über belegte Partikel-Objekt-Speicherplätze

SHOWBRS
Information über alle oder einen Brush(es) anzeigen

SHOWTXT
Information über alle oder eine Textur(en) anzeigen

SIZE
Objekt-Größe ausgeben

STATS
Information über Objekte ausgeben

STATS2
belegte Objektespeicherplätze ausgeben

WRITEATTS
Objektattribute ausgeben

WRITEAXIS
Größe, Position, ... des lokalen Objektkoordinatensystems ↔
ausgeben

WRITECLST
Farbe aller Objektflächen ausgeben

WRITEEDGES
Objektkanten ausgeben

WRITEFACES
Objektflächen ausgeben

WRITEPOINTS
Objektpunkte ausgeben

WRITEPPOS
Partikel-Positionen ausgeben

WRITEPROT
Partikel-Drehwinkel ausgeben

WRITEPSCL

Partikel-Größe ausgeben

WRITESGROUP

Flächen, die eine Subgroup bilden, anzeigen

1.31 6. Funktionen für Programmierer

CALC oder .

Mathematische Ausdrücke berechnen, Variablen definieren

FOR

FOR-Schleifen

IF

IF-Construct

ECOUNT

Kantenanzahl in Variable "ecount" speichern

FCOUNT

Flächenanzahl in Variable "fcount" speichern

GETPOINT

Punktposition in Variablen speichern

GETOCOUNT

Anzahl der Partikel ausgeben (Datei EXFILE)

GETPSIZE

Ausdehnung des Shape-Objekts ausgeben (Datei EXFILE)

GETPPOS

Positionen aller Partikel-Objekte ausgeben (Datei EXFILE)

GETPROT

Drehwinkel aller Partikel-Objekte ausgeben (Datei EXFILE)

GETPSCl

Skalierungsfaktoren aller Partikel-Objekte ausgeben (Datei EXFILE ↔)

OCOUNT

Partikel-Objektanzahl in Variable "ocount" speichern

PCOUNT

Punktanzahl in Variable "pcount" speichern

1.32 7. Verschiedene Funktionen (Fenster, Interface, ...)

ABOUT

Programminformationen

AVAIL
freien Speicher ausgeben

CALC oder .
Mathematische Ausdrücke berechnen, Variablen definieren

CD
aktuelles Verzeichnis wechseln

CLOSEWINDOWS
Preview-Fenster eines Objekts schließen

EXEC
Stapeldatei abarbeiten

ECHO
Zeichenkette und Variablen ausgeben

FOR
FOR-Schleifen

GET
Systemparameter ausgeben

IF
IF-Construct

INTERPOLATEDATA
Daten durch Splines interpolieren

LOADCONFIG
Einstellungen laden

LWSTAGING
Objekt-Einbindung in Lightwave

MEMORY
Speicherverbrauch und Adressen eines Objektes ausgeben

MEMORYP
Speicherverbrauch und Adressen eines Partikel-Objektes ausgeben

REQUEST
den Benutzer bestimmte Aktionen bestätigen lassen (in Batch-Files ↔
)

REXX
den Dust-ARexx-Modus aktivieren

SAVECONFIG
alle Einstellungen abspeichern

SET
Programm-Parameter ändern

SHOWVALUES
Anzeigen der benutzerdefinierten Konstanten

STAGING2
ISL2.0-Datei erzeugen (Imagine2.0)

STAGING3
ISL3.x-Datei erzeugen (Imagine3.x)

TIME
Bearbeitungszeit des letzten Befehl ausgeben

WINDOW
Vorschau-Fenster öffnen

WINDOWSEQ
gut arrangierte Vorschau-Fenster für eine Objektsequenz öffnen

WINDOWCLOSE
Vorschau-Fenster schließen

WINDOWDRAWMODE
Ändern des Zeichenmodus eines Vorschau-Fensters

WINDOWFRONT
ein Vorschau-Fenster in den Vordergrund bringen

WINDOWOUTLINED
Ändern des Outline-Flags eines Fensters

WINDOWPERSPECTIVE
Ändern des Perspective-Flags

WINDOWPOS
Ändern der Position eines Vorschau-Fensters

WINDOWREDRAW
Fensterinhalt neu zeichnen

WINDOWRESCALE
Fenster neu skalieren (bei keepscale=TRUE)

WINDOWROTX
Rotationswinkel um die X-Achse erhöhen

WINDOWROTZ
Rotationswinkel um die Z-Achse erhöhen

WINDOWSAVE
Speichern des angegebenen Fensters als IFF-Bild

WINDOWSIZE
Ändern der Größe eines Vorschau-Fensters

WINDOWZOOM
Zoom-Faktor erhöhen

!
DOS-Kommandozeile ausführen

; oder #
Kommentar

1.33 4.5 Beschreibung aller Programmparameter

Durch den SET-Befehl können folgende Einstellungen, die zur hier ←
nur
Hervorhebung groß geschrieben werden, getroffen/geändert werden:

SFORMAT
Format, in dem alle Objekte gespeichert werden

ALIGNP
Ausrichtung der Partikel entlang der Flächennormalen des ←
Strukturobjektes

ASPECT
Aspect der Vorschau-Fenster

BACKFACES
DOUBLESIDED-flag für Lightwave/
Flächen für Videoscape-Objekte doppelt erzeugen

OPTEDGES
Kanten optimieren beim Laden von Lightwave-/VS-Objekten

LOG
Erstellen eines Log-Files

LOGFILE
Dateiname des Log-Files

QUIET
Unterdrücken sämtlicher Textausgaben

COMPLETE
Kommando- und Parameter-Vervollständigung an/aus

ACTVAL
Schleifenzähler außerhalb von Schleifen

SAVESPHEREP
Automatische Erzeugung von Particle-Dateien beim Abspeichern von
Sphere-Objekten

SPLINETYPE
Spline-Art, die von Befehlen wie SMOOTH benutzt werden

SPLINEENDS
Art der Spline-Enden

SPLINESUBDIV
Anzahl der zu erzeugenden Punkte pro Segment

MAXANGLE

Größter Winkel zwischen zwei Kanten für den SMOOTH-Operator

INTERPMODE

Bewegungsmodus der Punkte für CDEFORM

STARTPCORR

Startpunkt-Korrektur für CDEFORM

FORCESWAP

Richtungskorrektur für CDEFORM

ECHO

überflüssige Textausgabe bei der Ausführung von Batch-files ↔
untersrücken

EXFILE

Dateiname für externe Binär-Dateien (siehe z.B. GETPPOS)

EXFORMAT

Format der externen Binär-Dateien ändern (wichtig z.B. für GCC)

KEEPASPECT

Aspect bei Vergrößerung eines Fensters beibehalten

CHECKMOUSE

Zeichenvorgang durch Drücken der linken Maustaste abbrechen

RANDOM

globaler Wert für PutSeed()

DRAWMODE

Darstellungsweise der Fenster

KEEPSCALE

Skalierungsfaktor bei Objektveränderung beibehalten

LEFT

linke Eckkoordinate der Fenster

OUTLINED

Outlined-Flag

ROTX

Drehwinkel um die x-Achse des Views

ROTZ

Drehwinkel um die z-Achse des Views

TOP

obere Eckkoordinate der Fenster

WIDTH

Breite der Fenster

ZOOM

Zoom-Faktor des Views

WARNINGS
Warnungen ein

WINDOWSTACK
Stackgröße der Zeichenprozesse

WINDOWPRI
Priorität der Zeichenprozesse

PAGER
gibt den Pfad des Programms an, das die Help-Texte anzeigen soll

HELPPDIR
gibt den Pfad an, in dem sich die Online-Help-Texte befinden

HELPPDIR2
gibt den Pfad an, in dem sich zusätzliche Texte befinden

BWLEFT
x-Koordinate des Befehlsabbruch-Fensters

BWTOP
y-Koordinate des Befehlsabbruch-Fensters

BREAKWIN
Befehlsabbruch-Requester unterdruecken

LWCMD1-3
Lightwave-Befehle, die vor jedem Frame ausgeführt werden

1.34 Was kann das Programm wirklich ?

1. Metamorphosen beliebiger Objekte (CDEFORM, MORPH, DEFORMMORPH, BUILDMORPH)

Dust kann zwischen zwei beliebigen Objekten eine Metamorphose durchführen. Das Programm bietet dazu vier Methoden an:

Locale Deformations-Metamorphose: (CDEFORM, CDEFORMINTERP)

Hierbei werden einzelne Kurvenzüge des Ausgangsobjects in Kurvenzüge des Endobjekts umgewandelt. Der Kurvenzug des Endobjekts kann dabei auch durch Interpolation zweier realer Kurvenzüge gewonnen werden.

Weil dieser Effekt nur das Ausgangsobjekt verformt, kann später die Metamorphose selbst im Renderer vorgenommen werden.

(z.B. unter Benutzung der STATES-Funktion von Imagine)

Die Idee dazu kommt von den 2D-Morph-Programmen, wie z.B. "MorphPlus". Hier definiert man, wie sich welche Region des Ausgangsbildes in welche Region des Endbildes umwandeln soll. Nur so kann man gute Ergebnisse erhalten, da der Computer nicht denken kann.

Es gibt einige Befehle und Programmparameter, die den Umgang mit dieser Funktion erleichtern.

Globale Deformations-Metamorphose: (DEFORMMORPH)

Dies ist ein sehr leistungsfähiger und langsamer Algorithmus, der oft sehr gute Ergebnisse liefert. Allerdings ändert sich die tatsächliche Anzahl erzeugter Objekte dynamisch, somit kann nur ein Minimum für die Anzahl der zu erzeugenden Frames angegeben werden, die tatsächliche Objektanzahl wird nach der Metamorphose in der Variablen "result" gespeichert.

Für gute Ergebnisse sollte die Unterschiede zwischen den Objekten klein sein. (Wenn Sie ein Dreirad in 25 Kugeln verwandeln wollen, so verwenden Sie besser die Dreiecks-Metamorphose.)

Dreiecks-Metamorphose: (MORPH, PMORPH)

Hier werden zunächst zwei neue Objekte gleicher Flächen- und Punktzahl aus den beiden Quellobjekten berechnet.

Danach kann die eigentliche Metamorphose entweder direkt in Imagine oder auch durch Dust durchgeführt werden.

Diese Methode ist SEHR rechenaufwendig, funktioniert aber bei allen Objekten.

Build-Metamorphose: (BUILDMORPH, BUILDMORPHRND)

Hierbei wird das eine Objekt aus Flächen zusammengesetzt während die Flächen des anderen nacheinander gelöscht werden. Dies kann linear oder zufällig erfolgen.

2. Objekt-Smoothing (SMOOTH)

Diese Funktion interpoliert die Oberfläche eines Objektes durch Splines in Richtung der Kanten, so wird z.B. aus einer "groben" Kugel eine "richtige" Kugel

3. Explosionen (EXPLODE)

Dust berechnet auch realistische Explosionen. Im Unterschied zu anderen Programmen werden dabei berücksichtigt:

- Gravitation
- Stokes'sche Reibung
- Drehimpulse

Das bedeutet u.a.:

- große Teile fliegen nicht so weit wie kleinere
- große Teile drehen sich langsamer als kleinere
- alle Teilchen landen infolge der Gravitation auf dem Boden (z-Koordinate 0)

ACHTUNG: Alle Objekte müssen sich im positiven z-Halbraum befinden, ansonsten bewegt das Programm sie vor der Explosion dorthin.

4. Wellen (WAVE1D,WAVE2D,WAVE3D)

Das Programm läßt über beliebige Objekte (harmonische) Wellen laufen. Folgende Wellentypen werden angeboten:

- transversale oder longitudinale eindimensionale Wellen
 - transversale oder longitudinale zweidimensionale Wellen
 - dreidimensionale Kugel- oder "gallertartige" Wellen.
-

Dabei sind folgende Parameter frei einstellbar:

- Wellenlänge
- Amplitude
- Wellenzentrum
- Dämpfung
- Phase

Für Einsteiger werden für alle Wellentypen Prozeduren angeboten, die die besten Parameter selbst ermitteln.

5.1. Interferenzen

Interferenzen erhält man, indem man mehrere Wellen nacheinander über ein Objekt laufen läßt (dank dem Superpositionsprinzip).
(siehe Tutorium 2)

5. Partikel-System (P20,O2P,O2S)

Das Programm erzeugt aus zwei beliebigen Objekten (s. später) Partikel-Objekte. Die Positionen, Drehwinkel und Skalierungsfaktoren der einzelnen Partikel können dabei jeweils als Binärdateien ausgegeben werden. Das bietet jedem Programmierer die Möglichkeit, eigene Partikel-Effekte selbst zu programmieren (lediglich die Positionen, Drehwinkel und Größen der Partikel müssen dabei berechnet werden); das Objekt-Handling (Erzeugen eines "echten" 3D-Objekts usw.) übernimmt Dust.

Als Effekte habe ich neben Partikel-Wellen und algorithmischen Veränderungen noch Partikel-Explosionen implementiert; den Quelltext dazu werde ich später auch noch etwas beschreiben (Tutorium 3).

Außerdem können als Partikel auch mathematische Kugeln verwendet werden, wobei das Programm die resultierenden Objekte als TDDD-Groups abspeichert.

5.1. Partikel-Wellen (PWAVE1D,PWAVE2D,PWAVE3D)

Hier werden die gleichen Prozeduren wie bei 3. verwendet, nur werden hier nicht die Objekt-Punkte, sondern die Partikel (also ganze Objekte) bewegt.

6. Algorithmische Veränderungen (*FUNC,*CFUNC,*P*FUNC)

Sie können die Punkte, die Flächenfarben und die Partikel algorithmisch verändern, wobei diese Effekte auch animiert und kombiniert werden können. Als Parameter können dabei verwendet werden:

- Ausgangswert (x0,y0,z0)
- t0 (Parameter bei Animationen).

Für jede Dimension kann/muß eine andere Funktion angegeben werden, damit ist wirklich alles machbar.

Wenn Sie z.B. die Funktion $\sin(x^2+y^2)$ darstellen möchten, so laden Sie eine Plane und geben ein: "func(2,30*sin(x0*x0/30+y0*y0/30)",0,z)".

7. Gravitation (PFALL,PFALL2)

Dust kann Objekte in sich zusammenfallen lassen. Folgende zwei Möglichkeiten gibt es:

- auf alle Punkte wirkt die gleiche Kraft
- auf jeden Punkt wirkt eine zum Abstand vom Nullpunkt des Fallprozesses proportionale Kraft.

Anwendungsbeispiel ist z.B. ein auf eine Ebene fallender Tropfen.

8. Realistische Wasserwellen (WATER, WATERFRAME, WATERZ, WATERZFRAME)

Hierbei werden Wellen, wie sie entstehen, wenn man einen Stein in einen See wirft, berechnet - dies sind dreidimensionale nicht-harmonische Wellen.

Die Parameter

- Amplitude
- Wellenlänge
- Quellpunkt
- Dämpfung
- Anzahl der Wellentäler
- Anzahl der Ausbreitungsvorgänge (=Maximalgröße der "Ringe")

Bei großen Amplituden führt die dreidimensionale Bewegung der Punkte zu Fehlern - zu diesem Zweck gibt es eine Prozedur (WATERZ), welche die Punkte nur in z-Richtung bewegt, was immer noch sehr gut aussieht.

