

Graphisoft GDL Handbuch

Graphisoft

Besuchen Sie die Graphisoft Website bei <http://www.graphisoft.com> für Informationen über ortsnahe Verkäufer und Verfügbarkeit der Produkte.

Graphisoft GDL Handbuch

Copyright © 2004 by Graphisoft, alle Rechte vorbehalten. Die Reproduktion, Umschreibung oder Übersetzung ohne vorherige schriftliche Erlaubnis ist streng verboten.

Warenzeichen

ArchiCAD ist ein eingetragenes Warenzeichen und PlotMaker, Virtual Building, StairMaker und GDL sind Warenzeichen von Graphisoft. Alle anderen Warenzeichen sind Warenzeichen ihrer entsprechenden Eigentümer.

Einführung

Dieses Handbuch ist die vollständige Referenz für die eigene Scriptsprache von Graphisoft, GDL (Geometric Description Language). Dieses Handbuch ist für solche Anwender gedacht, die ihre Möglichkeiten um die vorgestellten Konstruktionswerkzeuge und Objektbibliotheken, die in Graphisoft Software verfügbar sind, erweitern möchten. Es enthält eine detaillierte Beschreibung von GDL, einschließlich Syntaxdefinition, Befehlen, Variablen usw.

INHALT

GDL Programmierungs-Grundlagen	13	SPHERE.....	34
Starten	13	ELLIPS.....	34
Scripting	13	CONE.....	36
Struktur der Bibliothekselemente	13	PRISM.....	36
Analysieren, Zerlegen und Vereinfachen.....	14	PRISM_.....	37
Detaillieren	15	CPRISM_.....	40
Einstieg	15	BPRISM_.....	41
Weiterführende Befehle	16	FPRISM_.....	42
Befehle und Programmfunktionen für Fortgeschrittene	18	HPRISM_.....	44
GDL-Programmieren für Experten	19	SPRISM_.....	44
Wie das 3D-Bild erzeugt wird	19	SLAB.....	46
Der 3D-Raum.....	20	SLAB_.....	46
Wozu dienen Transformationen des Koordinatensystems?.....	20	CSLAB.....	47
Der GDL-Interpreter.....	20	CWALL_.....	47
Der Ablauf der GDL-Script Analyse.....	21	BWALL_.....	49
Grundelemente der GDL-Syntax	23	XWALL_.....	52
Anweisungen	23	XWALL_{2}	54
Zeilen	23	BEAM.....	55
Sprungmarke	23	CROOF_.....	56
Zeichen	23	MESH.....	59
Zeichenfolgen	24	ARMC.....	60
Identifizierer	24	ARME.....	61
Variablen	24	ELBOW	61
Parameter	25	Flächen-Gestalten in 3D	62
Einfache Typen	25	HOTSPOT	62
Derivierte Typen	25	LIN_.....	62
Transformationen des Koordinatensystems	27	RECT	63
2D-Transformationen	27	POLY	63
3D-Transformationen	28	POLY_.....	63
Verwaltung des Transformation-Stacks	31	PLANE.....	64
Dreidimensionale Elemente	33	PLANE_.....	64
Grundkörper	33	CIRCLE	64
Block	33	Bogen	65
BRICK	33	Körper aus Linienzügen	65
CYLIND.....	33	EXTRUDE.....	67
		PYRAMID	70

REVOLVE	72	SWEEPGROUP	119
RULED	75	Binäres 3D	119
RULED{2}	76	2D Elemente	121
SWEEP	78	Zeichnungselemente	121
TUBE	81	HOTSPOT2	121
TUBEA	85	LINE2	121
COONS	87	RECT2	122
MASS	90	POLY2	122
Elemente zur Unterstützung der Photorealistik	92	POLY2_	123
Light	92	POLY2_A	124
PICTURE	94	POLY2_B	124
3D-Textelemente	95	POLY2_B{2}	124
TEXT	95	ARC2	125
RIGHTTEXT	97	CIRCLE2	125
Grundelemente	97	SPLINE2	126
VERT	98	SPLINE2A	127
TEVE	98	PICTURE2	128
VECT	98	DRAWING3{2}	129
EDGE	99	Textelement	129
PGON	99	TEXT2	129
PIPG	100	RIGHTTEXT2	129
COOR	100	Binäres 2D	130
BODY	102	FRAGMENT2	130
BASE	104	FRAGMENT2	130
Verschneiden im 3D-Raum	104	3D Projektionen in 2D	130
CUTPLANE	104	PROJECT2	130
CUTPOLY	106	PROJEKT2{2}	131
CUTPOLYA	108	Zeichnungen in der Liste	132
CUTSHAPE	110	DRAWING2	132
CUTFORM	110	DRAWING3	133
Solid-Element-Befehle	112	DRAWING3{2}	133
GROUP	116	Grafische Bearbeitung	135
ENDGROUP	116	Bearbeitungsbefehle auf Hotspot-Basis	135
ADDGROUP	116	HOTSPOT	135
SUBGROUP	116	HOTLINE2	139
SECTGROUP	116	HOTARC2	140
SECTLINES	116	Statuscodes	141
PLACEGROUP	117	Statuswert-Syntax	141
KILLGROUP	117		

Zusätzliche Statuscodes	143	<i>DEFINE MATERIAL</i>	162
Vorgegebener erster Teil eines Polygonzuges: aktuelle Position und Tangente ist definiert	143	<i>DEFINE MATERIAL BASED_ON</i>	165
Segment, definiert durch den absoluten Endpunkt	143	<i>DEFINE TEXTURE</i>	166
Segment, definiert durch den relativen Endpunkt	144	Schraffuren	168
Segment, definiert durch Längen- und Richtungsangabe	144	<i>DEFINE FILL</i>	168
Tangentiales Segment, definiert durch Längenangabe	145	<i>DEFINE FILLA</i>	171
Angabe des Startpunktes	145	<i>DEFINE SYMBOL_FILL</i>	174
Schließen des Polygonzuges	146	<i>DEFINE SOLID_FILL</i>	175
Angabe der Tangente	146	<i>DEFINE EMPTY_FILL</i>	175
Angabe des Mittelpunktes	147	Linientypen	175
Tangentialem Bogen zum Endpunkt	147	<i>DEFINE LINE_TYPE</i>	175
Tangentialem Bogen, definiert durch Radius und Winkel	148	<i>DEFINE SYMBOL_LINE</i>	176
Kreisbogen, definiert durch Mittelpunkt und Punkt auf der Kreislinie (letzter Radius)	148	Stile	176
Kreisbogen, definiert durch Mittelpunkt und Winkel	149	<i>DEFINE STYLE</i>	176
Geschlossener Kreis, definiert durch Mittelpunkt und Radius ..	149	<i>DEFINE STYLE {2}</i>	177
Attribute	153	Paragraph	178
Anweisungen	153	Textblock	179
Befehle für 3D- und 2D-Skripts	153	Zusätzliche Daten	180
<i>[LET]</i>	153	<i>Externe Dateiabhängigkeit</i>	181
<i>RADIUS</i>	154	Nicht geometrische Skripts	183
<i>RESOL</i>	155	Das Eigenschaften-Skript	183
<i>TOLER</i>	156	<i>DATABASE_SET</i>	183
<i>PEN</i>	156	<i>DESCRIPTOR</i>	184
<i>LINE_PROPERTY</i>	157	<i>REF DESCRIPTOR</i>	184
<i>[SET] STYLE</i>	157	<i>COMPONENT</i>	184
<i>SET STYLE 0</i>	157	<i>REF COMPONENT</i>	185
Nur in 3D-Skripts verwendeten Anweisungen	157	<i>BINARYPROP</i>	185
<i>Modell</i>	157	<i>SURFACE3D ()</i>	185
<i>[SET] MATERIAL</i>	158	<i>VOLUME3D ()</i>	185
<i>SECT_FILL</i>	159	<i>POSITION</i>	185
<i>SHADOW</i>	160	<i>DRAWING</i>	186
Anweisungen nur für 2D-Skripts	161	Das Parameter-Skript	186
<i>DRAWINDEX</i>	161	Werte	187
<i>[SET] FILL</i>	161	Parameter	188
<i>[SET] LINE_TYPE</i>	162	<i>LOCK</i>	189
Inline Attributdefinition	162	<i>HIDEPARAMETER</i>	189
Material	162	Die Benutzeroberfläche (User Interface Script)	189
		<i>UI_DIALOG</i>	189
		<i>UI_PAGE</i>	189

UI_BUTTON	190	NOT	201
UI_SEPARATOR	190	Statistische Funktionen	201
UI_GROUPBOX	190	MIN	201
UI_PICT	190	MAX	201
UI_STYLE	191	RND	201
UI_OUTFIELD	191	Bit-Funktionen	201
UI_INFIELD	191	BITTEST	201
UI_INFIELD {2}	192	BITSET	202
Ausdrücke und Funktionen	195	Spezielle Funktionen	202
Ausdrücke (Expressions)	195	String-Funktionen	202
DIM	195	STR	202
VARDIM1(expr)	196	STR	202
VARDIM2(expr)	196	STR{2}	202
Operatoren	198	SPLIT	207
Arithmetische Operatoren	198	STW	208
Relationale Operatoren	198	STRLEN	208
Bool'sche Operatoren	198	STRSTR	208
Funktionen	199	STRSUB	209
Arithmetische Funktionen	199	Befehle zur Programmsteuerung	211
ABS	199	Befehle zur Programmsteuerung	211
CEIL	199	FOR	211
INT	199	NEXT	211
FRA	199	DO	212
ROUND_INT	199	IF	214
SGN	199	GOTO	215
SQR	199	GOSUB	215
Winkelfunktionen	200	RETURN	215
ACS	200	END / EXIT	216
ASN	200	BREAKPOINT	216
ATN	200	Arbeiten mit dem internen Parameterspeicher	216
COS	200	Macro-Objekte	220
SIN	200	Der Dialog-Befehl	221
TAN	200	Datei Operationen	222
PI	200	OPEN	222
Transzendente Funktionen	200	INPUT	222
EXP	200	VARTYPE	222
LGT	200	OUTPUT	223
LOG	201	CLOSE	223
Bool'sche Funktionen	201		

Verschiedenes	225	Rechteckige Türen/Fenster in gekrümmten Wänden	261
Globale Variablen	225	Nicht rechteckige Türen/Fenster in gerümmten Wänden	263
Allgemeine Umgebungsinformationen	225	Grafische Erzeugung von GDL-Objekten im Grundriß	265
Geschossinformationen	225	Befehle	266
Animationsinformationen	226	Häufige Befehle	266
Allgemeine Parameter von Elementen	227	Reservierte Befehle	266
Parameter von Objekten, Lampen, Türen und Fenstern	227	Befehle für den 3D-Bereich	266
Parameter von Objekten und Lampen	228	Befehle für den 2D-Bereich	268
Parameter von Objekten und Lampen - nur für Listen und Etiketten 228		Befehle für den 2D- und 3D-Bereich	268
Parameter von Objekten und Lampen - nur zum Auflisten und für Etiketten verfügbar	228	Nicht geometrische Scripts	268
Parameter von Fenstern, Türen und Wandenden	229	<i>Eigenschaften-Script</i>	268
Parameter von Fenstern und Türen - nur zum Auflisten und für Etiketten verfügbar	230	<i>Parameter-Text</i>	269
Parameter von Lampen - nur zum Auflisten und für Etiketten verfü- bar	230	<i>Interface-Script</i>	269
Bemaßungsparameter	231	Alphabetische Liste der aktuellen GDL-Befehle	270
Parameter von Wänden - nur für Türen/Fenster verfügbar	232	A	270
Parameter von Wänden - nur zum Auflisten und für Etiketten verfü- bar	233	B	270
Parameter von Stützen - nur zum Auflisten verfügbar	234	C	271
.....	234	D	274
Untergrenzparameter nur zur Auflistung verfügbar	236	E	276
Parameter von Decken - nur zum Auflisten verfügbar	237	F	276
Parameter von Dächern - nur zum Auflisten verfügbar	238	G	277
Schraffurparameter- nur zum Auflisten verfügbar	239	H	277
Parameter der Freiflächenform - nur zum Auflisten verfügbar ..	239	I	277
Benutzerdefinierbare globale Variablen	240	K	278
Alte Globale Variablen	242	L	278
Spezielle Funktionen	243	M	278
REQ	243	N	279
REQUEST	244	O	279
Türen und Fenster	253	P	280
GDL Programmierungs-Grundlagen	253	R	282
Erstellung von Bibliothekselementen des Türen/Fenster-Typs ..	254	S	284
Rechteckige Türen/Fenster in geraden Wänden	254	T	286
Nicht rechteckige Türen/Fenster in geraden Wänden	256	U	287
WALLHOLE	256	V	287
WALLNICHE	260	W	288
		X	289
		Konventionen für Parameternamen	290
		GDL Data I/O Add-On	290
		Beschreibung der Datenbank	290
		Öffnen einer Datenbank	290

Lesen der Werte aus der Datenbank	292
Eingeben von Werten in die Datenbank	293
Datenbank schließen	293
GDL DateTime Add-On	294
Kanal öffnen	294
Lesen der Information	296
Schließen des Kanals	296
GDL File Manager I/O Add-On	296
Spezifikation des Ordners	296
Datei/Ordernamen erhalten	297
Beenden Ordner Scanning	297
GDL Text I/O Add-On	298
Datei öffnen	298
Lesen der Werte	300
Schreiben der Werte	301
Schließen einer Datei	302
Eigenschaften GDL Add-On	303
OPEN	303
CLOSE	303
INPUT	304
OUTPUT	307
GDL XML Erweiterung	307
Das XML-Dokument öffnen	308
Das XML-Dokument ändern	309
Das XML-Dokument ändern	313
Index	319

GDL PROGRAMMIERUNGS-GRUNDLAGEN

GDL Graphic Description Language ist eine **parametrische Programmiersprache**, die BASIC ähnelt. Sie beschreibt 3D-Objekte, z. B. Türen, Fenster, Möbel, Strukturelemente, Treppen, sowie die 2D-Symbole, die diese auf dem Grundriss darstellen. Diese Elemente werden als **Bibliothekselemente** bezeichnet.

STARTEN

Die Anforderungen ihres Entwurfs, Ihre Erfahrungen in der Programmierung und Ihre Kenntnisse der beschreibenden Geometrie beeinflussen wahrscheinlich Ihren Einstieg in GDL.

Beginnen Sie Übung von GDL nicht mit komplizierten Objekten im Hinterkopf. Lernen Sie GDL stattdessen durch schrittweises Experimentieren um alle Funktionen ausnutzen zu können. Halten Sie sich an die folgenden Empfehlungen zum Kenntnisstand.

Beherrschen Sie eine Programmiersprache wie BASIC, können Sie GDL durch vorhandene Scripts kennenlernen. Indem Sie die mit Ihrer Software gelieferten Bibliothekselemente öffnen und die 2D- und 3D-Scripts betrachten, können Sie auch lernen. Zusätzlich können Sie Grundrisselemente im GDL-Format speichern und die resultierenden Scripts betrachten.

Beherrschen Sie BASIC nicht, haben aber mit Konstruktionseinheiten gespielt, können Sie sich durch Übung in GDL zurechtfinden. Wir empfehlen, einfache Befehle sofort auszuführen und die Ergebnisse im 3D-Fenster des Bibliothekselements zu überprüfen.

Für Einzelheiten über die Bibliothekselement-Bearbeitungsumgebung siehe "Parameterobjekte" auf Seite 222.

Graphisoft hat verschiedene Bücher zur GDL-Programmierung und zur Entwicklung von Objektbibliotheken veröffentlicht. "Einführung in die Objekterstellung mit ArchiCAD" ist eine hervorragende Anleitung für Einsteiger. Das "GDL Kochbuch" von David Nicholson Cole ist ein beliebtes Kursbuch für Einsteiger und fortgeschrittene GDL-Programmierer. "GDL Technical Standards" enthält die offiziellen Graphisoft Standards für professionelle Bibliotheksentwickler; dieses Dokument kann kostenlos von der Graphisoft Website heruntergeladen werden. Außerdem bietet das deutschsprachige GDL Forum auf www.archiforum.net viele Möglichkeiten, sich mit anderen über GDL auszutauschen.

SCRIPTING

Struktur der Bibliothekselemente

Jedes Bibliothekselement, das mit GDL definiert wurde, besteht aus den sog. **Scripts**. Diese Scripts sind Listen mit den eigentlichen GDL-Befehlen, welche die 3D-Darstellung und das 2D-Symbol erzeugen. Bibliothekselemente enthalten auch Angaben zur Massenermittlung in ArchiCAD.

Die Befehle des **Masterscripts** werden vor jedem Script ausgeführt (als ob es vor den anderen Scripts des Bibliothekselementes kopiert wäre).

Das **2D-Script** enthält eine parametrische 2D-Zeichnungsbeschreibung. Die **binären 2D-Daten** des Bibliothekselementes (der Inhalt des 2D-Symbolfensters) können durch den Befehl FRAGMENT2 aufgerufen werden. Ist das 2D-Script leer, werden die binären 2D-Daten zur Darstellung des Bibliothekselementes im Grundriß verwendet werden.

Das **3D-Script** enthält eine parametrische 3D-Modellbeschreibung. Die **binären 3D-Daten**, die während einer Import- oder Export Operation generiert werden, können durch den Befehl BINARY aufgerufen werden. .

Das **Eigenschaftenscript** enthält alle Bestandteile und Beschreibungen, die in den Massenermittlungen, Bestandteilen und Raumbüchern verwendet werden. Die in den **Bestandteils-** und **Beschreibungsschnitten** beschriebenen binären **Beschreibungsdaten** können durch den Befehl BINARYPROP aufgerufen werden. Ist das Beschreibungsscript und das Master-Script leer, dann werden die binären Beschreibungsdaten während des Listenprozesses benutzt.

Im **Benutzerinterface-Script** können mögliche Wertegruppen für die Parameter des Bibliothekselementes definiert werden.

Im **Parameter-Script** können mögliche Wertegruppen für die Parameter des Bibliothekselementes definiert werden.

Die im **Parameter-Teil** eingestellten Parameter werden als Grundwerte der Einstellungen des Bibliothekselementes beim Platzieren des Objektes im Grundriß verwendet.

Das **Vorschaubild** wird im Dialogfenster für die Einstellung von Bibliothekselementen beim Suchen der aktiven Bibliothek dargestellt. Es kann durch die Befehle PICTURE und PICTURE2 aus dem 3D- und 2D-Script aufgerufen werden.

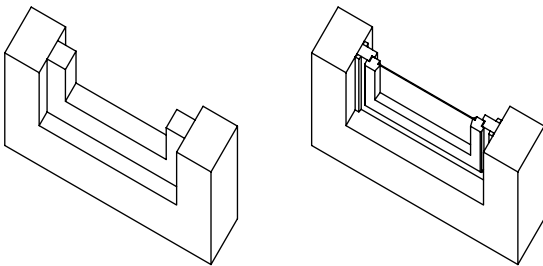
Im **Kommentar**-Teil kann irgendeine Bibliothekselement-bezogene textliche Information gespeichert werden.

ArchiCAD und ArchiFM stellen für die Entwicklung eines GDL-Scriptes eine hilfreiche Entwicklungsumgebung mit sofortiger Kontroll-Visualisierung, Syntax- und Fehlerprüfung zur Verfügung.

Analysieren, Zerlegen und Vereinfachen

Wie komplex sie auch immer sein mögen, die meisten Objekte, die Sie erstellen wollen, lassen sich in einfache geometrische Grundkörper zerlegen. Beginnen Sie mit einer kurzen Analyse und definieren Sie die geometrischen Einzelkörper, aus denen das zu erstellende Objekt besteht. Ist das Objekt aufgeteilt, werden seine Bestandteile in das GDL-Vokabular übersetzt. Aus diesen Einzelbestandteilen wird anschließend das Objekt zusammengesetzt..

Wie bei allen räumlichen Entwurfsprozessen, sind bei solchen Analysen ein gutes räumliches Vorstellungsvermögen und Grundkenntnisse in Geometrie von großer Hilfe.



Fensterdarstellungen in unterschiedlicher Detaillierung

Beginnen Sie mit Objekten, deren Dimensionen klar definiert sind und reduzieren Sie diese auf einfache, aber signifikante Formen. Durch einen angemessenen Einstieg sichern Sie sich den weiteren Erfolg im Lernprozeß. Sind Sie erst einmal mit den Grundformen vertraut, fällt Ihnen der Weg von der einfachen zur idealen Form hin umso leichter.

Dabei bedeutet "ideal nicht gleichzeitig "kompliziert". Auch die Darstellung eines Objektes kann zwischen "skizzenhaft" und "detailliert" variieren, je nach den gestalterischen Anforderungen eines Projektes. Das Fenster auf der linken Seite der oberen Abbildung fügt sich perfekt in ein Bauprojekt. Das rechte Fenster kann der Visualisierung den gewünschten Realismus und Detail verleihen, die in der späteren Projektsphase der Konstruktionsdokumentation verwendet werden können.

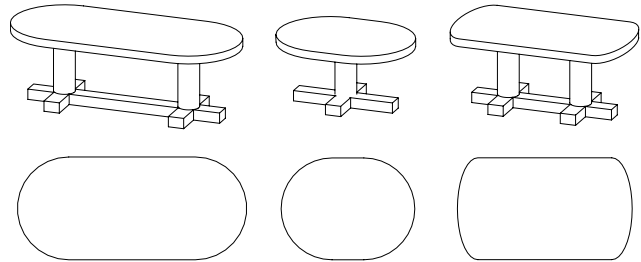
Detaillieren

Der Zweck, für den Objekte erstellt und parametrisiert werden, bestimmt den Grad der Ausarbeitung. Spezielle Objekte für eigene Studien werden sicherlich weniger detailliert werden, als solche, die allgemeine oder kommerzielle Verwendung finden sollen.

Ist die Bedeutung der Symbole im Grundriß weniger wichtig oder braucht eine Veränderung der Parameter im Grundriß nicht ersichtlich sein, so kann auf die ebenfalls parametrisierbaren 2D-Scripts verzichtet werden.

Selbst wenn eine Parameteränderung im Grundriß graphisch erscheinen soll, muß nicht unbedingt ein 2D-Script geschrieben werden. Sie modifizieren die Parameter im 3D-Script, übernehmen die Aufsicht des 3D-Modelles als neues Symbol und speichern das so geänderte Objekt unter einem neuen Namen ab. Eine Vielzahl von Objekten können so vom Original her abgeleitet werden.

Sehr komplexe und anspruchsvolle Bibliothekselemente bestehen aus parametrisierten 3D-Beschreibungen mit den dazu korrespondierenden 2D-Scripts. Änderungen in ihren Einstellungen betreffen somit nicht nur ihre dreidimensionale Darstellung, sondern auch das Erscheinungsbild im Grundriß.



Einstieg

Diese Befehle sind leicht zu verstehen und zu verwenden. Sie erfordern keinerlei Programmierkenntnisse, aber selbst mit dieser begrenzten Auswahl an Befehlen kann man sehr effektiv neue Objekte kreieren.

Einfache Formen

GDL-Körper sind die geometrischen Grundeinheiten, die zu komplexen Bibliothekselementen zusammengefügt werden. Sie sind die Konstruktionseinheiten von GDL. Sie ordnen eine Form im dreidimensionalen Raum durch Eingabe eines Befehls im GDL-Script an.

Ein Befehl zum Erzeugen eines Körpers besteht aus einem Schlüsselwort, das den Typ definiert und aus numerischen Werten oder alphabetischen Parametern, die die Dimensionen bestimmen.

Die Anzahl der Werte ist von Figur zu Figur verschieden.

Zu Beginn können Sie das Arbeiten mit Parametern vernachlässigen und nur mit festen Werten arbeiten.

Die Befehle zum Erzeugen der Grundelemente lauten:

Für den 3D-Raum:

BLOCK CYLIND SPHERE PRISM

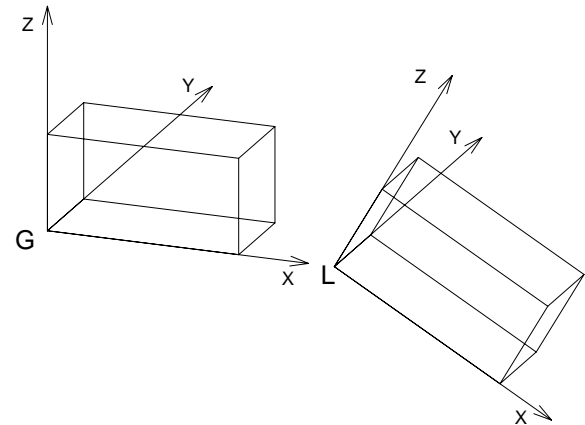
Für das 2D-Symbol:

LINE2 RECT2 POLY2 CIRCLE2 ARC2

Transformationen des Koordinatensystems

Das Verschieben des Koordinatensystemes ist vergleichbar dem Bewegen Ihrer Hand zu einem bestimmten Ort, bevor Sie einen Baustein platzieren. Durch Transformationen wird für die nächste Figur die Position, Orientierung und Skalierung festgelegt.

```
BLOCK 1, 0.5, 0.5  
ADDX 1.5  
ROTY 30  
BLOCK 1, 0.5, 0.5
```



Das 3D-Fenster eines jeden Bibliothekselementes kann auf Wunsch die Ausgangslage (G = global) und die momentane Lage (L = lokal) des 3D-Achsenkreuzes anzeigen.

Die einfachsten Transformationen des Koordinatensystems sind:

Für den 3D-Raum:

ADDX ADDY ADDZ
ROTX ROTY ROTZ

Für das 2D-Symbol:

ADD2 ROT2

Die ADD-Befehle bewegen die jeweils folgende Figur, während die ROT-Befehle diese um die angegebenen Achsen (x,y,z) drehen.

Weiterführende Befehle

Diese Befehle sind etwas komplexer. Nicht, weil sie schwieriger zu programmieren wären, sondern weil sie komplexere Figuren und Transformationen beschreiben.

Für den 3D-Raum:

```
ELLIPS CONE
POLY_ LIN_ PLANE PLANE_
PRISM_ CPRISM_ SLAB SLAB_ CSLAB_
TEXT
```

Für das 2D-Symbol:

```
HOTSPOT2 POLY2_ TEXT2 FRAGMENT2
```

Diese Befehle benötigen für gewöhnlich mehr Werte als die ersten Befehle. Einige von ihnen erfordern Statuswerte zur Kontrolle der Sichtbarkeit von Kanten und Oberflächen.

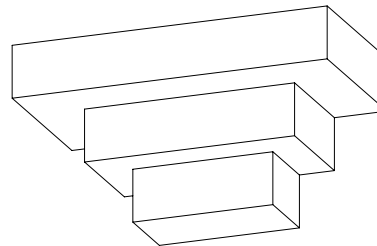
Transformationen des Koordinatensystems

Für den 3D-Raum:

```
MULX MULY MULZ
ADD MUL ROT
```

Für das 2D-Symbol:

```
MUL2
PRISM 4, 1, 3, 0,
      3, 3, -3, 3,
      -3, 0
ADDZ -1
MUL 0.666667, 0.666667, 1
PRISM 4, 1, 3, 0,
      3, 3, -3, 3,
      -3, 0
ADDZ -1
MUL 0.666667, 0.666667, 1
PRISM 4, 1, 3, 0,
      3, 3, -3, 3,
      -3, 0
```



Die Transformationen, die mit MUL beginnen, skalieren die nachfolgenden Figuren. Kreise werden zu Ellipsen verzerrt, Kugeln zu Ellipsoiden. Mit negativen Werten kombiniert, können sie für Spiegelungen eingesetzt werden. Die Befehle der letzten Reihe haben gleichzeitig Auswirkungen auf alle drei Dimensionen.

Befehle und Programmfunktionen für Fortgeschrittene

Der Schwierigkeitsgrad erhöht sich mit den folgenden Befehlen. Das erklärt sich zum einen durch die Geometrie des Körpers, zum anderen beginnt hier der Einstieg in die GDL-Programmiersyntax.

Für den 3D-Raum:

BPRISM_	BWALL_	CWALL_	XWALL_
CROOF_	FPRISM_	SPRISM_	
EXTRUDE	PYRAMID	REVOLVE	RULED
SWEEP	TUBE	TUBEA	COONS
MESH	MASS		
LIGHT	PICTURE		

Innerhalb dieser Gruppe gibt es Befehle, mit deren Hilfe räumliche Polygone erzeugt werden, die auf einem Grundpolygon aufbauen. Damit lassen sich sanft gekrümmte Oberflächen erstellen. Einige Befehle erfordern Verweise auf Materialtypen in der Parameterliste.

Wenn Sie 3D-Schnitte, Polygone und Formen benutzen, können Sie beliebige, komplexe Körper aus einfachen Formen erstellen. Die entsprechenden Befehle dafür sind CUTPLANE, CUTPOLY, CUTPOLYA, CUTSHAPE und CUTEND.

Für das 2D-Symbol:

PICTURE2	POLY2_A
SPLINE2	SPLINE2_A

Befehle zur Programmsteuerung

```
FOR NEXT
DO WHILE ENDWHILE
REPEAT UNTIL
IF THEN ELSE ENDIF GOTO GOSUB
RETURN END EXIT
```

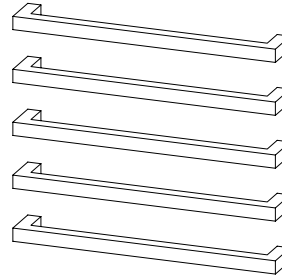
Diese Befehle sind jedem vertraut, der schon einmal ein Programm programmiert hat. Man erkennt jedoch die dahinter stehende Absicht auch ohne Programmiererfahrung.

Damit können sich wiederholende Scripts geschrieben werden, um z.B. viele Elemente mit wenig Aufwand zu platzieren. Auch Wenn-Dann-Bedingungen können definiert werden.


```

FOR I = 1 TO 5
PRISM_ 8, 0.05,
    -0.5, 0, 15,
    -0.5, -0.15, 15,
    0.5, -0.15, 15,
    0.5, 0, 15,
    0.45, 0, 15,
    0.45, -0.1, 15,
    -.45, -0.1, 15,
    -0.45, 0, 15
ADDZ 0.2
NEXT I

```



Parameter

Ersetzen Sie nun feste Zahlenwerte durch variable Namen. Ihre Objekte können dadurch flexibler eingesetzt werden. Wenn Sie an einem Projekt arbeiten, haben Sie durch das Dialogfenster des Bibliothekselementes Zugriff auf diese Größen.

Makro-Objekte

Sie müssen sich nicht nur auf die Auswahl an GDL-Körpern beschränken, die durch die Programmiersprache vorgegeben sind. Jedes schon vorhandene Bibliothekselement kann komplett als GDL-Objekt fungieren. Die Vorgehensweise ist der bei der Erzeugung von GDL-Körpern gleich..

GDL-Programmieren für Experten

Haben Sie sich die oben aufgeführten Programmfunktionen und Befehle erst einmal erarbeitet und die Zusammenhänge verstanden, dann sind Sie ohne Probleme im Stande sich der wenigen, noch verbleibenden Befehle anzunehmen.

Anmerkung: Die Speicherkapazität Ihres Rechners kann die Dateilänge Ihres GDL-Scriptes, die Tiefe der Makro-Aufrufe sowie die Anzahl der Transformationen begrenzen.

Detailliertere Informationen zu den GDL-Befehlen finden Sie im weiteren Verlauf dieses Handbuchs. Einen schnellen Überblick über die aktuellen Befehle und deren Parameterstruktur erhält man unter den entsprechenden Befehlen im Hilfemenü in ArchiCAD.

WIE DAS 3D-BILD ERZEUGT WIRD

Die 3D-Berechnungen basieren auf Fließkommazahlen, d.h. es gibt keine Begrenzung für die Größe eines Modells. Wie groß es auch sein mag, die Genauigkeit bleibt auch im kleinsten Detail erhalten

Das 3D-Modell, das Sie auf dem Bildschirm sehen, besteht aus **geometrischen Grundelementen**. Diese sind im Arbeitsspeicher Ihres Rechners in binärer Form gespeichert. Der 3D-Engine erzeugt die 3D-Darstellung entsprechend dem Grundriß, den Sie zuvor erstellt haben. Dieses Zusammenspiel zwischen den intelligenten, architektonischen Elementen im Grundriß und den 3D-Daten wird 3D-Konvertierung genannt.

Die geometrischen Grundelemente sind:

- alle **Eckpunkte** der 3D-Körper
- alle **Körperkanten**, welche die Eckpunkte verknüpfen
- alle **Oberflächen-Polygone**, die durch die Körperkanten beschrieben werden.

Diese Grundelemente bilden zusammen die **3D-Körper**. Aus ihnen setzt sich wiederum das gesamte 3D-Modell zusammen. Sämtliche 3D-Visualisierungen – glatte Oberflächen, Schattenwürfe, glänzende oder durchsichtige Materialien – basieren auf dieser Datenstruktur.

Der 3D-Raum

Das 3D-Modell wird innerhalb eines durch x-, y- und z-Achse definierten, also dreidimensionalen Raumes, dem Hauptkoordinatensystem erzeugt. Der dazugehörige Koordinaten-Ursprung wird **absoluter Nullpunkt** genannt.

Der absolute Nullpunkt wird, wenn Sie das Programm gestartet haben, ohne ein bereits bearbeitetes Dokument zu öffnen, in der unteren linken Ecke eines Arbeitsblattes durch ein dickes schwarzes Kreuz dargestellt. Zusätzlich legt er im Plan das Niveau $\pm 0,00$ fest, auf das sich alle anderen Geschoße aufbauen.

Angenommen, Sie platzieren ein Objekt im Grundriß, so wird anhand der x- und y-Koordinate des Hauptkoordinatensystems die Position definiert. Die Lage auf der z-Achse kann im Dialogfenster Objekteinstellungen oder unmittelbar bei der Anordnung in 3D festgelegt werden. Dieser Ort ist die Unterseite des Objekts und die Grundposition seines **örtlichen Koordinatensystems**. Die im GDL-Script erzeugten Körper beziehen sich auf diesen Nullpunkt.

Wozu dienen Transformationen des Koordinatensystems?

Alle geometrischen Elemente in GDL sind fest an ihr eigenes, lokales Koordinatensystem gebunden. So ist z.B. ein Eckpunkt eines Quaders, der mittels des Befehles BLOCK erzeugt wurde, fest im Nullpunkt verankert. Länge, Breite und Höhe werden entlang der x-, y- und z-Achsen bestimmt. Der Befehl BLOCK erfordert somit zur Bestimmung der Dimensionen des Quaders nur drei Parameter.

Wie kann man nun einen weiteren Block, der eine andere Position hat und um 45° Grad gedreht ist, erzeugen? In der Parameter-Struktur des BLOCK-Befehles gibt es dazu keine Möglichkeit. Die Parameter für Verschieben und Drehen fehlen.

Die Lösung liegt in der Verschiebung des Koordinatensystems in die gewünschte Lage, bevor man den BLOCK-Befehl anwendet. Mit Hilfe der Befehle zur Transformation des Koordinatensystems legt man vorher die Position und die Rotation des Körpers bezüglich der Raumachsen fest. Diese Transformationen wirken sich nur auf die, in der GDL-Syntax noch folgenden Körper aus. Die bereits erzeugten Körper bleiben davon unberührt.

Der GDL-Interpreter

Der GDL-Interpreter ist das Programmteil, welches das GDL-Script in die speziellen internen Befehle (z.B. für die Bildschirmdarstellung) des Computers übersetzt. Wird ein GDL-Script ausgeführt, erkennt der GDL-Interpreter Lage, Größe, Rotationswinkel, die vom Benutzer definierten Parameter und eventuell den "Gespiegelt"-Modus des Objektes. Dann wird das lokale Koordinatensystem an die richtige Position verschoben, um die weiteren GDL-Befehle aus dem Script des Bibliothekselementes bearbeiten zu können. Jedes Mal, wenn der Interpreter einen Befehl zur Erzeugung eines geometrischen Grundkörpers erhält, wandelt er ihn für die graphische Darstellung am Bildschirm um.

Wenn der Interpreter die Übersetzung ausgeführt hatte, wird das ganze 3D- Binär-Modell in dem freien Arbeitsspeicher gespeichert, wo Sie 3D-Projektionen, photorealistische Darstellungen oder Sonnenstudien erzeugen können.

ArchiCAD und ArchiFM enthalten einen Precompiler und einen Interpreter für GDL. Die Übersetzung von GDL-Scripts verwendet precompilierten Code. Das ermöglicht eine schnellere Umsetzung der Daten. Jedes Mal, wenn der GDL-Text verändert wird, wird ein neuer Code erzeugt.

Daten, die über andere Dateiformate (z.B. DXF, Zoom, Alias Wavefront) in ArchiCAD importiert wurden, werden in der Objektbibliothek in Binär-Form abgelegt. Der BINARY-Befehl bezieht sich auf diese Datenstrukturen.

Der Ablauf der GDL-Script Analyse

Anwender haben keine Übersicht darüber, in welcher Reihenfolge, die im Grundriß platzierten Bibliothekselemente berechnet werden. Die Reihenfolge der GDL Text-Übersetzung beruht auf der inneren Datenstruktur; überdies können diese Reihenfolge sowohl die "Rückgängig" und "Wiederholen"-Operationen als auch die Änderungen beeinflussen. Diese Regel gestattet aber Ausnahmen und zwar die speziellen GDL-Scripte der aktiven Bibliothek, deren Namen mit **"MASTER_GDL"** oder **"MASTEREND_GDL"** beginnen.

Scripte, deren Namen mit "MASTER_GDL" beginnen, werden vor einer 3D-Umwandlung, vor der Erstellung einer Schnitte/Ansichten, vor dem Starten eines Berechnungsprozesses und nach der Auswahl der aktiven Bibliothek dargestellt.

Scripte, deren Namen mit "MASTEREND_GDL" beginnen, werden nach einer 3D-Umwandlung-Szene, nach der Erstellung eines Schnitts oder Ansicht, nach dem Beenden eines Berechnungsprozesses und wenn die aktive Bibliothek zu ändern ist (Bibliotheken wählen, Projekte öffnen, Neues Projekt, Beenden) ausgeführt.

Diese Scripte werden nicht dargestellt, wenn Sie Bibliothekselemente bearbeiten. Beinhaltet Ihre Bibliothek eine oder mehrere Scripte von dieser Art, so werden all diese in einer nichtdefinierten Reihenfolge dargestellt.

MASTER_GDL und MASTEREND_GDL Scripte können Attribute-Definitionen, Initialisierungen der GDL-Anwender, Globale Variablen, 3D-Befehle (welche nur in 3D wirksam sind), Werteliste-Definitionen (siehe den Befehl VALUES im Kapitel Nichtgeometrische Scripte) und erweiterungsspezifische GDL-Befehle beinhalten (see the *"Werte" auf Seite 187*). Die in diesen Scripten definierten Attribute werden der aktuellen Gruppe von Attributen dazugeladen (Attribute mit den selben Namen werden nicht ersetzt, da Attribute, die aus GDL stammen und in dem Programm nicht bearbeitet werden, immer ersetzt werden).

GRUNDELEMENTE DER GDL-SYNTAX

In diesem Kapitel werden die grundlegenden Elemente der GDL-Syntax beschrieben, einschließlich Anweisungen, Sprungmarken, Identifizierern, Variablen sowie Parametern. Die typographischen Regeln werden auch detailliert behandelt.

Regeln der GDL Syntax

GDL unterscheidet nicht zwischen Groß- und Kleinschreibung, ausgenommen Strings zwischen Aufführungszeichen. Das logische Ende eines GDL-Scripts wird durch END, EXIT oder das physische Ende des Scripts gekennzeichnet.

ANWEISUNGEN

Ein GDL-Programm besteht aus Anweisungen. Eine Anweisung beginnt entweder mit einem Befehl (erzeugt einen GDL-Körper, transformiert das Koordinatensystem oder kontrolliert den Programmablauf), mit dem Namen eines Makros oder mit einer Variablen, gefolgt von einem '='-Zeichen und einem mathematischen Ausdruck.

ZEILEN

Die Anweisungen in den Textzeilen werden durch Steuerzeichen oder das Zeilenende voneinander getrennt.

Ein Komma (,) an der letzten Stelle zeigt an, daß die Anweisung in der nächsten Zeile fortgesetzt wird. Ein Doppelpunkt (:) wird für die Trennung verschiedener GDL-Anweisungen innerhalb einer Zeile benutzt. Nach einem Ausrufezeichen (!) können Sie eine beliebige Bemerkung in die Zeile schreiben. Sie wird nicht als Anweisung interpretiert. In GDL-Scripts können Sie ohne Auswirkung auf die Funktion beliebig viele Leerzeilen einsetzen. Leerschritte und Tabulatoren zwischen Operanten und Operatoren können verwendet werden. Wichtig ist jedoch, daß ein Leerzeichen oder ein Tabulator nach einem Befehl oder einem Makro gesetzt wird.

SPRUNGMARKE

Jede Zeile kann mit einer Sprungmarke beginnen. Eine Sprungmarke ist eine ganze Zahl, die von einem Doppelpunkt (:) gefolgt wird. Die folgenden Anweisungen können sich dann auf die Sprungmarke beziehen. Eine Sprungmarke darf nur einmal erscheinen. Der Programmablauf kann durch Angabe einer beliebigen Sprungmarke über die Befehle GOTO oder GOSUB gesteuert werden.

ZEICHEN

Ein GDL-Text setzt sich aus den Buchstaben des englischen Alphabetes, Ziffern und den folgenden Zeichen zusammen:

<Leerzeichen> _ (unterstrichen) ~ ! : , ; . + - * / ^ = < > <= >= # () [] { } \ @ & | (vertical bar) " ' ` ^ " " ' ` <end_of_line>

ZEICHENFOLGEN

Zeichenfolgen sind beliebige Zeichenfolgen, die sich zwischen Anführungszeichen ("',',') befinden oder Zeichen ohne Anführungszeichen, die in dem Script nicht als Identifizierer mit einem gegebenen Wert (Makro-Aufruf, Attributname, Dateiname) funktionieren. Zeichenfolgen ohne Anführungszeichen werden in Großbuchstaben konvertiert, daher sind Anführungszeichen empfohlen. Eine Zeichenfolge darf maximum 255 Zeichen beinhalten.

Das '\ ' Zeichen hat spezielle Kontrollwerte. Seine Bedeutung ist von dem nächsten Zeichen abhängig.

```
\\      '\ ' Zeichen selbst
\n      neue Zeile
\t      Tabulator
\Fortsetzen mit dem GDL-Text in der nächsten Zeile ohne neue Zeile
\andere nicht richtig, Ergebnisse in Zeichen
```

Beispiele:

```
"Das ist eine GDL-Zeichenfolge"
""Waschbecken 60*50""
' 'Verwenden Sie nicht unterschiedliche
Trennungszeichen'
```

IDENTIFIZIERER

Identifizierer sind spezielle Zeichen Variablen::

- sie dürfen nicht mehr als 255 Zeichen beinhalten ;
- sie beginnen mit einem Buchstaben des Alphabets oder einem '_' oder '~' Zeichen;
- sie bestehen aus Buchstaben, Ziffern und '_' oder '~' Zeichen.;
- Groß- und Kleinschreibung wird identisch berücksichtigt.

Identifizierer können GDL-Schlüsselwörter, globale oder lokale Variablen oder GDL-Texte (Namen) sein. Schlüsselwörter und globale Variable Namen werden vom

Programm festgestellt; alle anderen Identifizierer können als Variable Namen verwendet werden.

VARIABLEN

GDL-Programm kann numerische Variablen und Textvariablen (durch ihre Identifizierer), Ziffern und Zeichenfolgen handhaben.

Es gibt zwei Arten von Variablen: lokale und globale.

Alle Identifizierer, die keine Schlüsselwörter, globalen Variablen, Namen von Attributen, Makros, oder Dateien sind, werden als lokale Variablen betrachtet. Falls diese ohne Anfangswert bleiben, beträgt ihr Wert 0.0. Lokale Variablen werden beim Aufruf von GDL-Texten temporär verwendet. Ist ein GDL-Text abgearbeitet, so werden ihre Werte gelöscht.

Globale Variablen haben reservierte Namen (Liste der globalen Variablen ist im Anhang zu finden). *“Verschiedenes” auf Seite 225*). Diese werden während des Aufrufes von GDL-Texten nicht aufgestapelt. Ist ein GDL-Text abgearbeitet, so werden ihre Werte gelöscht. Die Werte globaler Variablen sind für alle GDL-Texte gültig, wie z. B. der Zeichnungsmaßstab. Die globale Variablen des Benutzers können in beliebige Scripts eingesetzt werden, aber werden erst in den nachfolgenden Scripts aktiv. Wenn Sie sich davon überzeugen wollen, daß das gewünschte Script umgesetzt wurde, fügen Sie diese Variablen in das MASTER_GDL Bibliothekselement ein. Die anderen globalen Variablen können in Ihren Scripts verwendet werden, um mit dem Programm zu kommunizieren.

Unter Verwendung des "=" Befehls können Sie numerische Variablen oder Textvariablen in lokale und globale Variablen einfügen.

PARAMETER

Identifizierer, die in der Parameterliste eines Bibliotheksteiles aufgelistet sind, werden Parameter genannt. Parameter Identifizierer dürfen 32 Zeichen nicht überschreiten. Innerhalb eines Scriptes gelten dieselben Regeln für Parameter wie für lokale Variablen.

Parameter von GDL-Textdateien werden durch Buchstaben von A bis Z identifiziert.

EINFACHE TYPEN

Es gibt zwei einfache Typen von Variablen, Parametern und Ausdrücke: numerische oder textliche.

Numerische Ausdrücke sind ständige Ziffern, numerische Variablen oder Parameter, Funktionen, die numerische Werte und deren beliebige Kombinationen in die Operation zurückstellen. Numerische Ausdrücke können Ganzzahlen oder reelle Zahlen sein. Ganzzahl Ausdrücke sind ständige Ziffern, numerische Variablen oder Parameter, Funktionen, die Ganzzahlwerte und deren beliebige Kombinationen in die Operation zurückstellen. Reelle Ausdrücke sind ständige Ziffern, numerische Variablen oder Parameter, Funktionen, die reelle Werte und deren beliebige Kombinationen (oder Ganzzahl Ausdrücke) in die Operation zurückstellen. Ein numerischer Ausdruck (eine Ganzzahl oder reelle Zahl) wird während des Zustellungsvorganges bestimmt und ist nur von den ständigen Ziffern, Variablen oder Parametern und den Operationen abhängig, von denen diese kombiniert werden. Reele und Ganzzahl Ausdrücke können überall gleich verwendet werden, wo ein numerischer Ausdruck benötigt wird. Es kann jedoch vorkommen, daß eine Kombination von diesen ein Präzisionsproblem verursacht. In diesem Fall erscheint eine Warnungsmeldung (Vergleich von reellen Zahlen oder reellen Zahlen und Ganzzahlen, die relationelle Operatoren '=' oder '<>', oder boolsche Operatoren AND, OR, EXOR; IF oder GOTO Anweisungen mit reellen Sprungmarken Ausdrücken verwenden).

String Ausdrücke sind ständige Strings, String Variablen oder Parameter, Funktionen, die Strings und deren beliebige Kombinationen in die Operation zurückstellen.

DERIVIERTE TYPEN

Variablen und Parameter können auch Reihen sein und Parameter können Wertelisten von einer einfachen Art sein.

Reihen sind ein- oder zweidimensionale Tabellen mit numerischen bzw. Textwerten, die direkt durch Indizes zugänglich sind.

Variablen und Parameter können auch Reihen sein und Parameter können *Wertelisten* von einer einfachen Art sein. Sie können den Parametern im Werteliste-Script des Bibliothekselementes oder im MASTER_GDL Script zugewiesen werden und werden in der Parameterliste als Popup-Menü erscheinen.

[aaa]

Eckige Klammern sagen aus, daß der Inhalt optional ist (sind diese fettgedruckt, müssen sie eingegeben werden wie gezeigt).

{n}

Befehlsversionsnummer

...

Vorherige Elemente können wiederholt werden

variable

Beliebiger GDL-Variable Name

prompt

Beliebige alphanumerische Zeichenfolge oder Wert

FETTGEDRUCKTE_ZEICHEN

GROSSGESCHRIEBENE_ZEICHEN

spezielle Zeichen

Muß eingegeben werden wie gezeigt

andere_kleingeschriebene_zeichen_in_der_parameterliste

Beliebiger GDL-Ausdruck

TRANSFORMATIONEN DES KOORDINATENSYSTEMS

In diesem Kapitel werden Transformationsstypen beschrieben, die in GDL verfügbar sind (Bewegen, Bemaßen, Drehen des Koordinatensystems), sowie die Art und Weise, wie sie interpretiert und verwaltet werden.

Über Transformationen

In GDL sind alle geometrischen Elemente streng an das lokale Koordinatensystem gebunden. GDL verwendet ein rechtshändiges Koordinatensystem. Ein Eckpunkt eines BLOCKs liegt beispielsweise im Nullpunkt verankert und seine Oberflächen befinden sich in der x-y-, x-z- und der y-z-Ebene.

Zwei Schritte sind nötig, um ein geometrisches Element in die gewünschte Position zu bringen. Zuerst schieben Sie das Koordinatensystem an die gewünschte Stelle, danach erzeugen Sie dort den Körper. Danach erzeugen Sie dort den Körper. Jede Verschiebung, Drehung oder Streckung des Koordinatensystems entlang oder um seine Achsen heißt Transformation.

Alle Transformationen werden nacheinander in einer Datei (Stack) regelrecht “aufgestapelt”; die Abarbeitung im Programm erfolgt dann in umgekehrter Reihenfolge. GDL-Scripts arbeiten mit diesem “Transformations-Stack”; dabei können sie neue, lokale Transformationen hinzufügen und auch wieder löschen. Gelöscht oder rückgängig gemacht werden können eine, mehrere oder alle Transformationen im aktuellen Script. Nachdem ein GDL-Script abgearbeitet wurde, werden alle lokalen Transformationen aus dem Stack gelöscht.

2D-TRANSFORMATIONEN

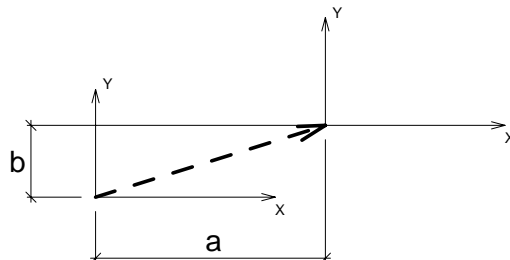
Diese sind die Äquivalenten von ADD, MUL und ROTZ 3D Transformationen im 2D-Raum..

ADD2

ADD2 x , y

Beispiel:

ADD2 a , b



MUL2

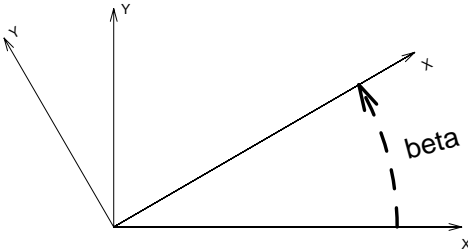
MUL2 x, y

ROT2

ROT2 α

Beispiel:

ROT2 β



3D-TRANSFORMATIONEN

ADDX

ADDX dx

ADDY

ADDY dy

ADDZ

ADDZ dz

Bewegt das lokale Koordinatensystem entlang der angegebenen Achse für dx , dy oder dz .

ADD

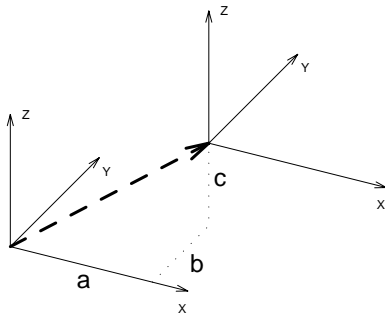
ADD dx, dy, dz

Ersetzt die Befehlsfolge ADDX dx: ADDY dy: ADDZ dz.

Der Befehl erzeugt nur einen Eintrag im Stack, mit DEL 1 kann er daher gelöscht werden.

Beispiel:

ADD a,b,c



MULX

MULX mx

MULY

MULY my

MULZ

MULZ mz

Skaliert das lokale Koordinatensystem entlang der angegebenen Achse. Negative Werte für mx, my, mz bedeuten eine gleichzeitige Spiegelung.

MUL

MUL mx, my, mz

Ersetzt die Befehlsfolge MULX mx : MULY my : MULZ mz. Der Befehl erzeugt nur einen Eintrag im Stack, mit DEL 1 kann er daher gelöscht werden.