9. Diverses

LATTICE erzeugt eine gitterähnliche Oberflächen-Struktur

ADDFACE erlaubt die Erstellung kompletter Objekte anhand mathematischer Formeln

RENAME benennt ganze Objektsequenzen um, dabei sind alle möglichen Richtungen erlaubt (so ist z.B. die Umkehrung der Objekt-Reihenfolge möglich)

STAGING2/STAGING3 erzeugt ISL2.0/ISL3.x-Staging-Dateien, sodaß nur das erste Objekt einer Sequenz von Hand eingeladen werden muß (in Imagine). (siehe Tutorium 1)

IWSTAGING erzeugt aus einer Lightwave-Szene neue Szene-Dateien und ein AREXX-Script, mit dessen Hilfe Lightwave alle Frames automatisch berechnet.

TXTDIR/BRSDIR ändert den Verzeichnisnamen aller Texturen/Brushes eines Objektes, sodaß diese mit den Texturen bequem irgendwo ausgelagert werden können

BUILD(RND) löscht nacheinander (zufallsgesteuert) Punkte und Flächen aus einem Objekt, bis es verschwindet.

DISTORT verschiebt zufallsgesteuert Punkte eines Objektes.

TRIANGULATE erzeugt zu jeder Fläche eigene Punkte und Kanten.

MERGE löscht überflüssige Punkte und Kanten schnell.

SCALEFACES skaliert Flächen bezüglich ihrer Mittelpunkte

...

1.35 Programm-Parameter ASPECT

Name: ASPECT
Wertebereich: 0.25..4.0
Beschreibung: Verhältnis von Bildschirmbreite zu -höhe
Beispiel: set (aspect,1.0) (Hires-Interlaced)

1.36 Programm-Parameter BWLEFT

Name: BWLEFT
Wertebereich: 0..2048
Beschreibung: x-Koordinate des Befehlsabbruch-Fensters (wird automatisch in die Konfiguration geschrieben)
Beispiel: set (bwleft,0)

1.37 Programm-Parameter BWTOP

Name: BWTOP
Wertebereich: 0..2048
Beschreibung: y-Koordinate des Befehlsabbruch-Fensters (wird automatisch in die Konfiguration geschrieben)
Beispiel: set (bwtop,16)

1.38 Programm-Parameter KEEPASPECT

Name: KEEPASPECT
Wertebereich: true/false
Beschreibung: Gibt an, ob nach einer Veränderung der Fenstergröße durch den Benutzer wieder das richtige Breiten-Höhen-Verhältnis eingestellt werden soll
Beispiel: set (keepaspect,t)

1.39 Programm-Parameter WINDOWSTACK

Name: WINDOWSTACK
Wertebereich: 12000..100000
Beschreibung: gibt die grÖÙe des Stack-Speichers der Zeichen-Prozesse an, sollte das Programm einmal bei sehr groÙen Objekten mit einem "Stack-Overflow" abbrechen, so muÙ dieser Wert hÖÙher gesetzt werden.
Beispiel: set (windowstack,18000)

1.40 Programm-Parameter WINDOWPRI

Name: WINDOWPRI
Wertebereich: -3..3
Beschreibung: Priorität der Zeichen-Prozesse an,
Beispiel: set(windowpri,1)

1.41 Programm-Parameter WARNINGS

Name: WARNINGS
Wertebereich: true/false
Beschreibung: gibt an, ob vor dem Überschreiben eines Objektes durch ein anders
gewarnt werden soll (default:false!)
Beispiel: set(warnings,t)

1.42 Programm-Parameter CHECKMOUSE

Name: CHECKMOUSE
Wertebereich: true/false
Beschreibung: Zeichenvorgang (Preview-Fenster) durch Drücken der linken
Maustaste abbrechen (CHECKMOUSE=TRUE)
Beispiel: set(checkmouse,t)

1.43 Programm-Parameter BREAKWIN

Name: BREAKWIN
Wertebereich: true/false
Beschreibung: Befehlsabbruch-Requester unterdruecken
Beispiel: set(breakwin,f)

1.44 Programm-Parameter RANDOM

Name: RANDOM
Wertebereich: 0..32677
Beschreibung: Jede Prozedur, die mit Zufallswerten arbeitet, initialisiert vor dem
Start den Zufallsgenerator mit diesem Wert. Das hat den Vorteil, daß
man bei gleichen Werten für RANDOM immer die gleichen Objekte/ Bewegungen
erhält
Beispiel: set(random,1234)

1.45 Programm-Parameter BACKFACES

Name: BACKFACES
 Wertebereich: true/false
 Beschreibung: gibt an,
 -ob beim Speichern von Objekten im Videoscape3D-Format
 die Flächen doppelt (Vorder- und Rückseite) erzeugt werden ↔
 sollen.
 Ansonsten sind alle Flächen in Videoscape nur von einer Seite
 aus sichtbar, was nur bei echten VS-Objekten zur korrekten
 Darstellung führt.
 -ob beim Speichern von Lightwave-Objekten das DOUBLESIDED-Flag
 gesetzt werden soll, die Bedeutung ist dieselbe wie bei
 Videoscape-Objekten, aber die Objekte werden hier nicht größer
 Beispiel: set(backfaces,t)

1.46 Programm-Parameter SAVESPHEREP

Name: SAVESPHEREP
 Wertebereich: true/false
 Beschreibung: gibt an, ob beim Speichern von Sphere-Objekten automatisch
 eine Particle-Object-Datei mit der Endung ".dpo" abgespeichert
 werden soll
 Beispiel: set(savespherep,t)

1.47 Programm-Parameter ALIGNP

Name: ALIGNP
 Wertebereich: true/false
 Beschreibung: gibt an, ob die Partikel entlang der Flächennormalen des ↔
 Strukturobjektes
 ausgerichtet werden sollen (beim Befehl O2P). Dies funktioniert
 im Face-Modus sehr gut, im Punkt-Modus überzeugen die Ergebnisse ↔
 nur
 bei glatten Objekten.
 Achtung: Im Punkt-Modus kann es auf langsameren Rechnern (28MHz) ↔
 bis zu
 einigen Minuten dauern.
 Beispiel: set(alignp,f)

1.48 Programm-Parameter EXFILE

Name: EXFILE
 Wertebereich: gültiger Dateiname, hier ist kein Leerstring erlaubt
 Beschreibung: Dies ist der Dateiname für externe Binärdateien, diese werden
 z.B. vom GETPPOS-Befehl erzeugt und sind nur wichtig für die
 Partikel-Effekt-Programmierung
 Beispiel: set(exfile,ram:DUSTOUT)

1.49 Programm-Parameter EXFORMAT

Name: EXFORMAT
Wertebereich: (FLOAT|LONG)
Beschreibung: Normalerweise schreibt Dust (reelle) Zahlen im Floating-Point-Format. Dieses ist aber bei den meisten Rechnern verschieden, so kann z.B. GCC nichts damit anfangen (vielleicht muß man auch nur eine der 1000 Compileroptionen setzen).
Da es schade wäre, GCC in Verbindung mit Dust nicht einsetzen zu können, biete ich hier ein zweites Format an: LONG.
Hierbei werden alle reellen Zahlen mit 65536.0 multipliziert und als LONGINT gespeichert.
Ein Beispiel für GCC (PExampleGCC.c) sollte sich irgendwo auf den Programmdisketten befinden.
Beispiel: set(exformat,long)

1.50 Programm-Parameter LEFT

Name: LEFT
Wertebereich: INTEGER
Beschreibung: linke Eck-Koordinate der Preview-Fenster
Beispiel: set(left,100)

1.51 Programm-Parameter TOP

Name: TOP
Wertebereich: INTEGER
Beschreibung: obere Eck-Koordinate der Preview-Fenster
Beispiel: set(top,11)

1.52 Programm-Parameter WIDTH

Name: WIDTH
Wertebereich: INTEGER
Beschreibung: Breite der Preview-Fenster
(die Höhe wird durch den Parameter ASPECT bestimmt)
Beispiel: set(width,200)

1.53 Programm-Parameter DRAWMODE

Name: DRAWMODE
Wertebereich: (WIRE,SOLID,COLOR,BBOX)
Beschreibung: Zeichenmodus der Preview-Fenster
Beispiel: set(drawmode,color)

1.54 Programm-Parameter ROTX

Name: ROTX
Wertebereich: real (in Grad)
Beschreibung: Rotation um die X-Achse des Views
Beispiel: set(rotx,12)

1.55 Programm-Parameter ROTZ

Name: ROTZ
Wertebereich: real (in Grad)
Beschreibung: Rotation um die Z-Achse des Views
Beispiel: set(rotz,-24)

1.56 Programm-Parameter ZOOM

Name: ZOOM
Wertebereich: real
Beschreibung: Zoom-Faktor des Views
Beispiel: set(zoom,0.7)

1.57 Befehl ABOUT

Befehl: ABOUT
Beschreibung: gibt Informationen zum Programm aus

1.58 Befehl ANIMCFUNC

Befehl: ANIMCFUNC(<objectID>,<frames>,<filename>,<tmin>,<tmax>,<R-expression>,<G-expression>,<B-expression>)
Beschreibung: Dies ist eine Funktion, die eine Folge von Objekten erzeugt, indem sie die Funktion CFUNC mit $t_0=t_{min}..t_{max}$ aufruft. Hierbei können alle Farbkomponenten gleichzeitig verändert werden ↔
.
Um eine Komponente unverändert zu belassen, muß der entsprechende Zahlenwert angegeben werden.
siehe auch CFUNC.
Beispiel: animcfunc(1,12,cobj,1.0,2.0,z0*t0,"(x0+y0)*t0",128\')

1.59 Befehl ANIMFUNC

Befehl: ANIMFUNC (<objectID>, <frames>, <filename>, <tmin>, <tmax>, <x-expression>, <y-expression>, <z-expression>)

Beschreibung: Dies ist eine Funktion, die eine Folge von Objekten erzeugt, indem sie die Funktion FUNC mit $t_0=t_{\min}..t_{\max}$ aufruft. Hierbei können alle Dimensionen des Objektes gleichzeitig verändert werden. Zu beachten ist, daß bei allen 3 Ausdrücken die Original-Positionen für (x_0, y_0, z_0) verwendet werden, d.h.: werden z.B. durch x-expression die x-Koordinaten aller Punkte verändert, so werden bei y-expression die alten verwendet usw. Um eine Dimension unverändert zu belassen, muß entweder "x0", "y0" oder "z0" bzw. "" als Ausdruck angegeben werden. siehe auch FUNC.

Bemerkung: Diese Funktion ermöglicht auch phantastische Metamorphosen

Beispiel: animfunc(1,12,obj,1.0,2.0,x0,y0,"30*sin(x0*x0/60*t0+y0*y0/60*t0)")

1.60 Befehl ANIMPPOSFUNC

Befehl: ANIMPPOSFUNC (<particleID>, <frames>, <filename>, <tmin>, <tmax>, <x-expression>, <y-expression>, <z-expression>, <Speicherformat>)

Beschreibung: Dies ist eine Funktion, die eine Folge von Objekten erzeugt, indem sie die Funktion PPOSFUNC mit $t_0=t_{\min}..t_{\max}$ aufruft. Hierbei können alle Dimensionen des Objektes gleichzeitig verändert werden. Zu beachten ist, daß bei allen 3 Ausdrücken die Original-Positionen für (x_0, y_0, z_0) verwendet werden, d.h.: werden z.B. durch x-expression die x-Koordinaten aller Punkte verändert, so werden bei y-expression die alten verwendet usw. Um eine Dimension unverändert zu belassen, muß entweder "x0", "y0" oder "z0" bzw. "" als Ausdruck angegeben werden. Als Speicherformat kann wie üblich "OBJ" oder "PARTICLE" angegeben werden. siehe auch PPOSFUNC, COPYPPOS, COPYPROT, COPYPSCL

Bemerkung: Diese Funktion ermöglicht auch phantastische Metamorphosen

Beispiel: animpposfunc(1,12,obj,1.0,2.0,x0,y0,"30*sin(x0*x0/60*t0+y0*y0/60*t0)",obj)

1.61 Befehl ANIMPROTFUNC

Befehl: ANIMPROTFUNC (<particleID>, <frames>, <filename>, <tmin>, <tmax>, <x-expression>, <y-expression>, <z-expression>, <Speicherformat>)

Beschreibung: Dies ist eine Funktion, die eine Folge von Objekten erzeugt, indem sie die Funktion PROTFUNC mit $t_0=t_{\min}..t_{\max}$ aufruft. Hierbei können alle Dimensionen des Objektes gleichzeitig verändert werden. Zu beachten ist, daß bei allen 3 Ausdrücken die Original-Positionen für (x_0, y_0, z_0) verwendet werden, d.h.: werden z.B. durch x-expression die x-Rotationswinkel aller Punkte verändert, so werden bei y-expression die alten verwendet usw. Um eine Dimension unverändert zu belassen, muß entweder

"x0", "y0" oder "z0" bzw. "" als Ausdruck angegeben werden.
 Als Speicherformat kann wie üblich "OBJ" oder "PARTICLE" angegeben ↔
 werden.

siehe auch PROTFUNC, COPYPPOS, COPYPROT, COPYPSCL

Beispiel: animprotfunc(1,24,obj,0.0,1.0,x0,y0,360*t0)",obj)

1.62 Befehl ANIMPSCLFUNC

Befehl: ANIMPSCLFUNC(<particleID>,<frames>,<filename>,<tmin>,<tmax>,
 <x-expression>,<y-expression>,<z-expression>,<Speicherformat>)

Beschreibung: Dies ist eine Funktion, die eine Folge von Objekten erzeugt,
 indem sie die Funktion PSCLFUNC mit $t_0=t_{\min}..t_{\max}$ aufruft.
 Hierbei können alle Dimensionen des Objektes gleichzeitig
 verändert werden. Zu beachten ist, daß bei allen 3 Ausdrücken
 die Original-Positionen für (x0,y0,z0) verwendet werden,
 d.h.: werden z.B. durch x-expression die x-Skalierungsfaktoren
 aller Punkte verändert, so werden bei y-expression die alten
 verwendet usw.

Um eine Dimension unverändert zu belassen, muß entweder
 "x0", "y0" oder "z0" bzw. "" als Ausdruck angegeben werden.

Als Speicherformat kann wie üblich "OBJ" oder "PARTICLE" angegeben ↔
 werden.

siehe auch PSCLFUNC, COPYPPOS, COPYPROT, COPYPSCL

Bemerkung: Diese Funktion ermöglicht auch phantastische Metamorphosen

Beispiel: animpsclfunc(2,36,pobj,-1,1,t0*2,t0*2,t0*2)",particle)

1.63 Befehl AVAIL

Befehl: AVAIL

Beschreibung: gibt den freien Arbeitsspeicher aus

1.64 Befehl AXALIGN0

Befehl: AXALIGN0(<objectID>)

Beschreibung: Setzen der Ausrichtung des Objektkoordinatensystems
 auf (0,0,0) (alle Drehwinkel 0), sehr
 nützlich, um Objekt-Sequenzen nachzubearbeiten

Beispiel: axalign0(4)

1.65 Befehl AXPOS

Befehl: AXPOS(<objectID>,<XPos>,<YPos>,<ZPos>)

Beschreibung: Ändern der Position des Objektkoordinatensystems, sehr
 nützlich, um Objekt-Sequenzen nachzubearbeiten

Beispiel: axpos(3,0.0,10.0,20.0)

1.66 Befehl AXSIZE

Befehl: AXSIZE(<objectID>,<Xsize>,<YSize>,<Zsize>)

Beschreibung: Ändern der Größe des Objektkoordinatensystems, sehr nützlich, um Objekt-Sequenzen nachzubearbeiten

Beispiel: axsize(3,32.0,50.0,32.0)

1.67 Befehl BUILD

Befehl: BUILD(<objectID>,<frames>,<filename>)

Beschreibung: Löscht nacheinander Flächen des Objektes mit der Nummer objektID, erzeugt dabei (maximal) frames Einzelobjekte mit den Dateinamen filename.001,filename.002,...

Beispiel: build(4,60,ram:obj)

1.68 Befehl BUILDRND

Befehl: BUILDRND(<objectID>,<frames>,<filename>)

Beschreibung: Löscht zufallsgesteuert Flächen des Objektes mit der Nummer objektID, erzeugt dabei (maximal) frames Einzelobjekte mit den Dateinamen filename.001,filename.002,...

Beispiel: buildrnd(4,60,ram:obj)

1.69 Befehl CD

Befehl: CD <filename>

Beschreibung: Wechseln des aktuellen Verzeichnisses

Beispiel: cd ram:

1.70 Befehl CENTERAXIS

Befehl: CENTERAXIS(<objectID>)

Beschreibung: Fixieren des Ursprungs des Objektkoordinatensystems in den Objektmittelpunkt

Beispiel: centeraxis(4)

1.71 Befehl CFUNC

Befehl: CFUNC(<objectID>,<expression>,<parameter t0>,<"R", "G" oder "B">)

Beschreibung: Algorithmische Veränderung von Flächenfarben, hier geben die Werte (x0,y0,z0) den Flächenmittelpunkt der betrachteten Fläche an.

Die berechneten Werte müssen zwischen 0 und 255 liegen, ←
andererseits

werden sie entsprechend verändert.
siehe auch FUNC
Beispiel: `cfunc(4, "z0*10", 2, r)`

1.72 Befehl CLOSEWINDOWS

Befehl: `CLOSEWINDOWS([<objectID>])`
Beschreibung: Schließen aller Preview-Fenster eines Objektes oder aller Fenster
Bemerkung: In manchen Fällen (ein anderes Programm mit größerer Priorität als die Preview-Tasks läuft im Hintergrund) ist dieser Befehl zunächst wirkungslos, dies ist kein Fehler o.ä.
Beispiele: `closewindows(5)`
`closewindows`

1.73 Befehl COLOR

Befehl: `COLOR(<objectID>, <red>, <green>, <blue>)`
Beschreibung: Ändern der Objektfarbe
Beispiel: `color(8, 128, 128, 0)`

1.74 Befehl COPY

Befehl: `COPY(<srcID>, <destID>)`
Beschreibung: Kopieren des Objektes mit der Nummer `srcID` an den Speicherplatz `destID`
Beispiel: `copy(1, 2)`

1.75 Befehl COPYATTS

Befehl: `COPYATTS(<srcID>, <destID>)`
Beschreibung: Kopieren von Objektattributen, dies ist eine sehr nützliche Funktion, will man die Attribute einer ganzen Objekt-Sequenz verändern
Beispiel: `copyatts(1, 2)`

1.76 Befehl COPYP

Befehl: `COPYP(<srcID>, <destID>)`
Beschreibung: Kopieren des Partikel-Objektes `srcID` nach `destID`
Beispiel: `copyp(1, 2)`

1.77 Befehl COPYPPOS

Befehl: COPYPPOS(<srcID>,<destID>)

Beschreibung: Kopieren der Partikel-Positionen von einem Partikel-Objekt zum anderen, dabei müssen die Partikelanzahlen übereinstimmen. Diese Funktion ist sehr nützlich, will man die Funktionen ANIMPPOSFUNC, ANIMPROTFUNC und ANIMPSCLFUNC miteinander kombinieren.

Beispiel: copyppos(1,2)

1.78 Befehl COPYPROT

Befehl: COPYPROT(<srcID>,<destID>)

Beschreibung: Kopieren der Partikel-Rotationswinkel von einem Partikel-Objekt ↔ zum

anderen, dabei müssen die Partikelanzahlen übereinstimmen. Diese Funktion ist sehr nützlich, will man die Funktionen ANIMPPOSFUNC, ANIMPROTFUNC und ANIMPSCLFUNC miteinander kombinieren.

Beispiel: copyprot(2,4)

1.79 Befehl COPYPSCL

Befehl: COPYPSCL(<srcID>,<destID>)

Beschreibung: Kopieren der Partikel-Skalierungsfaktoren von einem Partikel- ↔ Objekt zum

anderen, dabei müssen die Partikelanzahlen übereinstimmen. Diese Funktion ist sehr nützlich, will man die Funktionen ANIMPPOSFUNC, ANIMPROTFUNC und ANIMPSCLFUNC miteinander kombinieren.

Beispiel: copypscl(3,1)

1.80 Befehl DISTORT

Befehl: DISTORT(<objectID>[,<percent of objSize>])

Beschreibung: Verschiebt zufallsgesteuert die Punkte eines Objektes, dabei kann die Größe des maximalen Verschiebungsvektors als Prozentsatz von der maximalen Ausdehnung des Objektes angegeben werden

Beispiele: distort(5,20.0)
distort(2)

1.81 Befehl DITHER

Befehl: DITHER(<objectID>,<8-bit-value>)

Beschreibung: Ändern der Oberflächenfarbmischintensität

Beispiel: dither(4,255)

1.82 Befehl EXEC

Befehl: EXEC([<filename>])
 Beschreibung: Ausführen eines Batch-Files
 Beispiele: exec(ram:batch)
 exec()

1.83 Befehl EXPLODE

Befehl: EXPLODE(<srcID>,<destID>,<frames>,<filename>,
 <Zeitdauer>,<Fallbeschleunigung>,<Stokes'scher ↔
 Reibungskoeffizient>,
 <Anfangsgeschwindigkeit>,
 <maximale Anzahl von Rotationen des größten Teilchens>)
 Beschreibung: Erzeugen einer Explosion
 Beispiele: explode(1,180,hdl:objects/obj,12,-10.0,-0.0001,42,9)
 explode(1,48,obj,12.8,-10.0,-0.001,52,7)

1.84 Befehl EXPLODEFRAME

Befehl: EXPLODEFRAME(<srcID>,<frames>,<frame>,<dest2>,
 <Zeitdauer>,<Fallbeschleunigung>,<Stokes'scher ↔
 Reibungskoeffizient>,
 <Anfangsgeschwindigkeit>,
 <maximale Anzahl von Rotationen des größten Teilchens>)
 Beschreibung: Erzeugen des Einzelobjektes frame einer Explosion
 Beispiele: explodeframe(1,180,63,2,12,-10.0,-0.0001,42,9)
 explodeframe(1,48,24,3,12.8,-10.0,-0.001,52,7)

1.85 Befehl ECHO

Befehl: ECHO(<string>[,<Ausdruck>])
 Beschreibung: Ausgeben einer Zeichenkette, oder einer Zeichenkette und
 einen Zahlenwert, nützlich in Batch-Files
 Beispiele: echo(hello world)
 echo(a=,a)

1.86 Befehl FUNC

Befehl: FUNC(<objectID>,<expression>,<parameter t0>,<"X", "Y" oder "Z">)
 Beschreibung: Algorithmisches Verändern der Punktpositionen, expression ist
 dabei ein mathematische Funktion in x_0, y_0, z_0 und t_0 .
 Die Werte (x_0, y_0, z_0) sind die Position des betrachteten Punktes
 vor der Modifikation. (t_0 ist hier ein willkürlicher Parameter,
 der aber bei der Animation (z.B. ANIMFUNC) benötigt wird.)
 Der String "X", "Y" oder "Z" gibt die Dimension an, die ↔
 modifiziert

werden soll.

Beispiel: `func(2, "30*sin(x0+x0/30+y0*y0/30)", 0, z)`

Dieser Aufruf erzeugt ein Modell der Funktion $\sin(x^2+y^2)$, wenn als Ausgangsobjekt eine Plane verwendet wird; man kann Dust also auch als Funktionsplotter verwenden.

ACHTUNG: Enthält expression Kommata oder Klammern, so müssen ←
Anführungszeichen
angegeben werden.

1.87 Befehl GET

Befehl: `GET([<parameter>])`

Beschreibung: Anzeigen der aktuellen Programmparameter

Beispiel: `get(random)`

1.88 Befehl GETOCOUNT

Befehl: `GETOCOUNT(<particleID>)`

Beschreibung: Schreiben der Partikel-Anzahl eines Partikel-Objektes in die Datei EXFILE,
wichtig für externe Programmierung von Partikel-Effekten

Beispiel: `getocount(3)`

Dateiformat: die Zahl wird als INTEGER (int) ausgegeben, die Dateigröße ist also 2 bytes

1.89 Befehl GETPSIZE

Befehl: `GETPSIZE(<particleID>)`

Beschreibung: Schreiben der Ausdehnung (3 Komponenten) des Shape-Objekts eines Partikel-Objektes in die Datei Exfile, die tatsächliche Größe jedes Partikels bestimmt sich daraus durch Multiplikation mit den PSCL-Werten,

wichtig für externe Programmierung von Partikel-Effekten

z.B. hat eine Kugel mit dem Radius 50 die Größe 100 in allen

3 Richtungen, das Partikel mit dem PSCL-Wert von (0.5,1,-2)

hat dann die Ausdehnung (50,100,200),

Beispiel: `getpsize(4)`

Dateiformat: es werden die drei Koordinaten als REAL-Zahlen (float) ausgegeben, die Dateigröße ist also 12 bytes

1.90 Befehl GETPPOS

Befehl: `GETPPOS(<particleID>)`

Beschreibung: Schreiben der Positionen aller Partikel eines Partikel-Objektes in die Datei EXFILE,

wichtig für externe Programmierung von Partikel-Effekten

Beispiel: `getppos(1)`
Dateiformat: es werden jeweils die drei Koordinaten als REAL-Zahlen (float) ausgegeben,
die Dateigröße ist also $12 * (\text{Partikelanzahl})$ bytes;

1.91 Befehl GETPROT

Befehl: `GETPROT(<particleID>)`
Beschreibung: Schreiben der Rotationswinkel aller Partikel eines Partikel-Objektes in die Datei EXFILE,
wichtig für externe Programmierung von Partikel-Effekten
Beispiel: `getprot(3)`

1.92 Befehl GETPSCL

Befehl: `GETPSCL(<particleID>)`
Beschreibung: Schreiben der Skalierungsfaktoren aller Partikel eines Partikel-Objektes in die Datei EXFILE,
wichtig für externe Programmierung von Partikel-Effekten
Beispiel: `getpscl(3)`
Dateiformat: wie bei GETPPOS

1.93 Befehl HARDNESS

Befehl: `HARDNESS(<objectID>, <8-bit-value>)`
Beschreibung: Ändern der Oberflächenhärte
Beispiel: `hardness(3,128)`

1.94 Befehl JOIN

Befehl: `JOIN(<src1ID>, <src2ID>, <destID>)`
Beschreibung: Verbinden der Objekte `src1ID` und `src2ID` als Objekt `destID`
Beispiel: `join(1,2,3)`

1.95 Befehl JOINP

Befehl: `JOINP(<src1ID>, <src2ID>, <destID>)`
Beschreibung: Verbinden der Partikel-Objekte `src1ID` und `src2ID` als Objekt `destID` ↔
Achtung: Dabei wird das Shape-Objekt vom ersten Partikel-Objekt übernommen
Beispiel: `joinp(1,2,3)`

1.96 Befehl KILL

Befehl: KILL([<objectID>])

Beschreibung: Löschen eines oder aller im Speicher befindlichen Objekte

Beispiele: kill(12)
kill (löscht alle Objekte!)

1.97 Befehl KILLFREEPOINTS

Befehl: KILLFREEPOINTS(<objectID>)

Beschreibung: Löschen von unbenutzten Punkten

Beispiel: killfreepoints(16)

1.98 Befehl KILLEDGE

Befehl: KILLEDGE(<objectID>,<edgeID oder -1>)

Beschreibung: Löschen einer Kante, bei -1 wird die Kante durch den Zufalls-
generator bestimmt

Beispiele: killedge(4,24)
killedge(4,-1) (zufällig)

1.99 Befehl KILLFACE

Befehl: KILLFACE(<objectID>,<faceID>)

Beschreibung: Löschen einer Fläche, bei -1 wird die Fläche durch den Zufalls-
generator bestimmt

Beispiele: killface(5,25)
killface(5,-1) (zufällig)

1.100 Befehl KILLP

Befehl: KILLP([<particleID>])

Beschreibung: Löschen eines spezifizierten oder aller Partikel-Objekte

Beispiele: killp(1)
killp

1.101 Befehl KILLPOINT

Befehl: KILLPOINT(<objectID>,<pointID>)

Beschreibung: Löschen eines Punktes, bei -1 wird der Punkt durch den Zufalls-
generator bestimmt

Beispiele: killpoint(3,23)
killpoint(3,-1) (zufällig)

1.102 Befehl LOAD

Befehl: LOAD (<objectID>[, <filename>])

Beschreibung: Laden eines Objektes an die Speicherstelle objectID (TDDD-Format)

Beispiele: load(1,hd1:obj)
load(4)
load(3,)

1.103 Befehl MERGE

Befehl: MERGE (<objectID>)

Beschreibung: Löschen der überflüssigen Punkte und Kanten

Beispiel: merge(3)

1.104 Befehl MORPH

Befehl: MORPH (<srcID>, <destID>, <frames>, <filename>)

Beschreibung: Durchführen einer Dreiecksmetamorphose zwischen den Objekten srcID und destID, dabei wird die Prozedur PMORPH automatisch aufgerufen,

die Objekte srcID und destID werden dabei auch verändert

Beispiele: morph(4,1,60,ram:obj)
morph(1,2,33,)

1.105 Befehl MORPHFRAME

Befehl: MORPHFRAME (<srcID>, <destID>, <frames>, <frame>, <dest2ID>)

Beschreibung: Erzeugen eines einzelnen Morph-Objekts und kopieren dieses Objektes an den Speicherplatz dest2ID, auch hier wird die Prozedur PMORPH automatisch ausgeführt, die Objekte srcID und destID werden dabei auch verändert

Beispiel: morphframe(1,2,60,23,4)

1.106 Befehl MORPHATTS

Befehl: MORPHATTS (<firstObj>, <lastObj>, <basefilename>)

Beschreibung: Interpolieren der Attribute einer Objekt-Sequenz, die in der Form von Objekten auf der Festplatte vorliegt

Beispiel: morphatts(1,12,obj)