ROTX

ROTX α phax

ROTY

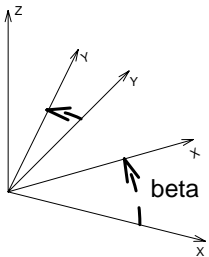
ROTY α phay

ROTZ

ROTZ α phaz

Dreht das lokale Koordinatensystem um einen Winkel α phax, α phay oder α phaz. Die Drehung um die angegebene Achse erfolgt entgegen dem Uhrzeigersinn.

Beispiel:



ROTZ β eta

ROT

ROT x, y, z, α pha

Dreht das lokale Koordinatensystem entgegen dem Uhrzeigersinn um den durch x, y, z definierten Vektor im Winkel α pha. Der Befehl erzeugt nur einen Eintrag im Stack, mit DEL 1 kann er daher gelöscht werden.

XFORM

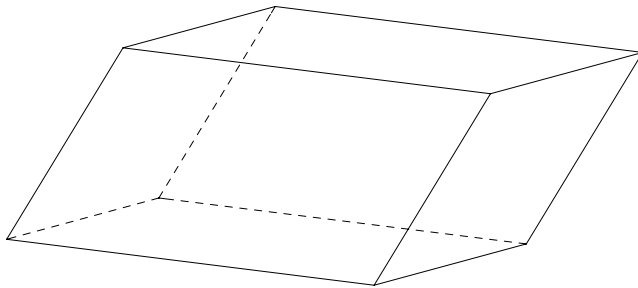
XFORM a11, a12, a13, a14,
a21, a22, a23, a24,
a31, a32, a33, a34

Definiert eine komplette Transformationsmatrix. Es wird hauptsächlich bei der automatischen Generierung des GDL-Codes. Der Befehl erzeugt nur einen Eintrag im Stack.

$$\begin{aligned}x' &= a_{11} * x + a_{12} * y + a_{13} * z + a_{14} \\y' &= a_{21} * x + a_{22} * y + a_{23} * z + a_{24} \\z' &= a_{31} * x + a_{32} * y + a_{33} * z + a_{34}\end{aligned}$$

Beispiel:

```
A=60
B=30
XFORM 2, COS(A), COS(B)*0.6, 0,
      0, SIN(A), SIN(B)*0.6, 0,
      0, 0, 1, 0
BLOCK 1, 1, 1
```



VERWALTUNG DES TRANSFORMATION-STACKS

DEL n

DEL n [, begin_with]

Löscht die vorangegangenen Einträge (n=Anzahl der Einträge) im Transformations-Stack.

Wird der Parameter begin_with nicht definiert, werden die durch n bestimmten, vorherigen Einträge im Transformations-Stack gelöscht. Das lokale Koordinatensystem wird in eine frühere Position gebracht.

Wird die Transformation begin_with definiert, werden die durch n definierten Einträge vorwärts gelöscht. Die Löschung beginnt mit dem durch start_with angezeigten Eintrag. Die Numerierung beginnt mit 1. Falls der Parameter begin_with nicht definiert wird und n einen negativen Wert hat, so erfolgt die Löschung in umgekehrter Reihe des Eintrages in den Stack.

Wenn mit n weniger Transformationen angegeben werden, als im GDL-Script enthalten sind, so wird nur die durch n definierte Anzahl von Transformationen gelöscht.

DEL TOP

DEL TOP

Löscht alle aktuellen Transformationen im aktuellen Script.

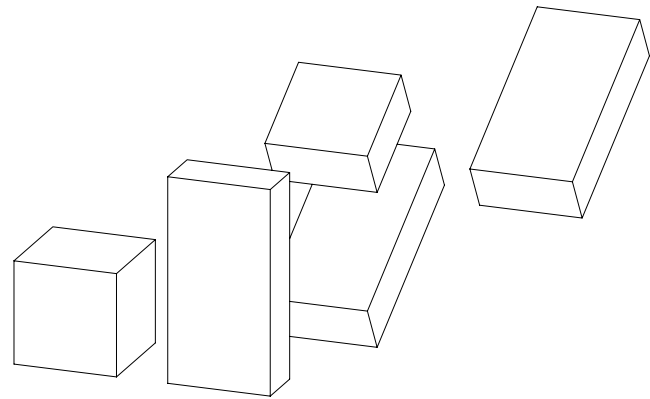
NTR

NTR

Kehrt zur tatsächlichen Nummer von Transformationen zurück.

Beispiel:

```
BLOCK 1, 1, 1
ADDX 2
ADDY 2.5
ADDZ 1.5
ROTX -60
ADDX 1.5
BLOCK 1, 0.5, 2
DEL 1, 1           !Löscht die ADDX 2
Transformation
BLOCK 1, 0.5, 1
DEL 1, NTR() -2    !Löscht die ADDZ 1.5
Transformation
BLOCK 1, 0.5, 2
DEL -2, 3          !Löscht die ROTX -60 and ADDY
2.5 Transformationen
BLOCK 1, 0.5, 2
```



DREIDIMENSIONALE ELEMENTE

In diesem Kapitel werden alle Befehle beschrieben, die in GDL zum Erstellen von 3D Formen dienen. Unter den Befehlen befinden sich Formen von den einfachsten bis zu den kompliziertesten, die aus Linienzügen erstellt werden. Außerdem werden hier Visualisierungselemente (Lichtquellen, Bilder) vorgestellt, wie auch die Definition von Texten, die dreidimensional dargestellt werden sollen. Darüber hinaus werden die Grundelemente der internen 3D-Datenstruktur, die aus Eckpunkten, Vektoren, Kanten und Körpern besteht, detailliert behandelt. Darauf folgt die Interpretation binärer Daten und Richtlinien für die Verwendung von Schnittflächen.

GRUNDKÖRPER

Block

BLOCK a , b , c

BRICK

BRICK a , b , c

Der erste Eckpunkt eines Blocks liegt im lokalen Nullpunkt. Die Kanten mit den Längen a , b und c sind entlang der x -, y und z -Achse ausgerichtet.

.

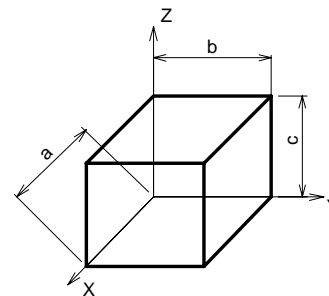
Nullwerte erzeugen Sonderformen eines Quaders (Rechteck, Linie).

Einschränkung der Parameter:

a , b , $c \geq 0$

CYLIND

CYLIND h , r



Ein senkrechter Zylinder, axial zur z-Achse mit der Höhe h und dem Radius r .

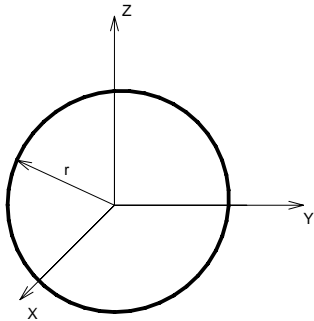
Ist $h = 0$, dann wird ein Kreis in der x-y-Ebene erzeugt.

Ist $r = 0$, dann wird eine Linie entlang der z-Achse erzeugt.

SPHERE

SPHERE r

Eine Kugel mit dem Mittelpunkt im Nullpunkt und dem Radius r .



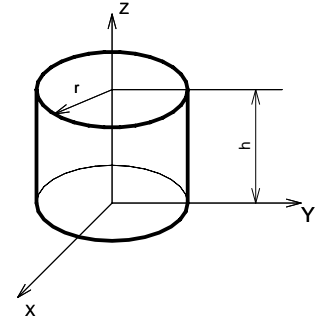
ELLIPS

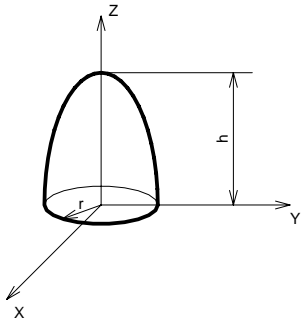
ELLIPS h, r

Halbiertes Ellipsoid. Die Grundfläche ist ein Kreis mit dem Radius r und liegt in der x-y-Ebene. Der Mittelpunkt liegt im lokalen Ursprung. Die Höhe h wird entlang der z-Achse abgetragen.

Beispiel:

ELLIPS r, r ! Halbkugel





CONE

CONE $h, r1, r2, \alpha1, \alpha2$

Kegelstumpf, wobei die Winkel $\alpha1$ und $\alpha2$ die Neigung der Schnittflächen zur z-Achse angeben; $r1$ und $r2$; von h wird die Deckfläche des Prismas unterhalb der x-y-Achse erzeugt.

Ist $h = 0$, werden die Werte von $\alpha1$ und $\alpha2$ vernachlässigt und es entstehen zwei konzentrische Kreise in der x-y-Ebene.

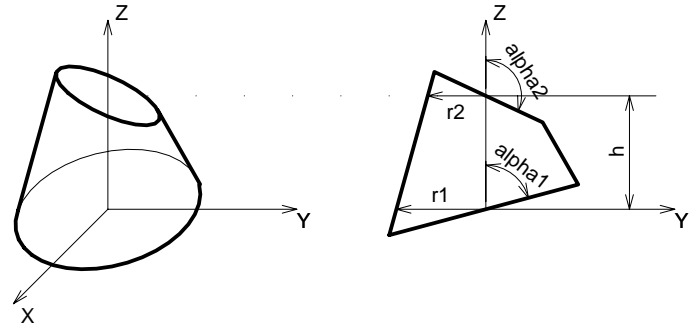
$\alpha1, \alpha2$ werden in Grad angegeben.

Einschränkung der Parameter:

$0 < \alpha1 < 180^\circ$ and $0 < \alpha2 < 180^\circ$

Beispiel:

CONE $h, r, 0, 90, 90$! spitzer Kegel



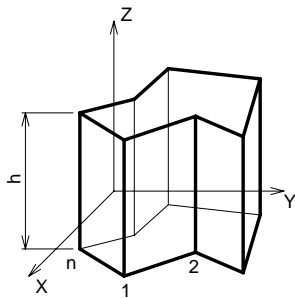
PRISM

PRISM $n, h, x1, y1, \dots, xn, yn$

Senkrecht Prisma, dessen Grundfläche ein Polygon in der x-y-Ebene ist (vgl. die Parameter von ["POLY" auf Seite 63](#) und ["POLY_" auf Seite 63](#)). Die Höhe entlang der z-Achse ist der absolute Wert von h . Für h können auch negative Werte eingesetzt werden. In diesem Fall wird die Deckfläche des Prismas unterhalb der x-y-Achse erzeugt.

Einschränkung der Parameter:

$n \geq 3$



PRISM_

PRISM_ *n*, *h*, *x1*, *y1*, *s1*, ... *xn*, *yn*, *sn*

Ähnlich der CPRISM_ Anweisung, beliebige horizontale Kanten und Seiten können jedoch vernachlässigt werden.

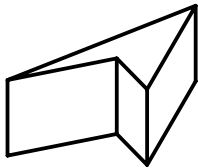
Einschränkung der Parameter:

n >= 3

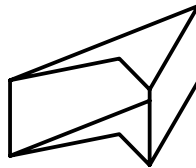
si: Statuscode, der das Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen ermöglicht. Sie können mithilfe spezieller Beschränkungen auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

Siehe *“Statuscodes” auf Seite 141 für mehr Details.*

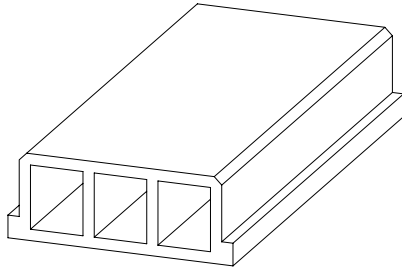
Beispiele:



```
PRISM_ 4,1,
0,0,15,
1,1,15,
2,0,15,
1,3,15
```



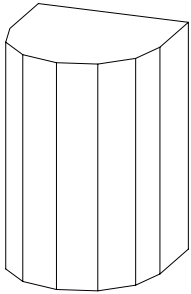
```
PRISM_ 4,1,
0,0,7,
1,1,5,
2,0,15,
1,3,15
```



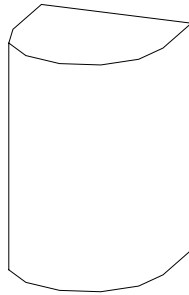
```

ROTX 90
PRISM_ 26, 1.2,
    0.3, 0, 15,
    0.3, 0.06, 15,
    0.27, 0.06, 15,
    0.27, 0.21, 15,
    0.25, 0.23, 15,
    -0.25, 0.23, 15,
    -0.27, 0.21, 15,
    -0.27, 0.06, 15,
    -0.3, 0.06, 15,
    -0.3, 0, 15,
    0.3, 0, -1,
    0.10, 0.03, 15,
    0.24, 0.03, 15,
    0.24, 0.2, 15,
    0.10, 0.2, 15,
    0.10, 0.03, -1,
    0.07, 0.03, 15,
    0.07, 0.2, 15,
    -0.07, 0.2, 15,
    -0.07, 0.03, 15,
    0.07, 0.03, -1,
    -0.24, 0.03, 15,
    -0.24, 0.2, 15,
    -0.1, 0.2, 15,
    -0.1, 0.03, 15,
    -0.24, 0.03, -1
                                !End of contour
                                !End of first hole
                                !End of second hole
                                !End of third hole

```



j7 = 0



j7 = 1

R=1

H=3

```
PRISM      9,      H,
  -R,      R,      15,
  COS(180)*R, SIN(180)*R, 15,
  COS(210)*R, SIN(210)*R, 15,
  COS(240)*R, SIN(240)*R, 15,
  COS(270)*R, SIN(270)*R, 15,
  COS(300)*R, SIN(300)*R, 15,
  COS(330)*R, SIN(330)*R, 15,
  COS(360)*R, SIN(360)*R, 15,
  R,      R,      15
```

ADDDX 5

```
PRISM      9,      H,
  -R,      R,      15,
  COS(180)*R, SIN(180)*R, 64+15,
  COS(210)*R, SIN(210)*R, 64+15,
  COS(240)*R, SIN(240)*R, 64+15,
  COS(270)*R, SIN(270)*R, 64+15,
  COS(300)*R, SIN(300)*R, 64+15,
  COS(330)*R, SIN(330)*R, 64+15,
  COS(360)*R, SIN(360)*R, 64+15,
  R,      R,      15
```

CPRISM_

CPRISM `top_material, bottom_material, side_material,`
`n, h, x1, y1, s1, ... xn, yn, sn`

Erweiterung des Befehles PRISM_. Die ersten 3 Parameter werden für die Angabe des Materialnamens bzw. -index für die Deck-, Grund- und Seitenflächen verwendet. Die anderen Parameter sind die gleichen wie im oben aufgeführten PRISM_-Befehl.

Einschränkung der Parameter:

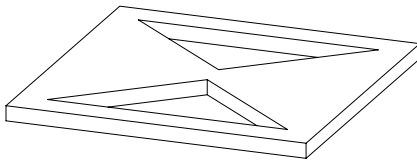
`n` \geq 3

Siehe auch *“Material” auf Seite 162*.

`si`: Statuscode, der das Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen ermöglicht. Sie können mithilfe spezieller Beschränkungen auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

Siehe *“Statuscodes” auf Seite 141 für mehr Details*.

Beispiel:



```
CPRISM_ "Iron", 0, T_,

13, 0.2,
0, 0, 15,
2, 0, 15,
2, 2, 15,
0, 2, 15,
0, 0, -1,
0.2, 0.2, 15,
1.8, 0.2, 15,
1.0, 0.9, 15,
0.2, 0.2, -1,
0.2, 1.8, 15,
1.8, 1.8, 15,
1.0, 1.1, 15,
0.2, 1.8, -1

! "Iron" is a predefined material.
! 0 is a general material.
! T_ is a global variable
! (a material index)

!end of the contour

!end of first hole

!end of second hole
```

BPRISM_

BPRISM_ top_material, bottom_material, side_material,
n, h, radius, x1, y1, s1, ... xn, yn, sn

Ein gleichmäßig gekrümmtes Prisma, das auf derselben Datenstruktur wie das ebene CPRISM_-Element aufbaut. Der einzige zusätzliche Parameter ist die Angabe des Radius.

Die Ableitung erfolgt aus dem entsprechenden CPRISM_-Element durch Krümmen der x-y-Ebene zu einem zu dieser Ebene tangentialen Zylinder. Die Körperkanten entlang der x-Achse werden zu Kreisbögen transformiert. Kanten entlang der y-Achse bleiben horizontal. Die Kanten entlang der z-Achse werden radial ausgerichtet.

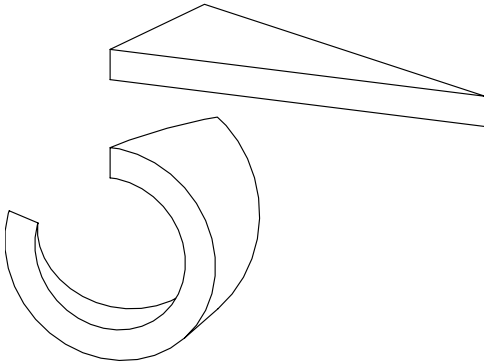
Siehe *"BWALL_"* auf Seite 49 für mehr Details.

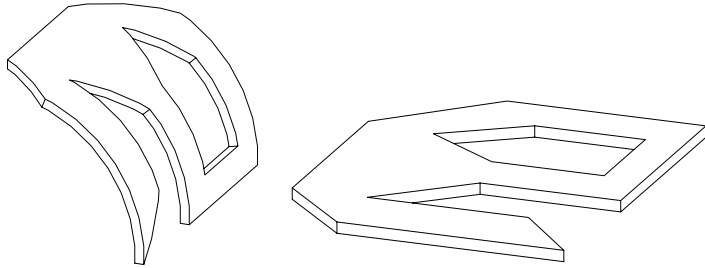
si: Statuscode, der das Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen ermöglicht. Sie können mithilfe spezieller Beschränkungen auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

Siehe *"Statuscodes"* auf Seite 141 für mehr Details.

Beispiele mit den korrespondierenden CPRISM_-Elementen:

```
BPRISM_ "Glass", "Glass", "Glass",  
        3, 0.4, 1,                ! radius = 1  
        0, 0, 15,  
        5, 0, 15,  
        1.3, 2, 15
```





```

BPRISM_ "Concrete", "Concrete", "Concrete",
17, 0.3, 5,
0, 7.35, 15,
0, 2, 15,
1.95, 0, 15,
8, 0, 15,
6.3, 2, 15,
2, 2, 15,
4.25, 4, 15,
8, 4, 15,
8, 10, 15,
2.7, 10, 15,
0, 7.35, -1,
4, 8.5, 15,
1.85, 7.05, 15,
3.95, 5.6, 15,
6.95, 5.6, 15,
6.95, 8.5, 15,
4, 8.5, -1

```

FPRISM_

```

FPRISM_ top_material, bottom_material, side_material, hill_material,
n, thickness, angle, hill_height,
x1, y1, s1,
...
xn, yn, sn

```

Der Aufbau dieses Befehles gleicht dem PRISM_-Befehl mit zusätzlichen Parametern von hill_material, angle und hill_height. Eine Anhöhe wird der Deckfläche des rechten Prismas hinzugefügt.

hill_material: das Seitenmaterial der Anhöhe

angle: der Neigungswinkel der Seitenkanten der Anhöhe. Einschränkung: $0 = \text{winkel} < 90$. Ist angle= 0, bilden die Seitenkanten der Anhöhe, aus einem orthogonalen Sichtpunkt gesehen, einen Viertelkreis mit einem durch die RESOL-Anweisung bestimmten Glättungsgrad. (siehe die Befehle: *"RADIUS" auf Seite 154*, *"RESOL" auf Seite 155* und *"TOLER" auf Seite 156*).

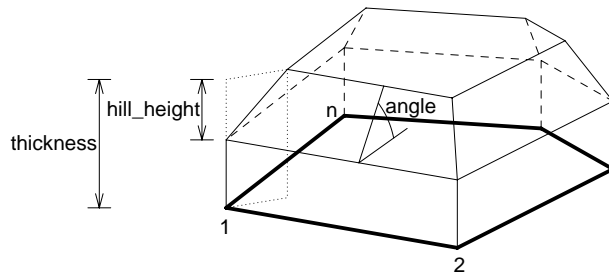
hill_height: Die Höhe der Anhöhet. Merken Sie sich, daß die Parameter der Stärke die totale Höhe von FPRISM_ bestimmen.

Einschränkung der Parameter:

$n \Rightarrow 3$, anhöhe_höhenwert < stärken >>

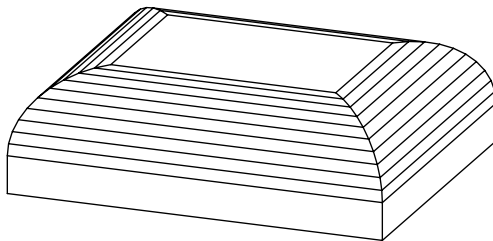
si: Statuscode, der das Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen ermöglicht. Sie können mithilfe spezieller Beschränkungen auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

Siehe "Statuscodes" auf Seite 141 für mehr Details.



Beispiele:

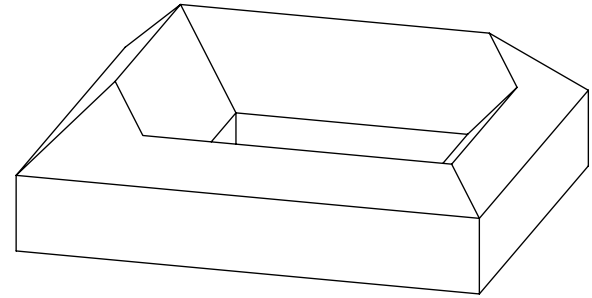
```
RESOL 10
FPRISM_ "Roof Tile", "Red Brick", "Face brick", "Roof Tile",
  4, 1.5, 0, 1.0, !angle = 0
  0, 0, 15,
  5, 0, 15,
  5, 4, 15,
  0, 4, 15
```



```

FPRISM_ "Roof Tile", "Red Brick", "Face brick",
    "Roof Tile",
    10, 2, 45, 1,
    0, 0, 15,
    6, 0, 15,
    6, 5, 15,
    0, 5, 15,
    0, 0, -1,
    1, 2, 15,
    4, 2, 15,
    4, 4, 15,
    1, 4, 15,
    1, 2, -1

```



HPRISM_

```

HPRISM_ top_material, bottom_material, side_
material, hill_material,
n, thickness, angle, hill_height,
x1, y1, s1,
...
xn, yn, sn

```

Ähnlich wie FPRISM_ mit einem zusätzlichen Parameter, der die Sichtbarkeit der Hügelkanten steuert.

status: steuert die Sichtbarkeit der Hügelkanten:

0: alle Hügelkanten sind sichtbar (FPRISM_)

1: Hügelkanten sind unsichtbar

SPRISM_

```

SPRISM_ top_material, bottom_material, side_material,
n, xb, yb, xe, ye, h, angle,
x1, y1, s1, ... xn, yn, sn

```

Erweiterung des Befehles CPRISM_ mit der Option, ein mit der x-y-Ebene nicht paralleles oberes Polygon zu haben. Die Definition der oberen Ebene ist ähnlich der Ebenedefinition des CROOF_-Befehls. Die Höhe des Prismas wird durch die Referenzlinie bestimmt. Oberes und unteres Polygon dürfen nicht verbunden werden.

Zusätzliche Parameter:

x_b , y_b , x_e , y_e : Start- und Endkoordinaten der Referenzlinie (Vektor),

angle: Der Drehwinkel des Oberpolygons rund um die gegebene, angerichtete Referenzlinie in Grad (entgegen dem Uhrzeigersinn),

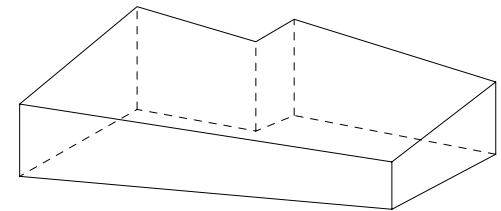
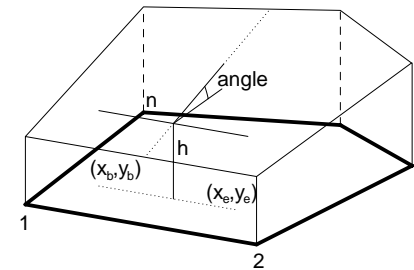
si: Statuscode, der das Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen ermöglicht. Sie können mithilfe spezieller Beschränkungen auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

Siehe *“Statuscodes” auf Seite 141* für mehr Details.

Anmerkung: alle berechneten z-Koordinaten der Eckpunkte des oberen Polygons müssen positiv oder 0 sein.

Beispiel:

```
SPRISM_ 'Grass', 'Earth', 'Earth',
        6,
        0, 0, 11, 6, 2, -10.0,
        0, 0, 15,
        10, 1, 15,
        11, 6, 15,
        5, 7, 15,
        4.5, 5.5, 15,
        1, 6, 15
```



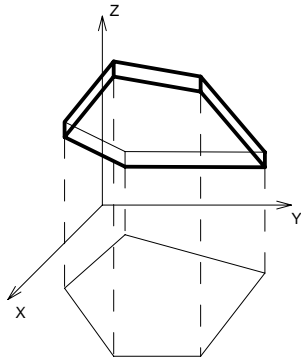
SLAB

SLAB $n, h, x1, y1, z1, \dots, xn, yn, zn$

Geneigtes Prisma. Die Seitenflächen stehen immer im Lot zur x-y-Ebene. Seine Grundfläche ist ein ebenes Polygon, das um eine zur x-y-Ebene parallele Achse gedreht ist. Für h können auch negative Werte eingesetzt werden. In diesem Fall liegt die Deckfläche unterhalb der Grundfläche. Das Programm prüft nicht, ob die Punkte wirklich in einer Ebene liegen. Wenn die Eckpunkte nicht in einer Ebene liegen, resultiert daraus eine fehlerhafte Schattierung.

Einschränkung der Parameter:

$n \geq 3$



SLAB_

SLAB_ $n, h, x1, y1, z1, s1, \dots, xn, yn, zn, sn$

Gleicher Aufbau wie der SLAB-Befehl. Die Sichtbarkeit jeder beliebigen Kante und Oberfläche kann hier unterdrückt werden. Der Befehl wird analog zum Befehl PRISM_ angewendet.

s_i : Statuscode, der das Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen ermöglicht. Sie können mithilfe spezieller Beschränkungen auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

Siehe *“Statuscodes” auf Seite 141* für mehr Details.

CSLAB_

CSLAB_ top_material, bottom_material, side_material,
n, h, x1, y1, z1, s1, ... xn, yn, zn, sn

Erweiterung des SLAB_-Befehles; die ersten 3 Parameter werden für die Angabe des Materialnamens bzw. -index für die Deck-, Boden- und Seitenflächen verwendet. Die anderen Parameter sind dieselben wie im oben aufgeführten SLAB_-Befehl.

Siehe auch *“Material” auf Seite 162* und die *“Spezielle Funktionen” auf Seite 243* > IND Beschreibung der Funktionen *“Verschiedenes” auf Seite 225*.

si: Statuscode, der das Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen. Sie können mithilfe spezieller Beschränkungen auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

Siehe *“Statuscodes” auf Seite 141* für mehr Details.

CWALL_

CWALL_ left_material, right_material, side_material,
height, x1, x2, x3, x4, t,
mask1, mask2, mask3, mask4,
n,
x_start1, y_low1, x_end1, y_high1, frame_shown1,
...
x_startn, y_lown, x_endn, y_highn, frame_shownn,
m,
a1, b1, c1, d1,
...
am, bm, cm, dm

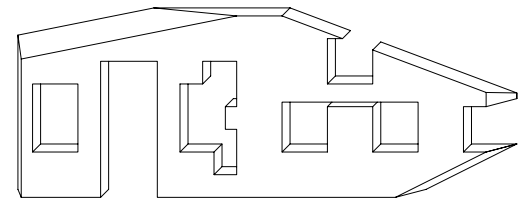
left_material, right_material, side_material:

Materialnamen für die linke, rechte und Aussenseite. (Die linke und die rechte Seiten der Wand folgen der x-Achse.)

Siehe auch *“Material” auf Seite 162* und die *“Spezielle Funktionen” auf Seite 243* > IND Beschreibung der Funktionen.

Die Konstruktionslinie der Wand verläuft immer auf der x- Achse. Die Seitenflächen liegen in der x-z-Ebene.

height: Die Höhe der Wand relativ zur Unterkante.



x_1, x_2, x_3, x_4 : Die auf die x-y-Ebene reflektierten Eckpunkte der Wand (siehe Abb. unten). Ist die Wand freistehend (keine Verschneidung), so sind $x_1 = x_4 = 0$ der Wandanfang und $x_2 = x_3 = \text{Länge der Wand bzw. Wandende}$.

t : Die Wandstärke.

$t < 0$ - der Wandkörper steht rechts der x-Achse,

$t > 0$ - der Wandkörper steht links der x-Achse und der Wandöffnungen,

$t = 0$ die Wand wird als Polygon und die Wandöffnungen als Rahmen dargestellt.

$\text{mask1}, \text{mask2}, \text{mask3}, \text{mask4}$: Bestimmen die Sichtbarkeit der Kanten und Oberflächen.

$\text{maski} = j_1 + 2*j_2 + 4*j_3 + 8*j_4$

hierbei können j_1, j_2, j_3, j_4 jeweils 0 oder 1 sein.

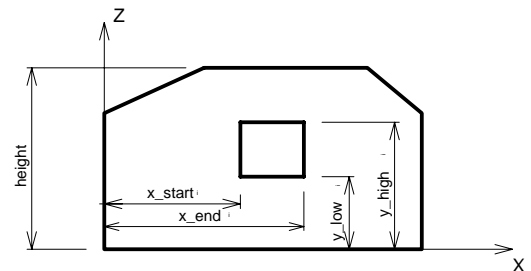
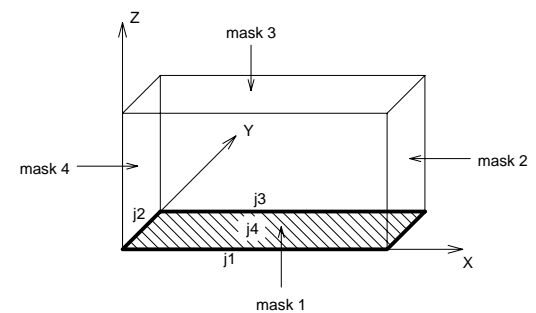
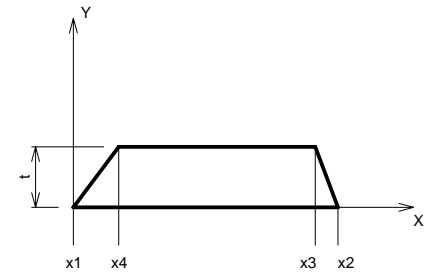
Die Werte j_1, j_2, j_3, j_4 geben an, ob die Kanten und Oberflächen vorhanden sind (1) oder nicht (0).

n : Anzahl der Öffnungen in einer Wand.

$x_{\text{starti}}, y_{\text{lowi}}, x_{\text{endi}}, y_{\text{highi}}$: Koordinaten der Öffnungen (siehe Abb. unten).

frame_showni : Dieser Parameter ist 1, wenn die Konturen einer Öffnung sichtbar sein sollen, ansonsten ist er 0.
Mit Hilfe negativer Werte kann die Sichtbarkeit jeder Kante einer Öffnung kontrolliert werden.

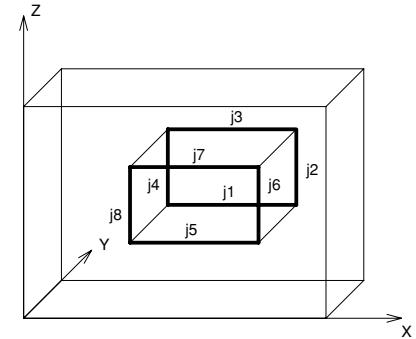
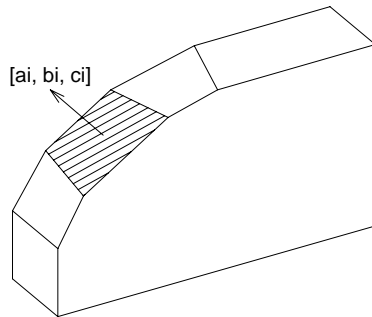
$\text{frame_showni} = -(1*j_1 + j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8)$ wo $j_1, j_2 \dots j_8$ kann entweder 0 oder 1 sein. Die Nummern von j_1 bis j_4 prüfen die Sichtbarkeit der Kanten der Öffnung auf der linken Seite der Wandoberfläche, während j_5 bis j_8 auf die Kanten auf der rechten Seite wirken, wie Sie es in der Abbildung unten sehen.



Eine Kante, die senkrecht zur Wandoberfläche liegt ist sichtbar, wenn sichtbare Kanten von seinen beiden Endpunkten gezeichnet wurden.

m: Anzahl der Schnittflächen.

a_i, b_i, c_i, d_i : Koeffizienten der Gleichung, die die Schnittebene angeben. Weitere definiert $[a_i*x + b_i*y + c_i*z = d_i]$. Die Teile der Wand, die Angaben unter dem Befehl `CWALL_` sich auf der positiven Seite der Schnittebene befinden ($a_i*x + b_i*y + c_i*z > d_i$), werden geschnitten und entfernt. .



BWALL_

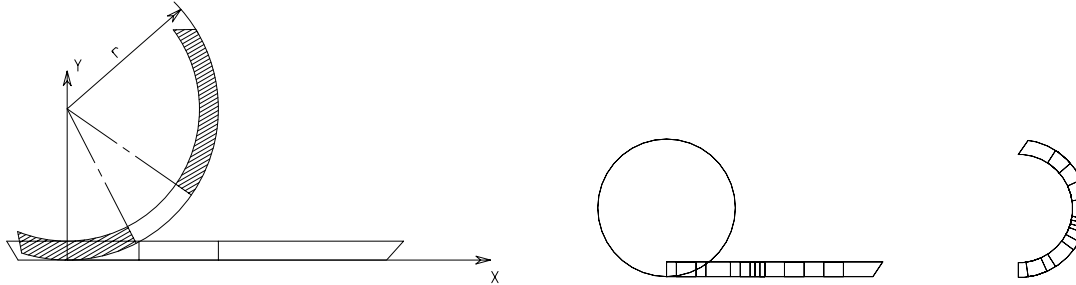
```
BWALL_ left_material, right_material, side_material,
height, x1, x2, x3, x4, t, radius,
mask1, mask2, mask3, mask4,
n,
x_start1, y_low1, x_end1, y_high1, frame_shown1,
...
x_startn, y_lown, x_endn, y_highn, frame_shownn,
m,
a1, b1, c1, d1,
...
am, bm, cm, dm
```

Gleichmäßig gekrümmte Wand mit derselben Datenstruktur wie die mit dem `CWALL_`-Befehl erzeugte gerade Wand. Der einzige zusätzliche Parameter ist die Angabe des Radius.

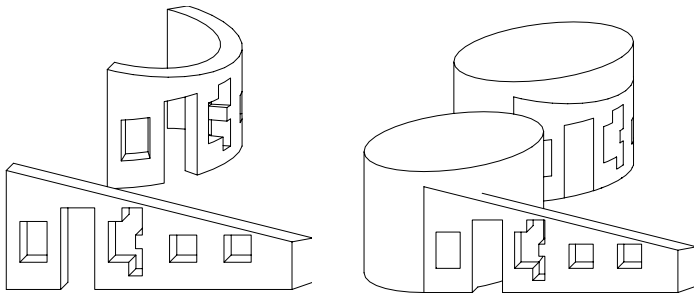
Die Ableitung erfolgt aus dem entsprechenden `CWALL_`-Element durch Krümmen der x-z-Ebene zu einem zu dieser Ebene tangentialen Zylinder. Die Körperkanten entlang der x-Achse werden zu Kreisbögen transformiert. Kanten entlang der y-Achse werden radial ausgerichtet.

Die Annäherung an die Krümmung erfolgt durch eine Anzahl von Segmenten, die wie bei Kugeln und Zylindern durch die aktuelle Auflösung bestimmt wird. (siehe die Befehle: *"RADIUS" auf Seite 154*, *"RESOL" auf Seite 155* und *"TOLER" auf Seite 156*).

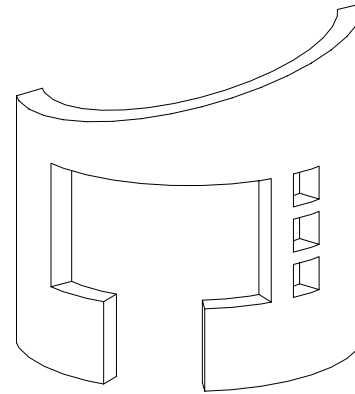
Siehe *"CWALL_" auf Seite 47* für mehr Details.



Beispiele: *BWALL_* und die entsprechende *CWALL_*.




```
ROTZ -60  
BWALL_ 1, 1, 1,  
        4, 0, 6, 6, 0,  
        0.3, 2,  
        15, 15, 15, 15,  
        5,  
        1, 1, 3.8, 2.5, -255,  
        1.8, 0, 3, 2.5, -255,  
        4.1, 1, 4.5, 1.4, -255,  
        4.1, 1.55, 4.5, 1.95, -255,  
        4.1, 2.1, 4.5, 2.5, -255,  
        1, 0, -0.25, 1, 3
```



XWALL_

```

XWALL_ left_material, right_material, vertical_material, horizontal_material,
height, x1, x2, x3, x4,
y1, y2, y3, y4,
t, radius,
log_height, log_offset,
mask1, mask2, mask3, mask4,
n,
x_start1, y_low1, x_end1, y_high1,
frame_shown1,
...
x_startn, y_lown, x_endn, y_highn,
frame_shownn,
m,
a1, b1, c1, d1,
...
am, bm, cm, dm,
status

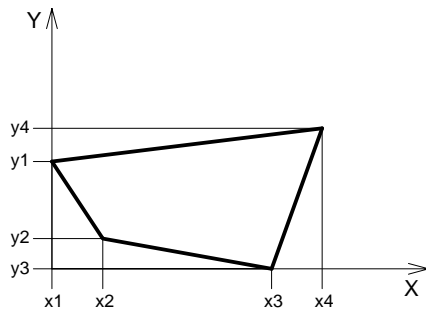
```

Erweiterte Wanddefinition auf der Basis der gleichen Datenstruktur wie das Element XWALL_.

Zusätzliche Parameter:

vertical_material, horizontal_material: Name oder Index der vertikalen/horizontalen Seitenmaterialien.

y1, y2, y3, y4: die projezierten Endpunkte der Wand, die auf 4.1, 1.55, 4.5, 1.95, -255, der x-y-Ebene liegen, wie unten gezeigt.



log_height, log_offset: zusätzliche Parameter für das Zusammensetzen einer Wand aus Blöcken.
Wirkt nur für gerade Wände.

status: steuert das Verhalten einer aus Blöcken zusammengesetzten Wand.

status = j1 + 2*j2 + 4*j3 + 32*j6 + 64*j7 + 128*j8 + 256*j9

j1: rechtes Seitenmaterial auf horizontale Kanten anwenden

j2: linkes Seitenmaterial auf horizontale Kanten anwenden

j3: mit halbem Block beginnen

j6: Textur an Wandkanten ausrichten

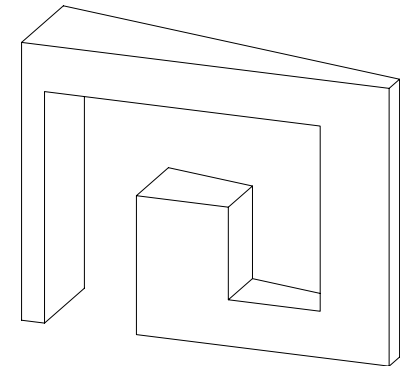
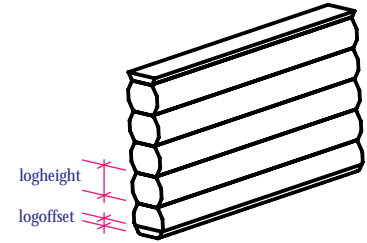
j7: doppelter Radius auf der gekrümmten Seite

j8: quadratischer Block auf der rechten Seite

j9: quadratischer Block auf der linken Seite

Beispiel:

```
XWALL_ "Whitewash", "Whitewash", "Whitewash", "Whitewash",
      3.0,
      0.0, 4.0, 4.0, 0.0,
      0.0, 0.0, 0.3, 1.2,
      1.2, 0.0,
      0.0, 0.0,
      15, 15, 15, 15,
      3,
      0.25, 0.0, 1.25, 2.5, -255,
      1.25, 1.5, 2.25, 2.5, -255,
      2.25, 0.5, 3.25, 2.5, -255, 0
```



XWALL_{2}

```
XWALL_{2} left_material, right_material, vertical_material, horizontal_material,  
height, x1, x2, x3, x4,  
y1, y2, y3, y4,  
t, radius,  
log_height, log_offset,  
mask1, mask2, mask3, mask4,  
n,  
x_start1, y_low1, x_end1, y_high1,  
sill_depth1, frame_shown1,  
...  
x_startn, y_lown, x_endn, y_highn,  
sill_depthn, frame_shownn,  
m,  
a1, b1, c1, d1,  
...  
am, bm, cm, dm,  
status
```

Erweiterte Wanddefinition auf der Basis der gleichen Datenstruktur wie das Element XWALL_.

Zusätzliche Parameter:

silldepthi: logische Tiefe der Öffnungsbrüstung. Wenn das Bit j9 des Parameters frame_showni gesetzt ist, umschließt das Material der Wandseite die Durchbruchpolygone, silldepthi definiert die dazwischen liegende Trennlinie.

$$\text{frame_showni} = -(1*j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8 + 256*j9 + 512*j10)$$

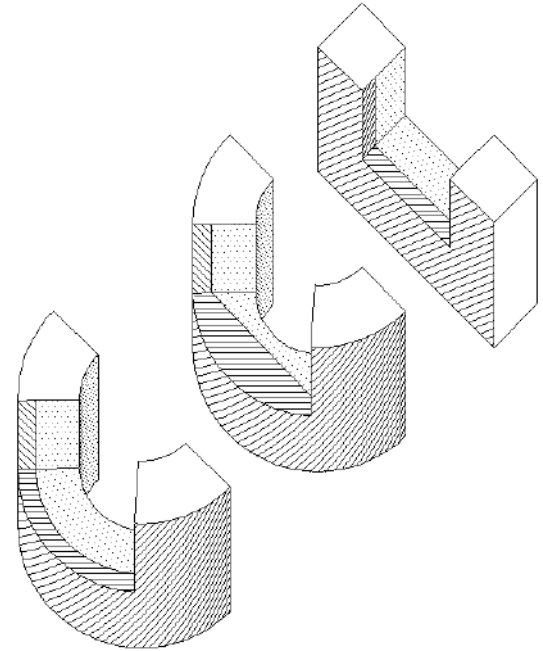
Es gibt zwei zusätzliche Werte zur Steuerung der Materialumschließung. Die Bedeutung der Werte j1, j2 ... j8 ist die gleiche wie die der Befehle CWALL_ und XWALL_. Der Wert j9 steuert das Material der Durchbruchpolygone. Wenn j9 den Wert 1 hat, übernimmt der Durchbruch das Seitenmaterial der Wand. Der Wert j10 steuert die Form der Trennlinie zwischen dem Durchbruchsmaterial der oberen und unteren Polygone des Durchbruchs bei einer gebogenen Wand. Wenn j10 den Wert 1 hat, ist die Trennlinie gerade, andernfalls gekrümmt.

Beispiel:

```

ROTZ 90
xWALL_{2} "C13", "C11", "C12", "C12",
    2, 0, 4, 4, 0,
    0, 0, 1, 1,
    1, 0,
    0, 0,
    15, 15, 15, 15,
    1,
    1, 0.9, 3, 2.1, 0.3, -(255 + 256),
    0,
    0
BODY -1
DEL 1
ADDX 2
xWALL_{2} "C13", "C11", "C12", "C12",
    2, 0, 2 * PI, 2 * PI, 0,
    0, 0, 1, 1,
    0, 0,
    15, 15, 15, 15,
    1,
    1.6, 0.9, 4.6, 2.1, 0.3, -(255 + 256),
    0,
    0
BODY -1
ADDX 4
xWALL_{2} "C13", "C11", "C12", "C12",
    2, 0, 2 * PI, 2 * PI, 0,
    0, 0, 1, 1,
    1, 2,
    0, 0,
    15, 15, 15, 15,
    1,
    1.6, 0.9, 4.6, 2.1, 0.3, -(255 + 256 + 512),
    0,
    0

```



BEAM

BEAM left_material, right_material, vertical_material, top_material, bottom_material,
 height, x1, x2, x3, x4,
 y1, y2, y3, y4, t,
 mask1, mask2, mask3, mask4

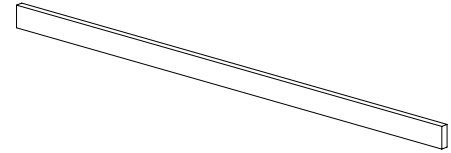
Unterzug Definition Die Parameter sind ähnlich denen des xWALL_-Elements.

Zusätzliche Parameter:

top_material, bottom_material: top and bottom materials

Beispiel:

```
BEAM 1, 1, 1, 1, 1,  
      0.3, 0.0, 7.0, 7.0, 0.0,  
      0.0, 0.0, 0.1, 0.1, 0.5,  
      15, 15, 15, 15
```



CROOF_

```
CROOF_ top_material, bottom_material, side_material,  
        n, xb, yb, xe, ye, height, angle, thickness,  
        x1, y1, alpha1, s1,  
        ...,  
        xn, yn, alphan, sn
```

Dachneigung mit beliebigen Winkel-Firsten.

top_material, bottom_material, side_material: Name/Index des Deck-, Boden-, und Seitenmaterials.

n: die Anzahl der Eckpunkte des Dachpolygons

xb, yb, xe, ye: Referenzlinie (Vektor).

height: die Höhe des Daches an der Referenzlinie (niedrigere Ebene).

angle: der Drehwinkel der Dachebene rund um die angegebene Referenzlinie in Grad (entgegen dem Uhrzeigersinn).

thickness: die Dachstärke senkrecht zur Dachebene gemessen.

x_i, y_i : die Koordinaten der Eckpunkte des unteren Dachpolygons.

α_{hai} : Der Winkel zwischen der Stirnseite, der zu der i -Kante des Daches gehört, und der Ebene, die senkrecht zur Dachebene ist $-90 < \alpha_{\text{hai}} < 90$. α_{hai} : Wenn man in Richtung auf die Kante des richtig orientierten Dachpolygons blickt, ist der Drehwinkel entgegen dem Uhrzeigersinn positiv.

Die Kanten des Dachpolygons werden dann richtig orientiert, wenn die Kontur in Obenansicht dem Uhrzeigersinn entgegengesetzt wird, und die Öffnungen sich im Uhrzeigersinn befinden.

si: Statuscode, der das Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen ermöglicht. Sie können mithilfe spezieller Beschränkungen auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

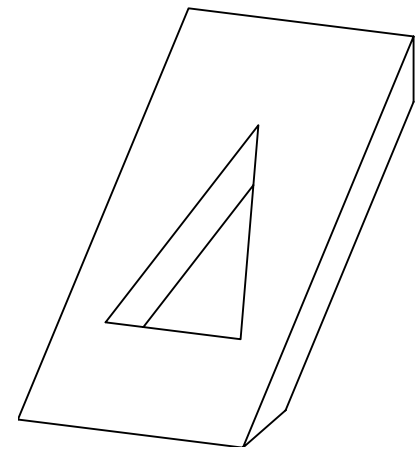
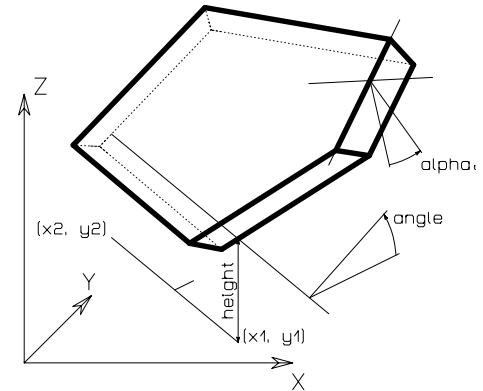
Siehe *“Statuscodes” auf Seite 141* für mehr Details.

Einschränkung der Parameter:

$n \geq 3$

Beispiele:

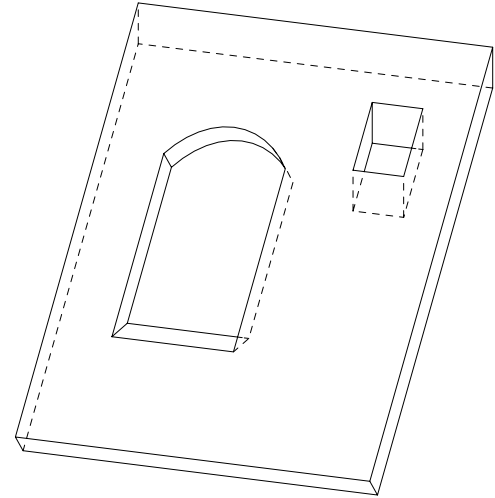
```
CROOF_ 1, 1, 1,           ! materials
      9,
      0, 0,
      1, 0,               ! reference line (xb,yb) (xe,ye)
      0.0,                ! height
      -30,                ! angle
      2.5,                ! thickness
      0, 0, -60, 15,
      10, 0, 0, 15,
      10, 20, -30, 15,
      0, 20, 0, 15,
      0, 0, 0, -1,
      2, 5, 0, 15,
      8, 5, 0, 15,
      5, 15, 0, 15,
      2, 5, 0, -1
```



```

L=0.25
R=(0.6^2+L^2)/(2*L)
A=ASN(0.6/R)
CROOF_ "Roof Tile", "Pine", "Pine",
      16, 2, 0, 0,
      0, 0, 45, -0.2*SQR(2),
      0, 0, 0, 15,
      3.5, 0, 0, 15,
      3.5, 3, -45, 15,
      0, 3, 0, 15,
      0, 0, 0, -1,
      0.65, 1, -45, 15,
      1.85, 1, 0, 15,
      1.85, 2.4-L, 0, 13,
      1.25, 2.4-R, 0, 900,
      0, 2*A, 0, 4015,
      0.65, 1, 0, -1,
      2.5, 2, 45, 15,
      3, 2, 0, 15,
      3, 2.5, -45, 15,
      2.5, 2.5, 0, 15,
      2.5, 2, 0, -1

```



MESH

MESH $a, b, m, n, \text{mask},$
 $z11, z12, \dots, z1m,$
 $z21, z22, \dots, z2m,$
 \dots
 $zn1, zn2, \dots, znm$

Eine räumliche Gitterstruktur, die auf einem Rechteck aufbaut, dessen Deckfläche mit einem gleichmäßigen Netz überzogen ist. Die gekrümmte Oberfläche wird durch Triangulation der Oberflächensegmente erreicht. Die Seiten des Basisrechtecks sind a und b ; die Anzahl an Punkten m und n wird entsprechend an der x - bzw. y -Achse aufgeteilt. z_{ij} ist die Höhe eines Scheitelpunktes.

Maskieren:

$\text{mask} = j1 + 4*j3 + 16*j5 + 32*j6 + 64*j7$

hierbei können $j1, j3, j5, j6, j7$ jeweils 0 oder 1 sein.

$j1$ (1): Grundfläche ist vorhanden.

$j3$ (4): Seitenflächen sind vorhanden.

$j5$ (16): Konturen der Grund- und Seitenflächen sind sichtbar.

$j6$ (32): Außenkonturen der Deckfläche sind sichtbar.

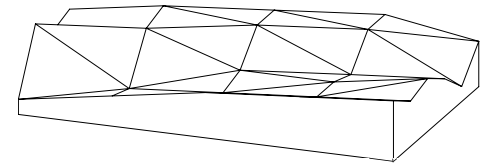
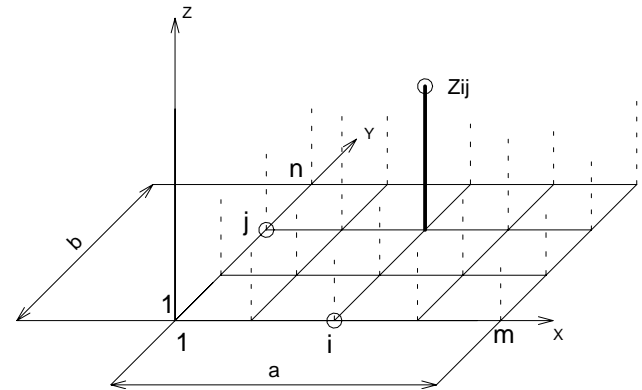
$j7$ (64): Innenkonturen der Deckfläche sind sichtbar, die Oberfläche ist in Segmente aufgeteilt.

Einschränkung der Parameter:

$m \geq 2, n \geq 2$

Beispiele:

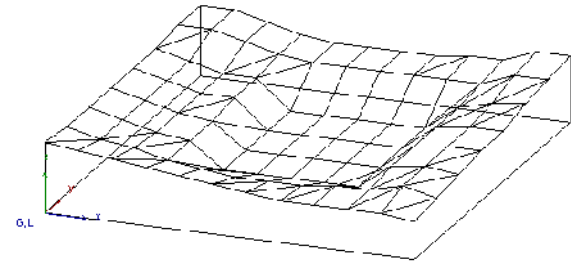
MESH 50, 30, 5, 6, 1+4+16+32+64,
 2, 4, 6, 7, 8,
 10, 3, 4, 5, 6,
 7, 9, 5, 5, 7,
 8, 10, 9, 4, 5,
 6, 7, 9, 8, 2,
 4, 5, 6, 8, 6



```

MESH 90,100, 12,8, 1+4+16+32+64,
 17,16,15,14,13,12,11,10,10,10,10, 9,
 16,14,13,11,10, 9, 9, 9,10,10,12,10,
 16,14,12,11, 5, 5, 5, 5, 5,11,12,11,
 16,14,12,11, 5, 5, 5, 5, 5,11,12,12,
 16,14,12,12, 5, 5, 5, 5, 5,11,12,12,
 16,14,12,12, 5, 5, 5, 5, 5,11,13,14,
 17,17,15,13,12,12,12,12,12,12,15,15,
 17,17,15,13,12,12,12,12,13,13,16,16

```



ARMC

ARMC r1, r2, l, h, d, alpha

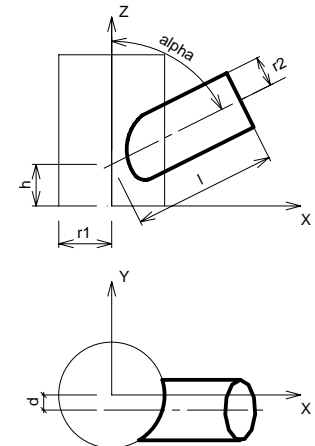
Teil einer Röhre, die an einer anderen Röhre ansetzt; die Verschneidungskurven werden berechnet und dargestellt. Der Winkel alpha wird in Grad angegeben.

Einschränkung der Parameter:

```

r1 >= r2 + d
r1 <= l*sin(alpha) - r2*cos(alpha)

```

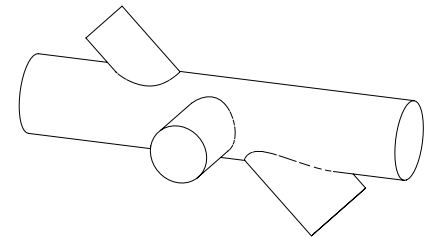


Beispiel:

```

ROTY 90
CYLIND 10,1
ADDZ 6
ARMC 1, 0.9, 3, 0, 0, 45
ADDZ -1
ROTZ -90
ARMC 1, 0.75, 3, 0, 0, 90
ADDZ -1
ROTZ -90
ARMC 1, 0.6, 3, 0, 0, 135

```



ARME

ARME l, r1, r2, h, d

Teil einer Röhre, die an einem Ellipsoid parallel zur z-Achse ansetzt; die Verschneidungskurven werden berechnet und dargestellt. Die Parameter sind aus der Abbildung ersichtlich.

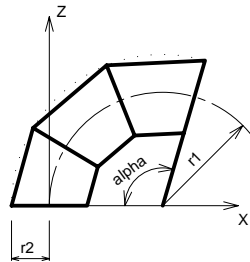
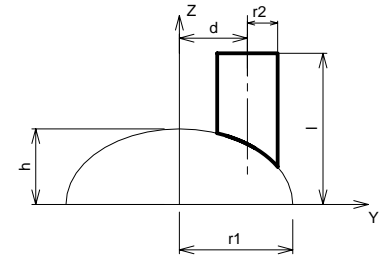
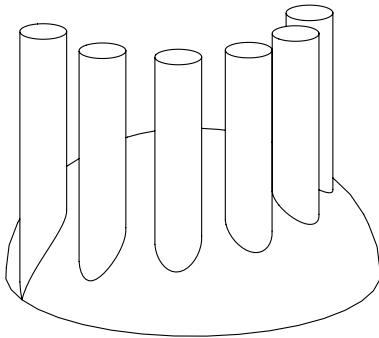
Einschränkung der Parameter:

$$r1 \geq r2 + d$$

$$l \geq h \cdot \sqrt{1 - (r2 - d)^2 / r1^2}$$

Beispiel:

```
ELLIPS 3,4
FOR i=1 TO 6
  ARME 6,4,0.5,3,3.7-0.2*i
  ROTZ 30
NEXT i
```



ELBOW

ELBOW r1, alpha, r2

Gekrümmter Zylinder mit der Mittelachse in der x-z-Ebene. Der Körper ist in Segmente unterteilt. Der Radius der Körperachse ist r1, der Winkel, der die Länge der Mittelachse festlegt, ist alpha und der Radius der Schnittfläche des Körpers bzw. eines Segment-Endstückes ist r2.

Einschränkung der Parameter:

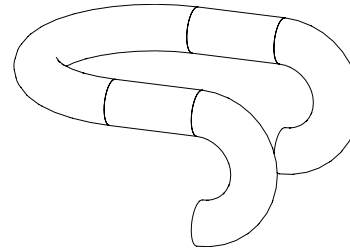
$$r1 > r2$$

Beispiel:

```

ROTY      90
ELBOW     2.5, 180, 1
ADDZ      -4
CYLIND     4, 1
ROTZ     -90
MULZ      -1
ELBOW     5, 180, 1
DEL        1
ADDX       10
CYLIND     4, 1
ADDZ        4
ROTZ       90
ELBOW     2.5, 180, 1

```



FLÄCHEN-GESTALTEN IN 3D

Die in diesem Schnitt vorgestellten Zeichnungselemente können in 3D-Scripts verwendet werden. Mit Hilfe dieser können Sie Punkte, Linien, Bogen, Kreisbogen und planare Polygone im 3D-Raum bestimmen.

HOTSPOT

HOTSPOT *x, y, z* [, *unID* [, *paramReference*, *flags*] [, *displayParam*]]

Ein 3D-Fixpunkt am Punkt (*x, y, z*).

unID ist der eindeutige Kennzeichner des Fixpunkts im 3D-Script. Er ist nützlich, wenn Sie eine variable Anzahl von Fixpunkten haben.

paramReference: Parameter, der von diesem Hotspot mit der Parameter-Bearbeitungsmethode auf der Basis grafischer Hotspots bearbeitet werden kann.

paramReference: Parameter, der durch diesen Fixpunkt über die Methode Parameterbearbeitung mithilfe grafischer Fixpunkte bearbeitet werden kann. Es können auch Elemente von Arrays übergeben werden.

Siehe *“Grafische Bearbeitung” auf Seite 135* über die Verwendung von HOTSPOT.

LIN_

LIN_ *x1, y1, z1, x2, y2, z2*

Erzeugt eine Linie zwischen den Eckpunkten P1(*x1,y1,z1*) und P2(*x2,y2,z2*).

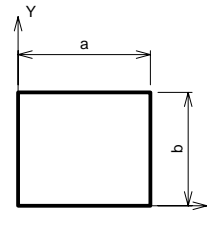
RECT

RECT a, b

Ein Rechteck in der x-y Ebene mit den Seiten a und b .

Einschränkung der Parameter:

$a, b, c \geq 0$



POLY

POLY $n, x_1, y_1, \dots, x_n, y_n$

Ein Polygon mit n Konturlinien in der x-y Ebene. Die Koordinaten der Eckpunkte sind $(x_i, y_i, 0)$.

Einschränkung der Parameter:

$n \geq 3$

POLY_

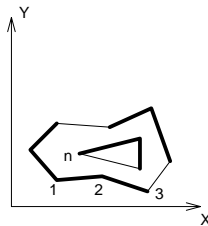
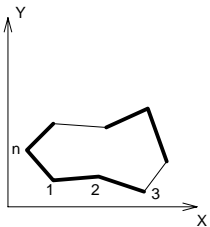
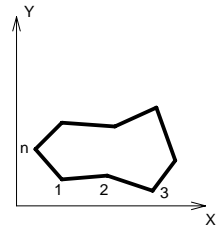
POLY_ $n, x_1, y_1, s_1, \dots, x_n, y_n, s_n$

Gleicht dem einfachen POLY-Befehl, jedoch kann die Sichtbarkeit jeder beliebigen Kante unterdrückt werden. Wenn $s_i = 0$, ist die Kante ausgehend vom Punkt (x_i, y_i) unsichtbar. Wenn $s_i = 1$, wird die Kante dargestellt.

$s_i = -1$ wird zur direkten Definition der Löcher benutzt.

Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie mithilfe spezieller Beschränkungen.

Siehe *“Zusätzliche Statuscodes” auf Seite 143* für mehr Details.



Einschränkung der Parameter:

$n \geq 3$

PLANE

PLANE $n, x_1, y_1, z_1, \dots, x_n, y_n, z_n$

Ein Polygon mit n Kanten in einer beliebigen Ebene. Die Koordinaten des Eckpunktes i sind (x_i, y_i, z_i) . Um ein korrektes Ergebnis beim Rendering und Schattenwurf zu erhalten, müssen die Eckpunkte des Polygons in einer Ebene liegen.

Einschränkung der Parameter:

$n \geq 3$

PLANE_

PLANE_ $n, x_1, y_1, z_1, s_1, \dots, x_n, y_n, z_n, s_n$

Der Aufbau dieses Befehles gleicht dem einfachen PLANE-Befehl, jedoch kann wie im POLY_-Befehl die Sichtbarkeit jeder beliebigen Kante unterdrückt werden.

Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie mithilfe spezieller Beschränkungen.

Siehe *“Zusätzliche Statuscodes” auf Seite 143*.

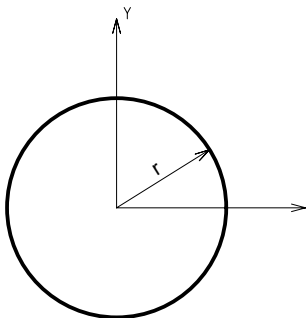
Einschränkung der Parameter:

$n \geq 3$

CIRCLE

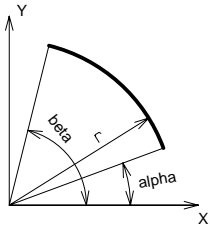
CIRCLE r

Ein Kreis in der x-y Ebene, dessen Mittelpunkt im Nullpunkt liegt; der Kreis hat den Radius r .



Bogen

ARC r, alpha, beta



Ein Kreisbogen (im Drahtmodell) bzw. ein Kreissektor (im Flächen- bzw. Volumenmodell), der durch die Winkel alpha und beta begrenzt ist. Er liegt in der x-y Ebene mit dem Mittelpunkt im lokalen Nullpunkt.

alpha und beta werden in Grad angegeben.

KÖRPER AUS LINIENZÜGEN

Diese Elemente erzeugen komplexe 3D-Körper. Ausgangsbasis dabei sind ein Polygonzug und eine dazugehörige Abbildungsvorschrift (Rotation, Projektion, Transformation). Die daraus resultierenden Körper sind teilweise Weiterführungen beschriebener Körper wie PRISM_ und CYLIND.

Körper, die aus einem Polygonzug generiert werden:

EXTRUDE
PYRAMID
REVOLVE

Körper aus zwei Linienzügen

RULED
SWEEP
TUBE
TUBEA

Der erste Polygonzug liegt immer in der x-y-Ebene. Die Eckpunkte werden über zwei Koordinaten bestimmt, der dritte Wert gibt den Status an (siehe unten). Der zweite Polygonzug (für RULED und SWEEP) ist eine Kurve im Raum. Ihre Eckpunkte werden über drei Koordinaten bestimmt..

Körper, der aus vier Polygonzügen generiert wird:

COONS

Körper, der aus beliebigen Anzahl von Polygonzügen generiert wird:

MASS

Generelle Einschränkungen für Polygonzüge:

- Benachbarte Eckpunkte dürfen sich nicht überlagern (ausgenommen RULED).
- Linienzüge dürfen sich nicht überschneiden. (Das Programm überprüft diese Bedingung nicht, das 3D-Modell wird jedoch fehlerhaft dargestellt).
- Polygonzüge können entweder offen oder geschlossen sein. Im zweiten Fall müssen End- und Anfangspunkt übereinstimmen.

Mask-Werte

Mask-Werte werden verwendet, um charakteristische Oberflächen und/oder Kanten des 3D-Körpers zu zeigen oder zu verbergen. Mask-Werte haben für jedes Element ihre spezifischen Eigenschaften. Genauere Angaben dazu finden Sie bei dem jeweiligen Elementtyp.

$\text{mask} = j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7$

hierbei können $j1, j2, j3, j4, j5, j6, j7$ 0 oder 1 sein.

$j1, j2, j3, j4$ geben an, ob die Oberflächen dargestellt (1) oder unterdrückt (0) werden.

$j5, j6, j7$ geben an, ob die Körperkanten sichtbar (1) oder unsichtbar (0) sind.

$j1$: Grundfläche.

$j2$: Deckfläche.

$j3$: Seitenfläche.

$j4$: weitere Seitenfläche.

$j5$: Konturen der Grundfläche.

$j6$: Konturen der Deckfläche.

$j7$: Konturen der Schnittfläche/Oberfläche sind sichtbar, die Oberfläche ist segmentiert.

Um alle Konturen und Oberflächen sichtbar zu machen, setzen Sie den Mask-Wert auf 127.

Status-Werte

Die Status-Werte ermöglichen die Darstellung der vorgeschriebenen Positionsänderung eines gegebenen Eckpunktes im Polygonzug. Der Weg, den der Punkt dabei beschrieben hat, kann je nach Status-Wert als Körperkontur dargestellt oder unterdrückt werden.

0: Vom Punkt ausgehende Kreisbögen/Körperkanten sind sichtbar.

1: Vom Punkt ausgehende Kreisbögen/Körperkanten werden bei der 3D-Darstellung nur für die Konturensuche berücksichtigt.

-1: nur für den EXTRUDE-Befehl: dieser Wert kennzeichnet den Endpunkt des umschließenden Polygons bzw. der Öffnung. Der folgende Wert wird dabei automatisch der Startpunkt der nächsten Öffnung.

Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der Polylinie mithilfe spezieller Beschränkungen.

Siehe *“Zusätzliche Statuscodes” auf Seite 143 für mehr Details.*

Um einen gleichmäßigen 3D-Körper zu erzeugen, setzen Sie alle Status-Werte auf 1. Der Status-Wert 0 erzeugt eine segmentierte Oberfläche. Andere Werte sind für spätere Weiterentwicklungen reserviert.

EXTRUDE

EXTRUDE *n*, *dx*, *dy*, *dz*, *mask*, *x1*, *y1*, *s1*,
..., *xn*, *yn*, *sn*

Allgemeines, durch ein Polygon der x-y-Ebene definiertes Prisma. Der Verschiebungsvektor aus der Grundfläche ist (*dx*, *dy*, *dz*).

Dieser Befehl ist eine Verallgemeinerung der PRISM- und SLAB-Anweisungen. Der Basispolygonzug muß nicht geschlossen sein, die Seitenkanten müssen nicht im Lot zur x-y-Ebene stehen. Die Grundfläche kann wie beim PRISM_-Befehl Öffnungen enthalten. Die Sichtbarkeit der Kanten kann kontrolliert werden.

n: Anzahl der Eckpunkte des Basispolygonzuges.

mask: kontrolliert die Existenz der Grund-, Deck- und (bei einem offenen Polygonzug) Seitenfläche..

si: status Status der seitlichen Kanten oder Kennzeichen für das Ende eines Polygons oder einer Öffnung. Sie können mit zusätzlichen Statuscodewerten auch Bögen und Segmente in der Polylinie definieren.

Einschränkung der Parameter::

n > 2

Mask-Werte

mask = *j1* + 2**j2* + 4**j3* + 16**j5* + 32**j6* wobei *j1*, *j2*, *j3*, *j5*, *j6* 0 oder 1 sein können.

j1 (1): Grundfläche ist vorhanden.

j2 (2): Deckfläche ist vorhanden.

j3 (4): Die das Prisma schließende) Seitenfläche ist vorhanden.

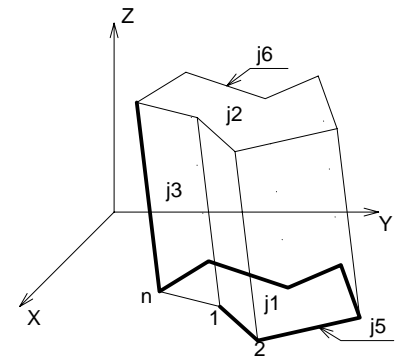
j5 (16): Konturen der Grundfläche sind vorhanden.

j6 (32): Außenkonturen der Deckfläche sind sichtbar.

Statuswerte:

0: Seitenkanten, ausgehend vom Punkt sind sichtbar.

1: Seitenkanten, ausgehend vom Punkt werden nur für die Konturensuche im 3D-Modell berücksichtigt.



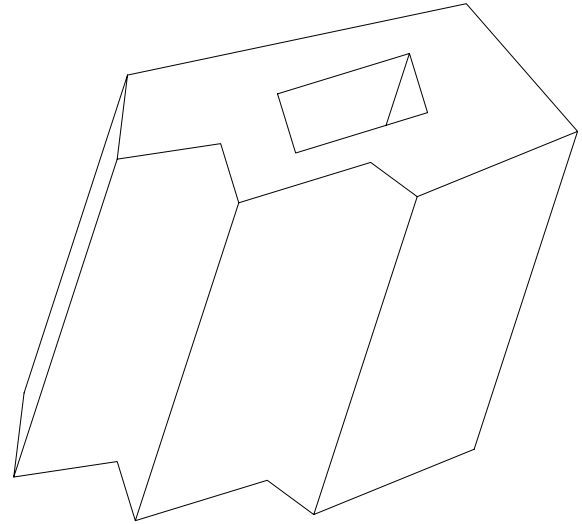
-1: kennzeichnet das Ende des umschließenden Polygons oder der Öffnung. Der folgende Eckpunkt ist dann automatisch der Startpunkt der nächsten Öffnung.

Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie mithilfe spezieller Beschränkungen.

Siehe *“Zusätzliche Statuscodes” auf Seite 143 für mehr Details.*

Beispiele:

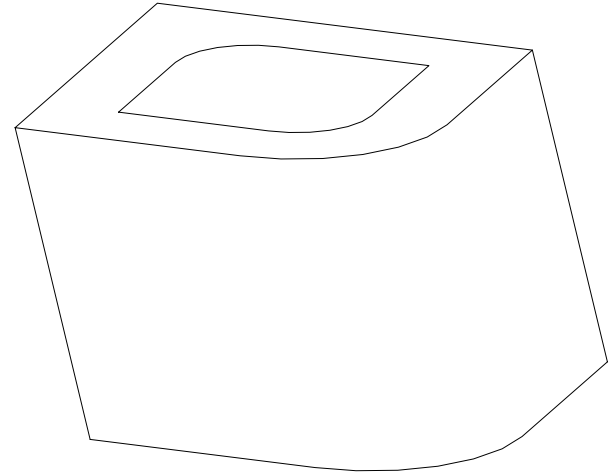
```
EXTRUDE 14, 1, 1, 4, 1+2+4+16+32,
0, 0, 0,
1, -3, 0,
2, -2, 1,
3, -4, 0,
4, -2, 1,
5, -3, 0,
6, 0, 0,
3, 4, 0,
0, 0, -1,
2, 0, 0,
3, 2, 0,
4, 0, 0,
3, -2, 0,
2, 0, -1
```



```

A=5: B=5
R=2: S=1
C=R-S
D=A-R
E=B-R
EXTRUDE 28, -1, 0, 4, 1+2+4+16+32,
0, 0, 0,
D+R*SIN(0), R-R*COS(0), 1,
D+R*SIN(15), R-R*COS(15), 1,
D+R*SIN(30), R-R*COS(30), 1,
D+R*SIN(45), R-R*COS(45), 1,
D+R*SIN(60), R-R*COS(60), 1,
D+R*SIN(75), R-R*COS(75), 1,
D+R*SIN(90), R-R*COS(90), 1,
A, B, 0,
0, B, 0,
0, 0, -1,
C, C, 0,
D+S*SIN(0), R-S*COS(0), 1,
D+S*SIN(15), R-S*COS(15), 1,
D+S*SIN(30), R-S*COS(30), 1,
D+S*SIN(45), R-S*COS(45), 1,
D+S*SIN(60), R-S*COS(60), 1,
D+S*SIN(75), R-S*COS(75), 1,
D+S*SIN(90), R-S*COS(90), 1,
A-C, B-C, 0,
R-S*COS(90), E+S*SIN(90), 1,
R-S*COS(75), E+S*SIN(75), 1,
R-S*COS(60), E+S*SIN(60), 1,
R-S*COS(45), E+S*SIN(45), 1,
R-S*COS(30), E+S*SIN(30), 1,
R-S*COS(15), E+S*SIN(15), 1,
R-S*COS(0), E+S*SIN(0), 1,
C, C, -1

```



PYRAMID

PYRAMID $n, h, \text{mask}, x_1, y_1, s_1, \dots, x_n, y_n, s_n$

Eine Pyramide, deren Basis ein Polygonzug in der x-y-Ebene ist. Die Spitze der Pyramide ist durch $(0, 0, h)$ festgelegt.

n : Anzahl der Eckpunkte des Polygonzuges.

mask : Kontrolliert die Existenz der Grund- und Seitenfläche (bei einem offenen Polygonzug).

s_i : Status der seitlichen Kanten.

Einschränkung der Parameter:

$h > 0$ und $n > 2$

Maskieren:

$\text{mask} = j_1 + 4*j_3 + 16*j_5$

wobei j_1, j_3, j_5 0 oder 1 sein können.

j_1 (1): Grundfläche ist vorhanden.

j_3 (4): Die (das Prisma schließende) Seitenfläche ist vorhanden.

j_5 (16): Konturen der Grundfläche sind vorhanden.

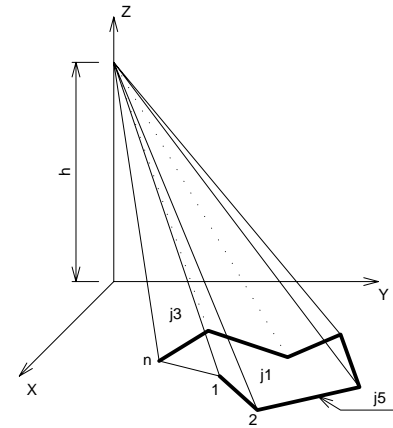
Statuswerte:

0: Seitenkanten, vom Eckpunkt ausgehend sind alle sichtbar.

1: Seitenkanten, ausgehend vom Punkt werden nur für die Konturensuche im 3D-Modell berücksichtigt.

Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie mithilfe spezieller Beschränkungen.

Siehe *“Zusätzliche Statuscodes”* auf Seite 143 für mehr Details.

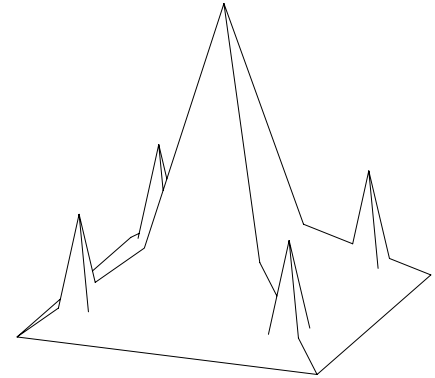


Beispiel:

```

PYRAMID 4, 1.5, 1+4+16,
        -2, -2, 0,
        -2, 2, 0,
        2, 2, 0,
        2, -2, 0
PYRAMID 4, 4, 21,
        -1, -1, 0,
        1, -1, 0,
        1, 1, 0,
        -1, 1, 0
ADDX -1.4
ADDY -1.4
GOSUB 100
ADDX 2.8
GOSUB 100
ADDY 2.8
GOSUB 100
ADDX -2.8
GOSUB 100
END
100:
PYRAMID 4, 1.5, 21,
        -0.25, -0.25, 0,
        0.25, -0.25, 0,
        0.25, 0.25, 0,
        -0.25, 0.25, 0
RETURN

```



REVOLVE

REVOLVE *n*, *alpha*, *mask*, *x1*, *y1*, *s1*, ... *xn*, *yn*, *sn*

Körper, der durch Rotation eines in der x-y-Ebene offenen Linienzuges um die x-Achse erzeugt wird.

n: Anzahl der Eckpunkte des Polygonzuges.

alpha: Drehwinkel. Angabe erfolgt in Grad.

mask: Kontrolliert die Existenz der Grund-, Deck- und (falls $\alpha < 360$) Seitenflächen..

si: Status der Begrenzungsbögen.

Einschränkung der Parameter:

$n \geq 2$

$y_i \geq 0.0$

y_i und y_{i+1} (d.h. der y-Wert zwei benachbarter Eckpunkte) dürfen nicht gleichzeitig Null sein.

Maskieren:

$mask = j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7$

hierbei können $j1, j2, j3, j4, j5, j6, j7$ can be 0 oder 1 sein.

$j1$ (1): Grundfläche ist vorhanden.

$j2$ (2): Deckfläche ist vorhanden.

$j3$ (4): Seitenfläche am Startwinkel ist vorhanden.

$j4$ (8): Seitenfläche am Endwinkel ist vorhanden.

$j5$ (16): Kanten der Seitenfläche (Startwinkel) sind sichtbar.

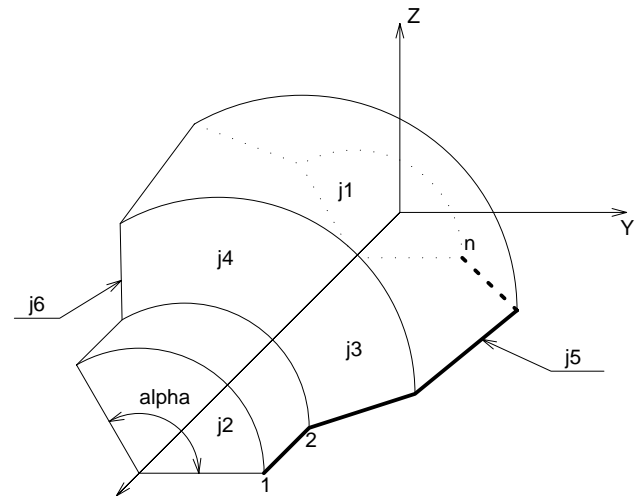
$j6$ (32): Kanten der Seitenfläche (Endwinkel) sind sichtbar.

$j7$ (64): Oberflächenkonturen sind sichtbar, die Oberfläche ist in Segmente aufgeteilt.

Statuswerte:

0: Kreisbögen, vom Eckpunkt ausgehend sind sichtbar.

1: Kreisbögen, vom Eckpunkt ausgehend werden nur für die Konturensuche im 3D-Modell berücksichtigt.

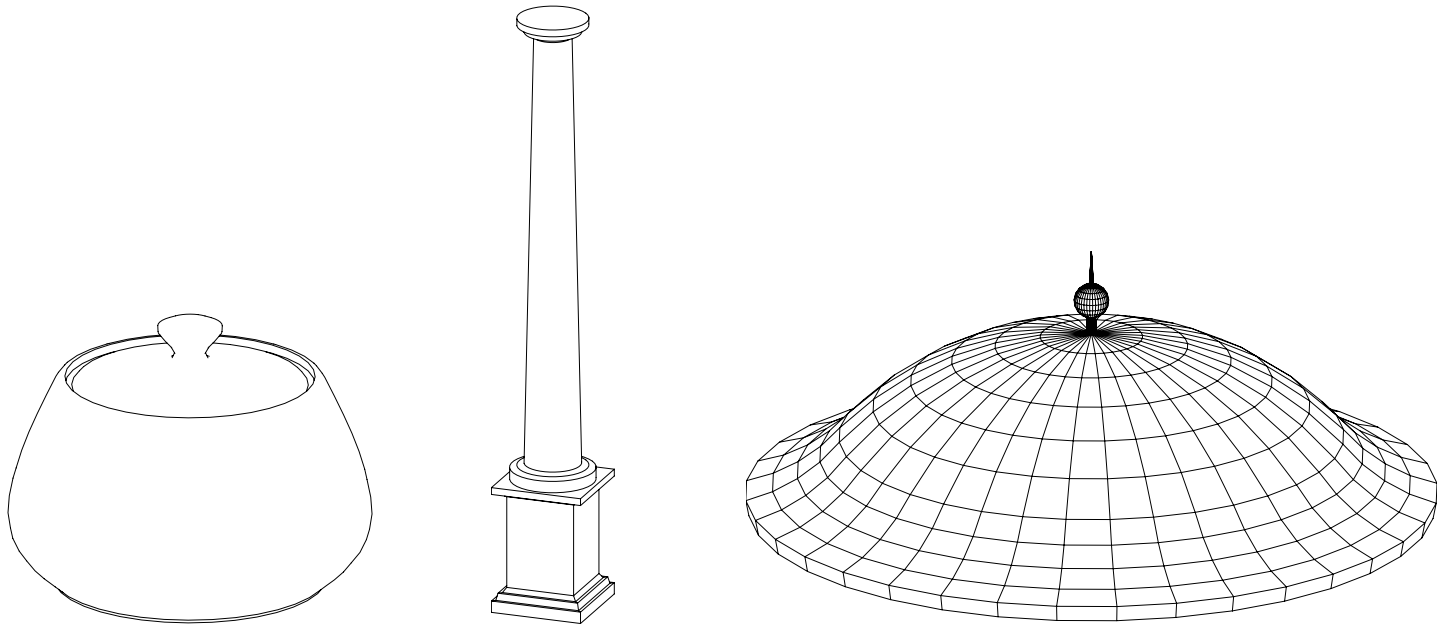


2: Wenn man ArchiCAD/ArchiFM oder Z-Buffer Rendering Engine nutzt und diese auf Glatte Oberflächen stellt, definiert die, zu diesem Punkt gehörige, kreisförmige Kante, eine Pause. Diese Lösung ist mit der Bestimmung von zusätzlichen Punkten gleichwertig. Die Berechnung wird vom Compiler ausgeführt. Andere photorealistische Methoden haben denselben Effekt, als würde man 0 verwenden.

Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie mithilfe spezieller Beschränkungen.

Siehe *“Zusätzliche Statuscodes”* auf Seite 143 für mehr Details.

Beispiele:



ROTY -90

```
REVOLVE 22, 360, 1+64,  
0, 1.982, 0,  
0.093, 2, 0,  
0.144, 1.845, 0,  
0.220, 1.701, 0,  
0.318, 1.571, 0,  
0.436, 1.459, 0,  
0.617, 1.263, 0,  
0.772, 1.045, 0,  
0.896, 0.808, 0,  
0.987, 0.557, 0,  
1.044, 0.296, 0,  
1.064, 0.030, 0,  
1.167, 0.024, 0,  
1.181, 0.056, 0,  
1.205, 0.081, 0,  
1.236, 0.096, 0,  
1.270, 0.1, 0,  
1.304, 0.092, 0,  
1.333, 0.073, 0,  
1.354, 0.045, 0,  
1.364, 0.012, 0,  
1.564, 0, 0
```


Lösung ohne die gleiche Lösung mit Status-Code 2:

```

ROTY -90
REVOLVE 26, 180, 16+32,
7, 1, 0,
6.0001, 1, 1,
6, 1, 0,
5.9999, 1.0002, 1,
5.5001, 1.9998, 1,
5.5, 2, 0,
5.4999, 1.9998, 1,
5.0001, 1.0002, 1,
5, 1, 0,
4.9999, 1, 1,
4.0001, 1, 1,
4, 1, 0,
3+COS(15), 1+SIN(15), 1,
3+COS(30), 1+SIN(30), 1,
3+COS(45), 1+SIN(45), 1,
3+COS(60), 1+SIN(60), 1,
3+COS(75), 1+SIN(75), 1,
3, 2, 1,
3+COS(105), 1+SIN(105), 1,
3+COS(120), 1+SIN(120), 1,
3+COS(135), 1+SIN(135), 1,
3+COS(150), 1+SIN(150), 1,
3+COS(165), 1+SIN(165), 1,
2, 1, 0,
1.9999, 1, 0,
1, 1, 0

```

RULED

```

RULED n, mask,
    u1, v1, s1, ... un, vn, sn,
    x1, y1, z1, ... xn, yn, zn

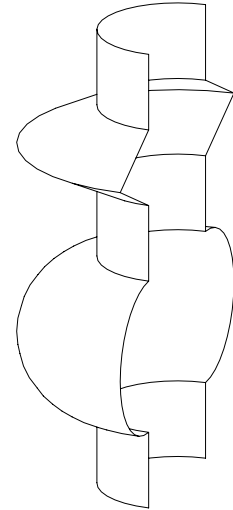
```

Status-Code 2:

```

ROTY -90
REVOLVE 18, 180, 48,
7, 1, 0,
6, 1, 2,
5.5, 2, 2,
5, 1, 2,
4, 1, 2,
3+COS(15), 1+SIN(15), 1,
3+COS(30), 1+SIN(30), 1,
3+COS(45), 1+SIN(45), 1,
3+COS(60), 1+SIN(60), 1,
3+COS(75), 1+SIN(75), 1,
3, 2, 1,
3+COS(105), 1+SIN(105), 1,
3+COS(120), 1+SIN(120), 1,
3+COS(135), 1+SIN(135), 1,
3+COS(150), 1+SIN(150), 1,
3+COS(165), 1+SIN(165), 1,
2, 1, 2,
1, 1, 0

```



RULED{2}

RULED{2} *n*, *mask*,
 u1, *v1*, *s1*, ... *un*, *vn*, *sn*,
 x1, *y1*, *z1*, ... *xn*, *yn*, *zn*

RULED erzeugt einen Körper, der auf einem Polygon der x-y-Ebene und einem frei im Raum definierten Polygon aufbaut. Beide haben die gleiche Anzahl an Eckpunkten. Zwei korrespondierende Punkte der beiden Polygone werden durch Geraden miteinander verbunden.

Dies ist das einzige GDL-Element, bei dem zwei sich überlagernde, benachbarte Punkte möglich sind.

Die zweite Version, RULED{2}, überprüft die Richtung (im oder gegen den Uhrzeigersinn), in welcher die Punkte des Deck- und Grundpolygons definiert wurden und kehrt die Richtung nötigenfalls um. (Der ursprüngliche Befehl RULED rechnet nur mit dem Grundpolygon, dies kann jedoch Fehler verursachen.

n: Anzahl der Eckpunkte in jedem Polygon.

mask: Kontrolliert die Existenz des Grund-, Deck- und abschließenden Seitenpolygons und die Sichtbarkeit der Konturen der beiden Hauptpolygone. Das Seitenpolygon verbindet den ersten und den letzten Eckpunkt der beiden Hauptpolygone, falls eines davon nicht geschlossen sind.

ui, *vi*: Koordinaten des Polygons der x-y-Ebene.

si: Status der seitlichen Kanten.

xi, *yi*, *zi*: Koordinaten der Eckpunkte des zweiten Polygons.

Einschränkung der Parameter::

n > 1

Mask-Werte

mask = *j1* + 2**j2* + 4**j3* + 16**j5* + 32**j6* + 64**j7*
 wobei *j1*, *j2*, *j3*, *j5*, *j6*, *j7* 0 oder 1 sein könnten.

j1 (1): Grundfläche ist vorhanden.

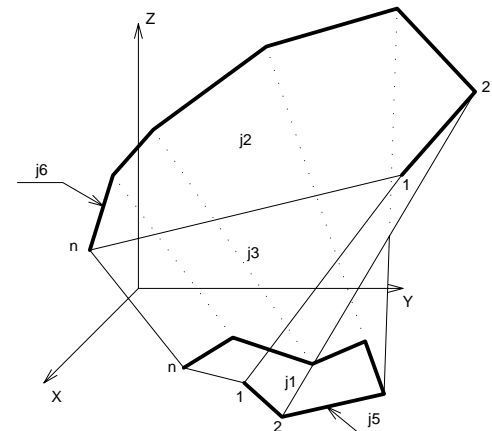
j2 (2): Deckfläche ist vorhanden (nicht wirksam, falls die Deckfläche keine Ebene ist).

j3 (4): Fläche des Schlußpolygons ist vorhanden (ein Viereck in einer Ebene oder zwei Dreiecke).

j5 (16): Konturen des Polygons der x-y-Ebene sind sichtbar.

j6 (32): Konturen des zweiten Polygons sind sichtbar.

j7 (64): Konturen der Oberflächen sind sichtbar, die Oberfläche ist in Segmente aufgeteilt.

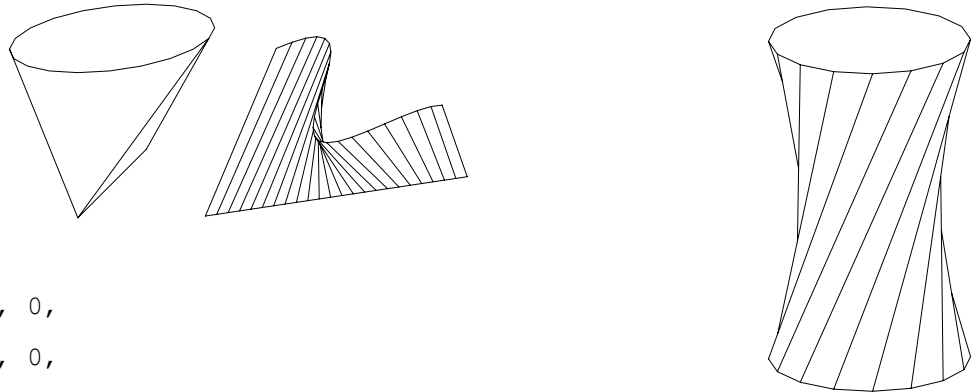


Statuswerte:

0: Seitenkanten, vom Eckpunkt ausgehend sind alle sichtbar.

1: Seitenkanten, ausgehend vom Punkt werden nur für die Konturensuche im 3D-Modell berücksichtigt.

Beispiele:



R=3

```

RULED 16, 1+2+4+16+32,
COS(22.5)*R, SIN(22.5)*R, 0,
COS(45)*R, SIN(45)*R, 0,
COS(67.5)*R, SIN(67.5)*R, 0,
COS(90)*R, SIN(90)*R, 0,
COS(112.5)*R, SIN(112.5)*R, 0,
COS(135)*R, SIN(135)*R, 0,
COS(157.5)*R, SIN(157.5)*R, 0,
COS(180)*R, SIN(180)*R, 0,
COS(202.5)*R, SIN(202.5)*R, 0,
COS(225)*R, SIN(225)*R, 0,
COS(247.5)*R, SIN(247.5)*R, 0,
COS(270)*R, SIN(270)*R, 0,
COS(292.5)*R, SIN(292.5)*R, 0,
COS(315)*R, SIN(315)*R, 0,
COS(337.5)*R, SIN(337.5)*R, 0,
COS(360)*R, SIN(360)*R, 0,
COS(112.5)*R, SIN(112.5)*R, 1,
COS(135)*R, SIN(135)*R, 1,
COS(157.5)*R, SIN(157.5)*R, 1,
COS(180)*R, SIN(180)*R, 1,
COS(202.5)*R, SIN(202.5)*R, 1,
COS(225)*R, SIN(225)*R, 1,
COS(247.5)*R, SIN(247.5)*R, 1,
COS(270)*R, SIN(270)*R, 1,
COS(292.5)*R, SIN(292.5)*R, 1,
COS(315)*R, SIN(315)*R, 1,
COS(337.5)*R, SIN(337.5)*R, 1,

```

```
COS(360)*R, SIN(360)*R, 1,  
COS(22.5)*R, SIN(22.5)*R, 1,  
COS(45)*R, SIN(45)*R, 1,  
COS(67.5)*R, SIN(67.5)*R, 1,  
COS(90)*R, SIN(90)*R, 1
```

SWEEP

```
SWEEP  n, m, alpha, scale, mask,  
        u1, v1, s1, ... un, vn, sn,  
        x1, y1, z1, ... xm, ym, zm
```

Körper, der durch das Zeichnen eines Polygonzuges entlang eines Pfades im Raum erzeugt wird.

Die Fläche des Grundpolygons wird entlang des frei im Raum definierten zweiten Polygonzuges geführt. Dieser Pfad hat seinen Startpunkt in der x-y-Ebene und wird über seine Raum-Koordinaten definiert. Wird diese Voraussetzung nicht getroffen, wird der Pfad entlang der z-Achse mit dem Startpunkt in der x-y-Ebene bewegt.

Eine Schnittfläche im Punkt (xi, yi, zi) steht im rechten Winkel zum Polygonsegment des Pfades zwischen den Punkten (xi-1,yi-1,zi-1) und (xi, yi, zi).

Mit Hilfe des Befehles SWEEP lassen sich z.B. Ausgießer von Teekannen oder andere komplexe Formen erschaffen.

n: Anzahl der Eckpunkte des Polygonzuges.

m: Anzahl der Eckpunkte des Pfad-Polygons.

alpha: Relativer Rotationswinkel der Grundfläche. Drehung erfolgt an jedem Eckpunkt des Pfades.

scale: Skalierungsfaktor der Grundfläche. Skaliert von Eckpunkt zu Eckpunkt.

mask: Bestimmt die Existenz der Grund- und Deckfläche und deren Konturen.

ui, vi: Koordinaten der Eckpunkte des Grundpolygons.

si: Status der seitlichen Kanten.

xi, yi, zi: Koordinaten der Eckpunkte des Pfad-Polygons.

Einschränkung der Parameter:

```
n  > 1  
m  > 1  
z1 < z2
```

Maskieren:

$\text{mask} = j1 + 2*j2 + 4*j3 + 16*j5 + 32*j6 + 64*j7$
 wobei $j1, j2, j3, j5, j6, j7$ 0 oder 1 sein könnten.

$j1$ (1): Grundfläche ist vorhanden.

$j2$ (2): Deckfläche ist vorhanden.

$j3$ (4): Die das Prisma Seitenfläche ist vorhanden.

$j5$ (16): Konturen der Grundfläche sind vorhanden.

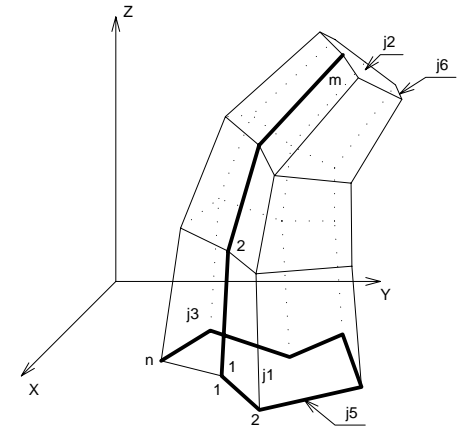
$j6$ (32): Außenkonturen der Deckfläche sind sichtbar.

$j7$ (64): Schnittkonturen sind sichtbar, die Oberfläche ist artikuliert.

Statuswerte:

0: Seitenkanten, vom Eckpunkt ausgehend sind alle sichtbar.

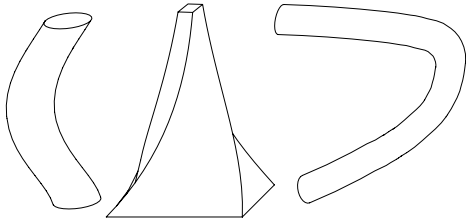
1: Seitenkanten, ausgehend vom Punkt werden nur für die Konturensuche im 3D-Modell berücksichtigt.



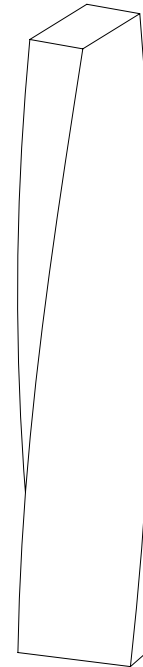
Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie mithilfe spezieller Beschränkungen.

Siehe *“Zusätzliche Statuscodes” auf Seite 143 für mehr Details.*

Beispiele:



```
SWEEP 4, 12, 7.5, 1, 1+2+4+16+32,
      -0.5, -0.25, 0,
      0.5, -0.25, 0,
      0.5, 0.25, 0,
      -0.5, 0.25, 0,
      0, 0, 0.5,
      0, 0, 1,
      0, 0, 1.5,
      0, 0, 2,
      0, 0, 2.5,
      0, 0, 3,
      0, 0, 3.5,
      0, 0, 4,
      0, 0, 4.5,
      0, 0, 5,
      0, 0, 5.5,
      0, 0, 6
```



TUBE

TUBE $n, m, \text{mask},$
 $u1, w1, s1,$
 \dots
 $un, wn, sn,$
 $x1, y1, z1, \text{angle1},$
 \dots
 $xm, ym, zm, \text{anglem}$

Körper, der durch Führen eines Polygonzuges entlang eines räumlichen Polygonzuges ohne Verzerren der Schnittflächen erzeugt wird. Die inneren Verbindungsflächen werden in der u-w-Ebene des unmittelbar definierten u-v-w Koordinatensystems gedreht..

V-Achse: entspricht der Tangente des den Körper erzeugenden Pfades im entsprechenden Punkt,

W-Achse: senkrecht zur v-Achse und nach oben ausgerichtet, nimmt Bezug auf die lokale z-Achse,

U-Achse: senkrecht zur v- und zur w-Achse und bildet mit den beiden ein bezugsgerechtes kartesisches Koordinatensystem (Rechte-Hand-Regel).

Ist die V-Achse vertikal, so kann die Ausrichtung der W-Achse nicht korrekt bestimmt werden. Die W-Achse wird im vorhergehenden Punkt verwendet, um eine horizontale Ausrichtung zu definieren.

Das Schnittfläche-Polygon der Röhre in der Mitte der Pfad-Segmente gemessen, ist mit dem Basispolygon immer identisch ($u1, w1, \dots un, wn$). Die Schnittpolygone in den Verbindungspunkten werden in der halbierten Ebene der Verbindungssegmente plziert. Das Basispolygon muß geschlossen werden.

n : Anzahl der Eckpunkte des Basispolygonzuges.

m : Anzahl der Eckpunkte des Pfad-Polygons.

ui, wi : Koordinaten der Eckpunkte des Grundpolygons.

si : Status der seitlichen Kanten.

xi, yi, zi : Koordinaten der Eckpunkte des Pfad-Polygons.

Anmerkung: Der Pfad enthält zwei Punkte mehr als die Anzahl der generierten Schnitte. Der erste und der letzte Punkt geben die Lage der vorderen und hinteren Oberfläche des TUBE an. Diese Punkte spielen nur bei der Bestimmung der senkrechten Oberflächen eine Rolle und sind keine Eckpunkte des Pfades. Die Orientierung der Oberflächen stimmt mit der Orientierung der Oberflächen überein, die an den Kanten in nächster Nähe zu den beiden Endpunkten erstellt wären, wenn die TUBE in von diesen angezeigte Richtung fortgesetzt wäre.

anglei : Drehwinkel der Schnittflächen.

Mask-Werte

$\text{mask} = j1 + 2*j2 + 16*j5 + 32*j6 + 64*j7$ wobei $j1, j2, j5, j6, j7$ 0 oder 1 sein können.

j1 (1): Basisfläche ist vorhanden.

j2 (2): Schlußfläche ist vorhanden.

j5 (16): Konturen der Basisfläche (bei x_2, y_2, z_2) sind sichtbar.

j6 (32): Konturen der Schlußfläche (bei $x_{m-1}, y_{m-1}, z_{m-1}$) sind sichtbar.

j7 (64): Schnittkonturen sind sichtbar, die Oberfläche ist artikuliert.

Einschränkung der Parameter:

$n > 2$ und $m > 3$

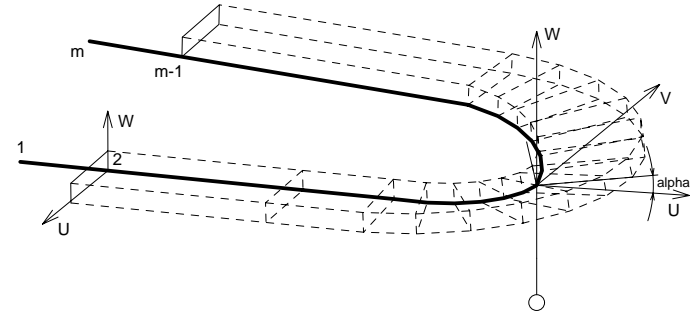
Statuswerte:

0: Seitenkanten, vom Eckpunkt ausgehend sind alle sichtbar.

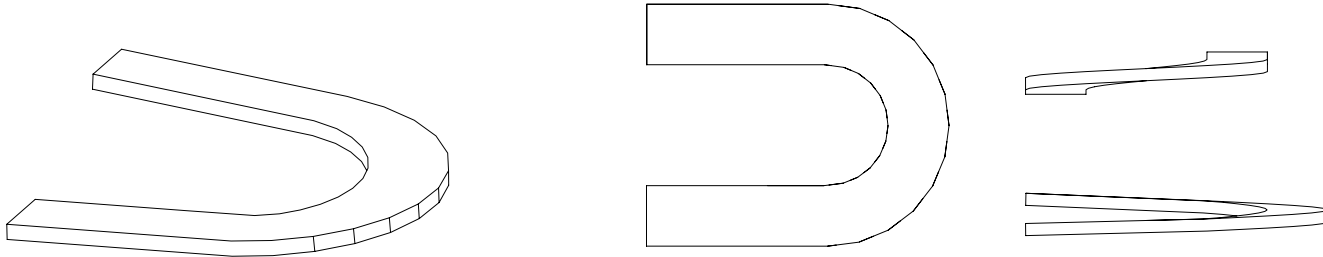
1: Seitenkanten, ausgehend vom Punkt werden nur für die Konturensuche im 3D-Modell berücksichtigt.

Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie mithilfe spezieller Beschränkungen.

Siehe *“Parameterobjekte” des ArchiCAD 9 Referenzbuches*.