1.107 Befehl DEFORMMORPH

Befehl: DEFORMMORPH(<srcID>,<destID>,<minimum of frames>,<filename>)
 Beschreibung: Durchführen einer Deformations-Metamorphose, für gute Ergebnisse ist eine gewisse Ähnlichkeit beider Objekte Voraussetzung, die tatsächliche Anzahl erzeugter Objekte wird in der Variablen "result" gespeichert.
 Beispiele: deformmorph(2,1,60,obj)
 deformmorph(1,2,30,)

1.108 Befehl O2P

Befehl: O2P(<structureID>,<shapeID>,<particleID>,<"P" oder "F">)
 Beschreibung: Konvertieren zweier Objekte in ein Partikel-Objekt particleID. Das Objekt structureID bestimmt dabei die Größe und die Positionen der einzelnen Partikel, die gleich dem Objekt shapeID sind. Dust bietet zur Berechnung der Position und Größe der Partikel zwei Methoden an:
 -FACE: in jeden Flächenmittelpunkt des Struktur-Objektes wird ein Shape-Objekt gesetzt, die Größe wird dabei unter Verwendung des Flächeninhaltes bestimmt
 -POINT: jeder Punkt des Struktur-Objektes "bekommt" ein Shape-Objekt, dessen Größe aus dem Vergleich der Objekt-Volumina bestimmt wird
 Beispiele: o2p(1,2,1,p)
 o2p(3,4,1,f)

1.109 Befehl P2O

Befehl: P2O(<particleID>,<objectID>)
 Beschreibung: Konvertieren des Partikel-Objektes particleID in ein normales Objekt objectID
 Beispiele: p2o(3,4)

1.110 Befehl PEXPLODE

Befehl: PEXPLODE(<particleID>,<frames>,<Dateiname>,<Zeitdauer>,<Gravitationskonstante>,<Zähigkeit des Mediums (Stokes'sche Reibung)>,<Anfangsgeschwindigkeit der Teilchen>,<maximale Rotationsanzahl des größten Teilchens>,<Speicherformat (TDDD|VS|PARTICLE)>
 Beschreibung: Eine Partikel-Explosion
 Beispiele: pexplode(1,180,hdl:objects/obj,12,-10.0,-0.0001,42,9,obj)
 pexplode(1,48,obj,12.8,-10.0,-0.001,52,7,particle)

1.111 Befehl PPOSFUNC

Befehl: PPOSFUNC (<particleID>, <expression>, <parameter t0>, <"X", "Y" oder
"Z">) ←

Beschreibung: Algorithmische Veränderung von Partikel-Positionen,
siehe auch FUNC.

Beispiel: pposfunc (3, "t0*sin(x0)", 2, z)

1.112 Befehl PROTFUNC

Befehl: PROTFUNC (<particleID>, <expression>, <parameter t0>, <"X", "Y" oder
"Z">) ←

Beschreibung: Algorithmische Veränderung von Partikel-Rotationswinkel,
siehe auch FUNC

Beispiel: protfunc (1, "cos(t0*(x0+y0))", 0, x)

1.113 Befehl PSCLFUNC

Befehl: PSCLFUNC (<particleID>, <expression>, <parameter t0>, <"X", "Y" oder
"Z">) ←

Beschreibung: Algorithmische Veränderung von Partikel-Skalierungsfaktoren,
siehe auch FUNC

Beispiel: psclfunc (1, "sin(x0*z0)", 0, y)

1.114 Befehl PSTATS

Befehl: PSTATS ([<particle>])

Beschreibung: Informationen über ein bestimmtes oder alle Partikel-Objekte
ausgeben ←

Beispiele: pstats
pstats(3)

1.115 Befehl PSTATS2

Befehl: PSTATS2

Beschreibung: Informationen über belegte Partikel-Objekt-Speicherplätze
auf einer Bildschirmseite ausgeben (belegte durch "***" ←
gekennzeichnet)

1.116 Befehl PFALL

Befehl: PFALL (<objectID>, <frames>, <filename>)

Beschreibung: Wirkung von Gravitation auf alle Punkte eines Objektes in
z Richtung, die Bewegung endet bei dem Punkt mit der kleinsten
z-Koordinate, auf alle Punkte wirkt die gleiche Beschleunigung

Beispiel: pfall(4, 64,)

1.117 Befehl PFALL2

Befehl: PFALL2

Beschreibung: Wirkung von Gravitation auf alle Punkte eines Objektes in z Richtung, die Bewegung endet bei dem Punkt mit der kleinsten z-Koordinate, auf alle Punkte wirkt eine andere Beschleunigung

Beispiel: pfall2(3,30,hdl:object)

1.118 Befehl POSITIVE

Befehl: POSITIVE(<objectID>,<"X" oder "Y" oder "Z">)

Beschreibung: Bewegen eines Objektes in den positiven Halbraum

Beispiel: positive(1,y)

1.119 Befehl PWAVE1D

Befehl: PWAVE1D(<objectID>,<frames>,<filename>,<"T" oder "L">,<Datei-Format>,<

[<amplitude>,<wavelength>,<source>,<damping>,<phase in Grad>])

Beschreibung: Erzeugt eine eindimensionale ebene Transversal- oder Longitudinal-Partikel-Welle, als Datei-Format kann angegeben werden:

"OBJ" (Objekt im SFORMAT)

"PARTICLE" (Particle)

(siehe auch WAVE1D)

Beispiele: pwaveld(1,40,obj,t,obj,12.0,24,0,1.0,0.0)

pwaveld(1,33,ram:ob,t,particle)

pwaveld(1,33,ram:ob,l,obj)

1.120 Befehl PWAVE1DFRAME

Befehl: PWAVE1DFRAME(<particleID>,<frames>,<frame>,<destID>,<"T" oder "L">,<

[<amplitude>,<Wellenlänge>,<Quelle>,<Dämpfung>,<Phase in Grad>])

Beschreibung: Erzeugen eines Partikel-Objekt-Wellen-Einzelobjekts (siehe PWAVE1D)

Beispiele: pwaveldframe(1,40,33,2,t,12.0,24,0,1.0,0.0)

pwaveldframe(1,33,12,5,t)

pwaveldframe(1,33,12,6,l)

1.121 Befehl PWAVE2D

Befehl: PWAVE2D(<objectID>,<frames>,<filename>,<"T" oder "L">,<Datei-Format>,<amplitude>,<wavelength>,<sourceX>,<sourceY>,<damping>,<phase in Grad>])

Beschreibung: Erzeugen einer ebenen Transversal- oder Longitudinal-Partikel-Welle

in der x-y-Ebene.

(siehe auch PWAVE1D und WAVE2D)

Beispiele: pwave2d(1,40,obj,l,obj,12.0,24,10,-10,1.6,60.0)
 pwave2d(1,60,,l,obj)
 pwave2d(1,60,ram:obj,t,particle)

1.122 Befehl PWAVE2DFRAME

Befehl: PWAVE2DFRAME(<particleID>,<frames>,<frame>,<destID>,<"T" oder "L">,<amplitude>,<Wellenlänge>,<QuelleY>,<QuelleY>,<Dämpfung>,<Phase in Grad>])

Beschreibung: Erzeugen eines Partikel-Objekt-Wellen-Einzelobjekts (siehe PWAVE2D)

Beispiele: pwave2dframe(1,40,23,4,l,12.0,24,10,-10,1.6,60.0)
 pwave2dframe(1,60,30,2,l)
 pwave2dframe(1,60,24,3,t)

1.123 Befehl PWAVE3D

Befehl: PWAVE3D(<objectID>,<frames>,<filename>,<"S" or "F">,<Datei-Format>,<amplitude>,<wavelength>,<sourceX>,<sourceY>,<sourceZ>,<damping>,<phase in Grad>])

Beschreibung: Erzeugen einer dreidimensionalen Partikel-Welle (siehe auch PWAVE1D und WAVE3D)

Beispiele: pwave3d(1,40,obj,s,obj,14.0,32.0,10,-10,12,1.2,-30.0)
 pwave3d(1,60,hdl:objects,f,tddd)

1.124 Befehl PWAVE3DFRAME

Befehl: PWAVE3DFRAME(<particleID>,<frames>,<frame>,<destID>,<"S" or "F">,<amplitude>,<Wellenlänge>,<QuelleY>,<QuelleY>,<QuelleZ>,<Dämpfung>,<Phase in Grad>])

Beschreibung: Erzeugen eines Partikel-Objekt-Wellen-Einzelobjekts (siehe PWAVE1D)

Beispiele: pwave3dframeframe(1,40,24,2,s,14.0,32.0,10,-10,12,1.2,-30.0)
 pwave3dframe(1,60,33,4,f)

1.125 Befehl RANDOMPPOS

Befehl: RANDOMPPOS (<particleID>, <amount>)
Beschreibung: Verschieben der Partikel etwa (zufallsgesteuert) um den Wert +-amount
Beispiel: randomppos (1,20.0)

1.126 Befehl RANDOMPROT

Befehl: RANDOMPROT (<particleID>, <amount in Grad>)
Beschreibung: Rotieren der Partikel etwa (zufallsgesteuert) um den Winkel +-amount
Beispiel: randomprot (2,30.0)

1.127 Befehl RANDOMPSCL

Befehl: RANDOMPSCL (<particleID>, <amount>)
Beschreibung: Skalieren der Partikel etwa (zufallsgesteuert) um den Wert amount
Beispiel: randompscl (1,0.5)

1.128 Befehl REFL

Befehl: REFL (<objectID>, <red>, <green>, <blue>)
Beschreibung: Ändern des Reflektionsvermögens
Beispiel: refl (3,68,68,75)

1.129 Befehl RENAME

Befehl: RENAME (<basename1>, <from1>, <to1>, <basename2>, <from2>, <to2>)
Beschreibung: Umbenennen von Objekt-Sequenzen, dies bietet auch eine sehr einfache Möglichkeit, um Objekt-Sequenzen zu spiegeln
Beispiele: rename (obj,1,20,obj2,10,30)
 rename (obj,20,1,obj2,10,30)
 rename (obj,1,20,obj2,30,10)
 rename (obj,20,1,obj2,30,10)
Anwendung: Die Objekt-Sequenz "hdl:obj.001", "hdl:obj.002", ..., "hdl:obj.020" soll in umgekehrter Reihenfolge benutzt werden:
 rename (hdl:obj,1,20,hdl:obj2,20,1)
 rename (hdl:obj2,1,20,hdl:obj,1,20);

1.130 Befehl REQUEST

Befehl: REQUEST(<text>,<positive>,<negative>)
 Beschreibung: Öffnen eines Requesters mit der Mitteilung text, wird der Button mit der Meldung negative gedrückt, so wird das Programm beendet. Der Text kann sich dabei über mehrere Zeilen erstrecken, als Zeilenumbruch dient das Symbol "\n". Enthält der Text Klammern oder Kommata, so müssen ←
 Anführungszeichen angegeben werden.
 Bemerkung: Es scheint so, daß mehrere Textzeilen nur dann möglich sind, wenn das Programm "RTPatch" im Hintergrund läuft.
 Beispiele: request(Ready to format your harddisk ?,yes,oh no)
 request("More\nthan\nnone\nline\nof\ntext.\n(Nice, isn't it)","Yo, ←
 No)

1.131 Befehl ROTATE

Befehl: ROTATE(<objectID>,<Winkel in Grad>,<"X" oder "Y" oder "Z">)
 Beschreibung: Rotieren der Objektpunkte um den Ursprung des Objektkoordinatensystems um eine ausgewählte Achse
 Beispiel: rotate(4,60,z)

1.132 Befehl ROUGHNESS

Befehl: ROUGHNESS(<objectID>,<8-bit-value>)
 Beschreibung: Ändern der Oberflächenrauigkeit
 Beispiel: roughness(2,12)

1.133 Befehl SAVE

Befehl: SAVE(<objectID>,<filename>)
 Beschreibung: Speichern des Objektes objektID im format, das der auf das der Parameter SFORMAT gesetzt ist
 Beispiele: save(1,hdl:obj)
 save(3,)

1.134 Befehl SAVECONFIG

Befehl: SAVECONFIG([<filename>])
 Beschreibung: Speichern aller Einstellungen (default: "S:.dustrc")
 Beispiele: saveconfig
 saveconfig(t:tmp.cfg)

1.135 Befehl LOADCONFIG

Befehl: LOADCONFIG([<filename>])
Beschreibung: Laden von Voreinstellungen (default: "S:.dustrc")
Beispiele: loadconfig
 loadconfig(t:tmp.cfg)

1.136 Befehl SAVEP

Befehl: SAVEP(<particleID>,<filename>)
Beschreibung: Ein Partikel-Objekt im DUST-Format speichern
Beispiele: savep(1,hdl:particle1)
 savep(3,)

1.137 Befehl SAVEPOBJ

Befehl: SAVEPOBJ(<particleID>,<filename>)
Beschreibung: Ein Partikel-Objekt als Objekt abspeichern,
das Format wird durch den globalen Parameter SFORMAT bestimmt
Achtung: Dieses Objekt kann NICHT wieder als PARTIKEL-Objekt
 geladen werden.
Beispiel: saveptobj(1,hdl:particle1.tddd)
 saveptobj(3,)

1.138 Befehl SCALE

Befehl: SCALE(<objectID>,<factor>,"X" oder "Y" oder "Z")
Beschreibung: Skalieren der Objektpunkte bezüglich des Ursprungs des
Objektkoordinatensystems entlang einer ausgewählten Richtung
Beispiel: scale(1,1.8,x)

1.139 Befehl SCALEFACES

Befehl: SCALEFACES(<objectID>,<frames>,<filename>,<factor>)
Beschreibung: Skalieren der Flächen eines Objektes bezüglich der
Flächenmittelpunkte, hiemit kann man auch Objekte verschwinden
lassen
Beispiele: scalefaces(1,54,ram:obj,0.5)
 scalefaces(1,54,ram:obj2,2.0)

1.140 Befehl SCALEP

Befehl: SCALEP(<particleID>,<amount>)
Beschreibung: Skalieren der Partikel eines Partikel-Objekts
Beispiele: scalep(1,0.5)

1.141 Befehl SET

Befehl: SET(<program-parameter>, <value>)
Beschreibung: Verändern der Programmparameter
Beispiel: set(aspect, 1.0)
 set(vdrawmode, solid)
 set(keepaspect, t)

1.142 Befehl SETCLST

Befehl: SETCLST(<objectID>, <faceID>, <red>, <green>, <blue>)
Beschreibung: Ändern der Farbe der Fläche faceID eines Objects
Beispiel: setclst(8, 53, 128, 128, 0)

1.143 Befehl SETPPOS

Befehl: SETPPOS(<particleID>, <particle>, <x>, <y>, <z>)
Beschreibung: Setzen der Position eines bestimmten Partikels,
wichtig für externe Programmierung von Partikel-Effekten
Beispiel: setppos(1, 3, 34.0, -2.0, 7.77)

1.144 Befehl SETPOINT

Befehl: SETPOINT(<objectID>, <pointID>, <x>, <y>, <z>)
Beschreibung: Setzen der Position eines bestimmten Punktes,
wichtig für externe Programmierung, das Programm "Crystal"
macht z.B. Gebrauch davon.
Beispiel: setpoint(2, 0, 34.0, -2.0, 7.77)

1.145 Befehl SETPROT

Befehl: SETPROT(<particleID>, <particle>, <x>, <y>, <z>)
Beschreibung: Setzen der Rotationswinkel (in Grad) eines bestimmten Partikels,
wichtig für externe Programmierung von Partikel-Effekten
Beispiel: setprot(2, 3, 0.0, 45.0, -90.0)

1.146 Befehl SETPSCL

Befehl: SETPSCL(<particleID>, <particle>, <x>, <y>, <z>)
Beschreibung: Setzen der Skalierungsfaktoren eines bestimmten Partikels,
wichtig für externe Programmierung von Partikel-Effekten
Beispiel: setpscl(2, 3, 2.0, 2.0, 2.0)
Dateiformat: wie bei GETPPOS

1.147 Befehl SHININESS

Befehl: SHININESS (<objectID>, <8-bit-value>)
 Beschreibung: Ändern des Oberflächenglanzes
 Beispiel: shininess (2,22)

1.148 Befehl SIZE

Befehl: SIZE (<objectID>)
 Beschreibung: Ermitteln der minimalen Bounding-Box
 Beispiel: size (8)

1.149 Befehl SPEC

Befehl: SPEC (<objectID>, <red>, <green>, <blue>)
 Beschreibung: Ändern der Intensität des Lichtquellen-Spiegelungseffektes
 Beispiel: spec (6,255,255,255)

1.150 Befehl STAGING2

Befehl: STAGING2 (<basename>, <fromObject>, <toObject>, <fromFrame>, <toFrame <->
 >, <filename>)
 Beschreibung: Dieser Befehl erzeugt Auschnitte aus ISL2.0-Staging-Dateien im
 ASCII-Format (für Imagine2.0)
 Beispiele: staging2 (ram:obj1,1,20,33,43,ram:st1)
 staging2 (ram:obj2,30,1,44,74,ram:st2)
 Anwendung: Die Objekte "ram:obj.001" bis "ram:obj.049" sollen von Frame 1
 bis Frame 49, die Objekte "ram:ob2.001" bis "ram:ob2.029" sollen
 von Frame 50 bis Frame 78 in umgekehrter Reihenfolge erscheinen.