Beispiele:



```
TUBE 4, 18, 16+32,
      2.0, 0.0, 0,
      0.0, 0.0, 0,
      0.0, 0.4, 0,
      2.0, 0.4, 0,
      -1, 0, 0, 0,
      0, 0, 0, 0,
      4, 0, 0.1, 0,
      6, 0, 0.15, 0,
      6+4*SIN(15), 4 - 4*COS(15), 0.2, 0,
      6+4*SIN(30), 4 - 4*COS(30), 0.25, 0,
      6+4*SIN(45), 4 - 4*COS(45), 0.3, 0,
      6+4*SIN(60), 4 - 4*COS(60), 0.35, 0,
      6+4*SIN(75), 4 - 4*COS(75), 0.4, 0,
      10, 4, 0.45, 0,
      6+4*SIN(105), 4 - 4*COS(105), 0.5, 0,
      6+4*SIN(120), 4 - 4*COS(120), 0.55, 0,
      6+4*SIN(135), 4 - 4*COS(135), 0.6, 0,
      6+4*SIN(150), 4 - 4*COS(150), 0.65, 0,
      6+4*SIN(165), 4 - 4*cos(165), 0.7, 0,
      6, 8, 0.75, 0,
      0, 8, 1, 0,
      -1, 8, 1, 0
```

```

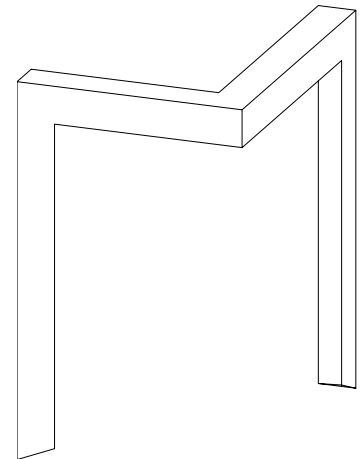
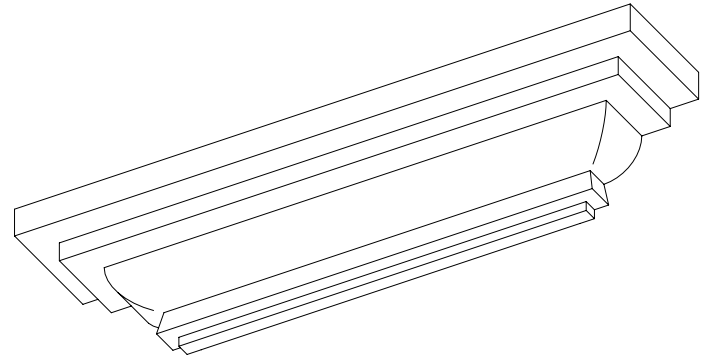
TUBE 14, 6, 1+2+16+32,
      0, 0, 0,
      0.03, 0, 0,
      0.03, 0.02, 0,
      0.06, 0.02, 0,
      0.05, 0.0699, 0,
      0.05, 0.07, 1,
      0.05, 0.15, 901,
      1, 0, 801,
      0.08, 90, 2000,
      0.19, 0.15, 0,
      0.19, 0.19, 0,
      0.25, 0.19, 0,
      0.25, 0.25, 0,
      0, 0.25, 0,
      0, 1, 0, 0,
      0, 0.0001, 0, 0,
      0, 0, 0, 0,
      -0.8, 0, 0, 0,
      -0.8, 0.0001, 0, 0,
      -0.8, 1, 0, 0

```

```

TUBE 3, 7, 16+32,
      0, 0, 0,
      -0.5, 0, 0,
      0, 0.5, 0,
      0.2, 0, -0.2, 0,
      0, 0, 0, 0,
      0, 0, 5, 0,
      3, 0, 5, 0,
      3, 4, 5, 0,
      3, 4, 0, 0,
      3, 3.8, -0.2, 0

```



TUBEA

TUBEA $n, m, \text{mask},$
 $u1, w1, s1,$
 \dots
 $un, wn, sn,$
 $x1, y1, z1,$
 \dots
 xm, ym, zm

TUBEA ist ein Körper, der durch Führen eines Polygonzuges entlang eines räumlichen Polygonzuges mit einem anderen Alogrythmus, als der der TUBE-Anweisung erzeugt wird.

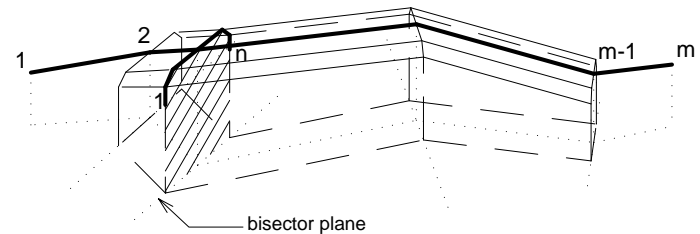
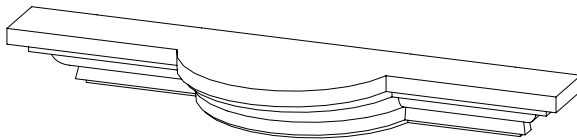
Das Schnittpolygon wird in jedem Verbindungspunkt der Pfad-Kurve generiert, es gleicht dem Basispolygon ($u1, w1, \dots un, wn$) und wird in der halbierten Ebene der Projektionen der Verbindungssegmente zu der lokalen x-y-Ebene plaziert. Das Basispolygon kann geöffnet werden: in diesem Fall werden die Schnittpolygone generiert, um die lokale x-y-Ebene zu erreichen, wie im Falle der REVOLVE-Oberflächen.

Die Schnittfläche der Röhre in der Mitte der Pfad-Segmente gemessen, kann anders sein als die des Basispolygons.

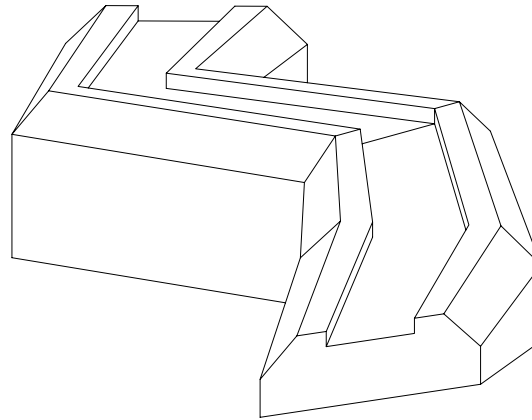
Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie mithilfe spezieller Beschränkungen.

Siehe *“Zusätzliche Statuscodes” auf Seite 143* für mehr Details.

Beispiele:



TUBEA 9, 7, 1 + 2 + 16 + 32,
 -1, 1, 0,
 0, 2, 0,
 0.8, 2, 0,
 0.8, 1.6, 0,
 0.8001, 1.6, 1,
 3.2, 1.6, 0,
 3.2, 2, 0,
 4, 2, 0,
 5, 1, 0,
 0, -7, 0,
 0, 0, 0,
 4, 0, 1,
 9, 3, 2.25,
 9, 10, 2.25,
 14, 10, 2.25,
 20, 15, 5



COONS

COONS $n, m, \text{mask},$
 $x_{11}, y_{11}, z_{11}, \dots x_{1n}, y_{1n}, z_{1n},$
 $x_{21}, y_{21}, z_{21}, \dots x_{2n}, y_{2n}, z_{2n},$
 $x_{31}, y_{31}, z_{31}, \dots x_{3m}, y_{3m}, z_{3m},$
 $x_{41}, y_{41}, z_{41}, \dots x_{4m}, y_{4m}, z_{4m}$

Oberflächenstruktur, die durch ihre vier durch Polygonzüge definierten Begrenzungskanten erzeugt wird.

Mask-Werte

$\text{mask} = 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7$
 wobei $j_3, j_4, j_5, j_6, j_7 \in \{0, 1\}$ sein können.

j_3 (4): Kontur der 1. Grenzlinie (x_1, y_1, z_1) ist sichtbar.

j_4 (8): Kontur der 2. Grenzlinie (x_2, y_2, z_2) ist sichtbar.

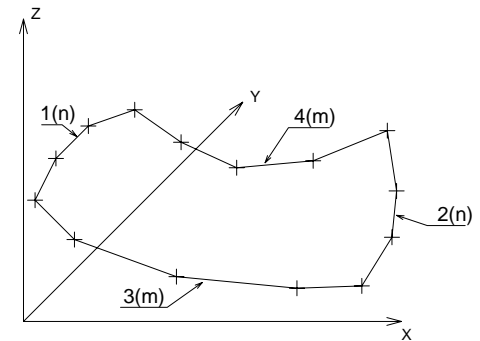
j_5 (16): Kontur der 3. Grenzlinie (x_3, y_3, z_3) ist sichtbar.

j_6 (32): Kontur der 4. Grenzlinie (x_4, y_4, z_4) ist sichtbar.

j_7 (64): Innenkonturen der Oberfläche sind sichtbar, die Fläche ist in Segmente aufgeteilt.

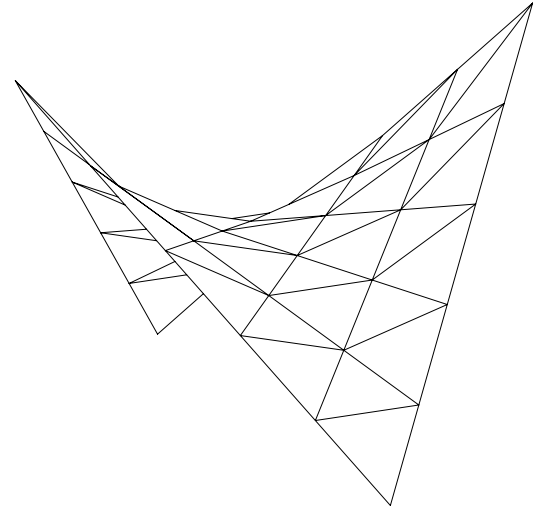
Einschränkung der Parameter:

$n, m > 1$



Beispiele:

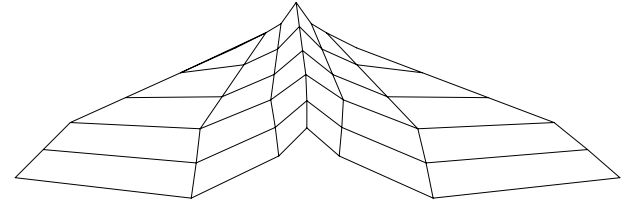
```
COONS  6, 6, 4+8+16+32+64,  
        ! 1st boundary, n=6  
        0, 0, 5,  
        1, 0, 4,  
        2, 0, 3,  
        3, 0, 2,  
        4, 0, 1,  
        5, 0, 0,  
        ! 2nd boundary, n=6  
        0, 5, 0,  
        1, 5, 1,  
        2, 5, 2,  
        3, 5, 3,  
        4, 5, 4,  
        5, 5, 5,  
        ! 3rd boundary, m=6  
        0, 0, 5,  
        0, 1, 4,  
        0, 2, 3,  
        0, 3, 2,  
        0, 4, 1,  
        0, 5, 0,  
        ! 4th boundary, m=6  
        5, 0, 0,  
        5, 1, 1,  
        5, 2, 2,  
        5, 3, 3,  
        5, 4, 4,  
        5, 5, 5
```



```

ROTZ  -90
ROTY  90
COONS  7, 6, 4+8+16+32+64,
! 1st boundary, n=7
1, 2, 0,
0.5, 1, 0,
0.2, 0.5, 0,
-0.5, 0, 0,
0.2, -0.5, 0,
0.5, -1, 0,
1, -2, 0,
! 2nd boundary, n=7
6, 10, -2,
6.5, 4, -1.5,
5, 1, -1.2,
4, 0, -1,
5, -1, -1.2,
6.5, -4, -1.5,
6, -10, -2,
! 3rd boundary, m=6
1, 2, 0,
2, 4, -0.5,
3, 6, -1,
4, 8, -1.5,
5, 9, -1.8,
6, 10, -2,
! 4th boundary, m=6
1, -2, 0,
2, -4, -0.5,
3, -6, -1,
4, -8, -1.5,
5, -9, -1.8,
6, -10, -2

```



MASS

```
MASS top_material, bottom_material, side_material,
      n, m, mask, h,
      x1, y1, z1, s1,
      ...
      xn, yn, zn, sn,
      xn+1, yn+1, zn+1, sn+1,
      ...
      xn+m, yn+m, zn+m, sn+m
```

Das Äquivalent der von dem Freiflächen-Werkzeug in ArchiCAD erstellten Form.

top_material, bottom_material, side_material: Name/Index des Deck-, Boden-, und Seitenmaterialien.

n: die Anzahl der Punkte des Massenpolygons.

m: die Anzahl der Punkte auf den Firsten.

h: die Höhe des Felsens (kann negativ sein).

x_i, y_i, z_i : die Koordinaten der Punkte.

si: ähnlich der PRISM_-Anweisung. Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie mithilfe spezieller Beschränkungen.

Siehe “Zusätzliche Statuscodes” auf Seite 143 für mehr Details.

Maskieren:

mask = j1 + 4*j3 + 16*j5 + 32*j6 + 64*j7
wobei j1, j3, j5, j6, j7 0 oderr 1 sein können.

j1 (1): Grundfläche ist vorhanden.

j3 (4): Seitenflächen sind vorhanden.

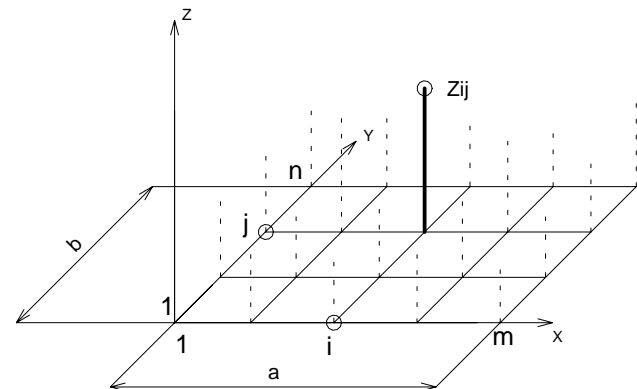
j5 (16): Konturen der Grund- und Seitenflächen sind sichtbar.

j6 (32): Außenkonturen der Deckfläche sind sichtbar.

j7 (64): Innenkonturen der Deckfläche sind sichtbar, die Oberfläche ist in Segmente aufgeteilt.

j8 (128): Alle Gratsparren sind scharf, aber die Oberfläche ist glatt.

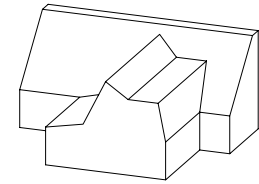
Einschränkung der Parameter:

$$n \geq 3, \quad m \geq 0$$


Beispiel:

```
MASS  "Whitewash", "Whitewash", "Whitewash",
      15, 12, 117, -5.0,
      0, 12, 0, 15,
      8, 12, 0, 15,
      8, 0, 0, 15,
      13, 0, 0, 13,
      16, 0, 0, 13,
      19, 0, 0, 13,
      23, 0, 0, 13,
      24, 0, 0, 15,
      24, 12, 0, 15,
      28, 12, 0, 15,
      28, 20, 8, 13,
      28, 22, 8, 15,
      0, 22, 8, 15,
      0, 20, 8, 13,
      0, 12, 0, -1,

      0, 22, 8, 0,
      28, 22, 8, -1,
      23, 17, 5, 0,
      23, 0, 5, -1,
      13, 13, 1, 0,
      13, 0, 1, -1,
      16, 0, 7, 0,
      16, 19, 7, -1,
      0, 20, 8, 0,
      28, 20, 8, -1,
      19, 17, 5, 0,
      19, 0, 5, -1
```



ELEMENTE ZUR UNTERSTÜTZUNG DER PHOTOREALISTIK

Light

LIGHT red, green, blue, shadow,
 radius, alpha, beta, angle_falloff,
 distance1, distance2,
 distance_falloff [ADDITIONAL_DATA name1 = value1,
 name2 = value2, ...]

Eine Lichtquelle, die farbiges Licht [rot, grün, blau] vom lokalen Nullpunkt entlang bzw. parallel zu der x-Achse ausstrahlt. Ausgangspunkt kann eine punkt- oder kreisförmige Lichtquelle sein. Die maximale Intensität liegt innerhalb eines gedachten Kegelstumpfes. Dieser wird durch die die Lichtquelle tangentialierenden, zur x-Achse parallelen Projektionslinien und einen mit dem Winkel alpha beschriebenen Ausfallbereich gekennzeichnet. (Der Wert Null gibt dem Lichtstrahl dabei eine scharfe Kante, höhere Werte ermöglichen einen weicheren Übergang.) Durch die Werte distance1 und distance 2 kann der Lichteffect auf einen Bereich begrenzt werden. Auch hier wird die Abschwächung der Intensität in Abhängigkeit von der Entfernung über einen Parameter angegeben (= distance_falloff). Dabei gibt der Wert Null dem Lichtstrahl eine konstante Intensität, höhere Werte bedeuten einen höheren Intensitätsverlust.

GDL-Transformationen beeinflussen nur den Ausgangspunkt der Lichtquelle und die Ausrichtung des Lichtes.

Der Schatten-Parameter bestimmt den Schattenwurf.

0: das Licht wirft keinen Schatten

1: das Licht wirft Schatten

Einschränkung der Parameter::

$\alpha \leq \beta \leq 80^\circ$

Kombinationen von Parametern mit speziellen Bedeutungen:

radius = 0, alpha = 0, beta = 0

Punktförmige Lichtquelle, strahlt Licht in alle Richtungen ab und erzeugt keinen Schattenwurf. Die shadow und angle_falloff Parameter werden ignoriert, für beide wird der Wert 0 angesetzt.

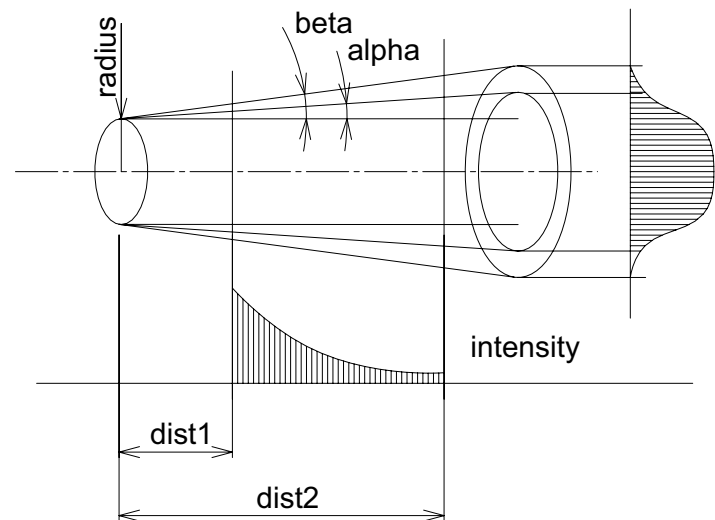
radius > 0, alpha = 0, beta = 0

Auf den angegebenen Radius fokussierte Lichtquelle.

Die Lichtdefinition kann optionale zusätzliche Definitionen enthalten nach dem Schlüsselwort

Das Schlüsselwort ADDITIONAL_DATA. Zusätzliche Daten haben einen Namen (namei) und einen Wert (valuei), der ein Ausdruck eines beliebigen Typs sein kann, auch ein Array. Wenn ein String Parametername mit substring "_file" beendet, wird sein Wert als Dateiname betrachtet und in das Archiv-Projekt eingefügt.

Unterschiedliche Bedeutungen zusätzlicher Daten können mit ArchiCAD oder mit Add-Ons zu ArchiCAD definiert und verwendet werden.

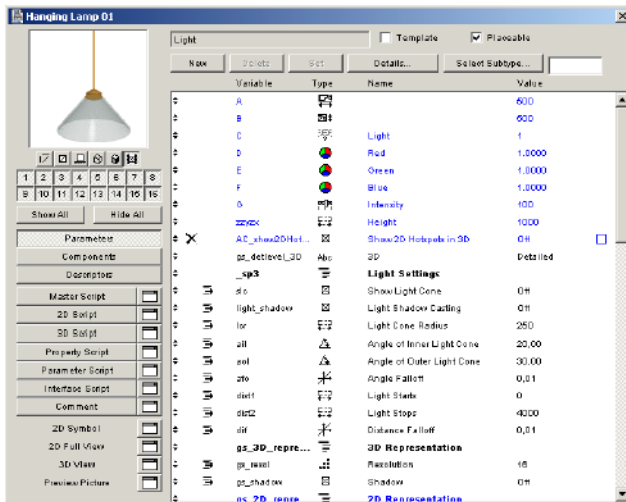


Siehe Bedeutung der LightWorks Add-On Parameter unter <http://www.graphisoft.com/support/developer/documentation/wide/LibraryDevKit/>.

Beispiel:

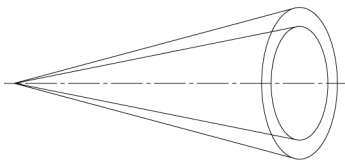
```
LIGHT 1.0,0.2,0.3,      ! RGB
      1,                ! shadow on
      1.0,              ! radius
      45.0,60.0,        ! angle1, angle2
      0.3,              ! angle_falloff
      1.0,10.0,         ! distance1, distance2
      0.2               ! distance_falloff
```

Bibliothekselement-Dialog für Lichtquellen in ArchiCAD/ArchiFM:

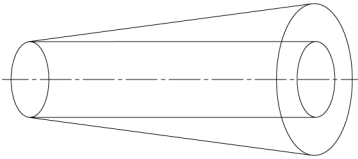


Teil des entsprechenden GDL-Scripts:

```
IF C = 0 GOTO 10
LIGHT G/100*D, G/100*E, G/100*F, ! RGB
...
10:
```



$r = 0, \alpha > 0, \beta > 0$



$r > 0, \alpha = 0, \beta > 0$



$r > 0, \alpha = 0, \beta = 0$

Lichtquellen mit verschiedenen alpha- und beta-Parametern.

PICTURE

PICTURE expression, a, b, mask

Ein Bildelement für die Photorealistik.

'expression' steht für einen Dateinamen, einen numerischen Ausdruck oder einen Index eines im Bibliothekselement abgelegten Bildes. Ein 0-Index ist ein spezieller Wert, der sich auf das Vorschaubild des Bibliothekselements bezieht. Andere Bilder können in Bibliothekselementen nur dann gespeichert werden, wenn das Projekt oder die ausgewählten Elemente, die die Bilder beinhalten, als GDL-Objekte gesichert werden.

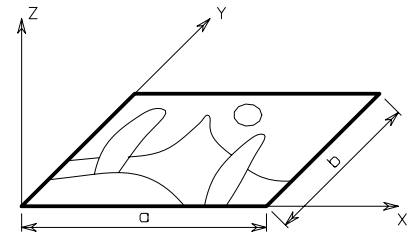
Indexierte Bildverweise können im MASTER_GDL-Script nicht verwendet werden, wenn Attribute zum aktuellen Attribut-Set dazugeladen werden. Das Bild wird in ein Rechteck eingepaßt, das in der 3D-Darstellung wie das RECT-Element behandelt wird.

mask = alpha + verzerrung

alpha: Steuerung des alpha-channel

0: Kein Verwenden des alpha-channel, das Bild ist ein Rechteck.

1: Verwenden des alpha-channel, Teile des Bildes sind transparent.

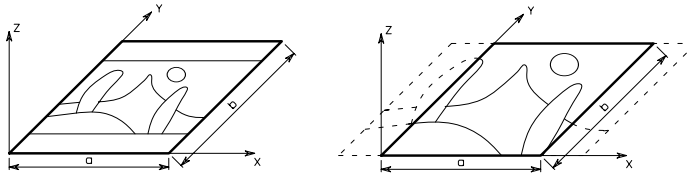
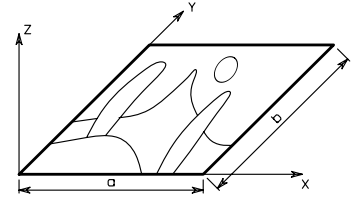


verzerrung: Kontrolle der Verzerrung

0: das Bild wird in den gegebenen Rahmen (evt. verzerrt) eingepaßt.

2: das Bild wird in Originalgröße und -proportionen in die Mitte des Rechtecks eingesetzt.

4: das Bild füllt komplett das Rechteck aus. Proportionen bleiben erhalten. Teile des Bildes sind evtl. verdeckt



3D-TEXTELEMENTE

TEXT

TEXT d, 0, expression

Eine 3D-Darstellung in dem eingestellten Stil des Wertes eines Text- oder numerischen Ausdrucks.

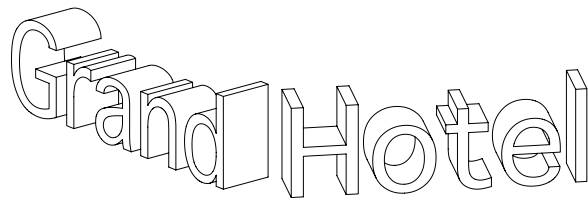
Siehe "[SET] STYLE" auf Seite 157 und "DEFINE STYLE" auf Seite 176.

d: Charakterstärke in Meter

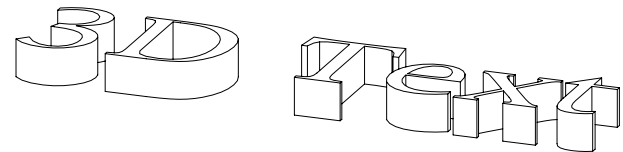
In dieser Version von GDL ist der zweite Wert stets Null.

Beispiele:

```
DEFINE STYLE "aa" "New York", 3, 7, 0
SET STYLE "aa"
TEXT 0.005, 0, "3D Text"
```



```
name = "Grand"
ROTX 90
```



```
ROTY -30  
TEXT 0.003, 0, name  
ADDX STW (name)/1000  
ROTY 60  
TEXT 0.003, 0, "Hotel"
```

Hinweis: Zu Kompatibilitätszwecken mit dem 2D GDL Script werden die Charakter-Höhen in DEFINE STYLE-Anweisungen immer in mm interpretiert.

RICHTTEXT

RICHTTEXT *x*, *y*,
height, 0, textblock_name

Dreidimensionale Darstellung von einem vorher definierten TEXTBLOCK. Für weitere Details, siehe *“Textblock” auf Seite 179*

x, *y*: X-Y Koordinaten der Richttext-Platzierung

Höhe: Charakterstärke in Meter.

textblock_name: Name eines früher definierten TEXTBLOCK

In der aktuellen Version von GDL ist der vierte Wert stets Null.

GRUNDELEMENTE

Die Grundeinheiten der 3D-Datenstruktur sind VERT, VECT, EDGE, PGON und BODY. Die Körper werden durch ihre Oberflächen und deren Verbindungen dargestellt. Diese Verknüpfungen liefern die Information zur Erzeugung des 3D-Schnittes.

Die Indizierung beginnt mit 1, jeder neue Körper oder ein BASE-Befehl setzt den Index auf 1 zurück. Für jede Kante wird der Index der angrenzenden Polygone (maximal 2) gespeichert. Kanten und ihre Ausrichtung werden durch den ersten und zweiten Eckpunkt definiert.

Die Beschreibung eines Polygons ist wiederum eine Liste dieser Körperkanten mit Angaben zu ihrer Ausrichtung und ihrer Indizes. Die Indizes können negative Vorzeichen haben. Das heißt, daß die Ausrichtung einer Kante dann gedreht wird. Polygone können Öffnungen besitzen. In der Auflistung der Kanten zeigt der Index Null eine neue Öffnung an. Eine Öffnung kann keine weiteren Öffnungen beinhalten. Eine Kante kann zu keinem, einem oder zwei Polygonen gehören. Bei geschlossenen Körpern ist die geometrische Ausrichtung eines Polygons (bezüglich der Kanten) dann korrekt, wenn die gemeinsame Kante zweier Polygone verschiedene Vorzeichen in der jeweiligen Liste hat.

Der Normal-Vektor eines Polygons wird separat gespeichert. Bei geschlossenen Körpern ist er von innen nach außen gerichtet. Die Auflistung der Körperkanten erfolgt, von der Außenseite her gesehen, gegen den Uhrzeigersinn (mathematisch positiv). Die Richtung der Öffnungen verläuft entgegengesetzt zum Basispolygon. Die Normal-Vektoren eines offenen Körpers müssen zur gleichen Seite des Körpers gerichtet sein.

Um die Innen- und Außenseite eines Körpers zu definieren, muß er geschlossen sein. Eine einfache Definition für einen geschlossenen Körper ist, daß jede Kante genau zwei benachbarte Polygone hat.

Die Leistungsfähigkeit der Algorithmen zur Berechnung von Schnitten, verdeckten Kanten oder beim Rendering ist bei offenen Körpern geringer. Jedes zusammengesetzte 3D-Element mit regulären Parametern ist in der internen Datenstruktur ein geschlossener Körper.

Die Berechnung der Konturlinien basiert auf den Status-Werten der Kanten und der angrenzenden Polygone. Sie wird bei Elementen mit gekrümmten Oberflächen automatisch ausgeführt. Bei den Basiselementen hat der Anwender die Möglichkeit, diese Werte selbst anzugeben.

Im Falle einer vereinfachten Definition (vect = 0 oder status < 0 in PGON) müssen die Grundelemente, auf die andere Elemente Bezug nehmen, diesen vorausgehen. Bei den Basiselementen hat der Anwender die Möglichkeit, diese Werte selbst anzugeben. In diesem Falle ist die erforderliche Ordnung:

```
VERT (TEVE)
EDGE
(VECT)
PGON (PIPG)
```

COOR
BODY

Die Suche der Kanten nach benachbarten Polygonen erfolgt während der Ausführung des Befehls BODY.

Die Nummerierung der VERT-, EDGE-, VECT- und PGON-Elemente erfolgt in Abhängigkeit zum letzten BASE-Befehl. (Wird entweder ausdrücklich vom Anwender oder automatisch vom Programm angegeben.)

Status-Werte werden benutzt, um bestimmte Informationen über die Grundelemente zu speichern. Jeder einzelne Wert hat seine spezielle Bedeutung, es gibt hier jedoch einige Ausnahmen.

Angegebene Werte können addiert werden. Andere als die unten angegebenen Kombinationen sind reserviert. Das Programm setzt als Voreinstellung den Status-Wert auf Null.

VERT

VERT x, y, z

Ein Punkt im x-y-z-Raum, definiert durch seine drei Koordinaten.

TEVE

TEVE x, y, z, u, v

Erweiterung der VERT-Anweisung mit einer Definition von Texturkoordinaten. Diese Anweisung kann anstelle der VERT-Anweisung benutzt werden, wenn benutzerdefinierte Texturkoordinaten anstelle der automatischen Texturverkleidung benötigt werden (siehe *“COOR” auf Seite 100* Anweisung).

x, y, z : Koordinaten eines Eckpunktes

u, v : Texturkoordinaten des Eckpunktes

(u, v) Koordinaten jedes Scheitelpunktes des gegenwärtigen Körpers müssen definiert werden und jeder Scheitelpunkt sollte nur eine Texturkoordinate haben. Werden die VERT- und TEVE-Anweisungen innerhalb der Körperdefinition gemischt verwendet, sind die (u, v) -Koordinaten unwirksam.

Anmerkung: die (u, v) -Texturkoordinaten haben nur in photorealistischen Darstellungen eine Auswirkung, nicht aber in den schattierten Darstellungen.

VECT

VECT x, y, z

Definition des Normal-Vektors eines Polygons durch drei Koordinaten. Im Falle einer vereinfachten Definition ($vect=0$ in PGON) können diese Befehle vernachlässigt werden.

EDGE

EDGE vert1, vert2, pgon1, pgon2, status

Definition einer Kante.

vert1, vert2: Indizes der Eckpunkte. Die Indizes vert1 und vert2 müssen verschieden sein und sich auf die vorher definierten VERTs beziehen.

pgon1, pgon2: Indizes der benachbarten Polygone. Null und negative Werte haben spezielle Bedeutung::

0: Seitliche oder alleinstehende Kante.

< 0: Mögliche Nachbarpolygone werden gesucht.

Status-Werte:

1: Unsichtbare Kante.

2: Kante einer gekrümmten Oberfläche.

Reservierte Status-Bits für zukünftige Anwendungen:

4: Erste Kante einer gekrümmten Oberfläche (nur im Zusammenhang mit 2).

8: Erste Kante einer gekrümmten Oberfläche (nur im Zusammenhang mit 2).

16: Die Kante ist ein Bogen..

32: Das erste Segment eines Bogens (nur im Zusammenhang mit 16).

64: Das letzte Segment eines Bogens (nur im Zusammenhang mit 16).

PGON

PGON n, vect, status, edge1, edge2, ... edgen

Polygondefinition

n: Anzahl der Kanten in der Parameterliste

vect: Index des Normal-Vektors. Er muß Bezug auf einen vorher definierten VECT nehmen.

Anmerkung: Wenn vect = 0, wird der Normal-Vektor während der Berechnung ermittelt.

Die Indizes edge1, edge2, ... kanten müssen auf die vorher definierten EDGE Bezug nehmen. Der Wert 0 kennzeichnet den Anfang oder das Ende einer Öffnungsdefinition.

Ein negativer Index kehrt die Richtung des gespeicherten Normal-Vektors oder der Kante des Polygons um. (Der gespeicherte Vektor oder die gespeicherte Kante selbst ändern sich nicht; andere Polygone können darauf Bezug nehmen, indem sie die Originalrichtung mit einem positiven Index verwenden.)

Status-Werte:

- 1: unsichtbares Polygon.
- 2: Polygon mit einer gekrümmten Oberfläche.
- 16: Konkaves Polygon.
- 32: Polygon mit Öffnungen.
- 64: Die Öffnungen sind konvex (nur im Zusammenhang mit 32).

Reservierte Status-Bits für zukünftige Anwendungen:

- 4: Erstes Polygon mit einer gekrümmten Oberfläche (nur im Zusammenhang mit 2).
- 8: Letztes Polygon mit einer gekrümmten Oberfläche (nur im Zusammenhang mit 2).

Wenn der Status-Wert negativ ist, berechnet das Programm den Status des Polygons (wie es das Programm bei konkaven Polygonen oder Polygonen mit Öffnungen automatisch ausführt).

$n = 0$ ist für spezielle Zwecke gestattet.

PIPG

PIPG expression, a, b, mask, n, vect,
status,
edge1, edge2, ... edgen

Definition eines Bild-Polygons Die ersten vier Parameter sind dieselben wie bei dem PICTURE-Befehl, die restlichen dieselben wie bei dem PGON-Befehl.

COOR

COOR wrap, vert1, vert2, vert3, vert4

Lokales Koordinatensystem eines BODY-Elementes für Schraffur- und Textur-Mapping.

wrap (umhüllung): Umhüllungsmodus + Projektionstyp

Umhüllungsmodi:

- 1: eben
- 2: kubisch
- 3: zylindrisch
- 4: sphärisch

5: gleicht dem zylindrischen Schraffur-Mapping, aber beim Rendering wird ein kreisförmiges Mapping auf der Deck- und Bodenflächen verwendet.

Projektionstypen

256: die Füllung geht immer vom Ursprung des lokalen Koordinatensystems aus.

1024: quadratische Texturprojektion (empfohlen)

2048: lineare Texturprojektion, die auf dem Durchschnittsabstand basiert.

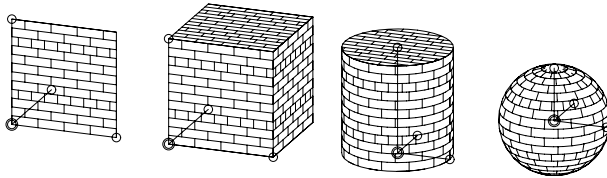
4096: lineare Texturprojektion, die auf den normalen Dreieckspunkten basiert.

Anmerkung: Die letzten drei Werten sind nur mit benutzerdefinierten Texturkoordinaten-Definitionen effektiv (siehe *“TEVE” auf Seite 98*).

vert1: Index eines VERT, definiert den Ursprung des lokalen Koordinatensystems.

vert2, vert3, vert4: Indizes der die drei Koordinatenachsen definierenden VERTs.

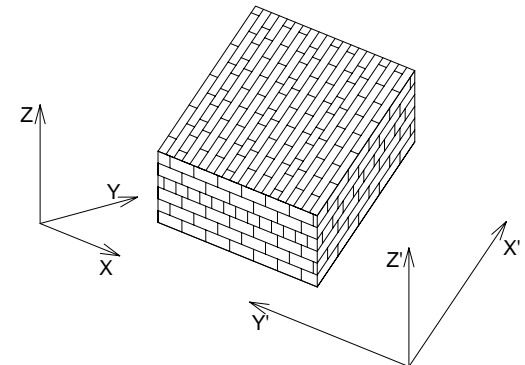
Ein Minus-Zeichen vor den VERT-Indizes wird verwendet, wenn diese nur das lokale Koordinatensystem definieren sollen.



Beispiel für benutzerdefinierte Texturachsen:

```
CSLAB_ "Face brick", "Face brick", "Face brick",
      4, 0.5,
      0, 0, 0, 15,
      1, 0, 0, 15,
      1, 1, 1, 15,
      0, 1, 1, 15

BASE
VERT   1, 0, 0      !#1
VERT   1, 1, 1      !#2
VERT   0, 0, 0      !#3
VERT   1, 0, 1      !#4
COORD  2, -1, -2, -3, -4
BODY   1
```



BODY

BODY status

Erzeugt einen Körper unter Verwendung der vorher genannten Grundelemente.

Status-Werte:

- 1: geschlossener Körper
- 2: Der Körper besitzt (eine) gekrümmte Oberfläche(n).
- 4: Flächenmodell: der Körper verhält sich wie ein Hohlkörper. Bei einem Schnitt durch den Körper erscheint keine Schnittfläche.
- 32: Der Körper erzeugt immer einen Schattenwurf, unabhängig von der automatischen Voreinstellung für Schattenwurf.
- 64: Körper erzeugt niemals Schattenwurf.

Ist weder der Wert 32 noch der Wert 64 angegeben, wird die automatische Voreinstellung für Schattenwurf verwendet.

Siehe *“SHADOW” auf Seite 160*.

Ist der Status des Körpers negativ, berechnet das Programm den Wert.

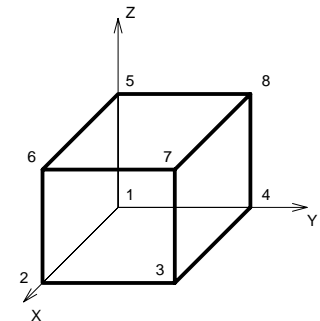
Beispiel:

1: vollständig

```

VERT 0.0, 0.0, 0.0      !#1
VERT 1.0, 0.0, 0.0      !#2
VERT 1.0, 1.0, 0.0      !#3
VERT 0.0, 1.0, 0.0      !#4
VERT 0.0, 0.0, 1.0      !#5
VERT 1.0, 0.0, 1.0      !#6
VERT 1.0, 1.0, 1.0      !#7
VERT 0.0, 1.0, 1.0      !#8
EDGE 1, 2, 1, 3, 0      !#1
EDGE 2, 3, 1, 4, 0      !#2
EDGE 3, 4, 1, 5, 0      !#3
EDGE 4, 1, 1, 6, 0      !#4
EDGE 5, 6, 2, 3, 0      !#5
EDGE 6, 7, 2, 4, 0      !#6
EDGE 7, 8, 2, 5, 0      !#7
EDGE 8, 5, 2, 6, 0      !#8
EDGE 1, 5, 6, 3, 0      !#9
EDGE 2, 6, 3, 4, 0      !#10
EDGE 3, 7, 4, 5, 0      !#11
EDGE 4, 8, 5, 6, 0      !#12
VECT 1.0, 0.0, 0.0      !#1
VECT 0.0, 1.0, 0.0      !#2

```



```

VECT 0.0, 0.0, 1.0      !#3
PGON 4, -3, 0, -1, -4, -3, -2 !#1      !VERT1,2,3,4
PGON 4, 3, 0, 5, 6, 7, 8    !#2      !VERT5,6,7,8
PGON 4, -2, 0, 1, 10, -5, -9 !#3      !VERT1,2,5,6
PGON 4, 1, 0, 2, 11, -6, -10 !#4      !VERT2,3,6,7
PGON 4, 2, 0, 3, 12, -7, -11 !#5      !VERT3,4,7,8
PGON 4, -1, 0, 4, 9, -8, -12 !#6      !VERT1,4,5,8
BODY 1                      !CUBE

```

2: (kein direkter Bezug zu den Polygonen, das Programm stellt selbst Berechnungen an)

```

VERT 0.0, 0.0, 0.0      !#1
VERT 1.0, 0.0, 0.0      !#2
VERT 1.0, 1.0, 0.0      !#3
VERT 0.0, 1.0, 0.0      !#4
VERT 0.0, 0.0, 1.0      !#5
VERT 1.0, 0.0, 1.0      !#6
VERT 1.0, 1.0, 1.0      !#7
VERT 0.0, 1.0, 1.0      !#8
EDGE 1, 2, -1, -1, 0      !#1
EDGE 2, 3, -1, -1, 0      !#2
EDGE 3, 4, -1, -1, 0      !#3
EDGE 4, 1, -1, -1, 0      !#4
EDGE 5, 6, -1, -1, 0      !#5
EDGE 6, 7, -1, -1, 0      !#6
EDGE 7, 8, -1, -1, 0      !#7
EDGE 8, 5, -1, -1, 0      !#8
EDGE 1, 5, -1, -1, 0      !#9
EDGE 2, 6, -1, -1, 0      !#10
EDGE 3, 7, -1, -1, 0      !#11
EDGE 4, 8, -1, -1, 0      !#12
PGON 4, 0, -1, -1, -4, -3, -2 !#1      !VERT1,2,3,4
PGON 4, 0, -1, 5, 6, 7, 8    !#2      !VERT5,6,7,8
PGON 4, 0, -1, 1, 10, -5, -9 !#3      !VERT1,2,5,6
PGON 4, 0, -1, 2, 11, -6, -10 !#4      !VERT2,3,6,7
PGON 4, 0, -1, 3, 12, -7, -11 !#5      !VERT3,4,7,8
PGON 4, 0, -1, 4, 9, -8, -12 !#6      !VERT1,4,5,8
BODY -1                      !CUBE

```

BASE

BASE

Setzt die Zähler für die geometrischen Grundelemente und Anweisungen (VERT, TEVE, VECT, EDGE, PGON und PIPG) auf Null. Im Rahmen komplexer Elementdefinitionen wird der Befehl automatisch ausgeführt.

VERSCHNEIDEN IM 3D-RAUM

CUTPLANE

```
CUTPLANE [x, y, z [, side]]  
    [statement1  
    ...  
    statementn]  
CUTEND
```

oder

```
CUTPLANE angle  
    [statement1  
    ...  
    statementn]  
CUTEND
```

Erstellt eine Schnittfläche und entfernt Teile der dazwischen aufgeführten Körper. CUTPLANE kann eine verschiedene Anzahl von Parametern haben.

Falls CUTPLANE die folgenden Anzahl von Parametern hat:

0: x-y Ebene ;

1: Schnittfläche durchkreuzt die x-Achse, Winkel ist zwischen der Schnittfläche und den x-y Ebenen;

2: Schnittfläche ist parallel zur z-Achse, durchkreuzt die x- und y -Achse bei den angegebenen Werten;

3: Schnittfläche durchkreuzt die x, y und z -Achsen bei den gegebenen Werten;

4: die drei ersten Parameter wie oben angeführt, mit dem folgenden Seitenwert:

side = 0: entfernt die Segmente oberhalb der Schnittebene (voreingestellt);

side = 1: entfernt die Segmente unterhalb der Schnittebene; im Falle von x-y, x-z, y-z, die Segmente in negativer Richtung der Achse.

Der Schnitt (ohne side Parameter) entfernt die Segmente oberhalb der Schnittfläche. Wenn die ersten drei Parameter die x-y, x-z oder y-z Ebene definieren, (z.B. 1.0, 1.0, 0.0 definiert die x-y Ebene), dann werden die Segmente in der positiven Richtung der dritten Achse entfernt.

Zwischen CUTPLANE und CUTEND kann eine beliebige Anzahl von Befehlen eingegeben werden. Außerdem können CUTPLANES in Makros eingefügt werden.

CUTPLANE Parameter beziehen sich auf das aktuelle Koordinatensystem

Die Umwandlungen (ROT, ADD, etc) zwischen CUTPLANE und CUTEND haben keine Wirkung auf diese bestimmte Schnittfläche, aber alle nachfolgenden CUTPLANES werden umgewandelt. So verwenden Sie die notwendigen Transformationen, um das CUTPLANE aufzubauen, dann löschen Sie all diese Transformationen bevor Sie die zu schneidenden Körper definieren.

Befehlspaare von CUTPLANE-CUTEND können sogar innerhalb von Schleifen (Loops) eingebaut werden. Fehlt das endgültige CUTEND, so wirkt sich CUTPLANE auf sämtliche Körper bis zum Ende des Scriptes aus.

CUTPLANES in Macros wirken sich nur auf den Inhalt des Macros aus, auch wenn das CUTEND fehlt.

Wird ein Makro zwischen CUTPLANE und CUTEND aufgerufen, dann werden die Körper aus dem Macro geschnitten.

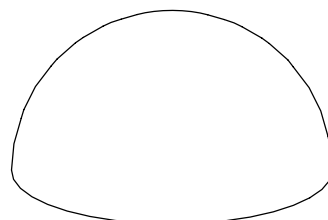
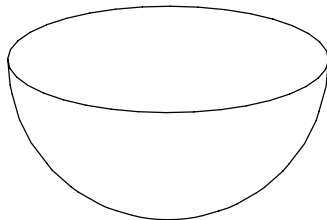
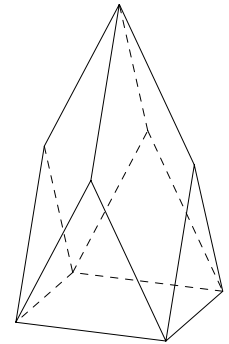
Aktuelle Material-, Stift- und Schrauffureinstellungen üben ihre Wirkung auf die Schnittflächen aus.

Hinweis: Wird CUTPLANE nicht mit CUTEND geschlossen, so könnten alle Körper vollkommen davon entfernt werden. Deswegen bekommen Sie stets eine Warnung über fehlende CUTENDs.

Wenn die Transformationen (ADD, ROT, etc), die nur zum Plazieren des CUTPLANE verwendet wurden, nicht entfernt worden sind, denken Sie evtl, daß CUTPLANE nicht richtig plazierte wurde, während wahrscheinlich die Körper komplett verschwinden.

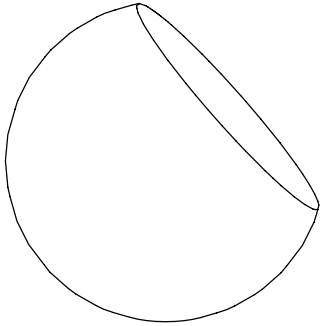
Beispiele:

```
CUTPLANE 2, 2, 4
CUTPLANE -2, 2, 4
CUTPLANE -2, -2, 4
CUTPLANE 2, -2, 4
  ADD -1, -1, 0
  BRICK 2, 2, 4
  DEL 1
CUTEND
CUTEND
CUTEND
CUTEND
```

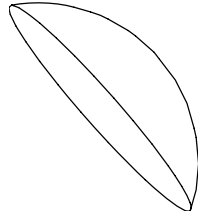


```
CUTPLANE
  SPHERE 2
CUTEND
```

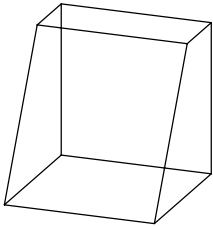
```
CUTPLANE 1, 1, 0, 1
  SPHERE 2
CUTEND
```



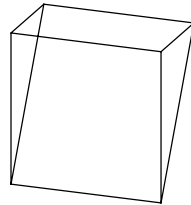
```
CUTPLANE 1.8, 1.8,  
          1.8  
SPHERE 2  
CUTEND
```



```
CUTPLANE 1.8, 1.8,  
          1.8, 1  
SPHERE 2  
CUTEND
```



```
CUTPLANE 60  
BRICK 2, 2, 2  
CUTEND
```



```
CUTPLANE -120  
BRICK 2, 2, 2  
CUTEND
```

CUTPOLY

```
CUTPOLY n,  
          x1, y1, ... xn, yn  
          [, x, y, z]  
          [statement1  
          statement2  
          ...  
          statementn]  
CUTEND
```


Ähnlicherweise wie beim CUTPLANE -Befehl nehmen die Parameter von CUTPOLY Bezug auf das aktuellen Koordinatensystem. Das Polygon muß konvex sein und kann sich nicht verschneiden. Der Schnitt erfolgt in der Richtung der z-Achse oder ein optionaler (x, y, z) Vektor kann bestimmt werden.

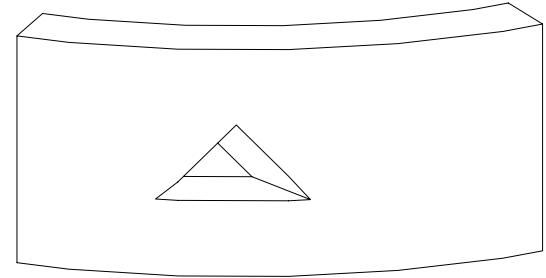
Die Parameter bestimmen eine unendliche "Röhre": das Polygon ist eine Schnittfläche der Röhre, die Schnittrichtung ist dieselbe wie die der Röhre. Alles innerhalb der Röhre wird entfernt

Beispiele:

```

ROTX 90
MULZ -1
CUTPOLY 3,
        0.5, 1,
        2, 2,
        3.5, 1,
        -1.8, 0, 1
      DEL 1
      BPRISM - "Red brick", "Red brick", "Face brick",
              4, 0.9, 7,
              0.0, 0.0, 15,
              6.0, 0.0, 15,
              6.0, 3.0, 15,
              0.0, 3.0, 15
CUTEND

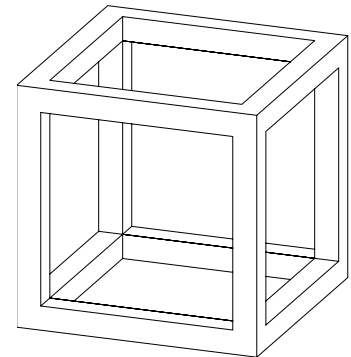
```



```

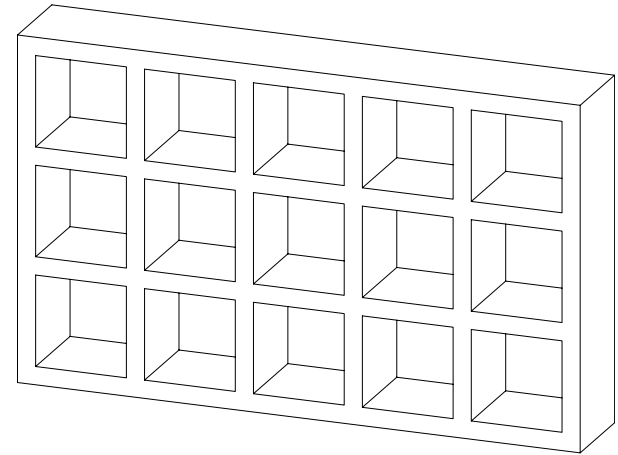
A=1.0
D=0.1
CUTPOLY 4,
        D, D,
        A-D, D,
        A-D, A-D,
        D, A-D
ROTX 90
CUTPOLY 4,
        D, D,
        A-D, D,
        A-D, A-D,
        D, A-D
      DEL 1
      ROTY 90
      CUTPOLY 4,
              D, D,
              A-D, D,
              A-D, A-D,
              D, A-D
      DEL 1
      BLOCK A, A, A
CUTEND

```



```
CUTEND
CUTEND
```

```
ROTX 90
FOR I=1 TO 3
  FOR J=1 TO 5
    CUTPOLY 4,
      0, 0, 1, 0,
      1, 1, 0, 1
    ADDX 1.2
  NEXT J
  DEL 5
  ADDY 1.2
NEXT I
DEL NTR()-1
ADD -0.2, -0.2, 0
BRICK 6.2, 3.8, 1
FOR K=1 TO 15
  CUTEND
NEXT K
DEL TOP
```



CUTPOLYA

```
CUTPOLYA n, status, d,
  x1, y1, mask1, ... xn, yn, maskn [,
  x, y, z]
  [statement1
  statement2
  ...
  statementn]
```

CUTEND

Hat den gleichen Aufbau wie die CUTPOLY-Definition, aber dies bietet die Möglichkeit, die Sichtbarkeit der Kanten des generierten Polygons kontrollieren zu können. Der Schnittsform ist eine unendliche Röhre mit der definierten Polygons-Schnittfläche. Das Ende des Schnittsformes kann in den Körper nicht hineinhängen.

status: steuert die Behandlung der erstellten Schnittpolygone

1: verwendet die Attribute des Körpers der generierten Polygone und Kanten.

22: generierte Schnittpolygone werden als normale Polygone behandelt.

d: der Abstand zwischen dem lokalen Ursprung und dem Ende der unendlichen Röhre.

d = 0 bedeutet einen Schnitt mit einer unendlichen Röhre.

maski: gleicht dem PRISM_-Befehl

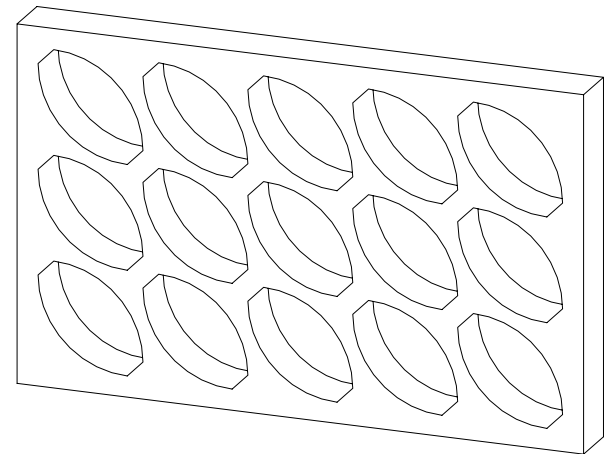
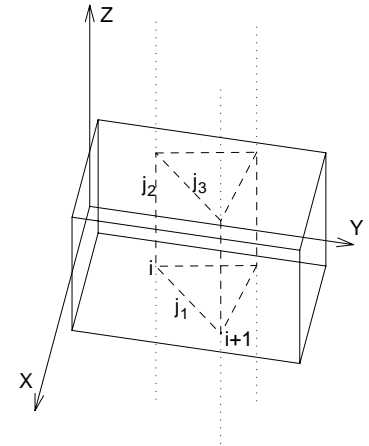
maski = $j1 + 2 * j2 + 4 * j3 + 64 * j7$

Beispiel:

```

ROTX 90
FOR I=1 TO 3
  FOR J=1 TO 5
    CUTPOLYA 6, 1, 0,
      1, 0.15, 5,
      0.15, 0.15, 900,
      0, 90, 4007,
      0, 0.85, 5,
      0.85, 0.85, 900,
      0, 90, 4007
    ADDX 1
  NEXT J
  DEL 5
  ADDY 1
NEXT I
DEL NTR() -1
ADD -0.2, -0.2, 0
BRICK 5.4, 3.4, 0.5
FOR K=1 TO 15
  CUTEND
NEXT K
DEL TOP

```



CUTSHAPE

```
CUTSHAPE d
    [statement1
    statement2
    ...
    statementn]
```

CUTEND

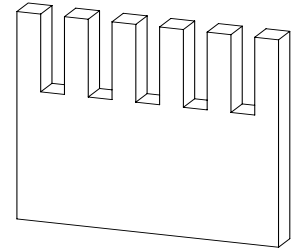
Schnittform Definition, ähnlich der Schnittfläche Definition..

Wenn $d = 0.0$, ist der Schnittkörper die X-Y-Ebene. Der Schnitt entfernt den Teil über der X-Y-Ebene.

$d < 0.0$ bedeutet einen L-förmigen Schnitt. Der Teil über der X-Y-Ebene wird durch $x \geq 0$ entfernt.

$d > 0$ bedeutet einen U-förmigen Schnitt. Ähnlich dem L-förmigen Schnitt wird der Teil über der X-Y-Ebene mit $0 < x < d$ entfernt..

```
FOR I = 1 TO 5
    ADDX 0.4 * I
    ADDZ 2.5
    CUTSHAPE 0.4
    DEL 2
    ADDX 0.4
NEXT I
DEL TOP
BRICK 4.4, 0.5, 4
FOR I = 1 TO 5
    CUTEND
NEXT I
```



CUTFORM

```
CUTFORM n, method, status,
    rx, ry, rz, d,
    x1, y1, mask1,
    ...
    xn, yn, maskn
```

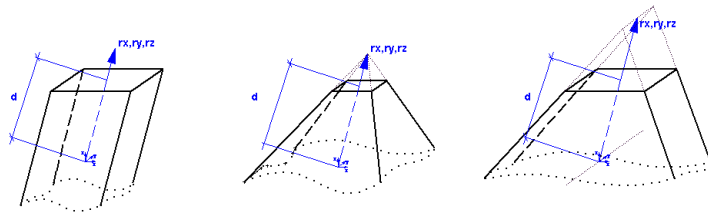
Ähnlich wie die CUTPOLYA Definition, aber mit der Möglichkeit zur Steuerung von Form und Umfang des Schnittkörpers.

method: Steuert die Form des Schnittkörpers

1: prismenförmig

2: pyramidenförmig

3: keilförmiger Schnittkörper. Die Richtung der Deckkante des Keils ist parallel zur Y-Achse und ihre Position liegt in rx, ry, rz (ry wird ignoriert.)



status: Steuert den Umfang des Schnittkörpers und die Behandlung der generierten Schnittpolygone.

status = $j1 + 2*j2 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8$

j1: verwendet die Attribute des Körpers der generierten Polygone und Kanten

j2: generierte Schnittpolygone werden als normale Polygone behandelt

j4, j5: definiert das Limit des Schnitts:

j4 = 0 und j5 = 0: endlicher Schnitt

j4 = 0 und j5 = 1: halb-unendlicher Schnitt

j4 = 1 und j5 = 1: unendlicher Schnitt

j6: generiert eine Boole'sche Schnittmenge mit dem Schnittkörper statt eines Boole'schen Unterschieds. (kann nur mit dem Befehl CUTFORM verwendet werden)

j7 : durch die Unterseite des Körpers generierte Kanten werden unsichtbar

j8 : durch die Oberseite des Körpers generierte Kanten werden unsichtbar

rx, ry, rz: definiert die Ausrichtung des Schnitts, wenn der Schnitt prismenförmig ist, oder die Spitze der Pyramide, wenn die Schnittmethode pyramidenförmig ist.

d: definiert den Abstand entlang rx, ry, rz bis zum Ende des Schnitts. Wenn der Schnitt unendlich ist, hat dieser Parameter keine Auswirkung. Handelt es sich um einen endlichen Schnitt, liegt der Beginn des Schnittkörpers am lokalen Koordinatensystem und der Körper endet in einem Abstand von d entlang der von rx, ry, rz definierten Richtung.

Ist der Schnitt halb-unendlich, liegt der Beginn des Schnittkörpers in einem Abstand von d entlang der von rx, ry, rz definierten Richtung, und die Ausrichtung des halb-unendlichen Schnitts liegt in der Gegenrichtung zu der von rx, ry, rz definierten Richtung.

mask: Definiert die Sichtbarkeit der Kanten des Schnittkörpers:

maski = $j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 64*j7$

j1: Das Polygon erzeugt eine sichtbare Kante beim Eintritt in den geschnittenen Körper

j2: Die Längskante der Schnittform ist sichtbar

j3: Das Polygon erzeugt eine sichtbare Kante beim Verlassen des geschnittenen Körpers

j4: Die untere Kante der Schnittform ist sichtbar

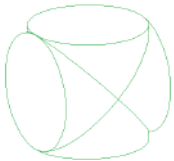
j5: Die obere Kante der Schnittform ist sichtbar

j7: Steuert die vom Blickpunkt abhängige Sichtbarkeit der Längskante

SOLID-ELEMENT-BEFEHLE

GDL kann spezielle 3D-Operationen zwischen Massivkörpern, die durch Gruppen dargestellt sind, durchführen. Diese Operationen können sein:

- - **ADDGROUP**: Bildet eine Boole'sche Verbindung aus zwei Massivkörpern;



- - **SUBGROUP**: Bildet einen Boole'schen Unterschied aus zwei Massivkörpern;



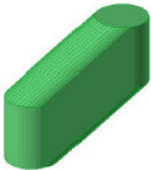
- - **ISECTGROUP**: Bildet eine Boole'sche Schnittmenge aus zwei Massivkörpern;



- - **ISECTLINES:** Berechnet die Schnittlinie aus zwei Massivkörpern;



- **SWEEPGROUP:** Wischt einen Massivkörper entlang eines Vektors.



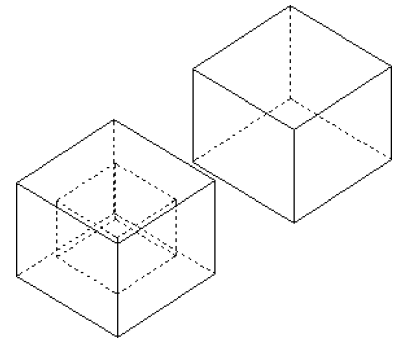
Ein GDL-Massivkörper besteht aus einem oder mehreren Blöcken, die in dem Modell als separate Körper erscheinen. Ein Block hat genau eine äußere Schale und kann Leerräume enthalten. (Leerräume lassen sich als “negative” innere Schalen innerhalb eines Blocks beschreiben.) Der Massivkörper in der Zeichnung unten besteht aus zwei Blöcken in einer Weise, dass einer davon einen Leerraum enthält. GDL-Körper wie BLOCK, SPHERE, etc. erscheinen als äußere Schalen in Gruppen. Mithilfe der folgenden Konstruktion kann der Benutzer mehrere Schalen in einem Massivkörper ablegen (beachten Sie die Anweisung BODY -1):

```
GROUP "myGroup"
  BLOCK 1,1,1
  BODY -1
  ADDX 1
  BLOCK 1,1,1
ENDGROUP
```

Der obige Massivkörper enthält zwei Blöcke; jeder dieser Blöcke besteht aus einer Schale. Leerräume können mithilfe einfacher Elemente definiert werden oder als Ergebnis eines Boole'schen Unterschieds (z. B. Subtraktion eines kleinen Würfels aus der Mitte eines großen) auftreten.

Siehe auch *“Grundelemente” auf Seite 97.*

Auch wenn Gruppenoperationen zur Verwendung mit Massivobjekten gedacht sind, können sie auch auf Oberflächen, Drahtmodellen oder Hybridmodellen angewendet werden. (Hybridmodelle sind im Wesentlichen Oberflächen, die Kanten mit benachbarten Flächen enthalten können.) Die Ergebnisse der Operationen mit solchen Modellen sind in den folgenden Tabellen zusammengefasst:



Vereinigung (Basis-Werkzeug)

	Massivbasis	Oberflächenbasis	Drahtmodellbasis	Hybridbasis
Massivwerkzeug	Massivergebnis	Oberflächenenergebnis (dazuladen)	Drahtmodellenergebnis (dazuladen)	Hybridergebnis (dazuladen)
Oberflächenwerkzeug	Oberflächenenergebnis (dazuladen)	Oberflächenenergebnis (dazuladen)	Hybridergebnis (dazuladen)	Hybridergebnis (dazuladen)
Drahtmodellwerkzeug	Drahtmodellenergebnis (dazuladen)	Hybridergebnis (dazuladen)	Drahtmodellenergebnis (dazuladen)	Hybridergebnis (dazuladen)
Hybridwerkzeug	Hybridergebnis (dazuladen)	Hybridergebnis (dazuladen)	Hybridergebnis (dazuladen)	Hybridergebnis (dazuladen)

Abzug (Basis \ Werkzeug)

	Massivbasis	Oberflächenbasis	Drahtmodellbasis	Hybridbasis
Massivwerkzeug	Massivergebnis	Oberflächenenergebnis	Drahtmodellenergebnis	Hybridergebnis
Oberflächenwerkzeug	Massivbasis (keine Auswirkung)	Oberflächenbasis (keine Auswirkung)	Drahtmodellbasis (keine Auswirkung)	Hybridbasis (keine Auswirkung)
Drahtmodellwerkzeug	Massivbasis (keine Auswirkung)	Oberflächenbasis (keine Auswirkung)	Drahtmodellbasis (keine Auswirkung)	Hybridbasis (keine Auswirkung)
Hybridwerkzeug	Massivbasis (keine Auswirkung)	Oberflächenbasis (keine Auswirkung)	Drahtmodellbasis (keine Auswirkung)	Hybridbasis (keine Auswirkung)

Schnittmenge (Basis -Werkzeug)

	Massivbasis	Oberflächenbasis	Drahtmodellbasis	Hybridbasis
Massivwerkzeug	Massivergebnis	Oberflächenenergebnis	Drahtmodellenergebnis	Hybridergebnis
Oberflächenwerkzeug	Oberflächenenergebnis	keine Auswirkung	keine Auswirkung	keine Auswirkung
Drahtmodellwerkzeug	Drahtmodellenergebnis	keine Auswirkung	keine Auswirkung	keine Auswirkung
Hybridwerkzeug	Hybridergebnis	keine Auswirkung	keine Auswirkung	keine Auswirkung

Schnittlinien (Basis-Werkzeug)

	Massivbasis	Oberflächenbasis	Drahtmodellbasis	Hybridbasis
Massivwerkzeug	Drahtmodellergebnis	Drahtmodellergebnis	keine Auswirkung	Drahtmodellergebnis
Oberflächenwerkzeug	Drahtmodellergebnis	keine Auswirkung	keine Auswirkung	keine Auswirkung
Drahtmodellwerkzeug	keine Auswirkung	keine Auswirkung	keine Auswirkung	keine Auswirkung
Hybridwerkzeug	Drahtmodellergebnis	keine Auswirkung	keine Auswirkung	keine Auswirkung

Wischen

Massivkörper	Oberfläche	Drahtmodell	Hybrid
gültiges Ergebnis	Oberflächenbasis (keine Auswirkung)	Drahtmodellbasis (keine Auswirkung)	Hybridbasis (keine Auswirkung)

Oberfläche können durch Verwendung des Befehls MODEL SURFACE explizit generiert werden oder implizit durch Auslassen der Polygone mit nicht benachbarten Flächen aus dem Modell. Drahtmodelle werden durch Verwendung der Anweisung MODEL WIRE erzeugt oder durch Definieren von Objekten ohne Flächenpolygone. Hybridmodelle können nur indirekt generiert werden durch Auslassen der Polygone mit nicht benachbarten Flächen aus dem Modell.

In der Mehrzahl der Fälle ist das Modell ein Massivmodell. GDL-Körper erscheinen in Gruppendefinitionen als Schalen; um eine schnelle und zuverlässige Operation zu erzielen, ist daher die geometrische Genauigkeit der generierten Schalen ein kritischer Faktor. Durch die Verarbeitung degenerierter Objekte wird die GDL Engine geladen, und die Ausführung der gewünschten Operation dauert länger. Die wichtigste Regel hinsichtlich der effizienten Verwendung der GDL-Gruppenoperationen lässt sich wie folgt formulieren: *Modellieren durch Einhalten der vorhandenen physischen Präsenz räumlicher Objekte*. In der Praxis lässt sich dies durch folgende Richtlinien ausdrücken:

- - Vermeiden selbstschneidender Objekte.
- - Vermeiden selbstberührender Objekte (kleine Abstände anwenden).
- - Vermeiden von Nullgrößen bei Objekten (geringe Stärke anwenden).

Entsprechend diesen Grundsätzen sollten für Schalen (durch Körper definiert) die folgenden Regeln eingehalten werden, nicht jedoch für Massivkörper (durch Gruppen definiert). (Der mit dem Script in der Gruppenkonstruktion erzeugte Massivkörper ist korrekt modelliert, da sich die Schalen, aus denen er besteht, berühren, aber geometrisch korrekt sind.)

GROUP

GROUP "name"

Beginnt einer neuen Gruppendefinition. Alle Körper bis zur nächsten Anweisung **ENDGROUP** sind Bestandteil der Gruppe "Name". Gruppen werden nicht tatsächlich generiert (platziert), sie können in Gruppenoperationen verwendet oder explizit mit dem Befehl **PLACEGROUP** platziert werden. Gruppendefinitionen können nicht verschachtelt werden, es können jedoch Makroaufrufe eingeschlossen werden, die die Gruppendefinitionen und **PLACEGROUP**-Befehle enthalten, die wiederum weitere Gruppen verwenden.

Gruppennamen müssen innerhalb des aktuellen Skripts eindeutig sein. Transformationen, Schnitte außerhalb der Gruppendefinition haben keine Auswirkungen auf die Gruppenteile; Transformationen, Schnitte innerhalb haben keine Auswirkung auf die Körper außerhalb der Definition. Gruppendefinitionen sind für die Attributanweisungen **DEFINE** und **SET** transparent (Stifte, Material, Schraffuren); Attribute, die vor der Definition definiert/gesetzt oder die innerhalb der Definition definiert/gesetzt wurden, sind wirksam.

ENDGROUP

ENDGROUP

Ende einer Gruppendefinition.

ADDGROUP

ADDGROUP (g_expr1, g_expr2)

SUBGROUP

ADDGROUP (g_expr1, g_expr2)

SECTGROUP

ISECTGROUP (g_expr1, g_expr2)

SECTLINES

ISECTLINES (g_expr1, g_expr2)

Gruppenoperationen: Hinzufügen, Subtrahieren, Schnittmenge, Schnittlinien. Der Rückgabewert ist eine neue Gruppe, die mit dem Befehl **PLACEGROUP** platziert, in einer Variablen gespeichert oder als Parameter in einer anderen Gruppenoperation verwendet werden kann. Gruppenoperationen können mit zuvor definierten Gruppen ausgeführt werden oder mit Gruppen, die aus anderen Gruppenoperationen stammen. g_expr1, g_expr2 sind Ausdrücke des Typs Gruppe. Ausdrücke des Typs Gruppe sind entweder Gruppennamen (Zeichenfolge-Ausdrücke) oder Variablen des Typs Gruppe oder eine beliebige Kombination dieser Ausdrücke, die Gruppen ergeben.

PLACEGROUP

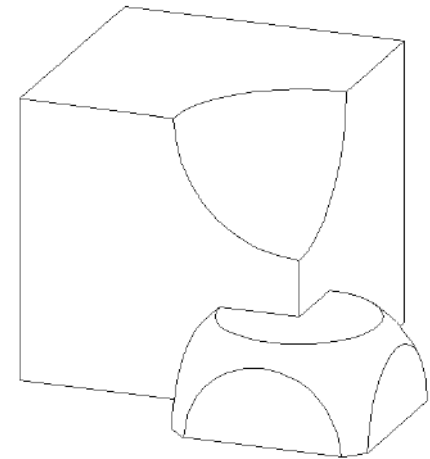
PLACEGROUP *g_expr*

Das Platzieren einer Gruppe ist die Operation, in der Körper tatsächlich generiert werden. Schnitte und Transformationen sind wirksam, der Gruppenausdruck wird ausgewertet, und die resultierenden Körper werden in den 3D-Datenstrukturen abgelegt.

KILLGROUP

KILLGROUP *g_expr*

Löscht die Körper der angegebenen Gruppe aus dem Speicher. Nach einer Operation KILLGROUP ist die Gruppe leer. Das Löschen erfolgt automatisch am Ende der Interpretation oder nach der Rückkehr aus dem Makroaufruf. Aus Gründen der besseren Leistung sollte dieser Befehl verwendet werden, wenn eine Gruppe nicht mehr benötigt wird.



Beispiel:

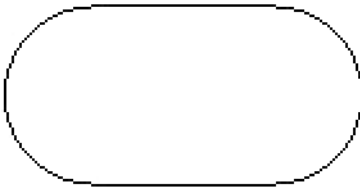
```
GROUP "box"
  BRICK 1, 1, 1
ENDGROUP
GROUP "sphere"
  ADDZ 1
  SPHERE 0.45
  DEL 1
ENDGROUP
GROUP "semisphere"
  ELLIPS 0.45, 0.45
ENDGROUP
GROUP "brick"
  ADD -0.35, -0.35, 0
  BRICK 0.70, 0.70, 0.35
  DEL 1
ENDGROUP
! Abzug der "sphere" aus dem "box" result_1=SUBGROUP("box","sphere") ! Schnittmenge der
"semisphere" und des "brick" result_2=ISECTGROUP("semisphere","brick") ! Hinzufügen des
generierten Resultats result_3=ADDGROUP(result_1,result_2) PLACEGROUP result_3
KILLGROUP "box"
KILLGROUP "sphere"
KILLGROUP "semisphere"
KILLGROUP "brick"
```

SWEEPGROUP

SWEEPGROUP (g_expr, x, y, z)

Ergibt eine Gruppe, die durch Wischen des Gruppenparameters entlang der angegebenen Ausrichtung erstellt wird. Der Befehl wirkt nur für Massivmodelle.

Beispiel:



```
GROUP "a"
  SPHERE 1
ENDGROUP
PLACEGROUP SWEEPGROUP ("a", 2, 0, 0)
```

BINÄRES 3D

BINARY

BINARY mode [, section]

Spezieller Befehl, um binäre inlinie Objekte in ein GDL Makro einzuschließen. Ein Satz von Kreuzungspunkten, Vektoren, Kanten, Polygonkörpern und Materialien wird aus einem speziellen Teil der Bibliothekselement-Datei eingelesen. Sie werden den aktuellen Transformationen unterworfen und mit den übrigen Daten im 3D-Modell verschmolzen. Die reinen Binär-Daten des Objektes können nicht vom Anwender editiert werden.

mode: Bestimmt die Verwendung von Stiftfarben- und Material-Attributen.

0: die aktuellen PEN und MATERIAL Einstellungen gelten.

1: die aktuellen PEN und MATERIAL Einstellungen sind nicht gültig. Das Bibliothekselement wird mit den gespeicherten Farben- und Material-Definitionen dargestellt. Das Erscheinen der Oberfläche ist konstant.

2: die gespeicherten PEN und MATERIAL Einstellungen werden benutzt, undefinierte Materialien werden durch aktuelle Einstellungen ersetzt.

3: die gespeicherten PEN und MATERIAL Einstellungen werden benutzt, undefinierte Materialien werden durch die gespeicherten voreingestellten Attribute ersetzt.

section: Index des Binär-Teiles, von 1 bis 16

Wenn Sie 0 für den Schnitt-Index verwenden, können Sie sich gleichzeitig auf sämtliche vorhandene Binär-Teile beziehen. .

Nur Schnitte (section) mit dem Index-Wert 1 können innerhalb von GDL abgespeichert werden, auch BINARY- Befehle ohne Schnittanmerkung beziehen sich darauf. Die anderen Schnitt-Indizes werden von Drittanbieter-Werkzeugen verwendet.

Wenn Sie Dateien öffnen, die von der Datenstruktur abweichen (z.B. DXF, ZOOM), wird ihre 3D-Beschreibung ins Binär-Format transformiert.

Sie können von dem Hauptfenster Bibliothekselemente über den Befehl Sichern als..abspeichern. Falls die Kontrollbox *Sichern in Binär-Format* in dem *Sichern als* Dialogfenster markiert ist, wird der GDL-Text des aktuellen Bibliothekselementes durch eine binäre Beschreibung ersetzt.

Hinweis: Wenn Sie ein 3D -Modell nach einem Schnitt in Binär-Format abspeichern, wird das geschnittene Modell gesichert. Auf diese Weise können Sie geschnittene Körper erstellen.

Sie können Ihr Bibliothekselement nur dann im Binär-Format abspeichern, wenn dessen 3D-Modell bereits erstellt wurde, d.h. wenn Sie die 3D-Ansicht des Elementes vorher wenigstens einmal aufgebaut hatten.

Durch Ersetzen der GDL- Beschreibung Ihres Bibliothekselementes mit einer binären Beschreibung können Sie die 3D -Transformationszeit bedeutend reduzieren. Andererseits ist die binäre 3D-Beschreibung nicht parametrisch und benötigt mehr freien Platz auf der Festplatte als ein algorithmisches GDL- Programm.

2D ELEMENTE

In diesem Kapitel werden die Befehle vorgestellt, mit den zweidimensionale Formen von einfachen Linien und Bögen bis zu komplexen Polygonen und Splines erstellt und Textelemente in 2D definiert werden können. Außerdem behandelt es die Handhabung binärer Daten in 2D und die Projektion einer in einem 3D-Script erstellten Form in eine 2D-Ansicht, wodurch der Zusammenhang zwischen der zwei- und dreidimensionalen Erscheinung eines Objektes gewahrt bleibt. Durch weitere Befehle können graphische Elemente in Elementlisten eingefügt werden, die für Berechnungen erstellt werden.

ZEICHNUNGSELEMENTE

HOTSPOT2

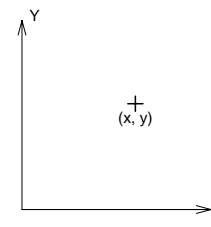
HOTSPOT x, y, z [, unID [, paramReference, flags] [, displayParam]]

unID ist der eindeutige Kennzeichner des Fixpunkts im 2D-Script. Er ist nützlich, wenn Sie eine variable Anzahl von Fixpunkten haben.

paramReference: Parameter, der von diesem Hotspot mit der Parameter-Bearbeitungsmethode auf der Basis grafischer Hotspots bearbeitet werden kann.

paramReference: Parameter, der durch diesen Fixpunkt über die Methode Parameterbearbeitung mithilfe grafischer Fixpunkte bearbeitet werden kann. Es können auch Elemente von Arrays übergeben werden.

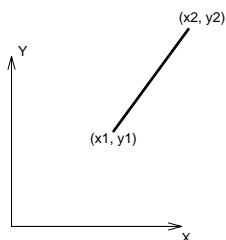
Siehe *“Grafische Bearbeitung” auf Seite 135* für weitere Informationen über HOTSPOT2.



LINE2

LINE2 $x1, y1, x2, y2$

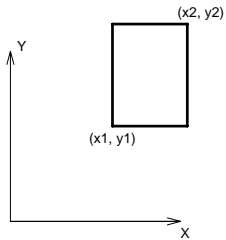
Definition von Linien zwischen zwei Punkten.



RECT2

RECT2 $x1, y1, x2, y2$

Definition von Rechtecken durch zwei Eckpunkten.



POLY2

PRISM $n, h, x1, y1, \dots, xn, yn$

Ein offenes oder geschlossenes Polygon mit n Eckpunkten.

Einschränkung der Parameter:

$n \geq 2$

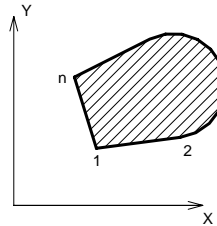
$frame_fill = j1 + 2*j2 + 4*j3$

hierbei können $j1, j2, j3$ jeweils 0 oder 1 sein.

$j1$ (1): nur Kontur

$j2$ (2): nur Schraffur

$j3$ (4): Schließen eines offenen Polygons



POLY2_

POLY2_ *n*, *frame_fill*, *x1*, *y1*, *s1*, ... *xn*, *yn*, *sn*

Gleicht dem einfachen POLY2-Befehl, jedoch kann die Sichtbarkeit jeder beliebigen Kante unterdrückt werden. Ist $s_i = 0$, ist die Kante ausgehend vom Punkt (x_i, y_i) unsichtbar. Ist $s_i = 1$, sollte der Kreuzungspunkt sichtbar sein. Mit $s_i = -1$ werden Durchbrüche direkt dargestellt. Sie können mit zusätzlichen Statuscodewerten auch Bögen und Segmente in der Polylinie definieren.

Einschränkung der Parameter:

$n \geq 2$

$\text{frame_fill} = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 32*j_6 + 64*j_7$

hierbei können j_1, j_2, j_3 jeweils 0 oder 1 sein.

j_1 (1): nur Kontur

j_2 (2): nur Schraffur

j_3 (4): Schließen eines offenen Polygons

j_4 (8): lokale Schraffurausrichtung

j_6 : Schraffur ist Bauteilschraffur (Grundeinstellung ist Zeichenschraffur)

j_7 : Schraffur ist Deckschraffur (nur falls $j_6 = 0$, Grundeinstellung ist Zeichenschraffur)

Statuswerte:

$s = j_1 + 16*j_5 + 32*j_6$

wobei j_1, j_5, j_6 0 oder 1 sein können.

j_1 (1): das nächste Segment ist sichtbar

j_5 (16): das nächste Segment ist eine Innenlinie (wenn 0, spezifische Linie)

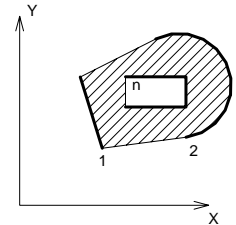
j_6 (32): das nächste Segment ist eine Kontur (nur wirksam, wenn j_5 nicht eingestellt ist)

-1: Ende einer Kontur

Grundeigenschaft für POLY2_ Linien ist 0 (spezifische Linie), der Befehl LINE_PROPERTY hat keinen Einfluss auf POLY2_ Kanten.

Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie mithilfe spezieller Beschränkungen.

Siehe *“Zusätzliche Statuscodes” auf Seite 143 für mehr Details.*



POLY2_A

```
POLY2_A n, frame_fill, fill_pen,  
        x1, y1, s1, ..., xn, yn, sn
```

POLY2_B

```
POLY2_B n, frame_fill, fill_pen,  
        fill_background_pen,  
        x1, y1, s1, ..., xn, yn, sn
```

Erweiterte Version des Befehls POLY2_B mit weiteren Parametern der Stift-Schraffur und des Hintergrundstiftes. Alle anderen Parameter entsprechen denen, die unter dem Befehl POLY2_ beschrieben wurden. Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie mithilfe spezieller Beschränkungen.

Siehe *“Zusätzliche Statuscodes” auf Seite 143 für mehr Details.*

POLY2_B{2}

```
POLY2_B{2} n, frame_fill, fill_pen,  
            fill_background_pen,  
            fillOrigoX, fillOrigoY,  
            fillAngle,  
            x1, y1, s1, ..., xn, yn, sn
```

Erweiterte Version des Befehls POLY2_B, in der Stift-Schraffur, Hintergrundstift, Ursprung und Ausrichtung definiert werden können.

$\text{frame_fill} = j1 + 2*j2 + 4*j3 + 8*j4$

hierbei können $j1, j2, j3, j4$ jeweils 0 oder 1 sein.

$j1$ (1): nur Kontur

$j2$ (2): nur Schraffur

$j3$ (4): Schließen eines offenen Polygons

$j4$ (8): lokale Schraffurausrichtung

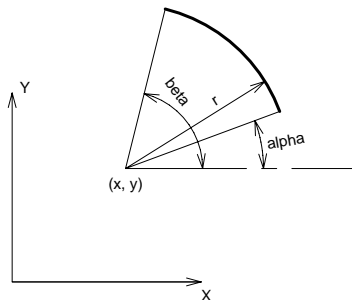
Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie mithilfe spezieller Beschränkungen.

Siehe *“Zusätzliche Statuscodes” auf Seite 143 für mehr Details.*

ARC2

ARC2 x, y, r, α, β

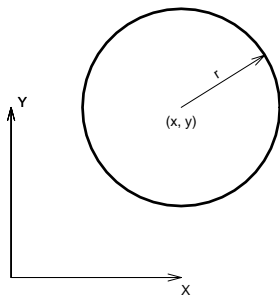
Ein Bogen zwischen den Winkeln α und β , einem Mittelpunkt bei (x, y) und dem Radius r .
 α und β werden in Grad angegeben.



CIRCLE2

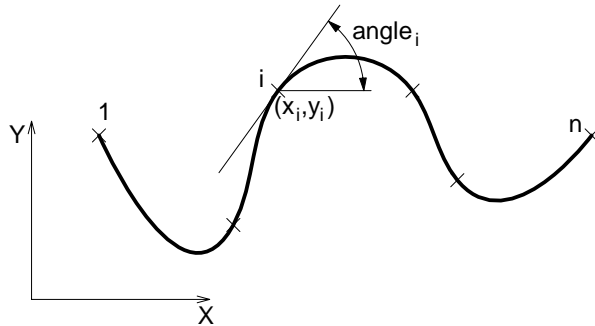
CIRCLE2 x, y, r

Ein Kreis mit einem Mittelpunkt bei (x, y) , und einem Radius r .



SPLINE2

SPLINE2 *n*, *status*, *x1*, *y1*,
angle1, ..., *xn*, *yn*, *anglen*



Einschränkung:

$n \geq 2$

Spline mit n Kontrollpunkten. Die Tangente des Splines am Kontrollpunkt (x_i, y_i) wird durch den Winkel i , bezogen auf die x-Achse, in Grad definiert.

Statuswerte:

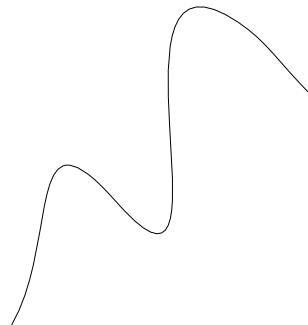
0: standard

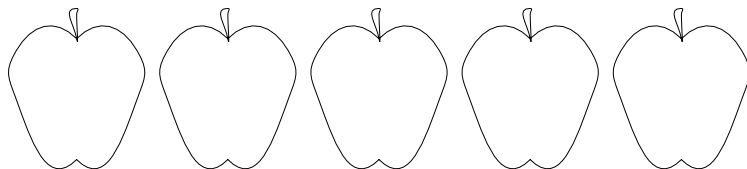
1: geschlossener Spline; der erste und letzte Punkt des Splines werden verbunden, wodurch dieses geschlossen wird.

2: automatisch geglätteter Spline; die Winkelwerte der Eckpunkte, die zwischen dem ersten und dem letzten Eckpunkt liegen, werden bei der Erzeugung des Splines nicht berücksichtigt. Es wird ein interner automatischer Glättalgorithmus benutzt.

Beispiele:

```
SPLINE2 5, 2,
         0, 0, 60,
         1, 2, 30,
         1.5, 1.5, -30,
         3, 4, 45,
         4, 3, -45
```





```

n = 5
FOR I = 1 TO n
  SPLINE2 4, 0,
    0.0, 2.0, 135.0,
    -1.0, 1.8, 240.0,
    -1.0, 1.0, 290.0,
    0.0, 0.0, 45.0
  MUL2 -1.0, 1.0
  SPLINE2 4, 0,
    0.0, 2.0, 135.0,
    -1.0, 1.8, 240.0,
    -1.0, 1.0, 290.0,
    0.0, 0.0, 45.0
  DEL 1
  SPLINE2 4, 0,
    0.0, 2.0, 100.0,
    0.0, 2.5, 0.0,
    0.0, 2.4, 270.0,
    0.0, 2.0, 270.0
  ADD2 2.5, 0
NEXT I

```

SPLINE2A

```

SPLINE2A n, status, x1, y1, angle1, length_previous1, length_next1,
  ...
  xn, yn, anglen, length_previousn,
  length_nextn

```

Erweiterung des Befehls SPLINE2 (Bézier Spline), der wegen seiner Komplexität vor allem bei der automatischen 2D-Scripterstellung eingesetzt wird.

Für weitere Details, siehe *“Splines zeichnen” im ArchiCAD 9 Referenzhandbuch*

Statuscodes

0: standard

1: geschlossener Spline; der erste und letzte Punkt des Splines werden verbunden, wodurch dieser geschlossen wird.

2: automatisch geglätteter Spline; die Parameter für angle, length_previous und length_next der Eckpunkte, die zwischen dem ersten und dem letzten Eckpunkt liegen, werden bei der Erzeugung des Splines nicht berücksichtigt. Es wird ein interner automatischer Glättalgorithmus benutzt.

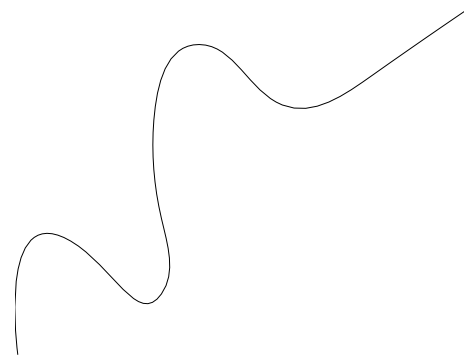
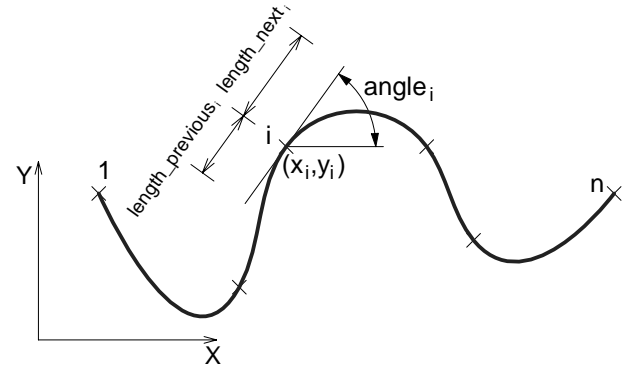
x_i, y_i : Kontrollpunkt-Koordinaten

length_previous_i, length_next_i: Längen der Tangenten des vorherigen und des folgenden Kontrollpunktes

angle_i: Richtungswinkel der Tangente

Beispiel:

```
SPLINE2A 9, 2,
0.0, 0.0, 0.0, 0.0, 0.0,
0.7, 1.5, 15, 0.9, 1.0,
1.9, 0.8, 72, 0.8, 0.3,
1.9, 1.8, 100, 0.3, 0.4,
1.8, 3.1, 85, 0.4, 0.5,
2.4, 4.1, 352, 0.4, 0.4,
3.5, 3.3, 338, 0.4, 0.4,
4.7, 3.7, 36, 0.4, 0.8,
6.0, 4.6, 0, 0.0, 0.0
```



PICTURE2

PICTURE expression, a, b, mask

DRAWING3{2}

PICTURE2{2} expression, a, b, mask

Kann in 2D ähnlich eingesetzt werden, wie der Befehl PICTURE in 3D. Im Unterschied zu 3D, haben die Maskierungswerte keine Auswirkungen auf die 2D-Bilder.

'expression' steht für einen Dateiname, einen numerischen Ausdruck oder einen Index eines im Bibliothekselement abgelegten Bildes. Ein 0-Index ist ein spezieller Wert, der sich auf das Vorschaubild des Bibliothekselements bezieht. Bei PICTURE2{2} bedeutet mask = 1, dass exakt weiße Pixel transparent sind. Andere Bilder können in Bibliothekselementen nur dann gespeichert werden, wenn das Projekt oder die ausgewählten Elemente, die die Bilder beinhalten, als GDL-Objekte gesichert werden.

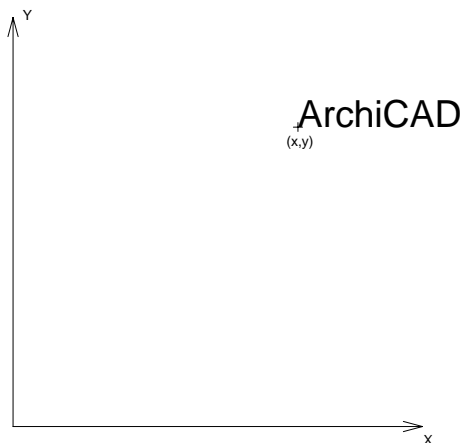
TEXTELEMENT

TEXT2

TEXT2 x, y, expression

Eine Textkonstante oder ein errechneter Wert. Die Position des Anfangsbuchstaben ist x,y. Die aktuellen Stil-Einstellungen werden verwendet.

Siehe auch Befehle *“[SET] STYLE” auf Seite 157* *“DEFINE STYLE” auf Seite 176*.



RIGHTTEXT2

RIGHTTEXT2 x, y, textblock_name

Platzieren Sie einen zuvor definierten TEXTBLOCK.

Für weitere Details, siehe *“Textblock” auf Seite 179*

x, y: X- und Y-Koordinaten der Richtext-Position

textblock_name: Name eines früher definierten TEXTBLOCK

BINÄRES 2D

FRAGMENT2

FRAGMENT2 fragment_index,
use_current_attributes_flag

Das 2D-Element mit dem angegebenen Index wird mit den aktuellen Transformationen in das 2D-Vollbild eingefügt.

use_current_attributes_flag: legt fest, ob die aktuellen Attribute benutzt werden oder nicht

0: Das 2D-Element wird mit den individuell dafür definierten Einstellungen für Farbe, Linientyp und Schraffurtyp dargestellt.

1: Die aktuellen Einstellungen des 2D-Scripts werden anstelle der individuellen Einstellungen des Elementes für die Darstellung verwendet.

FRAGMENT2

FRAGMENT2 ALL, use_current_attributes_flag

Alle 2D-Elemente werden mit den aktuellen Transformationen in das 2D-Vollbild eingefügt.

use_current_attributes_flag: legt fest, ob die aktuellen Attribute benutzt werden oder nicht

0: Das 2D-Element wird mit den individuell dafür definierten Einstellungen für Farbe, Linientyp und Schraffurtyp dargestellt.

1: Die aktuellen Einstellungen des 2D-Scripts werden anstelle der individuellen Einstellungen des Elementes für die Darstellung verwendet.

3D PROJEKTIONEN IN 2D

PROJECT2

PROJECT2 projection_code, angle, method

PROJEKT2{2}

PROJEKT2{2} projection_code, angle, method [, backgroundColor, fillOrigoX, fillOrigoY, fillldirection]

Erstellt eine Projektion des 3D-Skripts in demselben Bibliothekselement und addiert die erzeugten Linien zum parametrischen 2D Symbol. Die zweite Version von PROJEKT2{2} ermöglicht dem Benutzer zusammen mit einem vorherigen Befehl SET FILL die Steuerung der Hintergrundschraffur, des Ursprungs und die Steuerung der aus dem 2D-Script resultierenden Zeichnung.

projection_code: Projektionsart

3: Draufsicht

4: Seitenansicht

6: Vorderaxonometrie

7: Isometrische Axonometrie

8: Monometrische Axonometrie

9: Dimetrische Axonometrie

-3: Unteransicht

-6: Vorderseite Bodenansicht

-7: Isometrische Bodenansicht

-8: Monometrische Bodenansicht

-9: Dimetrische Bodenansicht

angle: Blickwinkel, gleicht mit dem Dialogfenster Blickpunkt & Projektionsart .

method: die ausgewählte Visualisierungsmethode

1: Drahtmodell

2: Verdeckte Kanten (analytisch)

3: Schattierung

16: Zusätzlicher Modifizierer, zeichnet Vektorschraffuren (nur wirksam bei verdeckten Linien und Schattierungs-Modus)

32: Zusätzlicher Modifizierer, verwendet aktuelle Attribute statt der Attribute aus 3D (nur wirksam im Schattierungsmodus)

64: Zusätzlicher Modifizierer, lokale Schraffurausrichtung (nur wirksam im Schattierungsmodus)

128: Zusätzlicher Modifizierer: Linien umfasst alle inneren Linie (nur mit 32 zusammen wirksam). Grundeinstellung ist generisch.

256: Zusätzlicher Modifizierer: Linien umfasst alle Konturen (nur mit 32 zusammen wirksam, wenn 128 nicht aktiviert wurde). Grundeinstellung ist generisch.

512: Zusätzlicher Modifizierer: alle Schraffuren sind Bauteilschraffuren (nur mit 32 zusammen wirksam). Grundeinstellung ist Zeichnungsschraffuren.

1024: Zusätzlicher Modifizierer: alle Schraffuren sind Deckschraffuren (nur mit 32 zusammen wirksam, wenn 256 nicht aktiviert wurde). Grundeinstellung ist Zeichnungsschraffuren.

BackgroundColor: Hintergrundfarbe der Schraffur

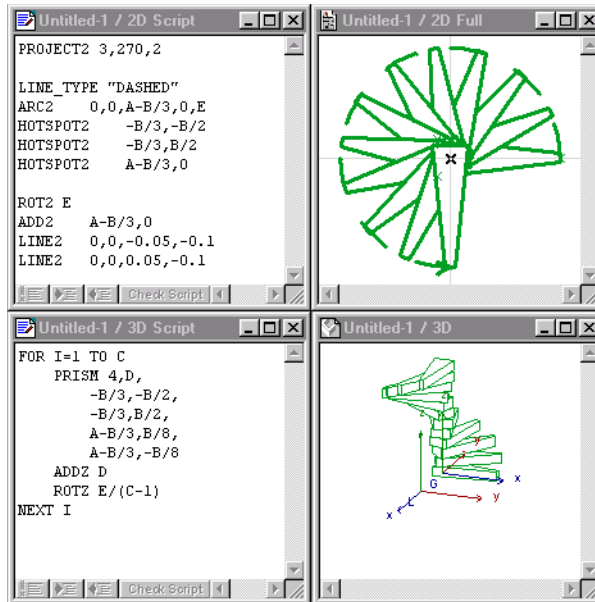
fillOrigoX: X-Koordinate des Schraffurursprungs

fillOrigoY: Y-Koordinate des Schraffurursprungs

filldirection: Ausrichtung des Schraffurwinkels

Anmerkung: SET FILL ist wirksam für PROJECT2{2}

Beispiel:



ZEICHNUNGEN IN DER LISTE

Diese Befehle sind nur wirksam, wenn eine Elementenliste in ArchiCAD erstellt wird.

Ist das Element ein spezielles Eigenschaften-Bibliothekselement und ist irgendwie mit einem im Grundriß plazierten Bibliothekselement (Objekt, Tür, Fenster oder Licht) verbunden und beinhaltet die folgenden Befehle in seinem 2D-Script, nimmt es auf den 2D- und 3D-Teil jenes Bibliothekselementes Bezug. Das ist eine virtuelle Referenz, die während des Listenprozesses unter Verwendung des 2D- oder 3D-Scriptes des gegenwärtig aufgelisteten Elementes aufgelöst wird.

DRAWING2

DRAWING2 [expression]

Erstellt abhängig von dem Wert der Variablen eine Zeichnung des Bibliothekselements (expression = 0, Vorgabe) oder das Etikett des Elements (expression = 1), das mit dem Eigenschafts-Objekt, das diesen Befehl enthält, assoziiert ist.

DRAWING3

DRAWING3 projection_code, angle, method

DRAWING3{2}

DRAWING3{2} projection_code, angle, method [,backgroundColor, origoX, origoY, fillldirection]

Erstellt ähnlich wie PROJECT2, eine Projektion des 3D-Scriptes des Bibliothekselementes, dass dem eigenschaftsbibliothekselement zugeordnet ist, dass diesen Befehl enthält. Alle Parameter gleichen den beim PROJECT2-Befehl und PROJECT2{2} beschriebenen Parametern.

"Neue Methode"-Flags in DRAWING3{2}:

3: Schattierung

32: aktuelle Attribute verwenden statt der Attribute aus 3D

64: lokale Schraffurausrichtung

GRAFISCHE BEARBEITUNG

Hotspot-basiertes interaktives grafisches Bearbeiten der GDL-Parameter des Typs Länge und Winkel.

BEARBEITUNGSBEFEHLE AUF HOTSPOT-BASIS

HOTSPOT

HOTSPOT *x*, *y*, *z* [, *unID* [, *paramReference*, *flags*] [, *displayParam*]]

HOTSPOT *x*, *y*, *z* [, *unID* [, *paramReference*, *flags*] [, *displayParam*]]

unID: eindeutige ID-Nr.; muss innerhalb der im Bibliothekselement definierten Hotspots eindeutig sein.

paramReference: Parameter, der von diesem Hotspot mit der Parameter-Bearbeitungsmethode auf der Basis grafischer Hotspots bearbeitet werden kann.

paramReference: Parameter, der durch diesen Fixpunkt über die Methode Parameterbearbeitung mithilfe grafischer Fixpunkte bearbeitet werden kann. Es können auch Elemente von Arrays übergeben werden.

Beispiele gültiger Argumente:

D, *Arr*[5], *Arr*[2*I+3][D+1], etc.

flags: Hotspottyp + Attribute,

Typ :

Typ: 1: Typ Längenbearbeitung, Basis-Hotspot

2: Typ Längenbearbeitung, beweglicher Hotspot

3: Typ Längenbearbeitung, Referenz-Hotspot (immer verborgen)

4: Typ Winkelbearbeitung, Basis-Hotspot

5: Typ Winkelbearbeitung, beweglicher Hotspot

6: Typ Winkelbearbeitung, Mittelpunkt des Winkels (immer verborgen)

7: Typ Winkelbearbeitung, Referenz-Hotspot (immer verborgen)

Attribut kann eine Kombination der folgenden Werte oder Null sein:

128: Hotspot verbergen (wichtig für Typen: 1,2,4,5)

256: Bearbeitbarer Basis-Hotspot (für Typen: 1,4)

512: Umkehrung des Winkels in 2D (für Typ 6)

Zur Bearbeitung eines Parameters des Typs Längenbearbeitung müssen drei Hotspots definiert werden mit den Typen 1, 2 und 3. Die positive Richtung der Bearbeitungslinie ist durch den Vektor vom Referenz-Hotspot zum Basis-Hotspot vorgegeben. Der bewegliche Hotspot muss entlang dieser Linie platziert werden in einem Abstand, der durch den Parameterwert, gemessen vom Basis-Hotspot, bestimmt ist.

Zur Bearbeitung eines Parameters des Typs Winkelbearbeitung müssen vier Hotspots definiert werden mit den Typen 4, 5, 6 und 7. Die Winklebene ist rechtwinklig zu dem Vektor vom Mittelpunkt-Hotspot zum Referenz-Hotspot. Die positive Richtung bei der Winkelmessung ist gegen den Uhrzeigersinn, wenn die Ebene vom Referenz-Hotspot aus betrachtet wird. In 2D ist die Ebene bereits gegeben, daher wird der Referenz-Hotspot ignoriert, und die positive Richtung der Winkelmessung ist standardmäßig gegen den Uhrzeigersinn. Dies kann in die Richtung "Im Uhrzeigersinn" geändert werden durch Einstellen des Attribut-Flags 512 für den Mittelpunkt-Hotspot (Typ 6). Für eine ausreichende Konsistenz müssen die Vektoren vom Mittelpunkt-Hotspot zu den beweglichen und Basis-Hotspots rechtwinklig zu dem Vektor vom Mittelpunkt zum Referenz-Hotspot sein. Der bewegliche Hotspot muss in einem Winkel platziert werden, der durch den zugehörigen Parameterwert, gemessen vom Basis-Hotspot um den Mittelpunkt-Hotspot herum, bestimmt ist.

Wenn mehrere Einstellungen von Hotspots zur Bearbeitung des gleichen Parameters definiert sind, sind Hotspots in der Reihenfolge der Ausführung der Hotspot-Befehle gruppiert. Wenn das bearbeitbare Attribut für einen Basis-Hotspot eingestellt ist, kann der Benutzer den Parameter auch durch Verschieben des Basis-Hotspots bearbeiten. Da der Basis-Hotspot im Koordinatenrahmen des Objekts fixiert sein sollte (d. h. seine Position muss unabhängig von dem zugeordneten Parameter sein), wird das gesamte Objekt zusammen mit dem Basispunkt gezogen bzw. gedreht. (Bei einer Änderung des Parameterwerts ändert der bewegliche Hotspot seine Position nicht.)

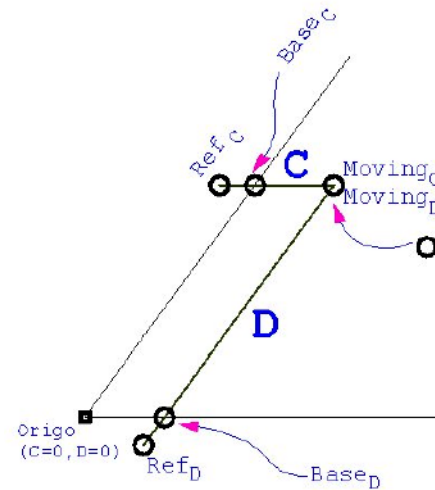
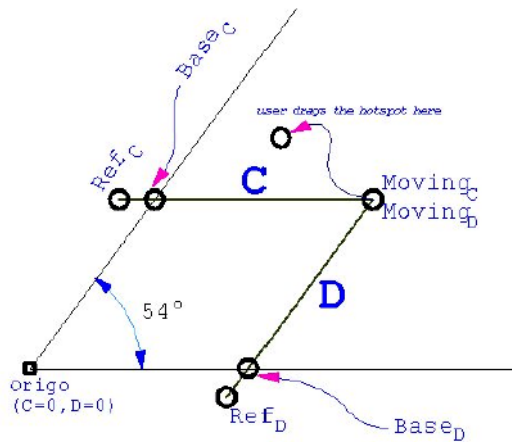
Zwei oder drei Hotspot-Sets des Typs Länge können kombiniert werden, um das Bearbeiten von zwei oder drei Parametern mit nur einem Verschiebevorgang zu ermöglichen. Werden zwei Hotspots kombiniert, ist die Bewegung des Hotspots nicht mehr auf eine Linie begrenzt, sondern auf die Ebene, die durch die beiden Linien der Sets von Längenbearbeitungs-Hotspots bestimmt ist. In 3D ermöglicht die Kombination der drei Sets von Längenbearbeitungs-Hotspots das Platzieren des Hotspots an einer beliebigen Stelle im Raum. Die beiden Linien dürfen nicht parallel zueinander sein, und die drei Linien dürfen nicht auf der gleichen Ebene liegen. Ein Bearbeiten kombinierter Parameter wird gestartet, wenn an der Position des ausgewählten Punkts zwei oder drei bearbeitbare Hotspots (bewegliche oder bearbeitbare Basis) mit unterschiedlichen zugeordneten Parametern liegen. Bei Parametern, die für eine kombinierte Bearbeitung konzipiert wurden, sind Basis- und Referenz-Hotspots im Koordinatenrahmen nicht fixiert, sondern werden bei der Änderung anderer Parameterwerte verschoben.

Siehe hierzu die Illustration und Beispiel 2.