1. Schritt:

Erzeugen des Staging-Files für das erste Objekt der Sequenz
 in Imagine (Kamera einstellen, Lichtquelle setzen,...)

2. Schritt:

Umwandeln des Staging-Files mittels dem Programm "destage"
 in das ASCII-Format

3. Schritt:

```
staging2 (ram:obj,1,49,1,49,ram:st.1)
staging2 (ram:obj,29,1,50,78,ram:st.2)
!join ram:st.1 ram:st.2 as ram:st
```

4. Schritt:

Editieren des von "destage" erzeugten Files:
 Einfügen des Files "ram:st" an der Stelle im Text,
 wo die Informationen über das erste Objekt der
 Sequenz stehen, anschließend diese Zeile löschen

5. Schritt:

Umwandeln des ASCII-Staging-Files in das Imagine-Format
 mittels "restage"

1.151 Befehl STAGING3

Befehl: STAGING3(<basename>,<fromObject>,<toObject>,<fromFrame>,<toFrame <←>
>,<filename>)

Beschreibung: Dieser Befehl erzeugt Auschnitte aus ISL3.x-Staging-Dateien im ASCII-Format (für Imagine3.x)

Beispiele: staging3(ram:obj1,1,20,33,43,ram:st1)
staging3(ram:obj2,30,1,44,74,ram:st2)

Anwendung: Die Objekte "ram:obj.001" bis "ram:obj.049" sollen von Frame 1 bis Frame 49, die Objekte "ram:ob2.001" bis "ram:ob2.029" sollen von Frame 50 bis Frame 78 in umgekehrter Reihenfolge erscheinen.

1. Schritt:

Erzeugen des Staging-Files für das erste Objekt der Sequenz in Imagine (Kamera einstellen, Lichtquelle setzen,...)

2. Schritt:

Umwandeln des Staging-Files mittels dem Programm "destage" in das ASCII-Format

3. Schritt:

```
staging3(ram:obj,1,49,1,49,ram:st.1)
staging3(ram:obj,29,1,50,78,ram:st.2)
!join ram:st.1 ram:st.2 as ram:st
```

4. Schritt:

Editieren des von "destage" erzeugten Files:
Einfügen des Files "ram:st" an der Stelle im Text, wo die Informationen über das erste Objekt der Sequenz stehen, anschließend diese Zeile löschen

5. Schritt:

Umwandeln des ASCII-Staging-Files in das Imagine-Format mittels "restage"

1.152 Befehl STATS

Befehl: STATS([<objectID>])

Beschreibung: Gibt Informationen über ein oder alle Objekte aus

Beispiele: stats(12)
stats

1.153 Befehl STATS2

Befehl: STATS

Beschreibung: Gibt den Status aller Objektspeicherplätze auf einer Bildschirmseite aus. (belegte durch "***" gekennzeichnet)

1.154 Befehl TIME

Befehl: TIME

Beschreibung: Gibt die Bearbeitungszeit des letzten Befehls aus

1.155 Befehl TRANS

Befehl: TRANS (<objectID>, <red>, <green>, <blue>)
 Beschreibung: Ändern der Transparenz
 Beispiel: trans(1,245,0,123)

1.156 Befehl TRANSLATE

Befehl: TRANSLATE (<objectID>, <amount>, <"X" oder "Y" oder "Z">)
 Beschreibung: Verschieben der Objektpunkte entlang einer ausgewählten Richtung
 Beispiel: translate(7,23.5,y)

1.157 Befehl TRIANGULATE

Befehl: TRIANGULATE (<objectID>)
 Beschreibung: Erzeugt zu jeder Fläche eigene Punkte und Kanten
 Beispiel: triangulate(12)

1.158 Befehl WAVE1D

Befehl: WAVE1D (<objectID>, <frames>, <filename>, <"T" oder "L">, [
 <amplitude>, <wavelength>, <source>, <damping>, <phase in Grad
 >])
 Beschreibung: Erzeugt eine eindimensionale ebene Transversal- oder Longitudinal
 -
 Welle,
 dabei bedeuten:
 source - das Zentrum, an dem die Dämpfung Null ist
 damping- der Dämpfungsfaktor, ein Dämpfungsfaktor von 1.0 ist
 so definiert, daß im Abstand von einer Wellenlänge vom
 Zentrum die Amplitude auf die Hälfte abgesunken ist
 T,L - Transversal- bzw. Longitudinal-Welle
 Da diese Funktion anfangs wohl etwas kompliziert zu bedienen sein
 scheint, können die Wellenparameter auch weggelassen werden, das
 Programm ermittelt dann selbst die besten.
 Beispiele: waveld(1,40,obj,t,12.0,24,0,1.0,0.0)
 waveld(1,33,ram:ob,t)
 waveld(1,33,ram:ob,l)

1.159 Befehl WAVE1DFRAME

Befehl: WAVE1DFRAME (<objectID>, <frames>, <frame>, <destID>, <"T" oder "L">, [
 <amplitude>, <wavelength>, <source>, <damping>, <phase in Grad>])
 Beschreibung: Erzeugen eines einzelnen Wellenobjekts
 Beispiele: waveldframe(1,40,12,2,t,12.0,24,0,1.0,0.0)
 waveldframe(1,33,12,4,t)

1.160 Befehl WAVE2D

Befehl: WAVE2D (<objectID>, <frames>, <filename>, <"T" oder "L">, [*amplitude*, <wavelength>, <sourceX>, <sourceY>, <damping>, <phase in degrees>])

Beschreibung: Erzeugen einer ebenen transversal- oder Longitudinalwelle in der x-y-Ebene, siehe auch WAVE1D

Beispiele: wave2d(1,40,obj,1,12.0,24,10,-10,1.6,60.0)
 wave2d(1,60,,1)
 wave2d(1,60,ram:obj,t)

1.161 Befehl WAVE2DFRAME

Befehl: WAVE2DFRAME (<objectID>, <frames>, <frame>, <destID>, <"T" oder "L">, [*amplitude*, <wavelength>, <sourceX>, <sourceY>, <damping \leftrightarrow>, <phase in Grad>])

Beschreibung: Erzeugen eines einzelnen Wellenobjekts

Beispiele: wave2dframe(1,40,20,2,1,12.0,24,10,-10,1.6,60.0)
 wave2dframe(1,60,33,4,1)

1.162 Befehl WAVE3D

Befehl: WAVE3D (<objectID>, <frames>, <filename>, <"S" or "F">, [*amplitude*, <wavelength>, <sourceX>, <sourceY>, <sourceZ>, <damping>, <phase in Grad>])

Beschreibung: Erzeugen einer dreidimensionalen Welle, siehe auch WAVE1D

Beispiele: wave3d(1,40,obj,f,14.0,32.0,10,-10,12,1.2,-30.0)
 wave3d(1,60,hdl:objects,s)

1.163 Befehl WAVE3DFRAME

Befehl: WAVE3DFRAME (<objectID>, <frames>, <frame>, <destID>, <"S" or "F">, [*amplitude*, <wavelength>, <sourceX>, <sourceY>, <sourceZ>, <damping>, <phase in Grad>])

Beschreibung: Erzeugen eines einzelnen Wellenobjekts

Beispiele: wave3dframe(1,40,20,4,f,14.0,32.0,10,-10,12,1.2,-30.0)
 wave3dframe(1,60,33,6,s)

1.164 Befehl WINDOW

Befehl: WINDOW (<objectID>, [<left>, <top>, <width>, <drawmode>, <rotX>, <rotZ>, <zoom>, <keepscale>, <outlined>])

Beschreibung: Öffnen eines Preview-Fensters zum Objekt objectID,

Als drawmode kann dabei angegeben werden:

WIRE, SOLID, GREY, COLOR, oder FACE
 (Durch Drücken der <a>-Taste im Fenster werden jeweils die aktuellen
 Werte der Variablen rotX, rotZ, Zoom und die Fenstergröße ausgegeben.
 Somit kann man für jedes Objekt die besten Werte erst interaktiv ermitteln
 und dann in eine Batch-Datei schreiben.)

Der Parameter KEEPSCALE(boolean) legt fest, ob die Skalierung des Fensters neu berechnet werden soll (false), wenn das Objekt, das dem Fenster zugewiesen wird, verändert worden ist.
 (Die Skalierung wird immer so berechnet, das das Objekt bei einem Zoom-Faktor von 1/0.6 genau das Fenster ausfüllt. Deshalb sind Objekt-Translationen und -Skalierungen in alle drei Richtungen in den Fenstern bei KEEPSCALE=false nicht zu bemerken.)

Bei der Animationsberechnung sollte deshalb KEEPSCALE immer auf true gesetzt werden.

OUTLINED gibt an, ob bei Verwendung eines der Modi GREY, COLOR oder FACE die Flächen umrandet dargestellt werden sollen.
 Diese Einstellung kann im Fenster durch Drücken der <o>-Taste verändert werden.

Werden die Spezifikationen weggelassen, so nimmt das Programm die Werte LEFT, TOP, WIDTH, DRAWMODE, ROTX, ROTZ, ZOOM, VKEEPSCALE und VOUTLINED.

Beispiele: window(1,40,60,400,wire,12,-24,1.0,f,t)
 window(5)
 window(3,0,0,0,lightsourced,16,-36,0.6,t,f)

1.165 Befehl WRITEATTS

Befehl: WRITEATTS(<objectID>
 Beschreibung: Ausgeben der Objekt-Attribute
 Beispiel: writeatts(2)

1.166 Befehl WRITEAXIS

Befehl: WRITEAXIS(<objectID>
 Beschreibung: Ausgeben der Größe, Position, ... des Objektkoordinatensystems

Beispiel: `writeaxis(4)`

1.167 Befehl WRITECLST

Befehl: `WRITECLST(<objectID>[,<start>])`
Beschreibung: Ausgeben der Farben aller Objektflächen,
Beispiele: `writeclst(1)`
`writeclst(2,34)`

1.168 Befehl WRITEEDGES

Befehl: `WRITEEDGES(<objectID>[,<start>])`
Beschreibung: Ausgeben der Objektkanten,
Beispiele: `writeedges(1)`
`writeedges(3,1256)`

1.169 Befehl WRITEFACES

Befehl: `WRITEFACES(<objectID>[,<start>])`
Beschreibung: Ausgeben der Objektflächen,
Beispiele: `writefaces(4)`
`writefaces(3,123)`

1.170 Befehl WRITEPOINTS

Befehl: `WRITEPOINTS(<objectID>[,<start>])`
Beschreibung: Ausgeben der Objektpunkte,
Beispiele: `writepoints(3)`
`writepoints(4,632)`

1.171 Befehl WRITEPPOS

Befehl: `WRITEPPOS(<particleID>[,<start>])`
Beschreibung: Ausgeben der Partikel-Positionen,
Beispiele: `writeppos(2)`
`writeppos(3,12)`

1.172 Befehl WRITEPROT

Befehl: `WRITEPROT(<particleID>[,<start>])`
Beschreibung: Ausgeben der Partikel-Rotationswinkel,
Beispiele: `writeprot(3)`
`writeprot(2,244)`

1.173 Befehl WRITEPSCL

Befehl: WRITEPSCL(<particleID>[,<start>])
Beschreibung: Ausgeben der Partikel-Größe (Skalierungsfaktoren),
Beispiele: writepscl(2)
 writepscl(3,52)

1.174 Befehl ;

Befehl: ; oder #
Beschreibung: Kommentar
Beispiele: ;Kommentar
 #Kommentar

1.175 Befehl !

Befehl: !<DOS-Befehlszeile>
Beschreibung: Ausführen einer DOS-Befehlszeile
Beispiel: !delete ram:obj#?

1.176 Befehl MEMORYP

Befehl: MEMORYP(<particleID>)
Beschreibung: Anzeigen des Speicherverbrauches und der Adressen der einzelnen
 Komponenten eines Partikel-Objektes
Beispiel: memoryp(3)

1.177 Befehl MEMORY

Befehl: MEMORY(<objectID>)
Beschreibung: Anzeigen des Speicherverbrauches und der Adressen der einzelnen
 Komponenten eines Objektes
Beispiel: memory(22)

1.178 Befehl LOADSEQ

Befehl: LOADSEQ(<1stID>,<lastID>,<filename>,<1stFile>)
Beschreibung: Laden der Objekte "filename.(1stFile)", "filename.(1stFile+1)", ←
 ...
 an die Speicherplätze 1stId bis lastId, Umkehrung der Reihenfolge
 möglich
Beispiel: loadseq(1,12,obj,1)

1.179 Befehl SAVESEQ

Befehl: SAVESEQ(<1stID>,<lastID>,<filename>,<1stFile>)

Beschreibung: Speichern der Objekte 1stId bis lastId als "filename.(1stFile)", "filename.(1stFile+1)", ..., Umkehrung der Reihenfolge möglich
Das Format wird durch den globalen Parameter SFORMAT bestimmt

Beispiel: saveseq(120,1,pobj,1)

1.180 Befehl SAVEPSEQ

Befehl: SAVEPSEQ(<1stID>,<lastID>,<filename>,<1stFile>,<saveFormat>)

Beschreibung: Speichern der Partikel-Objekte 1stId bis lastId als "filename.(1stFile)", "filename.(1stFile+1)", ..., Umkehrung der Reihenfolge möglich

Beispiele: savepseq(1,12,pobj,1,obj)
savepseq(1,12,pobj,1,particle)

1.181 Befehl WINDOWSEQ

Befehl: WINDOWSEQ(<1stID>,<lastId>,
[,<borderLeft>,<borderTop>,<borderWidth>,<borderHeight <↔>])

Beschreibung: Öffnen von Vorschau-Fenstern für die Objekte 1stId bis lastId, die Fenster werden in Zeilen arrangiert.
Die Fenstergröße wird so gewählt, daß die Fenster das Rechteck (border) genau ausfüllen, wird kein Rechteck angegeben, so wird der gesamte Bildschirm ausgefüllt.
Ideal für Präsentationen.
Als Fenster-Spezifikation werden die Werte VROTX, VROTZ, VZOOM, VDRAWMODE, VKEEPSCALE und VOUTLINED verwendet.