Beispiel 1, Winkelbearbeitung in 2D:

```
LINE2 0, 0, A, 0
LINE2 0, 0, A*COS(angle), A*SIN(angle)
ARC2 0, 0, 0.75*A, 0, angle
HOTSPOT2 0, 0, 1, angle, 6
HOTSPOT2 0.9*A, 0, 2, angle, 4
HOTSPOT2 0.9*A*COS(angle), 0.9*A*SIN(angle), 3,
          angle, 5
```

Beispiel 2, kombinierte Längenbearbeitung mit 2 Parametern:



```
RECT2 0, 0, A, B
RECT2 0, 0, sideX, sideY
HOTSPOT2 sideX, 0, 1, sideY, 1
HOTSPOT2 sideX, -0.1, 2, sideY, 3
HOTSPOT2 sideX, sideY, 3, sideY, 2
HOTSPOT2 0, sideY, 4, sideX, 1
HOTSPOT2 -0.1, sideY, 5, sideX, 3
HOTSPOT2 sideX, sideY, 6, sideX, 2
```

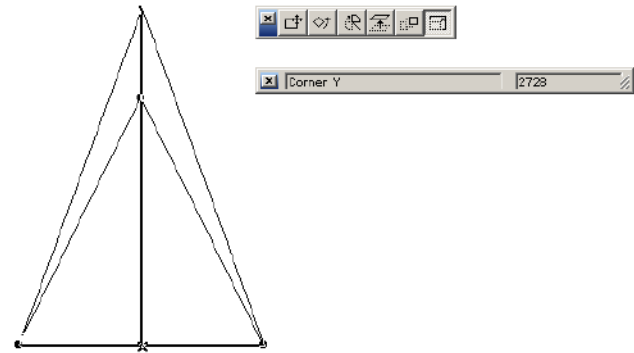
Beispiel 3, einfache Längenbearbeitung mit 1 Parameter:

2D SCRIPT:

```
HOTSPOT2 -1, 0, 1
HOTSPOT2 1, 0, 2
HOTSPOT2 0, 0, 3, corner_y, 1+128
HOTSPOT2 0, -1, 4, corner_y, 3
HOTSPOT2 0, corner_y, 5, corner_y, 2
LINE2 -1, 0, 1, 0
LINE2 -1, 0, 0, corner_y
LINE2 1, 0, 0, corner_y
```

3D SCRIPT:

```
HOTSPOT -1, 0, 0, 1
HOTSPOT -1, 0, 0.5, 2
HOTSPOT 1, 0, 0, 3
HOTSPOT 1, 0, 0.5, 4
HOTSPOT 0, 0, 0, 5, corner_y, 1+128
HOTSPOT 0, -1, 0, 6, corner_y, 3
HOTSPOT 0, corner_y, 0, 7, corner_y, 2
HOTSPOT 0, 0, 0.5, 8, corner_y, 1+128
HOTSPOT 0, -1, 0.5, 9, corner_y, 3
HOTSPOT 0, corner_y, 0.5, 10, corner_y, 2
PRISM_ 4, 0.5,
      -1, 0, 15,
      1, 0, 15,
      0, corner_y, 15,
      -1, 0, -1
```



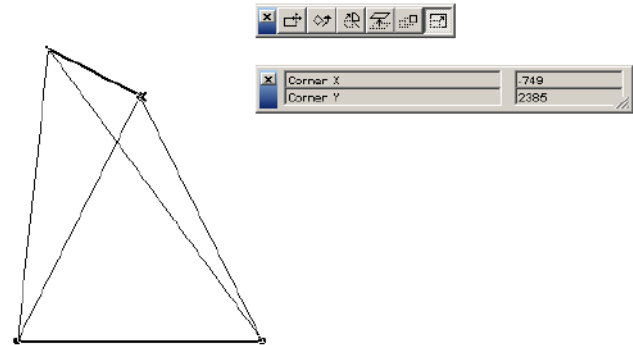
Beispiel 4: kombinierte Längenbearbeitung mit 2 Parametern:

2D SCRIPT:

```
HOTSPOT2 -1, 0, 1
HOTSPOT2 1, 0, 2
HOTSPOT2 corner_x, 0, 3, corner_y, 1+128
HOTSPOT2 corner_x, -1, 4, corner_y, 3
HOTSPOT2 corner_x, corner_y, 5, corner_y, 2
HOTSPOT2 0, corner_y, 3, corner_x, 1+128
HOTSPOT2 -1, corner_y, 4, corner_x, 3
HOTSPOT2 corner_x, corner_y, 5, corner_x, 2
LINE2 -1, 0, 1, 0
LINE2 -1, 0, corner_x, corner_y
LINE2 1, 0, corner_x, corner_y
```

3D SCRIPT:

```
HOTSPOT -1, 0, 0, 1
HOTSPOT -1, 0, 0.5, 2
HOTSPOT 1, 0, 0, 3
HOTSPOT 1, 0, 0.5, 4
HOTSPOT corner_x, 0, 0, 5, corner_y, 1+128
HOTSPOT corner_x, -1, 0, 6, corner_y, 3
HOTSPOT corner_x, corner_y, 0, 7, corner_y, 2
HOTSPOT 0, corner_y, 0, 8, corner_x, 1+128
HOTSPOT -1, corner_y, 0, 9, corner_x, 3
HOTSPOT corner_x, corner_y, 0, 10, corner_x, 2
HOTSPOT corner_x, 0, 0.5, 11, corner_y, 1+128
HOTSPOT corner_x, -1, 0.5, 12, corner_y, 3
HOTSPOT corner_x, corner_y, 0.5, 13, corner_y, 2
HOTSPOT 0, corner_y, 0.5, 14, corner_x, 1+128
HOTSPOT -1, corner_y, 0.5, 15, corner_x, 3
HOTSPOT corner_x, corner_y, 0.5, 16, corner_x, 2
PRISM_ 4, 0.5,
-1, 0, 15,
1, 0, 15,
corner_x, corner_y, 15,
-1, 0, -1
```



HOTLINE2

HOTLINE2 x1, y1, x2, y2

Definition der Statuslinie zwischen zwei Punkten.

HOTARC2

HOTARC2 *x, y, r, startangle, endangle*

Ein Bogen mit dem Mittelpunkt bei (x, y) zwischen Start- und Endwinkel und dem Radius r .

STATUSCODES

Die auf den folgenden Seiten vorgestellten Statuscodes ermöglichen Benutzern das Erstellen von Segmenten und Bögen in planaren Polylinien mithilfe spezieller Beschränkungen.

Planare Polylinien bilden mit den Statuswerten der Eckpunkte die Ausgangsbasis für viele GDL-Elemente:

POLY_, PLANE_, PRISM_, CPRISM_, BPRISM_, FPRISM_, HPRISM_, SPRISM_, SLAB_, CSLAB_, CROOF_,
EXTRÜDE_, PYRÄMID_, REVOLVE_, SWEEP_, TUBE_, TUBEA_

Statuscodes ermöglichen:

- die Steuerung der Sichtbarkeit der Kanten planarer Polylinien
- die Definition von Durchbohrungen in der Polylinie
- die Steuerung der Sichtbarkeit von seitlichen Kanten und Oberflächen
- das Erstellen von Segmenten und Bögen in der Polylinie

STATUSWERT-SYNTAX

Die Zahl *si* ist eine binäre Ganzzahl (zwischen 0 und 127 oder und) oder -1.

$s = j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7$ [+ *a_code*]

hierbei können *j1*, *j2*, *j3*, *j4*, *j5*, *j6*, *j7* can be 0 oder 1 sein.

Die Werte *j1*, *j2*, *j3*, *j4* geben an, ob die Kanten und Oberflächen vorhanden sind (1) oder nicht (0).

j1: untere horizontale Kante

j2: vertikale Kante

j3: obere horizontale Kante

j4: Seite

j5: horizontale Kante in Linienausschluss (nur für PRISM_ Formen)

j6: vertikale Kante in Linienausschluss (nur für PRISM_ Formen)

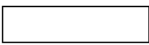
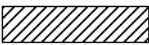
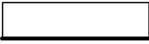
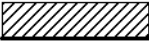


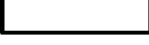



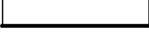


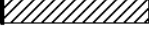
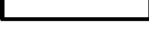

j7: spezieller zusätzlicher Statuswert, der nur gültig ist, wenn *j2* 1 ist und die vom Blickpunkt abhängige Sichtbarkeit der aktuellen vertikalen Kante kontrolliert

j2 = 0: die vertikale Kante ist immer unsichtbar.

j2 = 1 und *j7* = 1: die vertikale Kante ist nur sichtbar, wenn diese eine aus der aktuellen Blickrichtung betrachtete Kontur ist.

j2 = 1 und *j7* = 0: die vertikale Kante ist immer sichtbar

Mögliche Statuswerte (fette-Linie = sichtbare Kante):

<i>unsichtbare Oberfläche</i>	sichtbare Oberfläche
0 	8 
1 	9 
2 	10 
3 	11 
4 	12 
5 	13 
6 	14 
7 	15 

a_code: Zusätzlicher Statuscode (optional), der das Erstellen von Segmenten und Bögen in der Polylinie ermöglicht.

si=-1 wird benutzt, um Öffnungen innerhalb eines Prismas zu definieren. Dieser Wert kennzeichnet das Ende der Außenkontur und den Anfang der Öffnung innerhalb der Kontur. Ebenso kann dieser Wert das Ende einer Öffnung und den Beginn der nächsten anzeigen. Die Koordinaten, die diesem Wert voranstehen, müssen identisch mit den Koordinaten des Anfangspunktes der Außenkontur oder der Öffnung sein. Haben Sie den Maskenwert -1 benutzt, muss der letzte Maskenwert der Parameterliste -1 sein, der das Ende der letzten Öffnung angibt.

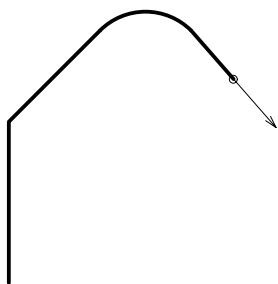
Öffnungen dürfen nicht miteinander verbunden sein und sich nicht überschneiden, sonst kann keine korrekte photorealistische Darstellung bzw. Schattendarstellung erzeugt werden.

ZUSÄTZLICHE STATUSCODES

Die folgenden zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie mithilfe spezieller Beschränkungen. Diese kennzeichnen das nächste Segment oder Kreisbögen. Die Ursprünglichen Mask- bzw. Statuswert(e) sind nur dort wirksam, wo sie festgelegt wurden (a "+s" wird nach dem zusätzlichen Wert eingeschlossen).

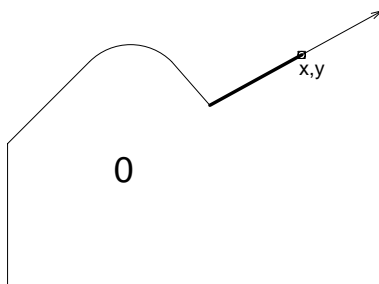
Hinweis: Die Auflösung von Kreisbögen wird durch die im Kapitel "Attribute" beschriebenen Anweisungen kontrolliert. Im Falle der POLY2_-Anweisung, wenn die Auflösung größer als 8 ist, generiert diese reale Kreisbögen. Sonst werden alle generierten Kreisbögen segmentiert.

Vorgegebener erster Teil eines Polygonzuges: aktuelle Position und Tangente ist definiert



Segment, definiert durch den absoluten Endpunkt

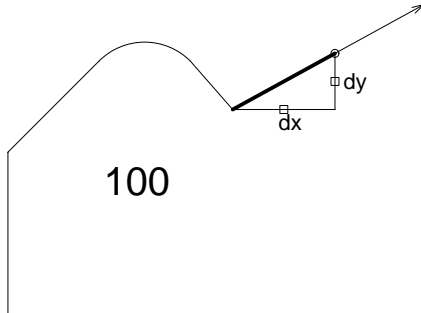
x, y, s
wobei $0 < s < 100$



Segment, definiert durch den relativen Endpunkt

$dx, dy, 100+s,$

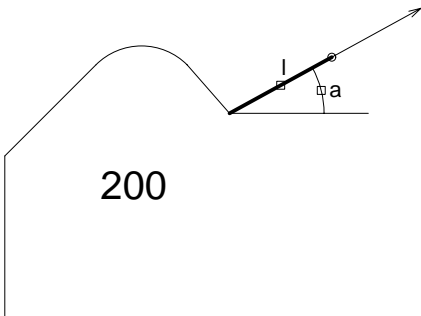
wobei $0 < s < 100$



Segment, definiert durch Längen- und Richtungsangabe

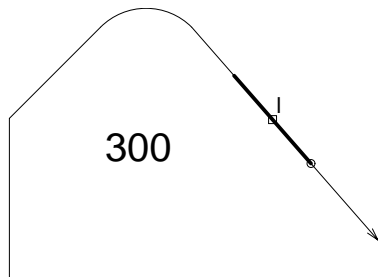
$l, a, 200+s,$

wobei $0 < s < 100$



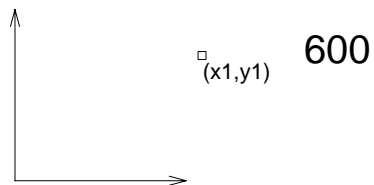
Tangentiales Segment, definiert durch Längenangabe

$l, 0, 300+s,$
wobei $0 < s < 100$



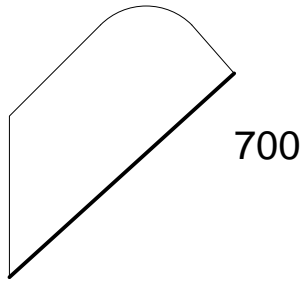
Angabe des Startpunktes

$x1, y1, 600,$



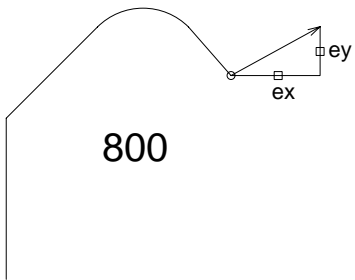
Schließen des Polygonzuges

0, 0, 700,



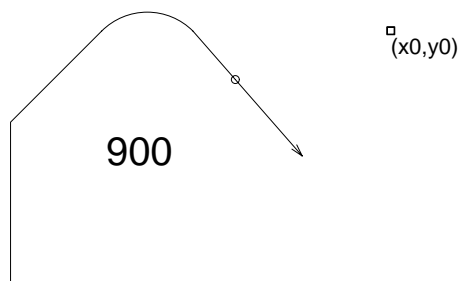
Angabe der Tangente

ex, ey, 800,



Angabe des Mittelpunktes

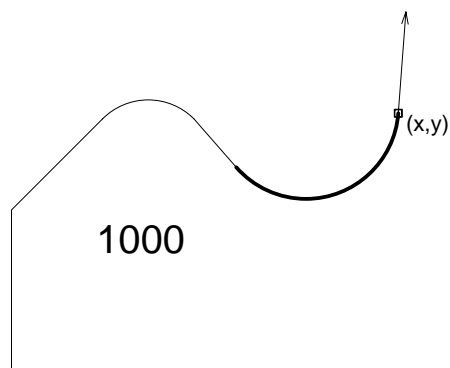
$x0, y0, 900,$



Tangentialer Bogen zum Endpunkt

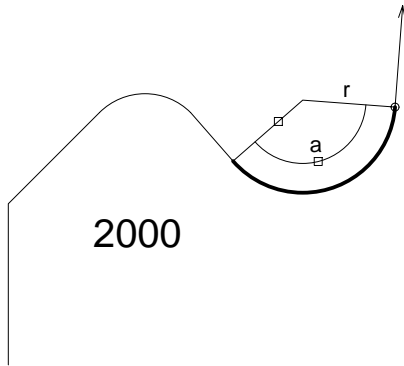
$x, y, 1000+s,$

wobei $0 < s < 100$



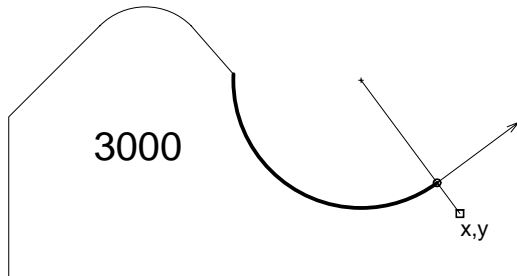
Tangentialer Bogen, definiert durch Radius und Winkel

$r, a, 2000+s,$
wobei $0 < s < 100$



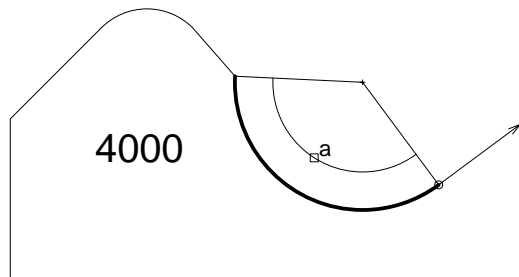
Kreisbogen, definiert durch Mittelpunkt und Punkt auf der Kreislinie (letzter Radius)

$x, y, 3000+s,$
wobei $0 < s < 100$



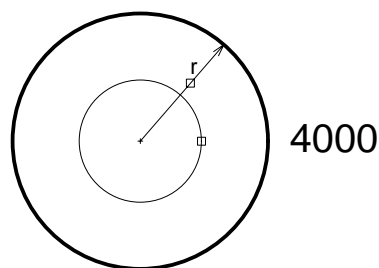
Kreisbogen, definiert durch Mittelpunkt und Winkel

$0, a, 4000+s,$
wobei $0 < s < 100$



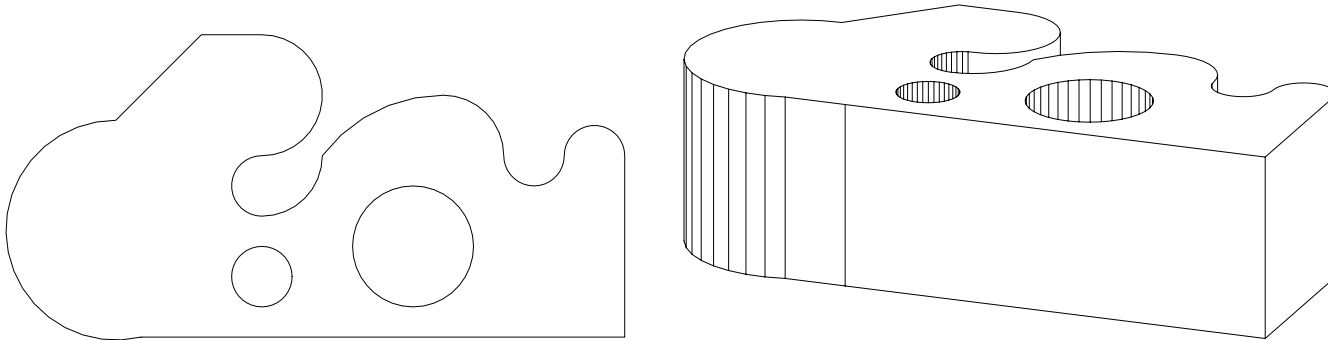
Geschlossener Kreis, definiert durch Mittelpunkt und Radius

$r, 360, 4000+s,$
wobei $0 < s < 100$



In diesem Fall bezieht sich der Status auf den ganzen Kreis.

Alle Winkelangaben in Grad. Unterdrückte Koordinaten werden durch 0 gekennzeichnet (bei den Status-Werten für 300, 700, 4000) und können einen beliebigen Wert haben.

Beispiele:

```

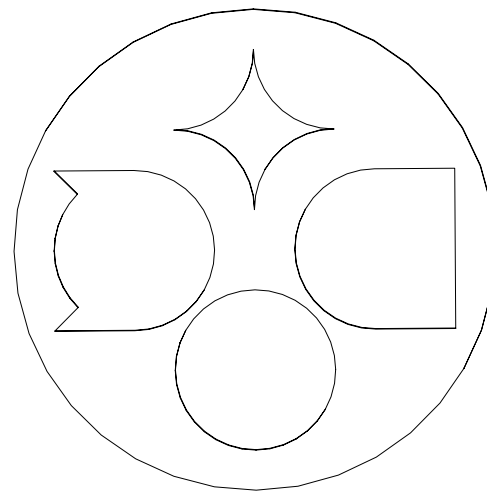
EXTRUDE 21, 0, 0, 3, 1+2+4+16+32,
0, 0, 0,
7, 0, 0,
7, 3, 1,
6, 3, 1000,      ! tangential arc to endpoint
5, 3, 1001,      ! tangential arc to endpoint
1, 90, 2000,     ! tangential arc by radius and angle
2, 3, 1001,      ! tangential arc to endpoint
1, 3, 900,       ! set centerpoint
1, 2, 3000,      ! arc using startpoint, centerpoint and point on final radius
1, 2.5, 900,     ! set centerpoint
0, -180, 4001,   ! arc using start point, centerpoint and angle
1, 5, 1000,      ! tangential arc to endpoint
-1, 0, 100,      ! segment by (dx, dy)
2, 225, 200,     ! segment by (len, angle)
-1, 0, 800,      ! set tangent
-1, 0, 1000,     ! tangential arc to endpoint
0, 0, -1,        ! end of contour
1, 1, 900,       ! set centerpoint
0.5, 360, 4000,  ! full circle by centerpoint and radius
3.5, 1.5, 900,   ! set centerpoint
1, 360, 4001     ! full circle by centerpoint and radius

```

```

EXTRUDE 2+5+10+10+2, 0, 0, 3, 1+2+4+16+32,
0, 0, 900,
3, 360, 4001,
2.5, -1, 0,
2.5, 1, 0,
1.5, 1, 1,
1.5, -1, 1001,
2.5, -1, -1,
0, 2.5, 600,
0, -1, 800,
1, 1.5, 1001,
-1, 0, 800,
0, 0.5, 1001,
0, 1, 800,
-1, 1.5, 1001,
1, 0, 800,
0, 2.5, 1001,
0, 2.5, 700,
-1.5, 0, 900,
-2.5, 0, 600,
-2.5, 1, 3000,
-2.5, 1, 0,
-1.5, 1, 0,
-1.5, -1, 1001,
-2.5, -1, 0,
SQR(2)-1, 45, 200,
-2.5, 0, 3000,
-2.5, 0, 700,
0, -1.5, 900,
1, 360, 4000

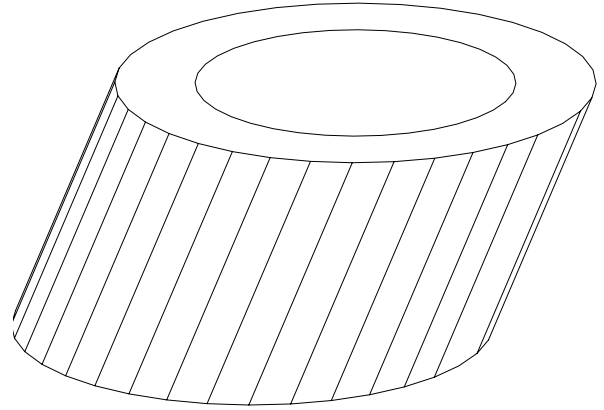
```



```

EXTRUDE 3, 1, 1, 3, 1+2+4+16+32,
0, 0, 900,
3, 360, 4001,
2, 360, 4000

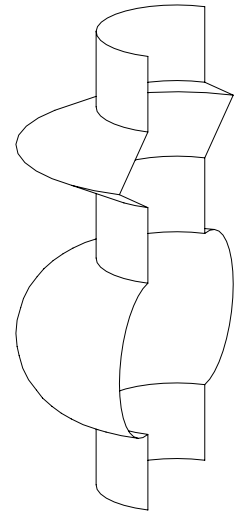
```



```

ROTY -90
REVOLVE 9, 180, 16+32,
7, 1, 0,
6, 1, 0,
5.5, 2, 0,
5, 1, 0,
4, 1, 0,
3, 1, 900,      ! set centerpoint
0, 180, 4001,   ! arc using startpoint, centerpoint and angle
2, 1, 0,
1, 1, 0

```



ATTRIBUTE

In dem ersten Abschnitt dieses Kapitels werden Anweisungen vorgestellt, die sich auf die Interpretation von GDL Befehlen auswirken. Anweisungen können die Glätte bestimmen, die bei zylindrischen Elementen, für ihren Darstellungsmodus in der 3D-Ansicht oder für die Zuweisung von Attributen (Farbe, Material, Textstil, usw.) bei den sukzessiven Gestalten verwendet werden. Die Inline-Attributbeschreibung wird im zweiten Abschnitt behandelt. Mit Hilfe dieser Funktion können Sie ihren Objekten individuelle Materialien, Texturen, Schraffurmuster, Linientypen und Textstile zuordnen, die im aktuellen Attributsatz Ihres Projekts nicht vorhanden sind.

ANWEISUNGEN

Anweisungen beeinflussen die Interpretation nachfolgender GDL-Befehle. Ihr Einfluß wirkt bis zur nächsten Anweisung oder bis zum Ende eines 3D-Scriptes. Aufgerufene Makros übernehmen die aktuellen Einstellungen, die Änderungen wirken sich lokal aus. Nach Beendigung des Scripts werden die Werte vor dem Makro-Aufruf wiederhergestellt.

Befehle für 3D- und 2D-Scripts

[LET]

[**LET**] varnam = n

Wertzuweisung Die LET-Anweisung ist optional. Die Variable speichert den errechneten Wert von n.

RADIUS

RADIUS radius_min, radius_max

Glättet die gekrümmten Oberflächen von zylindrischen Elementen und Kreisbögen in Polygonzügen.

Ein Kreis mit dem Radius r wird folgendermaßen dargestellt:

- wenn $r < \text{radius_min}$ als Sechseck,
- wenn $r \geq \text{radius_max}$ als 36-eckiges Polygon,
- wenn $\text{radius_min} < r < \text{radius_max}$ als ein $(6+30*(r-\text{radius_min})/(\text{radius_max}-\text{radius_min}))$ -eckiges Polygon.

Die Bogendarstellung erfolgt dazu analog.

Nach einer RADIUS-Anweisung $\alpha\psi$ verlieren die vorhergehenden RESOL- und TOLER-Anweisungen in Wirkung.

Einschränkung der Parameter::

$r_{\min} \leq r_{\max}$

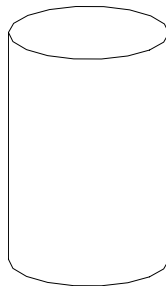
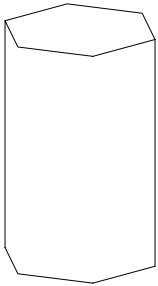
Beispiele:

RADIUS 1.1, 1.15

CYLIND 3.0, 1.0

RADIUS 0.9, 1.15

CYLIND 3.0, 1.0



RESOL

RESOL n

Glättet die gekrümmten Oberflächen von zylindrischen Elementen und Kreisbögen in Polygonzügen. Kreise werden in n-seitige reguläre Polygone umgewandelt.

Die Bogendarstellung erfolgt dazu analog.

Nach einer RESOL-Anweisung verlieren die vorhergehenden RESOL- und TOLER-Anweisungen in Wirkung.

Einschränkung der Parameter::

$n \geq 3$

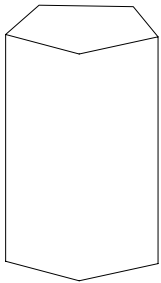
Grundeinstellung:

RESOL 36

Beispiele:

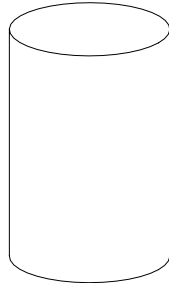
RESOL 5

CYLIND 3.0, 1.0



RESOL 36

CYLIND 3.0, 1.0



TOLER

TOLER d

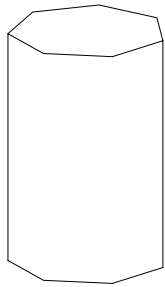
Glättet die gekrümmten Oberflächen von zylindrischen Elementen und Kreisbögen in Polygonzügen. Die Abweichung vom Bogen, also der Abstand zwischen dem theoretischen Bogen und der erzeugten Sehne, ist kleiner als d.

Nach einer TOLER-Anweisung verlieren die vorhergehenden RESOL- und TOLER-Anweisungen in Wirkung.

Beispiele:

TOLER 0.1

CYLIND 3.0, 1.0



TOLER 0.01

CYLIND 3.0, 1.0



Anmerkung: Die Anweisungen RADIUS, RESOL und TOLER glätten die gekrümmten Oberflächen von 3D-Elementen (CIRCLE, ARC, CYLIND, SPHERE, ELLIPS, CONE, ARMC, ARME, ELBOW, REVOLVE) und Kreisbögen in 2D-Polygonzügen, in denen gekrümmte Kanten verwendet werden.

Siehe *“Zusätzliche Statuscodes”* auf Seite 143.

PEN

PEN n

Legt die Stiftfarbe fest.

Einschränkung der Parameter::

$0 < n \leq 255$

Grundeinstellung:

PEN 1

falls es im Script keine PEN Anweisung gibt.

(Für Bibliothekselemente werden voreingestellte Werte aus dem Dialogfenster Bibliothekselemente-Einstellungen eingelesen. Falls sich das Script auf einen existierenden Index bezieht, so wird die PEN 1-Anweisung die Grundeinstellung.)

LINE_PROPERTY

LINE_PROPERTY expr

Legt die Eigenschaften aller nachfolgend erstellten Linien im 2D-Script fest (RECT2, LINE2, ARC2, CIRCLE2, SPLINE2, SPLINE2A, POLY2, FRAGMENT2commands), bis der nächste Befehl LINE_PROPERTY folgt. Grundwert ist generisch.

Mögliche Statuswerte:

0: alle Linien sind generische Linien

1: alle Linien sind innliegend

2: alle Linien sind Konturen

[SET] STYLE

[SET] STYLE name_string

[SET] STYLE index

Alle nachfolgend erzeugten Texte werden mit diesen Stil-Einstellungen dargestellt, bis die neue SET STYLE-Anweisung folgt.

Der Index ist eine Konstante, die sich auf einen vorher definierten Stiltypen-Stack in der internen Datenstruktur bezieht. Diese Struktur wird während der GDL-Analyse modifiziert und kann auch innerhalb des Programms verändert werden. Die Benutzung des Index anstelle des Stilnamens ist notwendig, wenn die IND-Funktion verwendet wurde.

Grundeinstellung:

SET STYLE 0

Ist keine SET STYLE-Anweisung im GDL-Script angegeben, wird die Voreinstellung SET STYLE 0 (aktueller Font, Höhe 5 mm, Ankerpunkt 1, Standard) angenommen.

Nur in 3D-Scripts verwendeten Anweisungen

Modell

MODEL WIRE

MODEL SURFACE

MODEL SOLID

Bestimmt die Darstellungsform im aktuellen Script.

MODEL WIRE: Drahtmodell, ohne Oberflächen und Volumen. Die Objekte sind transparent.

MODELSURFACE, MODELSOLID: Die Generierung einzelner Flächen baut auf den Relationen der Körperoberflächen zueinander auf. Daher erzeugen beide Anweisungen dieselbe interne 3D-Datenstruktur. Die Objekte sind nicht transparent.

Der einzige Unterschied wird bei einem Schnitt durch den Körper deutlich:

MODEL SURFACE: Das Innere des Körpers wird sichtbar ,

MODEL SOLID: Neue Oberflächen erscheinen.

Grundeinstellung:

MODEL SOLID

Die folgenden drei Quader sollen die drei Darstellungsmethoden veranschaulichen:

MODEL WIRE

BLOCK 3,2,1

ADDY 4

MODEL SURFACE

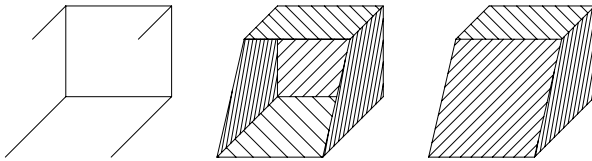
BLOCK 3,2,1

ADDY 4

MODEL SOLID

BLOCK 3,2,1

Darstellung nach einem Schnitt:



[SET] MATERIAL

[SET] MATERIAL name_string

[SET] MATERIAL index

Alle nach dieser Anweisung erzeugten Oberflächen erhalten diese Materialzuweisung, bis eine neue MATERIAL-Anweisung die vorhergehende ersetzt. Die Oberflächen der

BPRISM_, CPRISM_, FPRISM_, HPRISM_
 SPRISM_, CSLAB_, CWALL_, BWALL_, XWALL_,
 CROOF_, MASS

Körper sind von dieser Regel ausgenommen.

Der Index ist eine Konstante, die auf die Materialtypen der internen Datenstruktur Bezug nimmt. Diese Struktur wird während der GDL-Analyse modifiziert und kann auch innerhalb des Programms verändert werden. Die Benutzung des Index anstelle des Stilnamens ist notwendig, wenn die IND-Funktion verwendet wurde.

index 0 hat eine besondere Bedeutung: Oberflächen verwenden die aktuelle Stift-Farbe und werden matt dargestellt.

Grundeinstellung:

MATERIAL 0

falls es im Script keine MATERIAL-Anweisung gibt.

(Für Bibliothekselemente werden voreingestellte Werte aus dem Dialogfenster Bibliothekselemente-Einstellungen eingelesen. Falls sich das Script auf einen nicht existierenden Index bezieht, so wird die Material-Einstellung 0 der voreingestellte Wert.)

Siehe auch die Beschreibung der IND-Funktion in "Verschiedenes" > "Spezielle Funktionen" auf Seite 243.

SECT_FILL

SECT_FILL fill, fill_background_pen,
fill_pen, contour_pen

Definiert die Schraffur für die 3D-Elemente, die später im Fenster Schnitte/Ansichten erzeugt werden.

fill: Name oder Indexnummer der Schraffur

fill_background_pen: Nummer des Schraffurhintergrundstifts

fill_pen: Nummer des Schraffurstifts

contour_pen: Nummer des Schraffurkonturenstifts

SHADOW

SHADOW keyword_1[, keyword_2]

Steuert den Schattenwurf der Elemente in der photorealistischen Darstellung und im Vektorschattenwurf.

keyword_1: ON, AUTO or OFF

keyword_2: ON or OFF

ON: alle nachfolgenden Elemente erzeugen in jedem Fall Schatten.

OFF: alle nachfolgenden Elemente erzeugen in keinem Fall Schatten.

AUTO: Der Schattenwurf wird automatisch festgelegt.

- Stellt man für verdeckte Körper SHADOW OFF ein, so wird für die 3D-Berechnung Speicherplatz und Rechenzeit gespart.
- Die Einstellung SHADOW ON gewährleistet aber, daß selbst das kleinste Detail einen guten Schattenwurf erzeugt.

Das optionale zweite Schlüsselwort (keyword) steuert das Erscheinen der Schatten auf den Oberflächen.

- SHADOW keyword_1, OFF vektorielle Schatten auf den folgenden Oberflächen.
- SHADOW keyword_1, ON schaltet vektorielle Schatten zurück.

Grundeinstellung:

SHADOW AUTO

SHADOW OFF

BRICK 1, 1, 1

ADDX 2

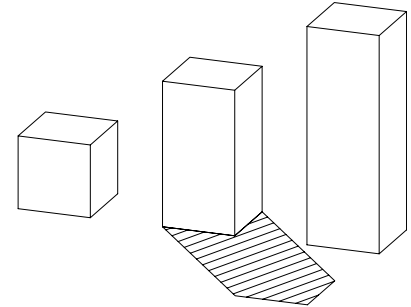
SHADOW ON

BRICK 1, 1, 2

ADDX 2

SHADOW OFF

BRICK 1, 1, 3



Anweisungen nur für 2D-Scripts

DRAWINDEX

DRAWINDEX number

Definiert die Darstellungsreihenfolge von 2D-Skriptelementen. Elemente mit einem niedrigeren Darstellungsindex werden zuerst gezeichnet.

Einschränkung der Parameter:

`0 < number <= 50`

(In der aktuellen GDL-Version sind nur die DRAWINDEX-Werte 10, 20, 30, 40 und 50 gültig. Andere Werte werden auf diese gerundet.)

Wenn es keine DRAWINDEX-Anweisung gibt, gilt folgende Standarddarstellungsreihenfolge:

- 1 Abbildungen
- 2 Schraffuren
- 3 Linien
- 4 Textelemente

[SET] FILL

[SET] FILL name_string

[SET] FILL index

Alle danach erzeugten 2D-Polygone werden mit der angegebenen Schraffur dargestellt, bis eine neue SET FILL-Anweisung erfolgt.

Der Index ist eine Konstante, die sich auf die Schraffurtypen in der internen Datenstruktur bezieht. Diese Struktur wird während der GDL-Analyse modifiziert und kann auch innerhalb des Programms verändert werden. Die Angabe des Index anstelle des Schraffurnamens ist nur dann erforderlich, wenn die IND-Funktion verwendet.

Grundeinstellung:

`SET FILL 0`

z.B. keine Schraffur, wenn das Script keine SET FILL-Anweisung enthält.

Siehe auch die IND-Funktion unter ["Verschiedenes"](#) > ["Spezielle Funktionen"](#) auf Seite 243.

[SET] LINE_TYPE

[SET] LINE_TYPE name_string

[SET] LINE_TYPE index

Alle nachfolgend erstellten 2D-Linien werden mit diesem Linientyp dargestellt, bis die nächste Anweisung SET LINE_TYPE folgt.

Der Index ist eine Konstante, die sich auf die Linientypen der internen Datenstruktur bezieht. Diese Struktur wird während der GDL-Analyse modifiziert und kann auch mit dem Programm verändert werden. Die Benutzung des Index anstelle des Stilnamens ist notwendig, wenn die IND-Funktion verwendet wurde.

Grundeinstellung:

SET LINE_TYPE 1

z.B. durchgezogene Linie, wenn das Script keine SET LINE_TYPE-Anweisung enthält.

Siehe auch die IND-Funktion unter ["Verschiedenes"](#) > ["Spezielle Funktionen"](#) auf Seite 243.

INLINE ATTRIBUTDEFINITION

Attribute können in den Material-, Schraffur- und Linientyp-Dialogfenster erstellt werden. Jedes GDL-Script kann auf diese Grundrissattribute bezogen werden.

Attribute können auch in GDL-Scripts definiert werden. Es gibt zwei verschiedene Fälle:

- Attribut-Definition im MASTER_GDL-Script. Das MASTER_GDL-Script wird interpretiert, wenn die Bibliothek, die das Script enthält, in den Speicher eingelesen wird. Die MASTER_GDL-Attribute werden den Grundriß-Attributen dazugeladet; Attribute mit denselben Namen werden nicht ersetzt. Ist das MASTER_GDL-Script geladen, kann sich jedes Script darauf beziehen.
- Attribut-Definition in Bibliothekselementen. Die hier definierten Materialien und Texturen können im Script und in seinen eventuellen Unterprogrammen benutzt werden. Die im 2D-Script definierten und benutzten Schraffuren und Linientypen verhalten sich genau so, als ob sie im MASTER_GDL-Script definiert worden wären.

Über den Befehl **Check GDL Script** im Script-Fenster können Sie sich davon überzeugen, ob die Material-, Schraffur-, Linientyp- oder Stilparameter richtig sind

Sollte der Material-, Schraffur-, Linientyp, oder Stil in der 3D-Darstellung der Bibliothekselemente vom vorgesehenen abweichen, ohne dass eine Fehlermeldung erscheint, bedeutet dies wahrscheinlich, dass einer oder mehrere Parameterwerte falsch sind. Mit Hilfe der detaillierten Meldungen des Befehls **Check GDL Scripts** finden Sie diese Parameter.

Material

DEFINE MATERIAL

DEFINE MATERIAL name type, parameter1,
parameter2, ... parametern

Hinweis: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe *“Zusätzliche Daten” auf Seite 180 für weitere Details.*

Jedes GDL-Script kann Materialien einschließen, die dann später über den Namen aufgerufen werden können. Dieses Material kann nur für 3D-Elemente in diesem Script oder seinen eventuellen Unterprogrammen benutzt werden.

name: Name des Materials.

type: Typ des Materials. Die tatsächliche Anzahl der Parameter, die das Material definieren, ist unterschiedlich und abhängig vom Typ. Die Bedeutungen und Einschränkung der Parameter werden in den folgenden Beispielen gezeigt.

0: allgemeine Definition, n=16

1: einfache Definition, n=9 (weitere Parameter sind Konstanten oder werden aus gegebenen Werten berechnet.)

2-7: vordefinierte Materialtypen, n=3

Die drei Werte sind die RGB-Komponenten der Oberflächenfarbe. Parameter sind Konstanten oder über Farbangaben berechnet.

2: matt

3: Metall

4: Kunststoff

5: Glas

6: glänzend

7: konstant

10: allgemeine Definition mit Schraffurparameter, n=17

11: einfache Definition mit Schraffurparameter, n=10

12-17: vordefinierte Materialtypen mit Schraffurparameter, n=4

20: allgemeine Definition mit Schraffur- und Texturparametern und Farbindex der Schraffur, n=19

21: einfache Definition mit Schraffurparameter, Farbe, Index der Schraffur und Index der Texturparameter, n=12

22-27: vordefinierte Materialtypen mit Schraffurparameter, Farbe, Index der Schraffur und Index der Texturparameter, n=6

Spezielle Bedeutungen für Typen 20-27:

Ist die Stiftnummer Null, so werden die vektoriellen Schraffuren mit dem aktiven Stift erstellt.

Der Null Wert für Textur-Index ermöglicht die Definition von Materialien ohne vektorielle Schraffuren oder Texturen.

Beispiele:

```
DEFINE MATERIAL "water" 0,
    0.5284, 0.5989, 0.6167,
!   surface RGB [0.0..1.0]
```

```
    1.0, 0.5, 0.5, 0.9,  
!   ambient, diffuse, specular, transparent  
!   coefficients [0.0..1.0]  
2.0,  
!   shining [0.0..100.0]  
    1,  
!   transparency attenuation [0.0..4.0]  
    0.5284, 0.5989, 0.6167,  
!   specular RGB [0.0..1.0]  
    0, 0, 0,  
!   emission RGB [0.0..1.0]  
    0.0  
!   emission attenuation [0.0..65.5]  
DEFINE MATERIAL "asphalt" 1,  
    0.1995, 0.2023, 0.2418,  
!   surface RGB [0.0..1.0]  
    1.0, 1.0, 0.0, 0.0,  
!   ambient, diffuse, specular, transparent  
!   coefficients [0.0..1.0]  
    0,  
!   shining [0..100]  
    0  
!   transparency attenuation [0..4]  
DEFINE MATERIAL "matte red" 2,  
    1.0, 0.0, 0.0  
!   surface RGB [0.0..1.0]  
DEFINE MATERIAL "Red Brick" 10,  
    0.878294, 0.398199, 0.109468,  
    0.58, 0.85, 0.0, 0.0,  
    0,  
    0.0,  
    0.878401, 0.513481, 0.412253,  
    0.0, 0.0, 0.0,  
    0,  
IND(FILL, "common brick")  
!   fill index  
DEFINE MATERIAL "Yellow Brick+*" 20,  
    1, 1, 0,  
!   surface RGB [0.0 .. 1.0]  
    0.58, 0.85, 0, 0,  
!   ambient, diffuse, specular, transparent  
!   coefficients [0.0 .. 1.0]  
    0,  
!   shining [0.0 .. 100.0]
```

```

0,
! transparency attenuation [0.0 .. 4.0]
0.878401, 0.513481, 0.412253,
! specular RGB [0.0 .. 1.0]
0, 0, 0,
! emission RGB [0.0 .. 1.0]
0,
! emission attenuation [0.0 .. 65.5]
IND(FILL, "common brick"), 61,
IND(TEXTURE, "Brick")
! Fill index, color index, texture index

```

DEFINE MATERIAL BASED_ON

```

DEFINE MATERIAL name BASED_ON orig_name PARAMETERS name1 = expr1 [,
    ...][ADDITIONAL_DATA name1 = expr1 [, ...]]

```

Materialdefinition auf Basis vorhandener Materialien. Ausgewählte Parameter des Originalmaterials werden durch neue Werte überschrieben. Die übrigen Parameter bleiben unverändert. Wird dieser Befehl ohne aktuelle Parameter eingesetzt, bleibt das Originalmaterial erhalten und erhält nur einen neuen Namen. Auf die Parameterwerte eines Materials kann über die Funktion *“REQUEST{2} (“Material_info”, name_or_index, param_name, value_or_values)”* zugegriffen werden.

orig_name: Name des Originalmaterials (Name eines vorhandenen Material, das bereits in GDL oder einem Grundriss definiert wurde)

name1: Name des Materialparameters, der durch einen neuen Wert überschrieben wird. Namen, die sich auf Parameter der Materialdefinition beziehen:

```

gs_mat_surface_r, gs_mat_surface_g, gs_mat_surface_b (surface RGB [0.0..1.0])
gs_mat_ambient (ambient coefficient [0.0..1.0])
gs_mat_diffuse (diffuse coefficient [0.0..1.0])
gs_mat_specular (specular coefficient [0.0..1.0])
gs_mat_transparent (transparent coefficient [0.0..1.0])
gs_mat_shining (shininess [0.0..100.0])
gs_mat_transp_att (transparency attenuation [0.0..4.0])
gs_mat_specular_r, gs_mat_specular_g, gs_mat_specular_b (specular color RGB [0.0..1.0])
gs_mat_emission_r, gs_mat_emission_g, gs_mat_emission_b (emission color RGB [0.0..1.0])
gs_mat_emission_att (emission attenuation [0.0..65.5])
gs_mat_fill_ind (fill index)
gs_mat_fillcolor_ind (fill color index)
gs_mat_texture_ind (texture index)

```

expri: neuer Wert, der den ausgewählten Parameter des Materials überschreibt. Der Wertbereich stimmt mit der Materialdefinition überein.

Beispiel:

```
n = REQUEST{2} ("Material_info", "Brick-Face", "gs_mat_emission_rgb ", em_r, em_g, em_b)
em_r = em_r + (1 - em_r) / 3
em_g = em_g + (1 - em_g) / 3
em_b = em_b + (1 - em_b) / 3
DEFINE MATERIAL "Brick-Face light" BASED_ON "Brick-Face" PARAMETERS gs_mat_emission_r = em_r,
gs_mat_emission_g = em_g, gs_mat_emission_b = em_b

SET MATERIAL "Brick-Face"
BRICK a, b, zzyzx
ADDX a
SET MATERIAL "Brick-Face light"
BRICK a, b, zzyzx
```

DEFINE TEXTURE

DEFINE TEXTURE name expression, x, y, mask, angle

In jedem GDL-Script können Sie Texturen definieren, die über ihren Namen aufgerufen werden können. Die so in einem GDL-Script definierte Textur kann nur in diesem Script und in seinen eventuellen Unterprogrammen benutzt .

name: Name der Textur

expression: Der Schraffur zugeordnetes Bild. Ein Zeichenfolgenausdruck bedeutet ein Dateiname, ein numerischer Ausdruck oder einen Index eines im Bibliothekselement abgelegten Bildes. Ein 0-Index ist ein spezieller Wert, der sich auf das Vorschaubild des Bibliothekselements bezieht.

x: die logische Texturbreite

y: die logische Texturhöhe

mask: $j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8 + 256*j9$
wobei j1, j2, j3, j4, j5, j6, j7, j8, j9 0 oder 1 sein kann.

Alpha Kanal- Kontrollen (j1 j6):

j1:

alpha Kanal ändert die Transparenz der Textur

j2: Bump mapping (3D-Texturen) oder normale Oberflächen-Störung.

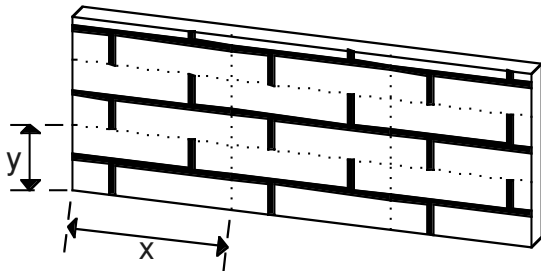
Bump mapping (3D-Texturen) verwendet Alpha Kanal zur Bestimmung der Amplitude der normalen Oberfläche.

j3: Alpha Kanal ändert die diffuse Farbe der Textur

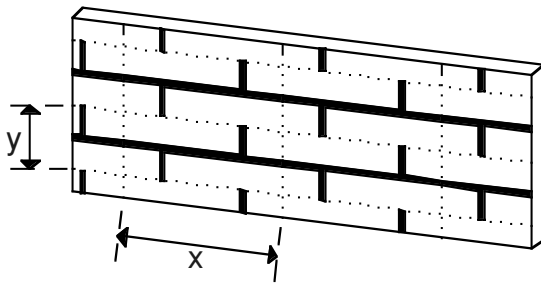
- j4: Alpha Kanal ändert die spiegelnde Farbe der Textur
- j5: Alpha Kanal ändert die Streulichtfarbe der Textur
- j6: Alpha Kanal ändert die Oberflächenfarbe der Textur

Verbindungskontrollen (j7 j9):

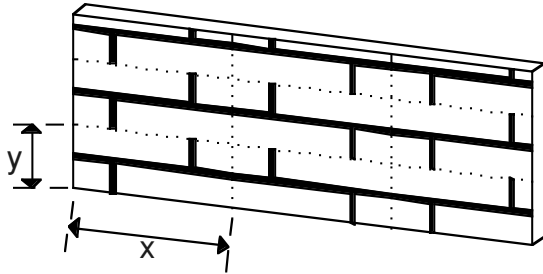
Ist der Wert Null, so wird der normale Modus gewählt:



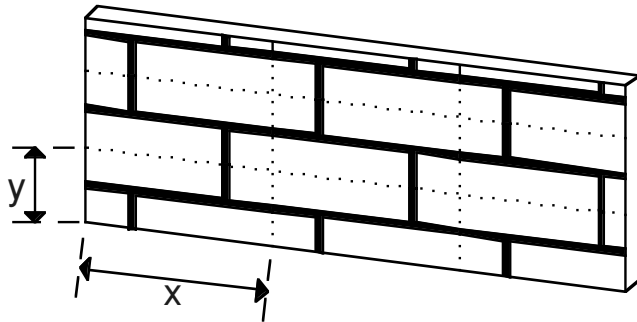
j7: die Textur wird zufälligerweise verschoben.



j8: Spiegelung in die 'x'-Richtung



j9: Spiegelung in die 'y'-Richtung



Winkel: Winkel der Drehung .

Beispiel:

```
DEFINE TEXTURE "Brick" "Brick.PICT", 1.35, 0.3, 256+128, 35.0
```

Schraffuren

DEFINE FILL

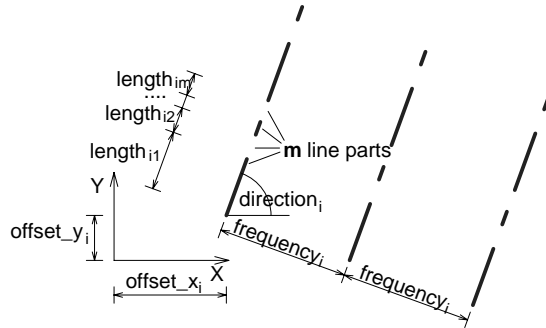
```
DEFINE FILL name [FILLTYPES_MASK fill_types,] pattern1, pattern2, pattern3, pattern4,
    pattern5, pattern6, pattern7, pattern8,
    spacing, angle, n,
    frequency1, direction1, offset_x1, offset_y1, m1,
    length11, ... length1m,
    ...
```

```
frequencyn, directionn, offset_xn,
lengthn1, ... lengthnm
```

Hinweis: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe *“Zusätzliche Daten” auf Seite 180* für mehr Details.

In jedem GDL-Script können Sie Schraffuren definieren, die später über ihren Namen aufgerufen werden können. Die so in einem GDL-Script definierte Textur kann nur in diesem Script und in seinen eventuellen Unterprogrammen benutzt .



name: Name der Schraffur

fill_types = j1 + 2 * j2 + 4 * j3

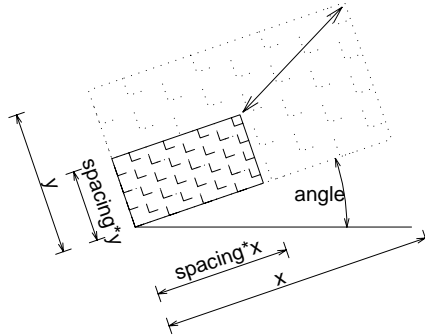
j1: Bauteilschraffuren

j2: Deckschraffuren

j3: Zeichenschraffuren

Ist j gesetzt, kann die definierte Schraffur in ArchiCAD ihrem Schraffurtyp entsprechend verwandt werden. Grundeinstellung ist alle Schraffuren (0).

Musterdefinition: pattern1, pattern2, pattern3, pattern4, pattern5, pattern6, pattern7, pattern8: 8 Zahlen zwischen 0 und 255 stellen Binär-Werte dar. Definiert das Bitmap-Muster der Schraffur.



spacing: Faktor Schraffurweite - definiert den globalen Faktor der Schraffurweite für die gesamte Schraffur. Alle Werte werden durch diese Nummer sowohl in die x-Richtung als auch in die y-Richtung multipliziert.

angle: Drehwinkel der (gesamten) Schraffur in Grad

n: Anzahl der Schraffurlinien

frequencyi: Frequenz der Schraffurlinie (der wirkliche Abstand zwischen zwei Linien ist schraffurweite*freqi)

directioni: Richtungswinkel der Schraffurlinien in Grad

offset_xi, offset_yi: Versatz der Schraffurlinien vom Ursprung

mi: Anzahl der Schraffurelemente

lengthij: Länge der einzelnen Schraffurelemente (die wirkliche Länge ist spacing * lengthij). Schraffurelemente sind aufeinanderfolgende Liniensegmente und Leerräume. Das erste Schraffurelement ist ein Strich, die Länge Null kennzeichnet einen Punkt.

Das **Bitmap-Muster** wird nur durch die Parameter pattern1... pattern8 definiert und benutzt, wenn das Menü Optionen / Reinzeichnungseinstellungen / Flächenschraffuren / Bitmap -Muster eingestellt wird. Um dies zu definieren, wählen Sie die kleinste Einheit der Schraffur aus und stellen Sie diese als Punkte und Leerräume unter Verwendung eines rechteckigen Konstruktionsrasters mit 8x8 Schrittweiten dar. Die 8 Musterparameter sind dezimale Darstellungen der binären Werte in den Konstruktionslinien (1 kennzeichnet einen Punkt, 0 einen Leerraum).

Die **Vektorschraffur** wird durch den zweiten Teil der Schraffurdefinition als eine Sammlung von Strichlinien definiert, die mit einer angegebenen Frequenz (frequencyi) wiederholt werden. Jede Linie der Sammlung wird durch ihre Richtung (directioni), ihren Versatz vom Ursprung (offset_xi, offset_yi) und die Definition der Strichlinie, die aufeinanderfolgende Segmente und Leerräume mit der angegebenen Länge (lengthij), definiert.

Hinweis: Mit dem Befehl DEFINE FILL können nur einfache Schraffuren definiert werden. Es gibt keine Möglichkeit zur Definition von Symbolschraffuren.

Beispiel:

```
DEFINE FILL "brick" 85, 255, 136, 255,
    34, 255, 136, 255,
    0.08333, 0.0, 4,
    1.0, 0.0, 0.0, 0.0, 0,
    3.0, 90.0, 0.0, 0.0, 2,
    1.0, 1.0,
    3.0, 90.0, 1.5, 1.0, 4,
    1.0, 3.0, 1.0, 1.0,
    1.5, 90.0, 0.75, 3.0, 2,
    1.0, 5.0
```

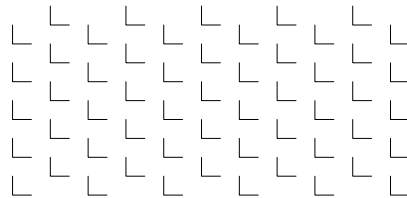
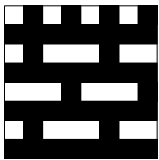
Bitmapmuster:

Pattern: Binary value:

pattern1 = 85	01010101
pattern2 = 255	11111111
pattern3 = 136	10001000	. .
pattern4 = 255	11111111
pattern5 = 34	00100010	. .
pattern6 = 255	11111111
pattern7 = 136	10001000	. .
pattern8 = 255	11111111

View:

Vectorial batch:



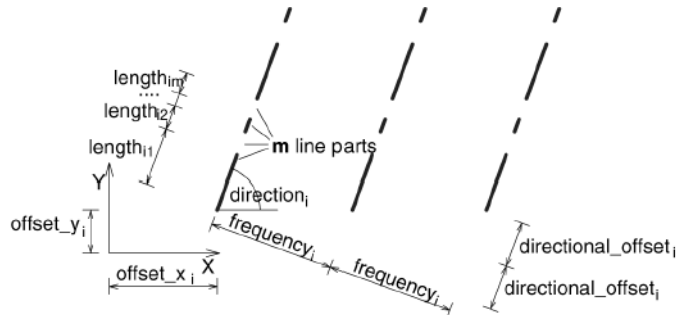
DEFINE FILLA

```
DEFINE FILLA name [FILLTYPES_MASK fill_types,] pattern1, pattern2, pattern3, pattern4, pattern5,
    pattern6, pattern7, pattern8, spacing_x, spacing_y, angle, n, frequency1,
    directional_offset1, direction1,
    offset_x1, offset_y1, m1, length11,
    ...
    length1m, ... frequencyn,
    directional_offsetn, directionn,
```

offset_xn, offset_yn, mn,
lengthn1, ... lengthnm

Hinweis: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe *“Zusätzliche Daten” auf Seite 180 für mehr Details.*



Ein erweiterter DEFINE FILL-Befehl.

Zusätzliche Parameter:

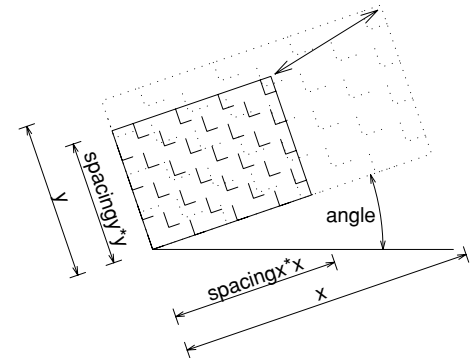
spacing_x, spacing_y: Faktor Schraffurweite in die x- und y-Richtungen. Diese beiden Parameter bestimmen einen globalen Faktor für die Schraffurweite der gesamten Schraffur. Alle Werte in der x-Richtung werden um die Schraffurweite x und alle Werte in der y-Richtung um die Schraffurweite y multipliziert.

directional_offset_i: Versatz des Anfangs der nächsten ähnlichen Schraffurlinie, die entlang der Linienrichtung gemessen wird. Jede Linie der Serie wird in einer durch frequency_i definierten Abstand mit einem durch offset gegebenen Versatz gezeichnet. Die wirkliche Länge des Versatzes wird durch spacing * directional_offset_i definiert.

Beispiel:

```
DEFINE FILLA "TEST" 8, 142, 128, 232,
    8, 142, 128, 232,
    0.5, 0.5, 0, 2,
    2, 1, 90, 0,
    0, 2, 1, 1,
    1, 2, 0, 0, 0,
    2, 1, 3

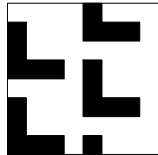
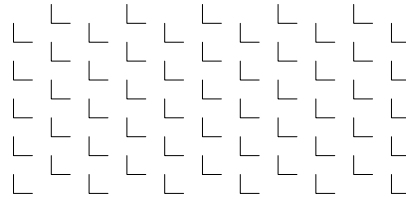
FILL "TEST"
POLY2 4, 6,
```



-0.5, -0.5, 12, -0.5,
12, 6, -0.5, 6

Bitmapmuster:

Pattern:	Binary value:
pat1 = 8	00001000 .
pat2 = 142	10001110
pat3 = 128	10000000 .
pat4 = 232	11101000
pat5 = 8	00001000 .
pat6 = 142	10001110
pat7 = 128	10000000 .
pat8 = 232	11101000... .

View:*Vectorial hatch:*

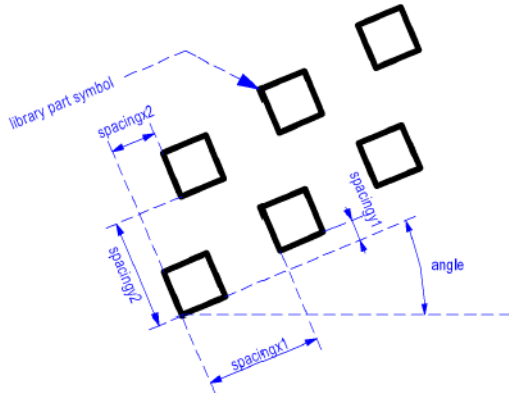
DEFINE SYMBOL_FILL

```

DEFINE SYMBOL_FILL name [FILLTYPES_MASK fill_types,]
    pat1, pat2, pat3, pat4, pat5, pat6, pat7, pat8,
    spacingx1, spacingy1,
    spacingx2, spacingy2, angle,
    scaling1, scaling2, macro_name PARAMETERS [name1 = value1, ... namen = valuen]
  
```

Hinweis: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe *“Zusätzliche Daten” auf Seite 180* für mehr Details.



Eine erweiterte Anweisung **DEFINE FILL**, die das Einschließen einer Bibliothekselementszeichnung in eine Schraffurdefinition ermöglicht. Die Verwendung von `macro_name` und den Parametern entspricht der beim Befehl **CALL**.

Parameter:

spacingx1, spacingx2: horizontale Abstände

spacingy1, spacingy2: vertikale Abstände

scaling1: horizontale Skalierung

scaling2: vertikale Skalierung

macro_name: der Name des Bibliothekselements

DEFINE SOLID_FILL

DEFINE SOLID_FILL name [**FILLTYPES_MASK** fill_types]

Definiert eine Vollschräffur.

Hinweis: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe *“Zusätzliche Daten” auf Seite 180 für weitere Details.*

DEFINE EMPTY_FILL

DEFINE EMPTY_FILL name [**FILLTYPES_MASK** fill_types]

Definiert eine leere Schräffur.

Hinweis: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe *“Zusätzliche Daten” auf Seite 180 für mehr Details.*

Linientypen

DEFINE LINE_TYPE

DEFINE LINE_TYPE name spacing, n,
length1, ... lengthn

Hinweis: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe *“Zusätzliche Daten” auf Seite 180 für weitere Details.*

In jedem GDL-Script können Linientypen definiert werden, die dann später über ihren Namen aufgerufen werden können. Der Linientyp, der auf diese Art und Weise in einem GDL-Script definiert wurde, kann nur in diesem Script oder in seinen eventuellen Unterprogrammen benutzt werden.

name: Name des Linientyps

spacing: Faktor Zwischenraum

n: Anzahl der Linienelemente

lengthi: Länge der Linienelemente (die wirkliche Länge ist $\text{spacing} * \text{lengthi}$). Linienelemente bestehen aus aufeinanderfolgenden Segmenten und Leeräumen. Das erste Schraffurelement ist ein Strich, die Länge Null kennzeichnet einen Punkt.

Anmerkung: Über die Anweisung `DEFINE LINE_TYPE`, können nur einfache Linientypen definiert werden, die nur aus aufeinanderfolgenden Strichen und Leerräumen bestehen. Es gibt keine Möglichkeit zur Definierung von Symbollinien.

Beispiel:

```
DEFINE LINE_TYPE "line - - ." 1,  
6, 0.005, 0.002, 0.001, 0.002, 0.0, 0.002
```

DEFINE SYMBOL_LINE

DEFINE SYMBOL_LINE name dash, gap, macro_name PARAMETERS [name1 = value1, ... namen = valuen]

Hinweis: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe *“Zusätzliche Daten” auf Seite 180 für mehr Details.*

Eine erweiterte Anweisung `DEFINE LINE`, die das Einschließen einer Bibliothekselementszeichnung in eine Liniendefinition ermöglicht. Die Verwendung von `macro_name` und den Parametern entspricht der beim Befehl `CALL`.

Parameter:

dash: Skalierung beider Linienkomponenten

gap: Abstand zwischen den einzelnen Komponenten

Stile

DEFINE STYLE

DEFINE STYLE name font_family, size, anchor, face_code

Sollte mit den Befehlen `TEXT2` und `TEXT` verwandt werden.

Jedes GDL-Script kann verschiedene Stilarten für Texte umfassen, die später über ihren Namen aufgerufen werden können. Der Textstil kann nur in diesem Script oder in eventuellen Unterprogrammen benutzt werden.

name: Name des Stils

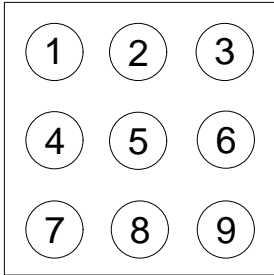
font_family: Name der verwendeten Schriftfamilie (z.B. Gramont).

size: Größe des Buchstabens "I" in mm oder m im Modellraum.

Wird der definierte Stil mit den Befehlen `TEXT2` und `TEXT` benutzt, bedeutet die Größe die Zeichenhöhe in Millimeter.

Bei Verwendung mit der `PARAGRAPH` -Taste in den Befehlen `RIGHTTEXT2` und `RIGHTTEXT` hängt die Höhenangabe in mm oder m vom festgelegten Höhenparameter der `TEXTBLOCK`-Definition ab. Gleichzeitig sind die Stilcodewerte für Kontur und Schatten nicht wirksam.

anchor: Code des Positionspunktes im Text



face_code: eine Kombination der folgenden Werte:

- 0 normal
- 1 **fett**
- 2 *kursiv*
- 4 unterstrichen
- 8 Kontur
- 16 Schatten

Anmerkung: Die Werte für Wert und Schatten sind nur auf dem Macintosh bis ArchiCAD 8.1 wirksam.

DEFINE STYLE {2}

DEFINE STYLE{2} name font_family, size, face_code

Neue Version der Stildefinition, die mit PARAGRAPH-Definitionen benutzt werden sollte.

name: Name des Stils

font_family: Name der verwendeten Schriftfamilie (z.B. Gramont).

size: Größe der Buchstaben in mm oder m im Modellraum.

face_code: eine Kombination der folgenden Werte:

- 0 normal
- 1 **fett**
- 2 *kursiv*
- 4 unterstrichen
- 32 hochgestellt
- 64 tiefergestellt
- 128 durchgestrichen

Wird der definierte Stil mit dem Befehl TEXT2 benutzt, wird die Zeichenhöhe in mm angegeben, während die Werte für Hoch- und Tieferstellung und Durchstreichen nicht wirksam sind. Wird er mit den PARAGRAPH-Tasten der Befehle RICHTEXT2 und RICHTEXT verwandt, hängt die Höhenangabe in mm oder meter vom Parameter fixierte_Höhe der TEXTBLOCK-Definition ab.

Paragraph

PARAGRAPH

```
PARAGRAPH name alignment, firstline_indent,
    left_indent, right_indent, line_spacing [,
    tab_size1, ...]
    [PEN index]
    [[SET] STYLE style1]
    [[SET] MATERIAL index]
    'string1'
    'string2'
    ...
    'string n'
    [PEN index]
    [[SET] STYLE style2]
    [[SET] MATERIAL index]
    'string1'
    'string2'
    ...
    'string n'
    ...
```

ENDPARAGRAPH

Jedes GDL-Script kann verschiedene Absatzdefinitionen umfassen, die später über ihren Namen aufgerufen werden können. Der definierte Absatz kann nur in diesem Script oder in eventuellen Unterprogrammen benutzt werden. Ein Absatz ist als eine willkürliche Anzahl von Zeichen (jeweils max. 256) mit unterschiedlichen Attributen definiert: Stil, Stift, und Material (3D). Sind in der Absatzdefinition keine Attribute angegeben werden aktuelle (oder Standard-) Attribute benutzt. Die neuen Linien, die in ein Absatz-String aufgenommen werden (durch das Sonderzeichen "\n") werden den String automatisch in gleiche Abschnitte unterteilen, die jeweils eine Zeile enthalten. Im Befehl TEXTBLOCK kann namentlich auf Absatzdefinitionen verwiesen werden. Alle Längenparameter (firstline_indent, left_indent, right_indent, tab_position) werden abhängig vom Parameter fixierte_Höhe der TEXTBLOCK-Definition in mm oder m angegeben.

name: Name des Absatzes

alignment: Ausrichtung der Absatz-Strings. mögliche Werte:

1: linksbündig, 2: zentriert, 3: rechtsbündig, 4: Blocksatz

firstline_indent: Einzug der ersten Linie, in mm oder m im Modellraum

left_indent: linker Einzug, in mm oder m im Modellraum

right_indent: rechter Einzug, in mm oder m im Modellraum

line_spacing: Zeilenabstandsfaktor. Der im aktuellen Stil definierte Grundabstand zwischen den Zeilen (Zeichengröße + Abstand zur nächsten Zeile) wird mit dieser Zahl multipliziert.

tab_positioni: aufeinander folgende Tabulatorpositionen (alle abhängig vom Absatzbeginn), in mm oder m im Modellraum. Tabulatoren in den Absatz-Strings springen an diese Position. Sind keine Tabulatorpositionen angegeben, werden die Grundwerte benutzt (12.7 mm).

Textblock

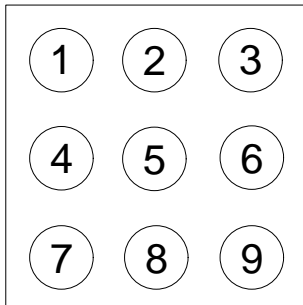
TEXTBLOCK

```
TEXTBLOCK name width, anchor, angle, width_factor, charspace_factor, fixed_height,  
    'string_expr1' [, 'string_expr2', ...]
```

Textblockdefinition. Jedes GDL-Script kann verschiedene Textblockdefinitionen umfassen, die später über ihren Namen aufgerufen werden können. Der definierte Textblock kann nur in diesem Script oder in eventuellen Unterprogrammen benutzt werden. Ein Textblock wird durch eine beliebige Anzahl von Strings oder Absätzen definiert, die mit *“RICHTEXT2” auf Seite 129* und *“RICHTEXT” auf Seite 97* angeordnet werden können. Der Befehl *“REQUEST (“TEXTBLOCK_INFO”, textblock_name, width, height)”* zeigt Informationen zur berechneten Höhe und Breite des Textblocks an.

width: Breite des Textblocks in mm oder m im Modellraum, bei 0 wird sie automatisch berechnet

anchor: Code des Positionspunktes im Text



angle: Drehwinkel des Textblocks in Grad

width_factor: Zeichenabstandsfaktor. Die Buchstabenbreite des aktuellen Stils wird um diese Zahl multipliziert.

charspace_factor: Buchstabenabstandsfaktor. Die horizontale Entfernung zwischen den Buchstaben wird mit dieser Zahl multipliziert.

charspace_factor: mögliche Werte

1: der platzierte TEXTBLOCK ist nicht maßstabsabhängig und alle spezifischen Längenparameter sind in mm angegeben

0: der platzierte TEXTBLOCK ist maßstabsabhängig und alle spezifischen Längenparameter m im Modellraum.

string_expri: bedeutet Absatzname, wenn dieser vorher definiert wurde, sonst einen einfachen String (mit Standardabsatzparametern).

Zusätzliche Daten

Attributdefinitionen können nach dem Schlüsselwort **ADDITIONAL_DATA** optionale zusätzliche Datendefinitionen enthalten. Die zusätzlichen Daten müssen nach den zuvor definierten Parametern des Attributbefehls eingegeben werden. Zusätzliche Daten haben einen Namen (Name;) und einen Wert (Wert;), der ein Ausdruck eines beliebigen Typs sein kann, auch ein Array. Endet ein Stringparametername mit “_file”, ist dessen Wert als Dateiname anzusehen und ist im Archivprojekt enthalte. In ArchiCAD oder Add-Ons können unterschiedliche Bedeutungen für zusätzliche Daten definiert werden.

Siehe Bedeutung der *LightWorks* Add-On Parameter unter <http://www.graphisoft.com/support/developer/documentation/wide/LibraryDevKit/>.

Zusätzliche Datendefinitionen sind bei den folgenden Befehlen verfügbar:

DEFINE MATERIAL

DEFINE MATERIAL parameters [ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE FILL

DEFINE FILL parameters [ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE FILL_A

DEFINE FILL_A parameters [ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE SYMBOL_FILL

DEFINE SYMBOL_FILL parameters [ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE LINE_TYPE

DEFINE LINE_TYPE parameters [ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE SYMBOL_LINE

DEFINE SYMBOL_LINE parameters [ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

Externe Dateiabhängigkeit

FILE_DEPENDENCE "name1" [, "name2", ...]

Sie können externe Dateien auflisten, auf die sich Ihr GDL-Script beziehen soll. Dateinamen sollten konstante Strings sein.

Alle hier angegebenen Dateien werden in das Archivprojekt eingefügt (wie in *CALL*-Anweisungen eingesetzte konstante Makronamen und die in verschiedenen GDL-Befehlen benutzten konstanten Bildnamen). Diese Befehl funktioniert nur auf dieser Ebene: sind die angegebenen Dateien Bibliothekselemente, werden deren aufgerufene Makros nicht eingeschlossen.

Dieser Befehl ist nützlich, wenn in benutzerdefinierten Positionen des GDL-Scripts auf externe Dateien verwiesen wird, z.B.: Dateiparameter *ADDITIONAL_DATA*, Daten in Dateioperationen.

NICHT GEOMETRISCHE SCRIPTS

Neben den 3D- und 2D-Script Fenstern, die das Erscheinen des GDL-Objektes definieren, sind weitere Scripts zum Hinzufügen von ergänzenden Informationen verfügbar. Diese sind das Eigenschaften-Script, das für Mengenberechnungen verwendet wird, das Parameterscript, das eine Liste von möglichen Werten für verschiedene Parameter beinhaltet, sowie das Script für die Benutzerschnittstelle, das zum Erstellen von eigenen Benutzerschnittstellen für die Parametereingabe dient. Die Befehle für alle diesen Script-Typen werden auf den folgenden Seiten detailliert beschrieben.

DAS EIGENSCHAFTEN-SCRIPT

Die Bibliothekselemente haben ein Fenster für das Eigenschaften-Script. Dieses Script ermöglicht das Erstellen von parameterabhängigen Bibliothekselementeigenschaften und die Festlegung ihrer Position in der Komponentenliste durch Anweisungen. Mit einigen Befehlen können Sie die lokalen Beschreibungen, die Bestandteile sowie die Zeichen im Script definieren (die im Beschreibungen-Fenster früherer ArchiCAD-Versionen erstellt wurden). Beschreibungen und Bestandteile können auch von externen Datenbankenbezogen werden. Die Länge des Codes darf 32 Zeichen nicht überschreiten.

Im Eigenschaften-Script können Sie alle GDL-Befehle verwenden, die keine Körper generieren.

DATABASE_SET

DATABASE_SET set_name [descriptor_name, component_name, unit_name, key_name, criteria_name, list_set_name]

Definition oder Auswahl des Datenbanksatzes. Wird dieser Befehl in ein MASTER_GDL-Script eingefügt, definiert er einen Datensatz, der Beschreibungs-, Komponenten-, Einheiten-, Schlüssel-, Kriterien- und Listendateien enthält.

Auf diesen Datensatznamen kann dann vom Eigenschaftenscript unter Verwendung des gleichen Befehls mit lediglich dem Parameterdatensatz als Vereinbarung Bezug genommen werden, indem der eigentliche Datensatz gewählt wird, auf den sich REF COMPONENTs und REF DESCRIPTORS beziehen. Der Name des Standarddatensatzes ist "Standardsatz" und wird benutzt, wenn kein anderer Satz ausgewählt wurde.

Die Standarddatensatz-Dateinamen sind:

DESCDATA, COMPDATA, COMPUNIT, LISTKEY, LISTCRIT, LISTSET.

Scripte können eine beliebige Anzahl von DATABASE_SET-Auswahlen enthalten.

set_name: Name des Datensatzes

descriptor_name: Name der Beschreibungsdatei

component_name: Name der Komponentendatei

unit_name: Name der Einheitendatei

key_name: Name der Schlüsseldatei

criteria_name: Name der Kriteriendatei

list_set_name: Name der Listendatei

DESCRIPTOR

DESCRIPTOR name [, code, keycode]

Definition der lokalen Beschreibungen. können eine beliebige Anzahl von DESCRIPTORS beinhalten.

name: kann länger sein als eine Zeile. Neue Zeilen können mit dem Zeichen '\n' und Tabulatoren mit '\t' definiert werden. Wenn Sie einem Zeilenende '\' hinzufügen, können Sie mit dem Text in der nächsten Zeile fortfahren, ohne eine neue Zeile zu addieren. Falls das '\' Zeichen innerhalb des Textes doppelt erscheint (\\), wird es seine Kontroll-Funktion verlieren und bedeutet einfach '\\'. Der Umfang des Textes (inkl. neuer Zeilen-Zeichen) darf 255 Zeichen nicht überschreiten: zusätzliche Zeichen werden von dem Compiler einfach geschnitten. Wenn Sie einen längeren Text benötigen, verwenden Sie mehrere DESCRIPTORS.

code: Text, definiert einen Code für die Beschreibungen

keycode: Zeichenfolge, Bezug auf einen Key in externer Datenbank.

Der Key wird der Beschreibung zugewiesen.

REF DESCRIPTOR

REF DESCRIPTOR code [, keycode]

Bezug durch einen Code-Text auf einer Beschreibung in externer Datenbank.

COMPONENT

.

COMPONENT

COMPONENT name, quantity, unit [, proportional_with, code, keycode, unitcode]

Definition des lokalen Bestandteiles. Scripte können beliebige Anzahl von COMPONENTs beinhalten.

name: Name des Bestandteiles (max. 128 Zeichen)

quantity: Menge, ein numerischer Ausdruck

unit: Text, verwendet für Einheitsbeschreibung

proportional_with: ein Code zwischen 1-6. Wenn es aufgelistet wird, wird das oben definierte Volumen des Bestandteiles automatisch mit dem für das gegenwärtig aufgelistete Element berechneten Wert multipliziert:

- 1: Element
- 2: Länge
- 3: Oberfläche A
- 4: Oberfläche B
- 5: Oberfläche
- 6: Volumen

code: Text, definiert einen Code für den Bestandteil

keycode: Zeichenfolge, Bezug auf einen Key in externer Datenbank. Der Key wird dem Bestandteil zugewiesen.

unitcode: Text, Bezug auf eine Einheit in einer externen Datenbank, die das Ausgabeformat des Bestandteil-Voluments kontrolliert. Dadurch wird der lokale definierte Einheitstext ersetzt.

REF COMPONENT

REF COMPONENT code [, keycode [, numeric_expression]]

Bezug durch einen Codetext auf einen Bestandteil in einer externen Datenbank. Der zu multiplizierende Wert wird in der Datenbank des Bestandteiles durch den hier beschriebenen, optionalen, numerischen Ausdruck überschrieben.

BINARYPROP

BINARYPROP

Binaryprop stellt einen Bezug zu den binären Daten (Bestandteile und Beschreibungen) her, die im Teil des Bibliothekselementes für Bestandteile/Beschreibungen definiert werden.

DATABASE_SET-Vereinbarungen haben keinen Einfluß auf die Binärdaten.

SURFACE3D ()

SURFACE3D ()

Mit der Funktion SURFACE3D () erhalten Sie die Oberfläche der 3D-Form des Bibliothekselements.

Warnung: Wenn Sie eine oder mehrere Formen mit den gleichen Parametern an der gleichen Stelle platzieren, erhalten Sie mit dieser Funktion die Gesamtsumme aller Oberflächen von Formen.

VOLUME3D ()

VOLUME3D ()

Mit der Funktion VOLUME3D () erhalten Sie das Volumen der 3D-Form des Bibliothekselements.

Warnung: Wenn Sie eine oder mehrere Formen mit den gleichen Parametern an der gleichen Stelle platzieren, erhalten Sie mit dieser Funktion die Gesamtsumme aller Volumen von Formen.

POSITION

POSITION position_keyword

Ist nur in der Massen-/Stückliste wirksam.

Ändert nur den Elementtyp, dem die folgenden Beschreibungen und Komponenten zugeordnet werden. Gibt es solche Anweisungen im Eigenschaften-Script nicht, werden Beschreibungen und Komponenten bei ihren Standardelementen aufgelistet.

Folgende sind die Codewörter:

WALLS
COLUMNS
BEAMS
DOORS
WINDOWS
OBJECTS
CEILS
PITCHED_ROOFS
LIGHTS
HATCHES
ROOMS
MESHES

Eine Anweisung bleibt für alle gelungenen DESCRIPTORS und COMPONENTs nur solange gültig, bis die nächste Anweisung zugeschrieben wird. Ein Script kann eine beliebige Anzahl von Anweisungen beinhalten.

Beispiel:

```
DESCRIPTOR "\tPainted box.\n\t Properties:\n\t\t\t - swinging doors\n\t\t\t - adjustable height\n\t\t\t - scratchproof"
REF DESCRIPTOR "0001"
s = SURFACE3D () !wardrobe surface
COMPONENT "glue", 1.5, "kg"
COMPONENT "handle", 2*c, "nb"!c number of doors
COMPONENT "paint", 0.5 * s, "kg"
POSITION WALLS
REF COMPONENT "0002"
```

DRAWING

DRAWING

DRAWING: Nimmt Bezug auf die im 2D-Script desselben Bibliothekselementes beschriebene Zeichnung. Verwenden Sie diesen Befehl, um die Zeichnungen in Ihre Materialliste plazieren zu können.

DAS PARAMETER-SCRIPT

Parameterlisten sind Sätze von möglichen numerischen oder String-Werten. Sie können auf die Parameter gemäß der Definition im Werteliste-Script des Bibliothekselements oder im MASTER_GDL-Script angewandt werden. Der Parameter muss einfach sein. Die Typenkompatibilität wird vom GDL-Compiler überprüft.

Das Parameterscript wird jedesmal interpretiert, wenn ein Parameterwert vom Typ Werteliste geändert werden soll. Die möglichen im Script definierten Werte erscheinen in einem Popup-Menü.

Werte

VALUES "fillparam_name" [**FILLTYPES_MASK** fill_types,] value_definition1 [, value_definition2, ...]

name: der Name des Parameters

maski = j1 + 2*j2 + 4*j3 *j4

j1: Bauteilschraffuren

j2: Deckschraffuren

j3: Zeichenschraffuren

Kann nur für Deckschraffurparameter benutzt werden. Ist j gesetzt, enthält das auf den Parameter "fillparam_name" bezogene Schraffur-Popup nur die Schraffuren des angegebenen Typs. Grundeinstellung ist alle Schraffuren (0).

value_definitioni: Wertdefinition, möglich ist:

expressioni: numerischer oder String-Ausdruck, oder

CUSTOM: Schlüsselwort, das bedeutet, daß ein beliebiger benutzerdefinierter Wert eingegeben werden kann, oder

RANGE left_delimiter [expression₁], [expression₂]
 right_delimiter [**STEP** step_start_value,
 step_value]

Bereichsdefinition mit optionaler Schrittweite

left_delimiter: [, größer-gleich '>=', oder (, größer '>'

expression1: untere Grenze

expression2: obere Grenze

right_delimiter:], kleiner-gleich '<=', oder), kleiner '<'

step_start_value: Startwert

step_value: Schrittwert

Beispiele:

```
VALUES "par1" 1, 2, 3
VALUES "par2" "a", "b"
VALUES "par3" 1, CUSTOM, SIN (30)
VALUES "par4" 4,RANGE(5, 10],12,RANGE(,20]
    STEP 14.5, 0.5, CUSTOM

! Beispiel für das Lesen aller String-Werte einer Datei
! und deren Verwendung in einer Werteliste
DIM sarray[]
! file in the library, containing parameter data
filename = "ProjectNotes.txt"
ch1 = OPEN ("text", filename, "MODE=RO, LIBRARY")
i = 1
j = 1
sarray[1] = ""
! collect all strings
DO
n = INPUT (ch1, i, 1, var)
IF n > 0 AND VARTYPE (var) = 2 THEN
sarray[j] = var
j = j + 1
ENDIF
i = i + 1
WHILE n > 0
CLOSE ch1
! Parameter-Popup mit Strings, die dem Abschnitt
VALUES "RefNote" entnommen wurden.
```

Parameter

```
PARAMETERS name1 = expression1 [,
    name2 = expression2, ...,
    namen = expressionn]
```

name: der Name des Parameters

expressioni: der neue Wert des Parameters

Mit diesem Befehl können die Parameterwerte eines Bibliothekselements durch das Parameterscript geändert werden.

Die Änderung wirkt sich nur auf die nächste Interpretation aus. Makrobefehle beziehen sich auf die Parameter des Aufrufenden. Wenn der Parameter eine Werteliste ist, wird der gewählte Wert entweder ein vorhandener Wert, der benutzerdefinierte Wert oder der erste Wert der Werteliste sein.

Zusätzlich enthält die globale Zeichenfolgevariable GLOB_MODPAR_NAME den Namen des letzten vom Anwender geänderten Parameters.

LOCK

LOCK name1 [, name2, ..., namen]

Schützt den benannten Parameter im Dialogfeld Einstellungen. Ein geschützter Parameter erscheint abgeblendet im Dialogfeld und sein Wert kann nicht vom Anwender geändert werden.

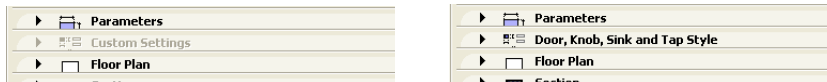
HIDEPARAMETER

HIDEPARAMETER name1 [, name2, ..., namen]

Verbirgt die benannten Parameter und ihre untergeordneten Parameter in dem Dialogfenster Einstellungen. Ein mit diesem Befehl verborgener Parameter im Parameter-Script verschwindet automatisch aus der Parameterliste.

DIE BENUTZEROBERFLÄCHE (USER INTEFACE SCRIPT)

Mit einigen GDL-Befehlen kann in einem Einstellungsdialogfeld für Bibliothekselemente eine neue Registerkarte definiert werden. Wenn Sie die Schaltfläche "Als Standard festlegen" im Editor für Bibliothekselemente anklicken, wird die grafische Bedienung standardmäßig in den Einstellungen des Elements verwendet. Parameter mit benutzerdefinierter Steuerung werden nicht automatisch in der Originalparameterliste ausgeblendet, sie können jedoch manuell im Editor für Bibliothekselement ausgeblendet werden.



Der Nullpunkt des Koordinatensystems befindet sich in der linken oberen Ecke. Größen und Koordinatenwerte werden in Pixel gemessen.

UI_DIALOG

UI_DIALOG title [, size_x, size_y]

Definiert den Titel des Dialogfelds. Die Größe des verfügbaren Bereichs ist jetzt auf 444 x 266 Pixel fixiert, die Parameter size_x und size_y werden nicht verwendet.

Einschränkung: Das Schnittstellenscript kann nur einen UI_DIALOG-Befehl enthalten.

UI_PAGE

UI_PAGE page_number

Seitenanweisung, definiert die Seite, auf der die Schnittstellenelemente platziert werden. Die Seitennumerierung beginnt bei 1. Blättern kann definiert werden, indem zwei Schaltflächen mit den Befehlen UI_NEXT und UI_PREV erstellt werden.

Wenn das Schnittstellenscript keinen UI_PAGE-Befehl enthält, wird jedes Element standardmäßig auf der ersten Seite platziert.

Achtung: Wenn die Kontinuität in der Seitenliste nicht gewahrt wird, wird eine neue Seite ohne Schaltflächen eingefügt, so daß es nicht möglich ist, von dieser Seite auf eine andere Seite zu blättern.

UI_BUTTON

UI_BUTTON type, text, x, y, width, height

Schaltflächendefinition auf der aktuellen Seite. Schaltflächen können zum Durchblättern der Seiten verwendet werden. Sie werden nicht automatisch erzeugt und müssen für jede Seite definiert werden.

type: Schaltflächentyp wie folgttext: der Text sollte auf der Schaltfläche erscheinen

x, y: die Position der Schaltfläche

width, height: Breite und Höhe in Pixel

UI_SEPARATOR

UI_SEPARATOR x1, y1, x2, y2

Erzeugt eine Trennungsrechteck.Das Rechteck wird zu einer vertikalen ($x1 = x2$) oder horizontalen ($y1 = y2$) Trennlinie

x1, y1: Koordinaten des Ausgangspunkts oben links (Koordinaten des Startpunktes der Linie)

x2, y2: x2, y2: Koordinaten des Endpunkts unten links (Koordinaten des Endpunktes der Linie)

UI_GROUPBOX

UI_GROUPBOX text, x, y, width, height

Ein Gruppenfeld ist eine rechteckige Trennung. Sie kann dazu verwendet werden, logisch miteinander verbundene Parameter optisch als Gruppe darzustellen.

text: der Titel des Gruppenfelds

x, y: die Position der linken oberen Ecke

width, height: Breite und Höhe in Pixel

UI_PICT

UI_PICT expression, x, y [,width, height]

Bildelement im Dialogfenster. Die Bilddatei muß sich in einer der geladenen Bibliotheken befinden.

expression: Dateiname oder Indexnummer des im Bibliothekselement gespeicherten Bildes. Der Index 0 bezieht sich auf das Vorschaubild des Bibliothekselements.

x, y: Position der linken oberen Ecke des Bildes .

width, height: wählbare Breite und Höhe in Pixel; standardmäßig werden die Originalwerte für Breite und Höhe des Bildes verwendet.

UI_STYLE

UI_STYLE *fontsize, face_code*

Alle UI_OUTFIELDSs und UI_INFIELDSs, die nach diesem Schlüsselwort erzeugt werden, tragen diesen Stil bis zum nächsten UI_STYLE-Befehl.

fontsize: einer der folgenden Fontgrößenwerte

- 0: klein
- 1: sehr klein
- 2: groß

face_code: ähnlich wie die Definition STYLE, die Werte können jedoch nicht kombiniert werden.

- 0: normal
- 1: **fett**
- 2: *kursiv*
- 4: unterstrichen
- 8: Kontur
- 16: Schatten

UI_OUTFIELD

UI_OUTFIELD *expression, x, y, width, height*

Erzeugt ein festes Textfeld.

expression: numerischer oder Zeichenfolge-Ausdruck

x, y: Position der linken oberen Ecke des Textblocks

width, height: Breite und Höhe des Texts in Pixel

UI_INFIELD

UI_INFIELD "name", *x, y, width, height* [,
version_flag, picture_name,
images_number,
rows_number, cell_x, cell_y,
image_x, image_y,
expression_image1, text1,
...,
expression_imagen, textn]

UI_INFIELD {2}

```
UI_INFIELD{2} name, x, y, width, height [,  
    version_flag, picture_name,  
    images_number,  
    rows_number, cell_x, cell_y,  
    image_x, image_y,  
    expression_image1, text1,  
    ...,  
    expression_imagen, textn]
```

Erzeugt ein Textbearbeitungs- oder Popup-Menü für die Parametereingabe. Ein Popup wird erzeugt, wenn der Parametertyp Werteliste, Material, Schraffur, Linientyp oder Stiftfarbe ist.

Wenn die wählbaren Parameter des Befehls vorhanden sind, können die Wertelisten alternativ als Miniaturansichten angezeigt werden. Miniaturansichten zeigen die gewählten Bilder und den damit assoziierten Text an, haben Bildlaufleisten an den Rändern und ermöglichen die Auswahl jeweils eines einzelnen Elements, genau wie in einem Popup-Menü.

Das Schnittstellenscript wird mit dem neuen Wert neu aufgebaut, nachdem ein Parameter geändert wurde.

name: Parametername als Zeichenfolge-Ausdruck für UI_INFIELD oder Parametername mit optionalen tatsächlichen Indexwerten bei Arrays für UI_INFIELD{2}

x, y: die Position des zu bearbeitenden Texts, des Popup oder des Steuerelements

width, height: Breite und Höhe in Pixel

version_flag: reserviert, immer 1

picture_name: Name der gemeinsamen Bilddatei, die eine Matrix verketteter Bilder enthält, oder leere Zeichenfolge

images_number: Anzahl von Bildern in der Matrix

rows_number: Anzahl von Zeilen in der Matrix

cell_x, cell_y: Breite und Höhe einer Zelle innerhalb der Miniaturansicht, einschließlich Bild und Text

image_x, image_y: Breite und Höhe des Bildes in der Zelle

expression_imagei: Index der Bildnummer i in der Matrix oder der individuelle Dateiname. Wenn ein allgemeiner Bilddateiname angegeben wurde, müssen hier Indizes verwendet werden. Kombinationen aus Indizes und individuellen Dateinamen können nicht verwendet werden.

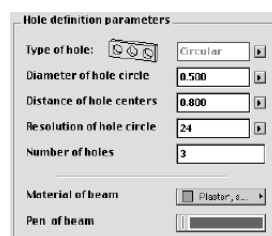
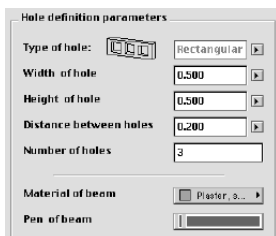
texti: Text in der Zelle i

Beispiel:

```

IF c THEN
  UI_DIALOG "Hole definition parameters"
  UI_OUTFIELD "Type of hole:",15,40,180,20
  UI_INFIELD "D",190,40,105,20
  IF D="Rectangular" THEN
    UI_PICT "rect.pict",110,33,60,30
    UI_OUTFIELD "Width of hole",15,70,180,20
    UI_INFIELD "E", 190,70,105,20
    UI_OUTFIELD "Height of hole",15,100,180,20
    UI_INFIELD "F", 190,100,105,20
    UI_OUTFIELD "Distance between
                  holes",15,130,180,20
    UI_INFIELD "G", 190,130,105,20
  ELSE
    UI_PICT "circle.pict",110,33,60,30
    UI_OUTFIELD "Diameter of hole
                  circle",15,70,180,20
    UI_INFIELD "J", 190,70,105,20
    UI_OUTFIELD "Distance of hole
                  centers",15,100,180,20
    UI_INFIELD "K", 190,100,105,20
    UI_OUTFIELD "Resolution of hole
                  circle",15,130,180,20
    UI_INFIELD "M", 190,130,105,20
  ENDIF
  UI_OUTFIELD "Number of holes",15,160,180,20
  UI_INFIELD "I", 190,160,105,20
ENDIF
UI_SEPARATOR 50,195,250,195
UI_OUTFIELD "Material of beam", 15,210,180,20
UI_INFIELD "MAT", 190,210,105,20
UI_OUTFIELD "Pen of beam", 15,240,180,20
UI_INFIELD "P", 190,240,105,20

```



AUSDRÜCKE UND FUNKTIONEN

Alle Parameter der GDL-Körper können das Ergebnis von Berechnungen sein. Sie können z.B. bestimmen, daß die Höhe eines Zylinders das Fünffache seines Radius sein soll. Ebenso kann auch - bevor Sie einen Würfel definieren - das gesamte Koordinatensystem jeweils um die Länge der halben Würfelseite verschoben werden, so daß der Ursprung immer im Mittelpunkt des Würfels und nicht mehr in der linken unteren Ecke liegt. Um diese Berechnungen ausführen zu können, bietet GDL die folgenden mathematischen Werkzeuge an: Ausdrücke, Operatoren und Funktionen.

AUSDRÜCKE (EXPRESSIONS)

Sie können zusammengesetzte Ausdrücke in GDL-Befehle integrieren. Ausdrücke können von numerischem und textlichem Typ sein. Sie können aus Konstanten, Variablen, Parametern oder Funktionen, die mit Operationen verknüpft sind, und deren Kombination bestehen. Runde Klammernpaare (()) (1. Priorität) zeigen an, daß der Ausdruck in der Klammer vorrangig berechnet werden muß.

Einfache Variablen können numerische Werte und Texte sogar im selben Script festlegen und können jeweils in numerischen und textlichen Ausdrücken verwendet werden. Texte ergebende Operationen KÖNNEN NICHT direkt als Makronamen in Makroaufrufen oder als Attributnamen in Material-, Schraffur-, Linientyp-, oder Stildefinitionen benutzt werden. Variablen mit Textwert werden als Strings behandelt und können überall eingesetzt werden, wo Textwerte erforderlich sind. Wird denselben Variablen im Script später ein numerischen Wert zugeordnet, können sie solange nur in numerischen Ausdrücken verwendet werden, bis ihnen erneut ein Textwert zugeordnet wird. Falls möglich, werden die Ausdruckstypen in der Precompilation überprüft.

GDL unterstützt ein- und zweidimensionale Anordnungen. Variablen werden nach einer Behauptung, in der ihre maximalen Bemaßungen bestimmt werden, zu *Anordnungen*.

DIM

```
DIM var1[dim_1], var2[dim_1][dim_2], var3[ ],  
      var4[ ][ ], var5[dim_1][ ],  
      var5[ ][dim_2]
```

Nach dem DIM-Schlüsselwort kann eine beliebige Anzahl von Variablennamen, durch Kommas getrennt, stehen. var1, var2, ... sind die Array-Namen, während die Nummern zwischen den Klammern die Dimensionen des Arrays (numerische Konstanten) angeben. Variablenausdrücke können nicht als Dimensionen verwendet werden. Wenn sie nicht angegeben sind, gilt der Array als dynamisch (eine oder zwei Dimensionen).

Parameter für Bibliothekselemente können auch Arrays sein. Ihre tatsächlichen Dimensionen sind im Dialogfenster "Bibliothekselement" angegeben. Parameter-Arrays brauchen im Script nicht deklariert zu werden und können standardmäßig dynamisch sein. Beim Verweisen auf das Bibliothekselement mithilfe einer CALL-Anweisung können die tatsächlichen Werte eines Array-Parameters einen Array mit zufälligen Dimensionen darstellen.

Auf die Elemente der Arrays kann überall im Script verwiesen werden; wenn es sich um Variablen handelt, ist solch ein Verweis nur nach der Deklaration möglich.

```
var1[num_expr] or var1
```

```
var2[num_expr1][num_expr2] or var2[num_expr1]  
or var2
```

Das Schreiben des Array-Namens ohne tatsächliche Indizes bedeutet einen Verweis auf den gesamten Array (oder auf eine Zeile eines zweidimensionalen Arrays); dies ist in einigen Fällen zulässig (CALL, PRINT, LET, PUT, REQUEST, INPUT, OUTPUT, SPLIT Anweisungen). Bei dynamischen Arrays gibt es keine Begrenzung hinsichtlich des tatsächlichen Indexwertes. Wird einem nicht vorhandenen dynamischen Array-Element ein Wert zugewiesen, so wird bei der Interpretation die erforderliche Menge Speicher zugeordnet und alle fehlenden Elemente werden auf 0 (numerisch) gesetzt.

Achtung! Dies kann in manchen Fällen zu einem unerwarteten Speicherüberlaufsfehler führen. Jeder Index - auch mit einem vermutlich falschen, sehr großen Wert - wird als gültig angesehen, da der Interpretierer die Fehlerbedingung nicht erkennen kann. Ein nicht vorhandenes dynamisches Array-Element ist 0 (numerisch).

Arrays mit einer festen Dimension werden auf Gültigkeit des tatsächlichen Index in der festen Dimension geprüft. Feste Arrays können keine vollständigen dynamischen Array-Werte zugeordnet werden. Dynamische Arrays, denen vollständige Array-Werte zugeordnet werden, nehmen jedoch diese Werte an. Dies gilt auch für einige Anweisungen, bei denen Verweise auf vollständige Arrays als Rückgabeparameter verwendet werden können. (REQUEST, INPUT, SPLIT).

Array-Elemente können in beliebigen numerischen oder Zeichenfolge-Ausdrücken verwendet werden. Diesen Elementen können Zeichenfolge- oder numerische Werte zugeordnet werden.

Indizes beginnen mit 1, und jeder numerische Ausdruck kann als Index verwendet werden.

Array-Elemente können verschiedene einfache Typen haben (numerisch, Zeichenfolge, Gruppe). Der Typ des gesamten Arrays ('Haupttyp') ist der Typ des ersten Array-Elements ([1] oder [1][1]). Parameter und globale variable Arrays können keinen gemischten Typ haben.

VARDIM1(expr)

VARDIM1 (expr)

VARDIM2(expr)

VARDIM2 (expr)

Diese Funktionen geben als ganze Zahlen die tatsächlichen Dimensionswerte für die als Parameter angegebenen (Array-)Ausdrücke an. Sie müssen verwendet werden, wenn Sie alle tatsächlichen Elemente eines dynamischen Arrays oder eines Array-Parameters korrekt verarbeiten wollen. Wenn kein Element eines dynamischen Arrays zuvor gesetzt wurde, ist der Rückgabewert 0. Für eindimensionale Arrays gibt VARDIM2 den Wert 0 zurück.

Beispiele für numerische Ausdrücke:

```
Z
5.5
(+15)
-X
A*(B+C)
SIN(X+Y)*Z
A+R*COS(I*D)
5' 4"
SQR (x^2 + y^2) / (1 - d)
a + b * sin (alpha)
height * width
```

Beispiele für textliche Ausdrücke:

```
"Konstante Zeichenfolge"
name + STR ("%m", i) + "." + ext
string_param <> "Mode 1"
```

Beispiele für Ausdrücke, die Anordnungswerte verwenden:

```
DIM tab [5], tab2 [3][4] ! declaration
tab [1] + tab [2]
tab2 [2][3] + A
PRINT tab

DIM f1 [5], v1[], v2[][]
v1[3] = 3 !v1[1] = 0, v1[2] = 0, array of 3
           ! elements
v2[2][3] = 23 ! all other elements(2 X 3) = 0
PRINT v1, v2

DIM f1 [5], v1[], v2[][]
FOR i = 1 TO VARDIM1(f1)
    f1 [i] = i
NEXT I
v1 = f1
v2 [1] = f1
PRINT v1, v2
```

OPERATOREN

Die untenstehenden Rechenoperatoren wurden in der Reihenfolge ihrer Priorität aufgelistet. Die Berechnung eines Ausdrucks beginnt mit der Rechenoperation der höchsten Priorität, ansonsten von links nach rechts.

Arithmetische Operatoren

^ (oder **)	Potenz	2. Priorität
*	Multiplikation	3. Priorität
/	Division	3. Priorität
MOD (oder %)	Modulo (verbleibender Teil)	3. Priorität
	$X \text{ MOD } Y = X - Y * \text{INT} (X/Y)$	
+	Addition	4. Priorität
-	Subtraktion	4. Priorität

Anmerkung: + (Addition) kann auch bei den Textausdrücken verwendet werden: das Ergebnis ist die Verknüpfung der Texte.

Das Ergebnis der '/' Division ist immer eine reale Zahl. Das Ergebnis der anderen Operationen hängt hingegen von der Art der Operatoren ab: sind alle Operatoren ganzzahlig, ist das Ergebnis ebenfalls ganzzahlig, sonst real .

Relationale Operatoren

=	ist gleich	5. Priorität
<	kleiner als	5. Priorität
>	größer als	5. Priorität
<=	kleiner als oder gleich	5. Priorität
>=	größer als oder gleich	5. Priorität
<> (oder #)	ungleich	5. Priorität

Anmerkung: + (Addition) kann auch bei den Textausdrücken verwendet werden: das Ergebnis ist die Verknüpfung der Texte. Das Ergebnis ist ganzzahlig, 1 oder 0. Es ist nicht empfehlenswert, die Operatoren, '=' (gleich), '<=' (kleiner gleich), '>=' (größer gleich), '<>' (oder #) (ungleich) mit realen Operanden zu benutzen, da dies Genauigkeitsprobleme verursachen kann.

Bool'sche Operatoren

AND (oder &)	UND-Verknüpfung	6. Priorität
OR (oder)	einschließlich ODERverknüpfung,	7. Priorität
EXOR (or @)	Logical exclusive or precedence	8

Anmerkung: Bool'sche Operatoren arbeiten mit ganzen Zahlen. In der Konsequenz bedeutet 0 deshalb "falsch" und alle anderen Zahlen "richtig". Der Wert eines logischen Ausdrucks ist ebenfalls ganzzahlig, z.B. 1 für "richtig" und 0 für "falsch". Bool'sche Operatoren sollten nicht mit realen Operanden benutzt werden, weil diese Operationen zu Genauigkeitsproblemen führen können.

FUNKTIONEN

Arithmetische Funktionen

ABS

ABS (x) ergibt den absoluten Wert von x (ganzzahlig, falls x ganzzahlig, sonst real).

CEIL

CEIL (x) Übergibt den kleinsten Ganzzahlwert, der nicht kleiner ist als x (immer ganzzahlig). (e.g., $\text{CEIL}(1.23) = 2$; $\text{CEIL}(-1.9) = -1$).

INT

INT (x) Ergibt den ganzzahligen Teil von x (immer ganzzahlig). (z.B. $\text{INT}(1.23) = 1$, $\text{INT}(-1.23) = -2$).

FRA

FRA (x) Ergibt den Bruchzahlenteil von x. (z.B., $\text{FRA}(1.23) = 0.23$, $\text{FRA}(-1.23) = 0.77$).

ROUND_INT

ROUND_INT (x) Ergibt den gerundeten ganzzahligen Teil von x. Der Ausdruck 'i = ROUND_INT (x)' entspricht dem folgenden Script:
 IF x < 0.0 THEN i = INT (x - 0.5) ELSE i = INT (x + 0.5)

SGN

SGN (x) Ergibt +1 wenn x positiv ist, -1 wenn x negativ ist ansonsten 0.