Beispiele: windowseq(1,12,0,11,640,280)
windowseq(2,8)

1.182 Befehl P2OSEQ

Befehl: P2OSEQ(<1stParticleID>,<lastParticleId>,<1stObjectID>)

Beschreibung: Konvertieren der Partikel-Objekte 1stParticleId bis lastParticleID in die Objekte 1stObjectId bis 1stObjectID+ABS(1stParticleID-lastParticleID), Umkehrung der Reihenfolge möglich

Beispiel: p2oseq(1,24,1)

1.183 Befehl CALC oder .

Befehl: CALC(<expression>) oder .<expression>
Beschreibung: Berechnung von mathematische Ausdrücken, z.B. der für eine Animation benötigten Festplattenkapazität oder Definition neuer globaler Variabel
Beispiele: calc("3*sin(pi/8)")
.a=3/5
.b=a*sin(45)
calc("b+a")

1.184 Programm-Parameter KEEPSCALE

Name: KEEPSCALE
Wertebereich: true/false
Beschreibung: Skalierungsfaktoren des Fensters bei Objektveränderung ↔ beibehalten
Beispiel: set(keepscale,f)

1.185 Programm-Parameter SFORMAT

Name: SFORMAT
Wertebereich: "TDDD", "LW", "MC4D", "VS" oder "NONE"
Beschreibung: Gibt das Objekt-Format an, in dem alle Objekte gespeichert werden
Beispiel: set(sformat,lw)

1.186 Programm-Parameter OUTLINED

Name: OUTLINED
Wertebereich: true/false
Beschreibung: Outline-Flag
Beispiel: set(outlined,t)

1.187 Programm-Parameter PAGER

Name: PAGER
Wertebereich: Pfad eines Textanzeigeprogramms
Beschreibung: Die Online-Help-Texte können auch an ein Textanzeigeprogramm übergeben werden (als Hintergrundtask). Besonders geeignet ist hierfür das Programm "Most", da man so beliebig viele Help- ↔ Fenster auf einem Screen (in effizienter Weise) haben kann.
Beispiele: set(pager,most)
set(pager,sys:utilities/multiview)

1.188 Programm-Parameter HELPDIR

Name: HELPDIR
Wertebereich: Pfadname, der mit "/" oder ":" enden KANN
Beschreibung: Verzeichnis, in dem die Help-Texte gesucht werden
Beispiele: set(helpdir,help:Dust)

1.189 Programm-Parameter HELPDIR2

Name: HELPDIR2
Wertebereich: Pfadname, der mit "/" oder ":" enden KANN
Beschreibung: Verzeichnis, in dem die Help-Texte (zuerst) gesucht werden, dies ist für benutzereigene Texte (z.B. in einer anderen Sprache gedacht)
Beispiele: set(helpdir2,help:Dust)

1.190 Befehl PMORPH

Befehl: PMORPH(<srcID>,<destID>)
Beschreibung: Erzeugen zweier neuer Objekte, die danach z.B. in Imagine oder mit der Prozedur MORPH "gemorphed" werden können
Beispiel: pmorph(1,2)

1.191 Befehl WINDOWCLOSE

Befehl: WINDOWCLOSE(<windowID>)
Beschreibung: Vorschau-Fenster schließen
Beispiel: windowclose(3)

1.192 Befehl WINDOWDRAWMODE

Befehl: WINDOWDRAWMODE(<windowID>,<drawmode>)
Beschreibung: Ändern des Zeichenmodus eines Vorschau-Fensters
Beispiel: windowdrawmode(5,solid)

1.193 Befehl WINDOWOUTLINED

Befehl: WINDOWOUTLINED(<windowID>,<(TRUE|FALSE)>)
Beschreibung: Ändern des Outline-Flags eines Fensters
Beispiel: windowoutlined(12,f)

1.194 Befehl WINDOWPERSPECTIVE

Befehl: WINDOWPERSPECTIVE (<windowID>, <(TRUE|FALSE)>)
Beschreibung: Ändern des Perspective-Flags
Beispiel: windowperspective(1,t)

1.195 Befehl WINDOWREDRAW

Befehl: WINDOWREDRAW (<windowID>)
Beschreibung: Fensterinhalt neu zeichnen
Beispiel: windowredraw(1)

1.196 Befehl WINDOWRESCALE

Befehl: WINDOWRESCALE (<windowID>>)
Beschreibung: Fenster neu skalieren (bei keepscale=TRUE)
Beispiel: windowrescale(9)

1.197 Befehl WINDOWROTX

Befehl: WINDOWROTX (<windowID>, <dalphaX in degrees>)
Beschreibung: Rotationswinkel um die X-Achse erhöhen
Beispiel: windowrotx(4,-30)

1.198 Befehl WINDOWROTZ

Befehl: WINDOWROTZ (<windowID>, <dalphaZ in degrees>)
Beschreibung: Rotationswinkel um die z-Achse erhöhen
Beispiel: windowrotx(4,-15)

1.199 Befehl WINDOWSAVE

Befehl: WINDOWSAVE (<windowID>, <filename>)
Beschreibung: Speichern des angegebenen Fensters als IFF-Bild
Beispiel: windowsave(2,ram:pic)

1.200 Befehl WINDOWZOOM

Befehl: WINDOWZOOM (<windowID>, <dzoom>)
Beschreibung: Zoom-Faktor erhöhen
Beispiel: windowzoom(45,0.1)

1.201 Befehl WINDOWFRONT

Befehl: WINDOWFRONT (<windowID>)
Beschreibung: ein Vorschau-Fenster in den Vordergrund bringen
Beispiel: windowfront (3)

1.202 Befehl WINDOWPOS

Befehl: WINDOWPOS (<windowID>, <XPos>, <YPos>)
Beschreibung: Ändern der Position eines Vorschau-Fenster
Beispiel: windowpos (12, 30, 200)

1.203 Befehl WINDOWSIZE

Befehl: WINDOWSIZE (<windowID>, <XSize>[, <YSize>])
Beschreibung: Ändern der Größe eines Vorschau-Fensters
Beispiele: windowsize (2, 400)
windowsize (2, 200, 181)

1.204 Befehl LOADVS

Befehl: LOADVS (<objectID>[, <filename>])
Beschreibung: Laden eines Videoscape-Objektes an die Speicherstelle objectID
(GEO1-Format)
Es werden alle Flächen, die keine Dreiecke sind, ignoriert.
Beispiele: loadvs (1, hdl:obj)
loadvs (4)
loadvs (3,)

1.205 Befehl LOADGROUPOBJ

Befehl: LOADGROUPOBJ (<objectID>, <filename>, <position>)
Beschreibung: Laden eines Objektes aus einer Objekt-Gruppe (Group-Object)
Beispiel: loadgroupobj (1, obj, 0)
Lädt das erste Objekt, dies ist gleich dem Befehl load (1, obj).

1.206 Befehl SHOWTDDD

Befehl: SHOWTDDD (<filename>)
Beschreibung: Zeigt die Hunks (Objekte) einer TDDD-Datei an, nützlich
für den Befehl LOADGROUPOBJ
Beispiel: showtdd (ram:obj)

1.207 Befehl BUILDMORPH

Befehl: BUILDMORPH (<srcID>, <destID>, <frames>, <filename>)
Beschreibung: Build-Morph (linear)
Beispiele: buildmorph(4,1,60,ram:obj)
 buildmorph(1,2,33,)

1.208 Befehl BUILDMORPHRND

Befehl: BUILDMORPHRND (<srcID>, <destID>, <frames>, <filename>)
Beschreibung: Build-Morph (zufallsgesteuert)
Beispiele: buildmorphrnd(4,1,60,ram:obj)
 buildmorphrnd(1,2,33,)

1.209 Befehl O2S

Befehl: O2S (<structureID>, <particleID>, <"P" or "F">)
Beschreibung: Erzeugen eines sphere-objects
Beispiel: o2s(3,1,f)

1.210 Befehl SAVESPHERES

Befehl: SAVESPHERES (<particleID>[, <filename>])
Beschreibung: ein sphere-object als TDDD-Group abspeichern
Beispiel: savespheres(1,hdl:sl)

1.211 Befehl LWSTAGING

Befehl: LWSTAGING (<baseObject>, <fromObject>, <toObject>, <fromFrame>, < ↔
toFrame>, <baseScene>)
Beschreibung: Dieser Befehl erzeugt aus der Scene-Datei <baseScene> (<toFrame> ↔
- <fromFrame>)
 neue Scene-Dateien, in welchen das Objekt <baseObject> ↔
 entsprechend
 <fromObject> und <toObject> ersetzt wird.
Beispiel: lwstaging(ram:obj1,1,20,33,43,hdl:scenel)
Bemerkung: Zum besseren Verständnis bitte Kapitel 8:
 "Einbindung der Objekte in Lightwave"
 lesen

1.212 Programm-Parameter QUIET

Name: QUIET
Wertebereich: true/false
Beschreibung: Unterdrücken sämtlicher Textausgabe, nützlich, wenn man
Dust von einem externen Programm steuern möchte
Beispiel: set(quiet,t)

1.213 Programm-Parameter LOG

Name: LOG
Wertebereich: true/false
Beschreibung: Schreiben sämtlicher Textausgabe in eine Datei
Beispiel: set(log,t)

1.214 Programm-Parameter LOGFILE

Name: LOGFILE
Wertebereich: string
Beschreibung: Dateiname des Log-Files
Beispiel: set(logfile,t:Dust.log)

1.215 Befehl FILETYPE

Befehl: FILETYPE([<filename>])
Beschreibung: Dieser Befehl gibt den Typ einer Objekt-Datei aus, wenn
es sich um eine Dust-übliche Datei handelt
(TDDD, Lightwave, Videoscape, Particle)
Beispiel: filetype(fl)

1.216 Programm-Parameter LWCMD1, LWCMD2, LWCMD3

Name: LWCMD1, LWCMD2, LWCMD3
Wertebereich: string
Beschreibung: spezielle Lightwave-ARexx-Kommandos, die vor jedem
Frame ausgeführt werden, dabei können MakeLoop-Platzhalter
verwendet werden,
Beispiel: Durch den Befehl "saveimages hdl:pic%" erzeugt
Lightwave die Bilder "pic.0001", "pic.0002", ...
Beispiel: set(lwcmd1,"saveimages hdl:pic%")

1.217 Befehl SAVEVS

Befehl: SAVEVS (<objectID>[,<filename>[,<color>]])
Beschreibung: Speichern des Objektes objektID im VideoScape3D-Format,
wird der Parameter color angegeben, so wird ein
einfarbiges Objekt in dieser Farbe erzeugt
siehe auch Programm-Parameter BACKFACES
Beispiele: savevs (1,hdl:obj,12)
savevs (3,)

1.218 Befehl SAVETDDD

Befehl: SAVETDDD (<objectID>[,<filename>])
Beschreibung: Speichern des Objektes objektID im Imagine-Format,
Beispiele: savetddd (1,hdl:obj)
savevs (3,)

1.219 Befehl SAVELW

Befehl: SAVELW (<objectID>[,<filename>[,<color>]])
Beschreibung: Speichern des Objektes objektID im Lightwave-Format,
siehe auch Programm-Parameter BACKFACES
Beispiele: savelw (1,hdl:obj)
savelw (3,)

1.220 Befehl SAVEMC4D

Befehl: SAVEMC4D (<objectID>[,<filename>[,<color>]])
Beschreibung: Speichern des Objektes objektID im MaxonCinema4D-Format,
Beispiele: savemc4d (1,hdl:obj)
savemc4d (3,)

1.221 Befehl REXX

Befehl: REXX
Beschreibung: Dieser Befehl aktiviert den Dust-ARexx-Modus, so werden
keine Befehle mehr von der Konsole entgegengenommen, bis
dieser Modus durch den ARexx-EXIT-Befehl verlassen wird.
Außer "EXIT" lassen sich alle Dust-Befehle durch den
ARexx-PARSE-Befehl aufrufen, sogar Batch-Dateien können
ausgeführt werden.

ARexx-Beispiele: rx "address 'Dust' 'EXIT' "
rx "address 'Dust' 'PARSE ?' "
rx "address 'Dust' 'PARSE pmorph(1,2) ' "

1.222 Befehl ADDFACE

Befehl: `ADDFACE(<objectID>,<x1>,<y1>,<z1>,<x2>,<y2>,<z2>,<x3>,<y3>,<z3>)`
Beschreibung: erzeugt eine Fläche mit den Eckpunkten (`<x1>,<y1>,<z1>`) (`<x2>,<y2>,<z2>`) und (`<x3>,<y3>,<z3>`), dieser Befehl kann zur Objektgenerierung anhand mathematischer Formeln (z.B. Erzeugung von "Schneckenhäusern" wie Shelly) benutzt werden
Existiert das Object `objectID` nicht, so wird es neu erzeugt.
Beispiel: `addface(1,1,2,3,10,20,30,100,200,300)`

1.223 Befehl BRSEXALIGN0

Befehl: `BRSEXALIGN0(<objectID>,<brushID>)`
Beschreibung: Ausrichten einer Brush-Axis auf 0,0,0
Beispiel: `brsexalign0(4,1)`

1.224 Befehl BRSEXPOS

Befehl: `BRSEXPOS(<objectID>,<brushID>,<XPos>,<YPos>,<ZPos>)`
Beschreibung: Verändern der Brush-Position des Brushes `<brushID>`
Beispiel: `brsexpos(1,4,0.0,20.0,10.0)`

1.225 Befehl BRSEXSIZE

Befehl: `BRSEXSIZE(<objectID>,<brushID>,<Xsize>,<Ysize>,<Zsize>)`
Beschreibung: Verändern der Brush-Größe des Brushes `<brushID>`
Beispiel: `brsexsize(3,0,32.0,50.0,32.0)`

1.226 Befehl BRSDIR

Befehl: `BRSDIR(<objectID>,<path>)`
Beschreibung: Verändern des Pfades aller Brushes eines Objects
Beispiel: `brsdir(1,"hdl:grafix/brushes/color")`

1.227 Befehl BRSEXNAME

Befehl: `BRSEXNAME(<objectID>,<brushID>,<name>)`
Beschreibung: Verändern des Namens eines Brushes
Beispiel: `brsexname(1,0,"hdl:pics/grey/clouds1")`

1.228 Befehl CENTERBRSAxis

Befehl: CENTERBRSAxis (<objectID>, <brushID>)
Beschreibung: Koordinatensystem eines Brushes zentrieren
Beispiel: centerbrsaxis (4,0)

1.229 Befehl CENTERTXTaxis

Befehl: CENTERTXTaxis (<objectID>, <textureID>)
Beschreibung: Koordinatensystem einer Textur zentrieren
Beispiel: centertextaxis (2,10)

1.230 Befehl COPYBRS

Befehl: COPYBRS (<srcID>, <destID>)
Beschreibung: Kopieren/Anhängen der Brushes von einem Objekt zum anderen
Beispiel: copybrs (1,2)

1.231 Befehl COPYTXT

Befehl: COPYTXT (<srcID>, <destID>)
Beschreibung: Kopieren/Anhängen der Texturen von einem Objekt zum anderen
Beispiel: copytxt (1,2)

1.232 Befehl KILLBRS

Befehl: KILLBRS (<objectID>, [<brushID>])
Beschreibung: einen oder alle Brushes eines Objektes löschen
Beispiele: killbrs (1)
 killbrs (2,3)

1.233 Befehl KILLTXT

Befehl: KILLTXT (<objectID>, [<textureID>])
Beschreibung: eine oder alle Texturen eines Objektes löschen
Beispiele: killtxt (1)
 killtxt (2,3)

1.234 Befehl TXTAXALIGN0

Befehl: TXTAXALIGN0 (<objectID>, <textureID>)
Beschreibung: Ausrichten einer Textur-Axis auf 0,0,0
Beispiel: txtaxalign0 (4,1)

1.235 Befehl TXTAXPOS

Befehl: `TXTAXPOS(<objectID>,<textureID>,<XPos>,<YPos>,<ZPos>)`
Beschreibung: Verändern der Textur-Position der Textur <textureID>
Beispiel: `txtaxpos(1,4,0.0,20.0,10.0)`

1.236 Befehl TXTAXSIZE

Befehl: `TXTAXSIZE(<objectID>,<textureID>,<Xsize>,<YSize>,<Zsize>)`
Beschreibung: Verändern der Textur-Größe der Textur <textureID>
Beispiel: `txtaxsize(3,0,32.0,50.0,32.0)`

1.237 Befehl TXTDIR

Befehl: `TXTDIR(<objectID>,<path>)`
Beschreibung: Verändern des Pfades aller Texturen eines Objects
Beispiel: `txtmdir(1,"hdl:grafix/Imagine/textures")`

1.238 Befehl TXTNAME

Befehl: `TXTNAME(<objectID>,<textureID>,<name>)`
Beschreibung: Verändern des Namens einer Textur
Beispiel: `txtname(1,0,"dh0:other_textures/tex1")`

1.239 Befehl TXTPARAM

Befehl: `TXTPARAM(<objectID>,<textureID>,<paramID>,<value>)`
Beschreibung: Verändern eines der 16 Textur-Parameter einer Textur
Beispiel: `txtparam(1,0,3,45.6)`

1.240 Befehl SHOWBRS

Befehl: `SHOWBRS(<objectID>,[<brushID>])`
Beschreibung: Information über alle oder einen Brush(es) anzeigen
Beispiele: `showbrs(1)`
`showbrs(2,3)`

1.241 Befehl SHOWTXT

Befehl: `SHOWTXT(<objectID>,[<textureID>])`
Beschreibung: Information über alle oder eine Textur(en) anzeigen
Beispiele: `showtxt(1)`
`showtxt(2,3)`

1.242 Befehl ROTATEAXIS

Befehl: Rotieren des lokalen Koordinatensystems eines Objekts
Beschreibung: ROTATEAXIS(<objectID>,<angle in degrees>,<"X", "Y" or "Z">)
Beispiel: rotateaxis(4,60,z)

1.243 Befehl ROTATEBRSAxis

Befehl: Rotieren des lokalen Koordinatensystems eines Brushes
Beschreibung: ROTATEBRSAxis(<objectID>,<brushID>,<angle in degrees>,<"X", "Y" ↔
or "Z">)
Beispiel: rotatebrsaxis(4,3,-40,x)

1.244 Befehl ROTATETXTAXIS

Befehl: Rotieren des lokalen Koordinatensystems einer Textur
Beschreibung: ROTATETXTAXIS(<objectID>,<textureID>,<angle in degrees>,<"X", "Y" ↔
or "Z">)
Beispiel: rotatetxtaxis(4,0,60,z)

1.245 Programm-Parameter OPTEDGES

Name: OPTEDGES
Wertebereich: true/false
Beschreibung: Sollen LightWave- oder VideoScape-Objekte später als Imagine-Objekte gespeichert werden, so sollte die Option auf TRUE gesetzt werden (Speicherplatzersparnis), wenn Sie dagegen ausschließlich mit LightWave arbeiten, so ist die Kantenoptimierung sinnlos, da sowieso nur Flächen erzeugt werden.
Beispiel: set(optedges,false)

1.246 Programm-Parameter COMPLETE

Name: COMPLETE
Wertebereich: true/false
Beschreibung: Dust besitzt eine Kommando- und Parameter-Vervollständigung, sodaß z.B. die Eingaben "l(1,obj)" und "load(1,obj)" oder "set(backf,t)" und "set(backfaces,t)" identisch sind. Dies kann durch diesen Parameter abgestellt werden.
Beispiel: set(complete,f)

1.247 Programm-Parameter ACTVAL

Name: ACTVAL
 Wertebereich: Zahl
 Beschreibung: Dieser Wert gibt den Wert des Schleifenzählers außerhalb von Schleifen an, der zur Formatierung von Strings mit den Formatierungsbefehlen benötigt wird
 Beispiel: set(actual,10)

1.248 Befehl WATER

Befehl: WATER(<objectID>,<frames>,<filename>,
 [<amplitude>,<wavelength>,<sourceX>,<sourceY>,
 <damping>,<times>,<rings>)
 Beschreibung: Erzeugen einer Wasserwelle, die z.B. dadurch entsteht, wenn man einen Stein in einen See wirft
 Der Parameter rings gibt an, wieviel Wellenberge sich ausbreiten, times gibt an, wie oft die Ausbreitung eines Wellenberges geschehen soll (Am Ende hat der erste Wellenberg eine Entfernung von times*wavelength vom Zentrum)
 Beispiele: water(1,40,obj,12.0,24,10,-10,1.6,4,1)
 water(1,60,ram:obj)

1.249 Befehl WATERFRAME

Befehl: WATERFRAME(<objectID>,<frames>,<frame>,<destID>,
 [<amplitude>,<wavelength>,<sourceX>,<sourceY>,
 <damping>,<times>,<rings>])
 Beschreibung: Erzeugen eines einzelnen Wasser-Wellen-Objekts
 Beispiele: waterframe(1,40,20,2,12.0,24,10,-10,1.6,4,2)
 waterframe(1,60,33,4)

1.250 Befehl WATERZ

Befehl: WATERZ(<objectID>,<frames>,<filename>,
 [<amplitude>,<wavelength>,<sourceX>,<sourceY>,
 <damping>,<times>,<rings>)
 Beschreibung: Erzeugen einer Wasserwelle, die z.B. dadurch entsteht, wenn man einen Stein in einen See wirft.
 Im Unterschied zu WATER wird hier aber nur die z-Koordinate der Objekte geändert, das sieht weicher aus, ist aber unrealistischer.
 Der Parameter rings gibt an, wieviel Wellenberge sich ausbreiten ↔
 ,
 times gibt an, wie oft die Ausbreitung eines Wellenberges geschehen soll (Am Ende hat der erste Wellenberg eine Entfernung von times*wavelength vom Zentrum)
 Beispiele: waterz(1,40,obj,12.0,24,10,-10,1.6,4,1)
 waterz(1,60,ram:obj)

1.251 Befehl WATERZFRAME

Befehl: WATERZFRAME (<objectID>, <frames>, <frame>, <destID>,
[<amplitude>, <wavelength>, <sourceX>, <sourceY>,
<damping>, <times>, <rings>])

Beschreibung: Erzeugen eines einzelnen Wasser-Wellen-Objekts wie WATERZ

Beispiele: waterzframe(1,40,20,2,12.0,24,10,-10,1.6,4,2)
waterzframe(1,60,33,4)

1.252 Befehl SETCOLSGROUP

Befehl: SETCOLSGROUP (<objectID>, <red>, <green>, <blue>, <groupname>)

Beschreibung: Farbe einer Subgroup/Surface setzen

Beispiel: setcolsgroup(1,205,12,16,uplip)

1.253 Befehl GETCOLSGROUP

Befehl: GETCOLSGROUP (<objectID>, <groupname>)

Beschreibung: Farbe einer Subgroup/Surface anzeigen

Beispiel: getcolsgroup(1,uplip)

1.254 Befehl WRITESGROUP

Befehl: WRITESGROUP (<objectID>, <groupname>)

Beschreibung: Flächen, die eine Subgroup bilden, anzeigen

Beispiel: writesgroup(1,lowlip)

1.255 Befehl ADDSGROUP

Befehl: eine Fläche einer Subgroup zufügen/Subgroup erzeugen

Beschreibung: ADDSGROUP (<objectID>, <faceID>, <groupname>)

Beispiel: addsgroup(1,33,lowlip)

1.256 Befehl SUBSGROUP

Befehl: eine Fläche aus einer Subgroup entfernen/Subgroup entfernen

Beschreibung: SUBSGROUP (<objectID>, <faceID> or -1, <groupname>)

Beispiele: subgroup(1,33,lowlip)
subgroup(1,-1,lowlip)

1.257 Tutorium 1 - MORPH und Imagine-States

Wir erzeugen ein States-Objekt, welches zwei Zustände enthält: eine Kugel und einen Würfel.

Zuerst müssen diese Objekte mit der Dust-PMORPH-Funktion aneinander angepaßt werden:

```
load(1,c1)
load(2,s1)
pmorph(1,2)
save(1,m1)
save(2,m2)
```

Danach können wir Imagine starten und die Objekte "m1" und "m2" in den Detail-Editor laden.

Nach dem Anwählen des Kugel wählen wir die Funktion States/States/Create und ändern den Namen "DEFAULT" in "BALL". Alle Optionen im Data-Type-Requester sollten selektiert worden sein.

Nun fügen wir der Kugel den Zustand des Würfels hinzu, indem wir die Funktion States/States/Import wählen und "PLANE" (bzw. den Namen des Würfels) im Objekt-Fenster selektieren. Der Name des neuen Zustands sollte "CUBE" sein.

(Auf jeden Fall sollte die Option "shape" im Data-Types-Window aktiviert worden sein, ich selektiere gewöhnlich alle)

Nun kann die Kugel gespeichert werden.

Um die States auszuprobieren, ist die Funktion States/States/Tween nützlich. In diesem Fall muß im States-Window der Zustand "CUBE" selektiert werden, der Defaultwert des Tweenings ist auf die Mitte eingestellt.

1.258 Befehl COPYCLST

Befehl: COPYCLST(<srcID>,<destID>)

Beschreibung: Kopieren der Flächenfarben von einem Objekt zum anderen, gut geeignet zum Kombinieren der Befehle MORPH und DEFORMMORPH, beide Objekte müssen die gleiche Flächenanzahl besitzen

Beispiel: copyclst(1,2)

1.259 Befehl MORPHSGROUP

Befehl: MORPHSGROUP(<srcID>,<destID>,<frames>,<filename>)

Beschreibung: Metamorphose zwischen gleichnamigen Subgroups zweier Objekte

Beispiel: morphsgroup(2,1,60,obj)

1.260 Befehl RENAMESGROUP

Befehl: RENAMESGROUP(<objectID>,<oldGroupname>,<newGroupname>)

Beschreibung: eine Subgroup umbenennen (zur Vorbereitung für MORPHSGROUP)

Beispiel: renamesgroup(1,uplip,mouth)

1.261 Befehl SHOWVALUES

Befehl: SHOWVALUES
Beschreibung: Anzeigen aller benutzerdefinierten Konstanten
Beispiel: a1=235.0
 b=34*sin(a1)
 showvalues

1.262 Befehl OCOUNT

Befehl: OCOUNT(<particleID>)
Beschreibung: Abspeichern der Partikel-Anzahl des spezifizierten
 Particle-Objektes in der Variable "ocount" (zum Programmieren)
Beispiel: ocount(21)
 .ocount

1.263 Befehl PCOUNT

Befehl: PCOUNT(<objectID>)
Beschreibung: Abspeichern der Punkt-Anzahl des spezifizierten
 Objektes in der Variable "pcount" (zum Programmieren)
Beispiel: pcount(14)
 .pcount

1.264 Befehl ECOUNT

Befehl: ECOUNT(<objectID>)
Beschreibung: Abspeichern der Kanten-Anzahl des spezifizierten
 Objektes in der Variable "ecount" (zum Programmieren)
Beispiel: ecount(3)
 .ecount

1.265 Befehl FCOUNT

Befehl: FCOUNT(<objectID>)
Beschreibung: Abspeichern der Flächen-Anzahl des spezifizierten
 Objektes in der Variable "fcount" (zum Programmieren)
Beispiel: fcount(71)
 .fcount

1.266 Befehl GETPOINT

Befehl: GETPOINT(<objectID>,<pointID>)
 Beschreibung: Abspeichern der Koordinaten des spezifizierten Punktes in den Variablen "px", "py" und "pz" (zum Programmieren)
 Beispiel: getpoint(1,133)
 .px
 .py
 .pz

1.267 Befehl LATTICE

Befehl: LATTICE(<objectID>,<scale>,<height>,<bscale>)
 Beschreibung: Flächen extrudieren, um gitterähnliche Objekte zu erzeugen. Dieser Effekt arbeitet ähnlich wie Imagine's "lattice"-Funktion, aber hier werden die Flächen zweimal extrudiert und skaliert, um "echte" Gitter zu erhalten.
 Beispiele: lattice(1,0.5,-10,0.2)
 lattice(1,0.3,3,0.3)
 Bemerkungen: 1. Negative height-Werte ergeben auch sehr schöne Effekte.
 2. Subgroups und Flächenfarben werden korrekt reproduziert

1.268 Befehl INSERTPOINT

Befehl: INSERTPOINT(<objectID>,<faceID>[,<x>,<y>,<z>])
 Beschreibung: Fügt einen Punkt in eine Fläche ein (ähnlich wie Imagine's "fracture"-Funktion)
 (default-Position: Zentriert innerhalb der Fläche)
 Beispiele: insertpoint(1,133)
 insertpoint(1,12,-6.0,0.0,12.0)
 Bemerkung: Subgroups und Flächenfarben werden korrekt reproduziert

1.269 Befehl IF

Befehl: IF(<expression>,<command>[,<alternative command>])
 Beschreibung: Simple IF-Construct, das mit geringstem Aufwand implementiert wurde
 Beispiele: if(1<2,echo(true),echo(false))
 if(xmin>t,.xmin=t)
 if(a,echo(a isnt 0),echo(a is 0))
 Bemerkungen: 1. Es sind nur einfache Ausdruecke, die die Boole'schen Operatoren "<=", ">=", "<", ">" und "=" enthalten, erlaubt.
 2. In C-Schreibweise umgeschrieben, lautet die Syntax:
 if(<expression>) <command>; else <alternative command>;

1.270 Programm-Parameter ECHO

Name: ECHO
Wertebereich: ON/OFF
Beschreibung: Unterdrückt die Ausgabe der Kommando-Zeilen auf dem Schirm bei der Ausführung von Batch-Files
Beispiel: set (echo,off)

1.271 Programm-Parameter SPLINETYPE

Name: SPLINETYPE
Wertebereich: "CUBICB", "QUADB", "CATMULLROM", "CUBICBEZIER" oder "QUADBEZIER"
Beschreibung: Spline-Typ, der von Prozeduren wie SMOOTH verwendet werden soll
Beispiel: set (splinetype,catmullrom)

1.272 Programm-Parameter SPLINEENDS

Name: SPLINEENDS
Wertebereich: "OPEN" oder "CLOSED"
Beschreibung: Art der Spline-Enden
Beispiel: set (splineends,closed)

1.273 Programm-Parameter SPLINESUBDIV

Name: SPLINESUBDIV
Wertebereich: Integer >=6
Beschreibung: Anzahl der Punkte, die pro Kontrollpunkt erzeugt werden
Beispiel: set (splinesubdiv,30)

1.274 Befehl INTERPOLATEDATA

Befehl: INTERPOLATEDATA([<src-filename>],[<dest-filename>])
Beschreibung: Dieser Befehl liest ein Datenfile ein, interpoliert diese Daten durch Splines, die durch die Programm-Parameter SPLINETYPE, SPLINEENDS und SPLINESUBDIV spezifiziert werden, und speichert die Ergebnisse wieder als Datenfile ab
Beispiele: interpolatedata()
interpolatedata(in)
interpolatedata(in,out)
Bemerkungen: 1. Dieser Befehl eignet sich auch, um einen Überblick über die von Dust unterstützten Splines zu bekommen ("gnuplot" benutzen)
2. Das Datenfile muß mindestens eine Spalte besitzen, es werden alle Spalten interpoliert

1.275 Befehl CUTSG

Befehl: CUTSG(<objectID>,<subgroup-name>)
 Beschreibung: Löschen der Punkte, Flächen und Kanten, die eine Subgroup bilden
 Beispiel: cutsg(1,G1)

1.276 Befehl EXTRACTSG

Befehl: EXTRACTSG(<objectID>,<destID>,<subgroup-name>)
 Beschreibung: Flächen, Punkte und Kanten einer Subgroup extahieren, um eine neues Objekt zu kreieren
 Beispiel: extractsg(1,2,G1)

1.277 Befehl SMOOTH

Befehl: SMOOTH(<objectID>,[<scale>],["noHalveSegs"])
 Beschreibung: Objekte weicher machen; dazu wird ein langsamer und komplizierter Algorithmus angewandt, der unglaubliche Ergebnisse liefert.

SCALE gibt den Skalierungsfaktor der Spline-Tangenten an, ein höherer Wert resultiert in einer höheren Krümmung der interpolierten Kurven.

Der Parameter "noHalveSegs" schaltet die Interpolation von komplizierten Segmenten ab durch Splines, die Punkte werden dann linear erzeugt und sollten später per Hand "gezogen" werden

Beispiele: smooth(1)
 smooth(1,1.4)
 smooth(1,nohalvesegs)
 smooth(1,1.6,nohalvesegs)

Bemerkungen: 1. Für nichtregistrierte Benutzer existiert ein Limit von 72 Flächen für das Ausgangsobjekt
 2. Subgroups und Flächenfarben werden reproduziert, Punkte und Kanten optimiert
 3. Die Prozedur wurde für folgende Spline-Parameter "designed":
 SPLINETYPE=CATMULLROM
 SPLINEENDS=OPEN
 SPLINESUBDIV=20

1.278 Befehl SMOOTHINNER

Befehl: SMOOTHINNER(<objectID>,[<scale>],["noHalveSegs"])
 Beschreibung: Objekte weicher machen; dazu wird ein langsamer und komplizierter Algorithmus angewandt, der unglaubliche Ergebnisse liefert. Die Randzone des Objektes bleibt hierbei unverändert, was das Wiederverbinden (join) von

getrennten Objektgruppen (split) möglich macht.

SCALE gibt den Skalierungsfaktor der Spline-Tangenten an, ein höherer Wert resultiert in einer höheren Krümmung der interpolierten Kurven.

Der Parameter "noHalveSegs" schaltet die Interpolation von komplizierten Segmenten ab durch Splines, die Punkte werden dann linear erzeugt und sollten später per Hand "gezogen" werden

Beispiele: smoothinner(1)
smoothinner(1,1.4)
smoothinner(1,nohalvesegs)
smoothinner(1,1.6,nohalvesegs)

- Bemerkungen:
1. Für nichtregistrierte Benutzer existiert ein Limit von 72 Flächen für das Ausgangsobjekt
 2. Subgroups und Flächenfarben werden reproduziert, Punkte und Kanten optimiert
 3. Die Prozedur wurde für folgende Spline-Parameter "designed":
SPLINETYPE=CATMULLROM
SPLINEENDS=OPEN
SPLINESUBDIV=20

1.279 Befehl SMOOTHSG

Befehl: SMOOTHSG(<objectID>,<subgroup>,[<scale>])

Beschreibung: Objekt-Untergruppe (Subgroup) weicher machen; dazu wird ein langsamer und komplizierter Algorithmus angewandt, der unglaubliche Ergebnisse liefert.

SCALE gibt den Skalierungsfaktor der Spline-Tangenten an, ein höherer Wert resultiert in einer höheren Krümmung der interpolierten Kurven.

Beispiele: smoothsg(1,lip)
smoothsg(1,ball,1.4)

- Bemerkungen:
1. Für nichtregistrierte Benutzer existiert ein Limit von 72 Flächen für die Subgroup
 2. Die Prozedur wurde für folgende Spline-Parameter "designed":
SPLINETYPE=CATMULLROM
SPLINEENDS=OPEN
SPLINESUBDIV=20

1.280 Programm-Parameter MAXANGLE

Name: MAXANGLE

Wertebereich: 0.0..360.0

Beschreibung: Größter Winkel (in Grad), den zwei Kanten einschließen dürfen, wenn durch sie eine Spline-Kurve durch den SMOOTH-Operator gelegt werden soll

Beispiel: set(maxangle,30)

1.281 Befehl COPYAXIS

Befehl: COPYAXIS (<srcID>, <destID>)

Beschreibung: Kopieren von Axis-Objektattributen
Dieser Befehl ist mitunter notwendig, da COPYATTS seit
Version 2.3 diese Attribute nicht mehr kopiert.

Beispiel: copyaxis(1,2)

1.282 Programm-Parameter STARTPCORR

Name: STARTPCORR

Wertebereich: integer

Beschreibung: Startpunktverschiebung bei geschlossenen Kurven zur
Verhinderung der Verdrehung von Polygonen beim
CDEFORM-Operator

Beispiel: set(startpcorr,-3)

1.283 Programm-Parameter FORCESWAP

Name: FORCESWAP

Wertebereich: boolean

Beschreibung: Bewirkt die Spiegelung der Ausgangskurve für
den CDEFORM-Operator. Dies ist nur notwendig wenn
die Dreiecke nicht verdreht sind, sondern sich
fast senkrecht schneiden.
(Sollte eigentlich nie passieren)

Beispiel: set(forceswap,true)

1.284 Programm-Parameter INTERPMODE

Name: INTERPMODE

Wertebereich: "MODEU" oder "MODEN"

Beschreibung: Bewegungsmodus für Punkte durch den CDEFORM-Operator,
"MODEU" bewirkt die Parametrierung der Kurven
nach ihrem Umfang, "MODEN" nach der Punktdichte
der zweiten Kurve (einfach ausprobieren)

Beispiel: set(interpmode,modeu)

1.285 Befehl CHECKOBJECT

Befehl: CHECKOBJECT (<objectId>)

Beschreibung: Testen eines Objektes und Entfernen ungültiger
Flächen, Punkte und Kanten

Beispiel: checkobject(1)

1.286 Befehl EXPANDSG

Befehl: EXPANDSG(<objectID>,<subgroup>,<newSubgroup>)

Beschreibung: Erweitern einer Subgroup um sie begrenzende Flächen
(wie in DPaint's, wenn man die o-Taste drückt)

Beispiel: expandsg(1,G1,G2)

Bemerkung: dieser Befehl ist sehr nützlich, wenn man für
CDEFORM Kurven definieren möchte

1.287 Befehl SHRINKSG

Befehl: SHRINKSG(<objectID>,<subgroup>,<newSubgroup>)

Beschreibung: Entfernen von Randflächen aus einer Subgroup
(wie in DPaint's, wenn man die O-Taste drückt)

Beispiel: shrink(1,G1,G2)

Bemerkung: dieser Befehl ist sehr nützlich, wenn man für
CDEFORM Kurven definieren möchte

1.288 Befehl SAMEPOS

Befehl: SAMEPOS(<objectID>,<subgroup>,<position>,<dimension>)

Beschreibung: eine Koordinate aller Punkte einer Subgroup auf einen
Wert setzen

Beispiel: samepos(1,G1,1.234,z)

1.289 Befehl CDEFORM

Befehl: CDEFORM(<src-object>,<dest-object>,<sg1>,<sg2>,<round>[,2 ←
Kontroll-Kurven])

<sg1> und >sg2> sind zwei Subgroups, die die Kurvenzüge definieren. Beide Objekte müssen diese Subgroups enthalten. Die Kurven selbst werden aus den Randkanten (boundary-edges) gewonnen. Es werden alle Randkanten, die in beiden Subgroups enthalten sind, verwendet. Wenn es möglich ist, die Kurve nur durch eine Subgroup zu definieren, so kann als Name der zweiten "" oder "NONE" angegeben werden. Soll das ganze Objekt als Subgroup verwendet werden, so muß "MAIN" oder "DEFAULT" angegeben werden.

Der <round>-parameter gibt an, welche Punkte interpoliert, und welche vom Zielobjekt direkt übernommen werden. Interpolierte Punkte, welche einen relativen Abstand von einem Punkt des Zielobjekt haben, der kleiner als <round> ist, werden direkt übernommen. <round> kann Werte von 0.0 (alle Punkte interpolieren) bis 1.0 (keine Punkte interpolieren) annehmen.

Die optionalen 4 Parameter "2 Kontroll-Kurven" sind die Namen zweier Kontroll-Kurven, die helfen, den optimalen Startpunkt bei geschlossenen Kurven zu finden.

Beschreibung: Einen Kurvenzug des Ausgangsobjektes entlang einer Kurve des Zielobjektes deformieren. Dies stellt den "Allround-Effekt" von Dust dar, erlaubt er Metamorphosen in höchster Qualität, animierte boolesche Effekte, ...

Beispiele: `cdeform(1,2,G1,G2,0.0)`
`cdeform(1,2,G1,,0.2)`
`cdeform(1,2,G1,MAIN,0.1,CG1,"",CG2,"")`

- Bemerkungen:**
1. Sie werden fragen: Was sind Kurven(züge) in meinem Polygon-Objekt ?
Tatsächlich bewegt Dust einige Punkte Ihres Ausgangsobjektes in die Nähe von Punkten des Zielobjektes. Damit dies "sauber", möglichst ohne die Verdrehung von Polygonen, geschieht, benötigt das Programm gutsortierte Kurvenzüge. Wenn die Punktzahl des Ausgangsobjektes größer als die des Zielobjektes ist, müssen Punkte interpoliert werden, was mit diesen Kurven möglich ist.
 2. Beide Kurven müssen vom gleichen Typ (offen (Kopf) oder geschlossen (Gesicht)) sein
 3. Nun wird es Ihnen möglich sein, einen Kopf in eine Kugel, ein Gesicht in eine Landschaft, einen "eckigen" Torus in einen "weichen" Torus in höchster Qualität umzuwandeln
 4. Meistens sind zwei Subgroups nötig, um eine Kurve exakt zu definieren
 5. Wenn Sie Nachbarkurven deformieren, kann es zur Verdrehung von Polygonen kommen. In diesem Fall gibt zwei Möglichkeiten:

a) Benutzung des STARTPCORR-Parameters

Beispiel:

Sie deformieren vier Kurven:

```
CDEFORM(1,2,G1,"",0.0)
CDEFORM(1,2,G2,"",0.0)
CDEFORM(1,2,G3,"",0.0)
CDEFORM(1,2,G4,"",0.0)
```

Danach seien die Dreiecke zwischen der 2. und der 3. Kurve nach "links" verdreht, die zwischen der 3. und der 4. Kurve nach "rechts"

Dann versuchen Sie es mit einem STARTPCORR-Wert von 1 für die 3. Kurve:

```
CDEFORM(1,2,G1,"",0.0)
CDEFORM(1,2,G2,"",0.0)
SET(STARTPCORR,1)
CDEFORM(1,2,G3,"",0.0)
SET(STARTPCORR,0)
CDEFORM(1,2,G4,"",0.0)
```

Ist das Ergebnis besser, versuchen Sie mit einem höheren Wert,

um das Optimum zu finden, anderenfalls mit einem negativen

Wert.

b) Benutzung von Kontroll-Kurven

Die Kontroll-Kurven sind immer die nächste und übernächste Vorgänger-Nachbar-Kurve.

```
CDEFORM(1,2,G1,"",0.0)
CDEFORM(1,2,G2,"",0.0)
CDEFORM(1,2,G3,"",0.0,G2,"",G1,"")
CDEFORM(1,2,G4,"",0.0)
```

Ohne Zweifel ist diese Methode besser, aber man benötigt eben immer zwei Vorgänger-Kurven !

7. siehe auch EXPANDSG, SHRINKSG and PREFS (INTERPMODE, FORCESWAP, STARTPCORR)

1.290 Befehl CDEFORMINTERP

Befehl: CDEFORMINTERP (<src-object>, <dest-object>, <sg1>, <sg2>, <sg3>, <sg4>, <sg5>, <sg6>, <scale>, <round>[, 2 Kontrol- ← Kurven])

<sg1> und <sg2> definieren die Kurve des Ausgangsobjektes, (<sg3> and <sg4>) und (<sg5> and <sg6>) die Kurven des Zielobjektes, zwischen denen interpoliert werden soll. <scale> gibt den Zustand zwischen den beiden Kurven (0.0=erste Kurve, 1.0=zweite Kurve, 0.5 dazwischen ...)

Beschreibung: Einen Kurvenzug eines Objektes entsprechend einem Kurvenzug eines anderen Objektes deformieren.. Der zweite Kurvenzug wird aus der Interpolation zweier realer Kurven des Zielobjektes gewonnen

Beispiele: cdeforminterp(1,2,G1,NONE,G1,NONE,G2,NONE,1.0,rr)
cdeforminterp(1,2,G1,G2,gg1,gg2,gg3,gg4,0.75)

- Bemerkungen: 1. Dieser Befehl eignet sich besonders dazu, ein Objekt mit vielen Punkten in eines mit sehr wenigen Punkten umzuwandeln
2. siehe CDEFORM