SQR

SQR (x) Ergibt die Quadratwurzel von x (immer real).

Winkelfunktionen

Winkelfunktionen für Sinus, Cosinus, Tangens, Arcsinus, Arccosinus und Arctangens

ACS

ACS (x) Ergibt den Arccosinus von x. ($-1.0 \leq x \leq 1.0$; $0^\circ \leq \text{ACS}(x) \leq 180^\circ$).

ASN

ASN (x) Ergibt den Arcsinus von x. ($-1.0 \leq x \leq 1.0$; $-90^\circ \leq \text{ASN}(x) \leq 90^\circ$).

ATN

ACS (x) Ergibt die Arctangente von x. ($-90^\circ \leq \text{ATN}(x) \leq 90^\circ$).

COS

COS (x) Ergibt den Cosinus von x.

SIN

SIN (x) Ergibt den Sinus von x.

TAN

TAN (x) Ergibt den Tangens von x.

PI

PI Ergibt die Ludolph Konstante . ($\pi = 3.1415926\dots$).

Anmerkung: Alle Ergebniswerte sind *real*.

Transzendente Funktionen

EXP

EXP (x) Ergibt die x^{te} Potenz von e ($e = 2.7182818$).

LGT

LGT (x) ergibt den dezimalen Logarithmus von x.

LOG

LOG (x) ergibt den natürlichen Logarithmus von x.

Anmerkung: Alle Ergebniswerte sind *real*.

Bool'sche Funktionen

NOT

NOT (x) Ergibt falsch (=0.0), wenn x wahr ist (0.0),
und wahr (=1.0), wenn x falsch (=0.0) ist.
(Logische Verneigung).

Anmerkung: Parameterwert sollte *ganzzahlig* sein.

Statistische Funktionen

MIN

MIN (x1,x2, ... xn) Ergibt den kleinsten Wert der Argumente. Die
Anzahl der Argumente ist nicht festgelegt.

MAX

MAX (x1,x2, ... xn) ergibt das größter der Argumente aus einer unedlichen Anzahl.

RND

RND (x) Ergibt immer einen realen Zufallswert zwischen
0.0 und x (x > 0.0).

Bit-Funktionen

BITTEST

BITTEST (x, b)

Gibt 1 zurück, wenn das b-Bit von x gesetzt ist, andernfalls 0.

BITSET

BITSET (x, b [, expr])

expr kann 0 oder ein anderer Wert sein; der Standardwert ist 1. Setzt das b-Bit von x auf 1 oder 0, je nach dem Wert des angegebenen Ausdrucks, und gibt das Ergebnis zurück. Parameterwerte sollten *ganzzahlig* sein, Ergebniswert ist *ganzzahlig*.

Spezielle Funktionen

Die speziellen Funktionen (neben globalen Variablen) können im Script verwendet werden, um mit dem Programm zu kommunizieren. Diese ermitteln entweder den laufenden Status und die verschiedenen Grundeinstellungen oder beziehen sich auf die aktuelle Umgebung des Bibliothekselementes. Frageaufrufe können auch zur Kommunikation mit GDL-Erweiterungen verwendet werden.

Es gibt zwei Typen der speziellen Funktionen: Aufrufe und IND-Funktion:

REQ (parameter_string)

REQUEST (question_name, name | index, variable1 [, variable2,...])

IND (TEXTURE, name_string)

IND (TEXTURE, name_string)

IND (TEXTURE, name_string)

IND (TEXTURE, name_string)

IND (TEXTURE, name_string)

Der Ergebniswert der Aufrufe ist immer die Anzahl der erfolgreich wiederhergestellten Werte (ganzzahlig), wohingegen der Typ der wiederhergestellten Werte bei jedem Aufruf festgelegt wird.

IND-Funktionen ergeben einen Attribut-Indexwert (ganzzahlig).

Weitere Einzelheiten siehe IND-Funktionsbeschreibung in *“Verschiedenes”* > *“Spezielle Funktionen”* auf Seite 243.

String-Funktionen

STR

STR (numeric_expression, length, fractions)

STR

STR (format_string, numeric_expression)

STR{2}

STR{2} (format_string, numeric_expression [, exta_accuracy_string])

Die erste Form der Funktion erstellt einen Text vom aktuellen Wert des numerischen Ausdrucks. Die Mindestanzahl für die die Ziffern des Textstrings ist `length`, während `fraction` die Ziffern nach der Fließkommazahl bezeichnet. Umfasst der konvertierte Wert mehr als nur "Länge"-Zeichen, wird er wie gewünscht erweitert. Hat er weniger Charakter,, wird er linksbündig ($\text{länge} > 0$) oder rechtsbündig ($\text{länge} < 0$) ausgerichtet.

Beispiel:

```
A=4.5
B=2.345
TEXT2 0, 2, STR(A, 8, 2) ! 4.50
TEXT2 0, 1, STR(B, 8, 2) ! 2.34
TEXT2 0, 0, STR(A*B, 8, 2) ! 10.55
```

Im zweiten und dritten Fall kann der Format-Text entweder eine Variable oder eine Konstante sein. Ist das Format leer, so wird es als Meter, mit einer Genauigkeit von drei Dezimalen interpretiert (es stellt 0 Öffnungen dar). Wenn die Rundungsintervall-Flags im format_2string gesetzt sind, gibt die STR(2) Funktion die entsprechende Zeichenfolge "Rundungsintervalle" im 3. Parameter zurück.

Der Format-String kann wie folgt aussehen:

```
%[0 or more flags][field_width][.precision] conv_spec
flags (for m, mm, cm, e, df, di, sqm, sqcm, sqf, sqi, dd, gr, rad, cum, l, cucm, cumm, cuf, cui, cuy, gal):
```

```
(none)    right justify (default)
-         left justify
+         explicit plus sign
(space)   in place of a + sign
'*' 0     extra accuracy Off (default)
'*' 1     extra accuracy .5
'*' 2     extra accuracy .25
'*' 3     extra accuracy .1
'*' 4     extra accuracy .01
'*' 5     extra accuracy .5
'*' 6     extra accuracy .25
```

```
flags (for m, mm, cm, df, di, sqm, sqcm, sqf, sqi, dd, fr, rad, cum, l, cucm, cumm, cuf, cui, cuy, gal):
```

```
'#'       don't display 0 wholes
```

```
flags (for ffi, fdi, fi):
```

```
'0'       display 0 inches
```

```
flags (for m, mm, cm, fdi, df, di, sqm, sqcm, sqf, sqi, dd, fr, rad, cum, l, cucm, cumm, cuf, cui, cuy, gal):
```

'~' blendet 0 Dezimalzahlen aus (nur wirksam, wenn die '#' Fahne nicht spezifiziert wurde)

'^' ändern Sie weder dezimale Trennzeichen noch Ziffern gruppierende Zeichen (falls nicht angegeben werden diese Zeichen ersetzt, wie im aktuellen System eingestellt)

field_width: unbezeichnete dezimale Zahl, die minimale Anzahl der zu erstellenden Zeichen

precision: unbezeichnete dezimale Zahl, die Anzahl der zu erstellenden Ziffern

conv_spec (conversion specifier): e: exponentiales Format (Meter)

m: Meters

mm: Millimeter

cm: Zentimeter

ffi: Fuß & Bruchzoll

fdi: Fuß & dezimaler Zoll

df: dezimaler Fuß

fi: Bruchzoll

di: dezimaler Zoll

für Flächen:

sqm: Quadratmeter

sqcm: Quadratzentimeter

sqmm: Quadratmillimeter

sqf: Quadratfuß

sqi: Quadratzoll

für Winkel:

dd: dezimaler Grad

dms: dezimaler Grad

gr: Grad

rad: Radiant

surv: Vermessungseinheit

für Raummaße:

cum: Kubikmeter

l: Liter

cucm: Kubikzentimeter

cumm: Kubikmillimeter

cuf: Kubikfuß

cui: Kubikzoll

cuy: cubic

gal: Gallone

Beispiele:

```
h = 23
nr = 0.345678
TEXT2 0, h, STR ("%m", nr) !0.346
TEXT2 0, h-1, STR ("%#10.2m", nr) !35
TEXT2 0, h-2, STR ("% .4cm", nr) !34.5678
TEXT2 0, h-3, STR ("%12.4cm", nr) !34.5678
TEXT2 0, h-4, STR ("% .6mm", nr) !345.678000
TEXT2 0, h-5, STR ("% +15e", nr) !+3.456780e-01
TEXT2 0, h-6, STR ("% ffi", nr) !1'-2"
TEXT2 0, h-7, STR ("%0.16 ffi", nr) !1'-1 5/8"
TEXT2 0, h-8, STR ("% .3fdi", nr) ! 1'-1.609"
TEXT2 0, h-9, STR ("% -10.4df", nr) ! 1.1341'
TEXT2 0, h-10, STR ("%0.64fi", nr) !13 39/64"
TEXT2 0, h-11, STR ("% +12.4di", nr) !+13.6094"
TEXT2 0, h-12, STR ("%#.3sqm", nr) ! 346
TEXT2 0, h-13, STR ("% +sqcm", nr) !+3,456.78
TEXT2 0, h-14, STR ("% .2sqmm", nr) ! 345,678.00
TEXT2 0, h-15, STR ("% -12sqf", nr) !3.72
TEXT2 0, h-16, STR ("%10sqi", nr) ! 535.80
```

```
alpha = 88.657
TEXT2 0, h-17, STR ("% +10.3dd", alpha) !+88.657°
TEXT2 0, h-18, STR ("% .1dms", alpha) !88°39'
TEXT2 0, h-19, STR ("% .2dms", alpha) !88°39'25"
TEXT2 0, h-20, STR ("%10.4gr", alpha) ! 98.5078G
TEXT2 0, h-21, STR ("%rad", alpha) !1.55R
TEXT2 0, h-22, STR ("% .2surv", alpha) !N 1°20'35" E
```

SPLIT

SPLIT (string, format, variable1 [, variable2, ..., variablen])

Spaltet den Zeichenfolge-Parameter in einen oder mehrere numerische oder textliche Teile dem Format entsprechend. Der Spaltungsprozeß stoppt, wenn der erste ungeeignete Teil vorkommt. Ergibt die Anzahl der erfolgreich eingelesenen Werte (*ganzzahlig*).

string: der zu spaltende Textstring

format: beliebige Kombination von konstanten Zeichenfolgen, %s und %n -s. Teile der Zeichenfolge müssen den konstanten Zeichenfolgen angepaßt sein, %s kennzeichnet jeden Zeichenfolge-Wert, der durch Leertasten oder Tabulatoren umbegrenzt wird, während %n einen numerischen Wert kennzeichnet.

variable_i: die Variablenamen, die die gespalteten Zeichenfolgeteile speichern.

Beispiel:

```
ss = "3 pieces 2x5 beam"
n = SPLIT (ss, "%n pieces %nx%n %s", num, ss1, size1, ss2, size2, name)
IF n = 6 THEN
    PRINT num, ss1, size1, ss2, size2, name      ! 3 pieces 2 x 5 beam
ELSE
    PRINT "ERROR"
ENDIF
```

STW

STW (string_expression)

Übergibt die (*reale*) Breite des Textes in Meter und stellt es in dem aktuellen Stil dar. Die Breite in Meter ist STW (string_expression) / 1000 * A_.

Beispiel:



```
DEFINE STYLE "own" "Monaco", 180000 / A_, 0, 0
SET STYLE "own"
string = "abcd"
width = STW (string) / 1000 * A_
n = REQUEST ("Height_of_style", "own", height)
height = height / 1000 * A_
text2 0,0, string
rect2 0,0, width, -height
```

STRLEN

STRLEN (string_expression)

Übergibt die (*ganzzahlige*) Länge des Strings (Zeichenanzahl)

STRSTR

STRSTR (string_expression1, string_expression2)

Übergibt die (*ganzzahlige*) Position der ersten Erscheinung der zweiten Zeichenfolge in der ersten Zeichenfolge. Wenn die erste Zeichenfolge die zweite nicht beinhaltet, kehrt die Funktion auf 0 zurück.

STRSUB

STRSUB (string_expression, start_position, characters_number)

Übergibt eine Sub-Zeichenfolge des Zeichenfolge-Parameters, der bei der durch den start_position Parameter angegebenen Position beginnt und seine Länge durch die charakter_nummer Charakter bestimmt wird.

Beispiel:

```
ss = ""
  n = REQUEST ("Linear_dimension", "",ss)
  unit = ""
IF STRSTR (ss, "m") > 0 THEN unit = "m"
IF STRSTR (ss, "mm") > 0 THEN unit = "mm"
IF STRSTR (ss, "cm") > 0 THEN unit = "cm"
TEXT2 0, 0, STR (ss, a) + " " + unit !1.00 m
string = "Flowers.PICT"
len = STRLEN (string)
n = STRSTR (string, ".")
TEXT2 0, -1, STRSUB (string, 1, n - 1) !Flowers
TEXT2 0, -2, STRSUB (string, len - 4, 5) !.PICT
```

BEFEHLE ZUR PROGRAMMSTEUERUNG

In diesem Kapitel werden die GDL-Befehle vorgestellt, die für die Steuerung von Schleifen und Subroutinen in Scripts verfügbar sind; Sie kennen auch das Konzept des Arbeitens mit dem internen Parameterspeicher kennen, das für die weitere Anwendung zum Speichern von Parameterwerten entworfen wurde. Hier wird auch beschrieben, wie man Objekte als Macros verwenden sollte und wie berechnete Ausdrücke auf dem Bildschirm angezeigt werden können.

BEFEHLE ZUR PROGRAMMSTEUERUNG

FOR

FOR variable_name = initial_value **TO** end_value [**STEP** step_value]

Erste Anweisung einer FOR-Schleife. Werden das Schlüsselwort STEP und der Schrittwert step_value nicht angegeben, wird automatisch der Wert 1 angenommen.

Die Verwendung einer globalen Variable als Schleifen-Wert ist nicht zulässig.

Beispiel:

```
FOR I=1 TO 10 STEP 2
    PRINT I
NEXT I
```

NEXT

NEXT variable_name

Letzte Anweisung einer FOR-Schleife.

Nach dem Erreichen der NEXT-Anweisung ändert die Variable ihren Wert bei jedem Durchgang, und zwar beginnend vom Anfangswert hin zum Endwert. Dabei werden jedesmal die zwischen den FOR- und NEXT-Anweisungen stehenden Befehle ausgeführt. Über die Schrittweite wird der Wert der Vergrößerung bzw. Verkleinerung der Variablen angegeben. Überschreitet der Schleifen-Wert den Endwert, wird der unmittelbar nachfolgende Befehl nach NEXT ausgeführt.

Anmerkung: Das Ändern des Schrittwertes step_value während der Erzeugung der Schleife übt keine Wirkung aus.

Die beiden folgenden Programmteile sind gleichbedeutend:

```
! 1st
  A = B
1: IF C > 0 AND A > D OR C < 0 AND A < D THEN 2
  PRINT A
  A = A + C
  GOTO 1
2:
! 2nd
  FOR A = B TO D STEP C
    PRINT A
  NEXT A
```

Das Beispiel oben zeigt, daß `step_value = 0` eine unendliche Schleife erzeugt.

Es ist nur ein `NEXT`-Befehl nach einem `FOR`-Befehl erlaubt. Es darf zwar eine Schleife mit einem `GOTO`-(oder `IF ... GOTO`-) Befehl verlassen und dann wieder in die Schleife zurückgekehrt werden, jedoch ist es nicht zulässig, unter Auslassen des Befehles `FOR` in eine Schleife einzusteigen.

DO

```
DO
  [statement1
  statement2
  ...
  statementn]
```

WHILE condition

Die Anweisungen zwischen den Schlüsselwörtern werden nur ausgeführt, solange die Bedingung wahr ist.

Die Bedingung wird nach jeder Ausführung der Anweisungen kontrolliert.

```
WHILE condition DO
  [statement1
  statement2
  ...
  statementn]
```

ENDWHILE

Die Anweisungen zwischen den Schlüsselwörtern werden nur ausgeführt, solange die Bedingung wahr ist.

Die Bedingung wird vor jeder Ausführung der Anweisungen kontrolliert.

```
CUTSHAPE d
    [statement1
    statement2
    ...
    statementn]
```

UNTIL condition

Die Anweisungen zwischen den Schlüsselwörtern werden ausgeführt, bis die Bedingung wahr ist.

Die Bedingung wird nach jeder Ausführung der Anweisungen kontrolliert.

Beispiel:

Die folgenden vier Reihenfolgen der GDL-Befehle sind gleichbedeutend:

```
! 1st
FOR i = 1 TO 5 STEP 1
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
NEXT i
! 2nd
i = 1
DO
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
    i = i + 1
WHILE i <= 5
! 3rd
i = 1
WHILE i <= 5 DO
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
    i = i + 1
ENDWHILE
! 4th
i = 1
REPEAT
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
    i = i + 1
UNTIL i > 5
```

IF

IF condition **THEN** label

IF condition **GOTO** label

IF condition **GOSUB** label

Sprungbefehl mit Bedingung. Wenn der Wert der Bedingung 0 ist, hat der Befehl keine Wirkung, andernfalls wird die Programmbearbeitung bei der angegebenen Adresse fortgesetzt.

Beispiele:

```
IF A THEN 28
IF I > J GOTO 200+I*J
IF I > 0 GOSUB 9000
IF condition THEN statement [ELSE statement]
oder
IF condition THEN [statement1
statement2
...
statementn]
[ELSE
statementn+1
statementn2
...
statementn+m]
ENDIF
```

Wenn Sie nur einen Befehl nach dem Schlüsselwort THEN und/oder ELSE in dieselbe Zeile einschreiben, dann wird ENDIF unnötig. Ein Befehl nach THEN oder ELSE in derselben Zeile bedeutet ein definitives ENDIF.

Falls sich eine neue Zeile nach dem Schlüsselwort THEN befindet, werden die eingegeben Befehle (alle bis zum Schlüsselwort ELSE oder ENDIF) nur dann ausgeführt, wenn der Ausdruck in der Bedingung wahr ist (anders als Null). Anderenfalls werden die Befehle, gefolgt von ELSE ausgeführt. Anderenfalls werden die Befehle, gefolgt von ELSE ausgeführt. Sollte das Schlüsselwort ELSE fehlen, so werden die Befehle nach ENDIF ausgeführt.

Beispiel:

```
IF a = b THEN height = 5 ELSE height = 7
IF needdoors THEN
    CALL    "door_macro" PARAMETERS
    ADDX    a
ENDIF
IF simple THEN
    HOTSPOT2 0, 0
    RECT2 a, 0, 0, b
ELSE PROJECT2 3, 270, 1
IF name = "Sphere" THEN
    ADDY b
    SPHERE 1
ELSE
    ROTX 90
    TEXT 0.002, 0, name
ENDIF
```

GOTO

GOTO label

Sprungbefehl ohne Bedingung. Das Programm springt sofort zu dem durch den Wert der Sprungmarke bestimmten Befehl.

Beispiel:

```
GOTO K+2
```

GOSUB

GOSUB label

Aufruf eines internen Unterprogrammes. Die Sprungmarke markiert den Einstieg in das Programm.

RETURN

RETURN

Das Programm kehrt aus dem internen Unterprogramm zurück.

END / EXIT

END

EXIT

Ende des aktuellen GDL-Scripts. Das Programm wird beendet oder kehrt zu der darüberliegenden Hierarchieebene zurück. Es können mehrere END- oder EXIT-Befehle in einem GDL-Dokument verwendet werden.

BREAKPOINT

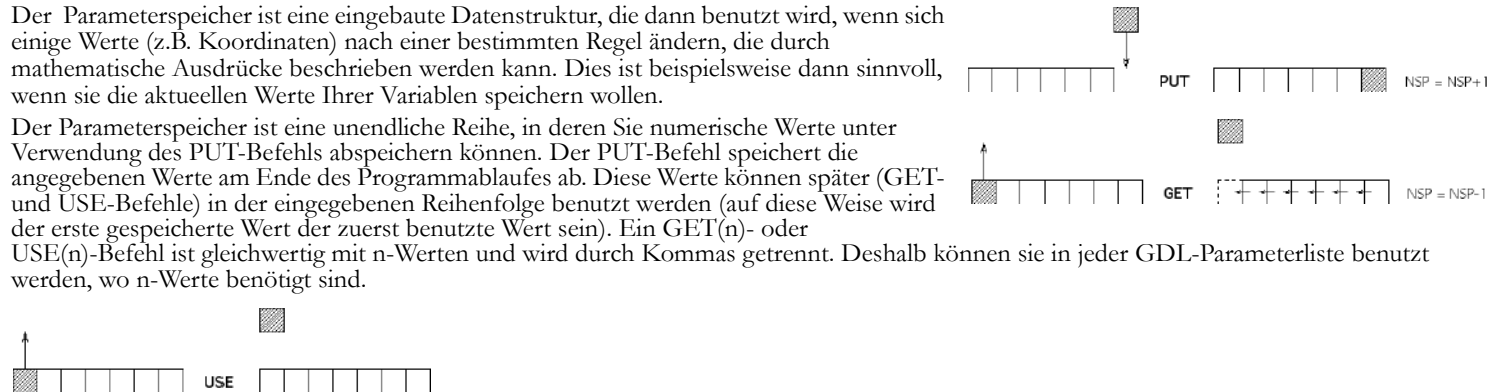
BREAKPOINT expression

Durch diesen Befehl können Sie einen Bruchpunkt im GDL-Script bestimmen. Der GDL-Debugger stoppt bei diesem Befehl, wenn der Parameterwert (ein numerischer Ausdruck) wahr ist (1) und die Option des Debuggers Bruchpunkte ermöglichen ausgewählt wird. In "normalem" Ausführungsmodus überschreitet der GDL-Interpreter einfach diesen Befehl.

ARBEITEN MIT DEM INTERNEN PARAMETERSPEICHER

Der Parameterspeicher ist eine eingebaute Datenstruktur, die dann benutzt wird, wenn sich einige Werte (z.B. Koordinaten) nach einer bestimmten Regel ändern, die durch mathematische Ausdrücke beschrieben werden kann. Dies ist beispielsweise dann sinnvoll, wenn sie die aktuellen Werte Ihrer Variablen speichern wollen.

Der Parameterspeicher ist eine unendliche Reihe, in deren Sie numerische Werte unter Verwendung des PUT-Befehls abspeichern können. Der PUT-Befehl speichert die angegebenen Werte am Ende des Programmablaufes ab. Diese Werte können später (GET- und USE-Befehle) in der angegebenen Reihenfolge benutzt werden (auf diese Weise wird der erste gespeicherte Wert der zuerst benutzte Wert sein). Ein GET(n)- oder USE(n)-Befehl ist gleichwertig mit n-Werten und wird durch Kommas getrennt. Deshalb können sie in jeder GDL-Parameterliste benutzt werden, wo n-Werte benötigt sind.



PUT

PUT expression [, expression, ...]

Speichert die angegebenen Werte in der angegebenen Reihenfolge in den internen Parameterspeicher.

GET

GET (n)

Holt sich die nächsten n-Werte des internen Parameterspeichers und löscht sie.

USE

USE (n)

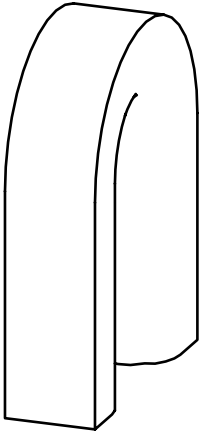
Holt sich die nächsten n-Werte des internen Parameterspeichers ohne sie zu löschen. Nachfolgende USE- und GET-Befehle können dieselbe Parametersequenz verwenden.

NSP

NSP

Holt sich alle gespeicherten Parameter in den internen Parameterspeicher zurück.

Beispiel für die Verwendung des internen Parameterspeichers:



```
R=2: B=6: C=4: D=10
N=12
```

```
S=180/N
FOR T=0 TO 180 STEP S
  PUT R+R*COS(T), C-R*SIN(T), 1
NEXT T
```

```
FOR I=1 TO 2
  EXTRUDE 3+NSP/3, 0,0,D, 1+16,
    0, B, 0,
    2*R, B, 0,
    USE(NSP),
    0, B, 0
  MULY -1
NEXT I
DEL 1
ADDZ D
REVOLVE 3+NSP/3, 180, 0,
  0, B, 0,
  2*R, B, 0,
  GET(NSP),
  0, B, 0
```


Die volle Beschreibung:

R=2: B=6: C=4: D=10

```

FOR I=1 TO 2
  EXTRUDE 16, 0,0,D, 1+16,
    0, B, 0,
    2*R, B, 0,
    2*R, C, 1,
    R+R*COS(15), C-R*SIN(15), 1,
    R+R*COS(30), C-R*SIN(30), 1,
    R+R*COS(45), C-R*SIN(45), 1,
    R+R*COS(60), C-R*SIN(50), 1,
    R+R*COS(75), C-R*SIN(75), 1,
    R+R*COS(90), C-R*SIN(90), 1,
    R+R*COS(105), C-R*SIN(105), 1,
    R+R*COS(120), C-R*SIN(120), 1,
    R+R*COS(135), C-R*SIN(135), 1,
    R+R*COS(150), C-R*SIN(150), 1,
    R+R*COS(165), C-R*SIN(165), 1,
    0, B, 1,
    0, B, 0
  MULY -1
NEXT I
DEL 1
ADDZ D
REVOLVE 16, 180, 0,
  0, B, 0,
  2*R, B, 0,
  2*R, C, 1,
  R+R*COS(15), C-R*SIN(15), 1,
  R+R*COS(30), C-R*SIN(30), 1,
  R+R*COS(45), C-R*SIN(45), 1,
  R+R*COS(60), C-R*SIN(50), 1,
  R+R*COS(75), C-R*SIN(75), 1,
  R+R*COS(90), C-R*SIN(90), 1,
  R+R*COS(105), C-R*SIN(105), 1,
  R+R*COS(120), C-R*SIN(120), 1,
  R+R*COS(135), C-R*SIN(135), 1,
  R+R*COS(150), C-R*SIN(150), 1,
  R+R*COS(165), C-R*SIN(165), 1,
  0, B, 1,
  0, B, 0

```

MACRO-OBJEKTE

Obwohl alle erforderlichen 3D-Objekte in mehr oder weniger komplexe Einzelelemente zerlegt werden können, ist es manchmal wünschenswert, diese komplexen Elemente für bestimmte Anwendungen spezifisch zu definieren. Diese individuell definierten Elemente heißen Macro.

CALL

CALL macro_name_string [,parameter_list]

CALL macro_name_string **PARAMETERS** [name1=value1 , ... namen=valuen]

CALL macro_name **PARAMETERS ALL**

Die Namen der Macros dürfen nicht länger als 31 Zeichen sein.

Macronamen können Zeichen-Konstanten, Zeichen-Variablen oder -Parameter sein. Zeichen-Operationen mit einem Macro-Aufruf können als Macronamen nicht verwendet werden.

Warnung: Wenn Zeichen-Variablen oder Zeichen-Parameter als Macronamen verwendet werden, wird das aufgerufene Macro in das Archivprojekt nicht eingeschlossen werden, auch wenn die Option "Alle Elemente von geöffneten Bibliotheken mitsichern" aktiv ist.

Der Macroname muss in Anführungszeichen („‘’,‘’,‘’,‘’,‘’) gesetzt werden, damit es der Definition von ID-Nr. entspricht, es beginnt z.B. mit einem Buchstaben oder ‘_’ oder dem ‘~’ Zeichen und enthält nur Buchstaben, Zahlen und die Zeichen ‘_’ und ‘~’. Andernfalls müssen die vorderen und hinteren Anführungszeichen der CALL-Anweisung übereinstimmen und sollten sich von allen anderen Zeichen des Macronamens unterscheiden.

Ein Macroname kann auch ohne das Schlüsselwort CALL als Befehl verwendet werden:

macro_name [parameter_list]

macro_name **PARAMETERS** [name1=value1, ... namen=valuen]

macro_name **PARAMETERS ALL**

Der erste Typ von Macros kann mit einfachen GDL-Textdateien unter der Voraussetzung wie jedes Bibliothekselement verwendet werden, daß seine Parameterliste nur aus Einzelbuchstaben-Parameter (A Z) besteht. Diese Form von Macro-Aufrufen kann mit den früheren Versionen benutzt werden, aber Graphisoft empfiehlt den zweiten Typ. Wozu dient die Parameterliste: der Wert des Parameters A wird der erste Wert in der Liste, der Wert des Parameters B wird der zweite Wert sein usw. Enthält das Macro (Bibliothekselement) keinen Einzelbuchstaben-Parameter, der dem Wert entspricht, dann setzt die Interpretation mit dem Suchen dieses Wertes fort, aber gibt eine Warnmeldung aus. Diese Methode lässt keine Zeichen-Typ Ausdrücke in der Wertliste zu.

Der zweite Typ von Macros kann nur mit vollkommen charakterisierten Bibliothekselementen und mit textlichen GDL-Dateien- benutzt werden. Nach dem Schlüsselwort PARAMETERS sollten Sie die Parameternamen des aufgerufenen Macros in beliebiger Reihenfolge auflisten, einen jeden in Anführungszeichen ‘=’ und mit einem Wert versehen. An dieser Stelle können Sie Zeichen-Typ Ausdrücke verwenden, aber seien Sie damit vorsichtig und ordnen Sie Zeichenwerte nur Zeichen-Typ Parametern des aufgerufenen Wertes zu. Ordnungsparametern müssen volle Ordnungswerte zugeordnet werden. Sollte ein Parametername in der Parameterliste des aufgerufenen Macros nicht gefunden werden, dann erscheint eine Warnung. Die nicht aufgelisteten Parameter des aufgerufenen Macros werden ihre ursprünglichen voreingestellten Werte, wie im Bibliothekselement als Macro definiert, bekommen.

Der dritte Typ kann nur mit Bibliothekselementen mit vollem Funktionsumfang verwendet werden. In diesem Fall brauchen die Parameter nicht einzeln angegeben zu werden; alle Parameter des Aufrufers werden an den aufgerufenen Macro übergeben. Für Parameter, die in dem aufgerufenen Macro nicht gefunden werden, wird der Standardwert verwendet.

Ein GDL-Macro hat seine Umgebung, die von der Reihenfolge des Aufrufs abhängig ist. Die aktuellen Werte von

MODEL, RADIUS, RESOL, TOLER, PEN, LINE_TYPE, MATERIAL, FILL, STYLE, SHADOW

Optionen und die aktuelle Transformation sind auch im Macro gültig. können die Werte benutzen oder modifizieren, dies übt jedoch nur lokale Wirkung aus. Es hat keine Auswirkungen auf diejenige Programmebene, aus welcher das Macro aufgerufen wurde.

Wenn Sie Parameter einem Macro-Aufruf zuordnen, bedeutet das eine neue Wertzuweisung für das Macro.

Parameter A und B werden immer zur Größenumstellung der Objekte verwendet.

Beispiele:

```
CALL "leg" 2, , 5 ! A = 2, B = 0, C = 5
leg 2, , 5
CALL "door-1" PARAMETERS height = 2, a = 25.5,
    name = "Director"
CALL "door-1" PARAMETERS
    ! Parametergrundwerte verwenden
"Tür-1" PARAMETERS
```

Kurz zusammengefaßt: Wenn Sie Parameter ohne langen Namen oder Zeichen-Typ benötigen, wird die Verwendung des Textes vom GDL-Typ empfohlen. Dieser GDL-Typ kann nur mit dem ersten Typ der Macro-Aufrufe aufgerufen werden, da es keine modifizierbare Parameterliste hat. Andererseits, wenn Sie Ihre Macro-Parameternamen nicht auf die Buchstaben von A bis Z beschränken, oder Zeichenfolgen in die Parameterliste einfügen möchten, muß Ihr Macro ein Bibliothekselement sein und dem zweiten Typ der GDL-Syntax entsprechend aufgerufen werden.

DER DIALOG-BEFEHL

Drucken

PRINT expression [, expression, ...]

Bringt alle Argumente in einem Dialogfenster auf den Bildschirm. Argumente können Zeichenfolgen oder numerische Ausdrücke beliebiger Anzahl in beliebiger Reihenfolge darstellen, sie werden durch Kommas getrennt.

Beispiele:

```
PRINT "loop-variable:", I
PRINT J, K-3*L
PRINT "Beginning of interpretation"
PRINT a * SIN (alpha) + b * COS (alpha)
PRINT "Parameter values: ", "a = ", a, ", b = ", b
PRINT name + STR ("%m", i) + "." + ext
```

DATEI OPERATIONEN

Über die nachstehenden Schlüsselwörter können Sie äußere Dateien zwecks Lesens und Schreibens öffnen sowie Werte aus /in GDL-Skripts einfügen und einholen. Dieser Prozeß bezieht notwendigerweise spezielle Add-Ons mit ein. TEXT-Dateien können durch das TEXT GDL I/O Add-On behandelt werden. Add-Ons für andere Dateitypen können von Drittanbietern entwickelt werden.

Siehe auch "GDL Text I/O Add-On" und "Verschiedenes".

OPEN

OPEN (filter, filename, parameter_string)

filter: Zeichenfolge, name einer vorhandenen Erweiterung

filename: Zeichenfolge, Dateiname

parameter_string: Zeichenfolge, enthält die bestimmten Trennungszeichen der aktuellen Erweiterung und Art des Öffnens. Der Inhalt wird in der Erweiterung interpretiert.

Öffnet eine Datei wie angewiesen. Sein Rückwert ist eine positive ganze Zahl, die die spezifische Datei identifiziert. Ergebnis-Wert ist eine positive ganze Zahl, die die bestimmte Datei wiedererkennt. Dieser Wert ist in den eingegebenen Fällen die Referenznummer der Datei *"FILE_DEPENDENCE "name1" [, "name2", ...]"*.

INPUT

INPUT (channel, recordID, fieldID, variable1 [, variable2, ...])

recordID, fieldID: Die Anfangsposition des Lesens vom textlichen oder numerischen Typ, dessen Inhalt über die Erweiterung interpretiert wird

Die Anzahl der angegebenen Parameter definiert die Anzahl der Werte von der Anfangs-Position, sie wird von der Datei eingelesen und vom Kanal-Wert wiedererkannt. Die Parameterliste muß wenigstens einen Wert enthalten. Diese Funktion fügt die roten Werte in die Parameter wie angewiesen ein. Diese Werte können vom numerischen- oder vom Zeichentyp sein, unabhängig von dem gespeicherten Parametertyp.

Der Ergebnis-Wert ist die Anzahl der eingelesenen Werte. Wenn man ans Dateizeichenende gelangt, dann beträgt es -1. .

VARTYPE

VARTYPE (expression)

Ergibt 1, wenn der Variablentyp numerisch ist, 2 bei einer Zeichenfolge.

Nützlich beim Lesen von Werten in Variablen mit dem Befehl INPUT, der den Variablentyp in Übereinstimmung mit den aktuellen Werten ändern kann.

Der Typ dieser Variablen wird während der Kompilierung nicht geprüft.

OUTPUT

OUTPUT channel, recordID, fieldID, expression1 [, expression2, ...]

recordID, fieldID: Die Anfangsposition des Lesens vom textlichen oder numerischen Typ, dessen Inhalt durch die Erweiterung interpretiert wird.

Schreibt ebenso viele Werte in die Datei ein, -die durch den Kanal-Wert der angegebenen Position wiedererkannt werden-, wie es definierte Bedingungen gibt. Es muß mindestens eine Bedingung vorhanden sein. Der Typ der Werte stimmt mit denen der Bedingungen überein.

CLOSE

CLOSE Kanal

Schließen Sie den Ordner identifiziert durch den Kanalwert.

VERSCHIEDENES

GDL kann einige Operationen in externen Dateien durch spezielle Add-Ons ausführen. Die hierfür benutzten Befehle werden in diesem Kapitel erläutert und mit einem Beispiel illustriert.

GLOBALE VARIABLEN

Globale Variablen ermöglichen es, spezielle Modellparameter zu speichern. Dadurch werden geometrische Informationen über die Zeichnungsumgebung des GDL-Makros zugänglich. So kann z.B. die Wandstärke abgelesen werden, wenn man ein Fenster definieren will, welches genau in diese Wand passen soll. Globale Parameter werden während des Makro-Aufrufes nicht im Stack abgelegt.

Allgemeine Umgebungsinformationen

GLOB_SCRIPT_TYPE	T~	Typ des aktuellen Scriptes
1-Descriptor-Script, 2-2D-Script, 3-3D-Script, 4-nicht implementiert, 5-Wertliste-Script, 6-Master-Script		
GLOB_CONTEXT		Erscheinungskontext
1-Bibliothekselement-Editor, 2-Grundriss, 3-3D-Ansicht, 4-Schnitt/ Ansicht, 5-Einstellungen-Dialogfenster, 6-Liste, 7 - Detailzeichnung, 22 - Rückmeldungs-Modus vom Grundriss, 23 - Rückmeldungs-Modus von einer 3D-Ansicht, 24 - Rückmeldungs-Modus von einem Schnitt/ einer Ansicht, 43 - Generieren als Bediener von einer 3D-Ansicht, 44 - Generieren als Bediener von einem Schnitt/ einer Ansicht, 46 - Generieren als Bediener von einer Liste		
GLOB_SCALE	A_	Zeichnungsmaßstab
dem aktuellen Fenster entsprechend		
GLOB_NORTH_DIR	U~	Nordausrichtung des Projekts
mit Bezug auf das Standardprojekt, entspricht das Koordinatensystem den Einstellungen im Dialogfenster Sonne...		
GLOB_DRAWING_BGD_PEN		Stift der Farbe des Zeichnungshintergrundes
der am besten zur Hintergrundfarbe des aktuellen passende Stift aus der aktuellen Palette		
GLOB_MODPAR_NAME		Name des letzten geänderten Parameters
Einstellungsdialog oder im Editor für Bibliothekselemente , einschließlich Werten, die durch editierbare Hotspots modifiziert werden. Nur in Parameter-Scripts gültig.		
GLOB_WORLD_ORIGO_OFFSET_X, GLOB_WORLD_ORIGO_OFFSET_Y		
Position des Projektursprungs bezogen auf den globalen Ursprung. Siehe <i>“Beispiel zur Illustration der globalen Variablen GLOB_WORLD_ORIGO_...:” auf Seite 241.</i>		

Geschossinformationen

GLOB_HSTORY_ELEV	B_	Höhenlage des Ursprungsgeschosses
Ursprungsgeschoss - Geschoss, in dem das Objekt platziert wird		
GLOB_HSTORY_HEIGHT	Q_	Höhe des Ursprungsgeschosses
Ursprungsgeschoss - Geschoss, in dem das Objekt platziert wird		
GLOB_CSTORY_ELEV	Q~	Höhenlage des Ursprungsgeschosses
Aktuelles Geschoss - Geschoss, das derzeit im Grundrissfenster angezeigt wird		
GLOB_CSTORY_HEIGHT	R~	Höhe des aktuellen Geschosses
Aktuelles Geschoss - Geschoss, das derzeit im Grundrissfenster angezeigt wird		
GLOB_CH_STORY_DIST	S~	relative Position des aktuellen Geschosses zum Referenzgeschoss
Aktuelles Geschoss - Geschoss, das derzeit im Grundrissfenster angezeigt wird		

Animationsinformationen

GLOB_FRAME_NR	N_	Anzahl der aktuellen Standorte in der Animation <i>nur für Animation gültig, 0 ist für Standbilder</i>
GLOB_FIRST_FRAME	O_	Index des ersten Standortes in der Animation <i>nur für Animation gültig, 0 ist für Standbilder</i>
GLOB_LAST_FRAME	P_	Index des letzten Standortes in der Animation <i>nur für Animation gültig, 0 ist für Standbilder</i>
GLOB_EYEPOS_X	K~	aktuelle Kameraposition (x) <i>gilt nur in der Perspektivprojektion für Animation und Standbilder</i>
GLOB_EYEPOS_Y	L~	aktuelle Kameraposition (y) <i>gilt nur in der Perspektivprojektion für Animation und Standbilder</i>
GLOB_EYEPOS_Z	M~	aktuelle Kameraposition (z) <i>gilt nur in der Perspektivprojektion für Animation und Standbilder</i>
GLOB_TARGPOS_X	N~	aktuelle Gegenstandsposition (x) <i>gilt nur in der Perspektivprojektion für Animation und Standbilder</i>
GLOB_TARGPOS_Y	O~	aktuelle Zielposition (x) <i>gilt nur in der Perspektivprojektion für Animation und Standbilder</i>
GLOB_TARGPOS_Z	P~	aktuelle Zielposition (z) <i>gilt nur in der Perspektivprojektion für Animation und Standbilder</i>
GLOB_SUN_AZIMUTH		Sonnenazimuth <i>entsprechend den Einstellungen im Dialogfenster Sonne...</i>
GLOB_SUN_ALTITUDE		Sonnenstand <i>entsprechend den Einstellungen im Dialogfenster Sonne...</i>

Allgemeine Parameter von Elementen

GLOB_LAYER		Ebene des Elementes
<i>Name der Ebene, der das Element zugewiesen wird</i>		
GLOB_ID		Benutzer-ID des Elementes
<i>ID, wie im Dialogfenster für Einstellungen eingegeben</i>		
GLOB_INTID		interne ID des Elementes
<i>die interne individuelle ID-Nummer, die durch das Programm generiert wird (kann durch den Benutzer nicht kontrolliert werden)</i>		
GLOB_ELEVATION	J_	Basishöhenlage des Elementes
<i>im Verhältnis zum Ursprungsgeschoss (Tür, Fenster, Schwellenhöhe ausgenommen, den aktuellen Einstellungen entsprechend)</i>		
GLOB_ELEM_TYPE		Elementtyp für Etiketten und Eigenschaftsobjekte, enthält den Typ des Elternelements
<i>0 - keiner (individuelles Etikett), 1-Objekt, 2-Lichtquelle, 3-Fenster, 4-Tür, 5-Wand, 6-Stütze, 7-Decke, 8-Dach, 9-Schraffur, 10-Freifläche, 11-Raumfläche, 12 - Unterzug</i>		

Parameter von Objekten, Lampen, Türen und Fenstern

SYMB_LINETYPE		Linientyp des Bibliothekselementes
<i>verwendet als der grundeingestellte Linientyp des 2D-Symbols</i>		
SYMB_FILL		Schraffurtyp des Bibliothekselementes
<i>angewandt auf geschnittene Oberflächen der Bibliothekselemente in den Schnitte/Ansichten-Fenstern</i>		
SYMB_FILL_PEN		Stift der Schraffur des Bibliothekselementes
<i>angewandt auf geschnittene Oberflächen der Bibliothekselemente in den Schnitte/Ansichten-Fenstern</i>		
SYMB_FBGD_PEN		Stift des Schraffurhintergrundes des Bibliothekselementes
<i>angewandt auf geschnittene Oberflächen der Bibliothekselemente in den Schnitte/Ansichten-Fenstern</i>		
SYMB_SECT_PEN		Stift des Bibliothekselementes im Schnitt
<i>angewandt auf Konturen geschnittener Oberflächen von Bibliothekselementen in den Schnitte/Ansichten-Fenstern</i>		
SYMB_VIEW_PEN	L_	grundeingestellter Stift des Bibliothekselementes
<i>verwendet für alle Kanten im 3D-Fenster und für alle sichtbaren Kanten in den Schnitte/Ansichten-Fenstern</i>		
SYMB_MAT	M_	Grundmaterial des Bibliothekselementes
SYMB_POS_X	X~	Position des Bibliothekselementes (x)
<i>im Verhältnis zum Projektursprung (Tür, Fenster und Wandende ausgenommen: im Verhältnis zum Startpunkt der beinhaltenden Wand)</i>		
SYMB_POS_Y	Y~	Position des Bibliothekselementes (y)
<i>im Verhältnis zum Projektursprung (Tür, Fenster und Wandende ausgenommen: im Verhältnis zum Startpunkt der beinhaltenden Wand). Hinweis: siehe "Türen und Fenster" auf Seite 253</i>		
SYMB_POS_Z	Z~	Position des Bibliothekselementes (z)

im Verhältnis zum Projektursprung (Tür, Fenster und Wandende ausgenommen: im Verhältnis zum Startpunkt der beinhaltenden Wand). Hinweis: siehe ["Türen und Fenster"](#) auf Seite 253

Parameter von Objekten und Lampen

SYMB_ROTANGLE	W~	Drehwinkel des Bibliothekselementes
<i>numerische Drehung - wird vom Dialogfenster Einstellungen aus um den aktuellen Ankerpunkt ausgeführt</i>		
SYMB_MIRRORED	V~	Bibliothekselement gespiegelt
<i>0-nicht, 1-gespiegelt (Spiegelung wird um den aktuellen Ankerpunkt ausgeführt). Immer 0 für Wandenden, ausgenommen, wenn der Ursprung des lokalen Koordinatensystems ein nichtrechteckiger Scheitelpunkt eines trapezförmigen Wandpolygons ist.</i>		

Parameter von Objekten und Lampen - nur für Listen und Etiketten

SYMB_A_SIZE	nominale Länge/Breite des Bibliothekselementes
<i>Länge von Objekt/Lampe, Breite von Fenster/Tür (festgesetzte Parameter)</i>	
SYMB_B_SIZE	nominale Breite/Höhe des Bibliothekselementes
<i>Breite von Objekt/Lampe, Breite von Fenster/Tür (festgesetzte Parameter)</i>	

Parameter von Objekten und Lampen - nur zum Auflisten und für Etiketten verfügbar

SYMB_Z_SIZE	Nominalhöhe des Bibliothekselementes
<i>wird der erste benutzerdefinierte Parameter im <code>xyz</code>-Format angegeben, dann wird dies für die nominale Höhe benutzt, andernfalls ist es 0</i>	

Parameter von Fenstern, Türen und Wandenden

WIDO_REVEAL_ON		Fenster/Tür-Anschlag ist aktiv
<i>0-Anschlag ist nicht aktiv, 1-Anschlag ist aktiv</i>		
WIDO_SILL	K_	Brüstung von Fenster/Tür
<i>Anschlagtiefe wie im Dialogfenster Anschlagtiefe des Registers Fenster/Tür-Einstellungen für nicht gerade Wände: in radialer Richtung bei Öffnungsecke von nominaler Größe</i>		
WIDO_SILL_HEIGHT		Nominale Brüstungshöhe Fenster/Tür
WIDO_RSIDE_SILL_HEIGHT		Brüstungshöhe Fenster/Tür an der Anschlagsseite
WIDO_OPRSIDE_SILL_HEIGHT		Brüstungshöhe Fenster/Tür an der Seite gegenüber der Anschlagsseite
WIDO_RIGHT_JAMB	B~	Anschlag von Fenster/Tür an der rechten Seite
<i>wie im Dialogfenster Anschlagseinstellungen eingegeben</i>		
WIDO_LEFT_JAMB		Anschlag von Fenster/Tür an der linken Seite
<i>wie im Dialogfenster Anschlagseinstellungen eingegeben</i>		
WIDO_THRES_DEPTH	C~	Brüstungs-/Schwellenbreite von Fenster/Tür
<i>wie im Dialogfenster Anschlagseinstellungen eingegeben</i>		
WIDO_HEAD_DEPTH	D~	Sturz von Fenster/Tür
<i>wie im Dialogfenster Anschlagseinstellungen eingegeben</i>		
WIDO_HEAD_HEIGHT		Nominale Sturzhöhe Fenster/Tür
WIDO_RSIDE_HEAD_HEIGHT		Sturzhöhe Fenster/Tür an der Anschlagsseite
WIDO_OPRSIDE_HEAD_HEIGHT		Sturzhöhe Fenster/Tür an der Seite gegenüber der Anschlagsseite
WIDO_REVEAL_SIDE	E~	Anschlagsseite ist der Öffnungsseite gegenüber
<i>1-ja, 0-nein - wenn man ein Element platziert, ist der Grundeinstellungswert 0 für Fenster und 1 für Türen</i>		
WIDO_FRAME_THICKNESS	F~	Rahmenstärke von Fenster/Tür
<i>Wird die Öffnungsseite von Türen/Fenstern geändert, werden sie durch diesen Wert automatisch gespiegelt, dann zurückplatziert</i>		
WIDO_POSITION	H~	Versatz von Tür/Fenster
<i>Winkel oder Abstand zwischen der Öffnungsachse oder der Wandende und des normalen Vektors am Anfangspunktes der Wand</i>		
WIDO_ORIENTATION		Orientierung der Öffnung von Fenster/Tür
<i>links/rechts - es funktioniert nur, wenn der Tür/Fenster dem lokalen Standard entsprechend erstellt wurde</i>		
WIDO_MARKER_TXT		Markertext von Fenster/Tür
<i>wie im Unterdialogfenster Bemaßungseinstellungen für Fenster/Türen vom Dialogfenster für Türen- und Fenstereinstellungen aus eingestellt</i>		
WIDO_SUBFL_THICKNESS		Dicke des Fußbodenaufbaus (Brüstungskorrektion)
<i>wie im Unterdialogfenster Bemaßungseinstellungen für Fenster/Türen vom Dialogfenster für Türen- und Fenstereinstellungen aus eingestellt</i>		
WIDO_PREFIX		Präfix der Brüstungshöhe von Fenster/Tür
<i>wie im Unterdialogfenster Bemaßungseinstellungen für Fenster/Türen vom Dialogfenster für Türen- und Fenstereinstellungen aus eingestellt</i>		
WIDO_CUSTOM_MARKER		grundeingestellter Markerschalter für Fenster/Tür
<i>11-Parameter können im 2D-Script benutzt werden, während die automatische Bemaßung nicht vorhanden ist</i>		
WIDO_ORIG_DIST	R_	Abstand des lokalen Ursprungs vom Ende der Wand
<i>Abstand des lokalen Ursprungs vom Zentralpunkt der gekrümmten Wand, 0 für gerade Wände. Negativ für Wandende am Endpunkt der gekrümmten Wand.</i>		
WIDO_PWALL_INSET		Brüstungseinzug

Parameter von Fenstern und Türen - nur zum Auflisten und für Etiketten verfügbar

WIDO_RSIDE_WIDTH	Fenster/Tür Breite der Öffnung an der Anschlagsseite
WIDO_OPRSIDE_WIDTH	/Tür Breite der Öffnung an der Seite gegenüber der Anschlagsseite
WIDO_RSIDE_HEIGHT	Fenster/Tür Höhe der Öffnung an der Anschlagsseite
WIDO_OPRSIDE_HEIGHT	Fenster/Tür Höhe der Öffnung an der Seite gegenüber der Anschlagsseite
WIDO_RSIDE_SURF	Fenster/Tür Öffnungsfläche an der Anschlagsseite
WIDO_OPRSIDE_SURF	Fenster/Tür Öffnungsfläche an der Seite gegenüber der Anschlagsseite
WIDO_N_RSIDE_WIDTH	Nominale Breite der Öffnung Fenster/Tür an der Anschlagsseite
WIDO_N_OPRSIDE_WIDTH	Nominale Breite der Öffnung Fenster/Tür an der Seite gegenüber der Anschlagsseite
WIDO_N_RSIDE_HEIGHT	Nominale Höhe der Öffnung Fenster/Tür an der Anschlagsseite
WIDO_N_OPRSIDE_HEIGHT	Höhe der Öffnung Fenster/Tür an der Seite gegenüber der Anschlagsseite
WIDO_N_RSIDE_SURF	Nominale Fenster/Tür Öffnungsfläche an der Anschlagsseite
WIDO_N_OPRSIDE_SURF	Nominale Fenster/Tür Öffnungsfläche an der Anschlagsseite
WIDO_VOLUME	Fenster/Tür Öffnungsvolumen
WIDO_GROSS_SURFACE	Fenster/Tür nominale Öffnungsfläche
WIDO_GROSS_VOLUME	Fenster/Tür nominale Öffnungsfläche

Parameter von Lampen - nur zum Auflisten und für Etiketten verfügbar

LIGHT_ON	Licht ist aktiv <i>0-Licht ist nicht aktiv, 1-Licht ist aktiv; wie im Dialogfenster für Lichtquellen-Einstellungen eingegeben (festgesetzte Parameter)</i>
LIGHT_RED	roter Bestandteil der Lichtfarbe <i>wie im Dialogfenster für Lichtquellen-Einstellungen eingegeben (festgesetzte Parameter)</i>
LIGHT_GREEN	grüner Bestandteil der Lichtfarbe <i>wie im Dialogfenster für Lichtquellen-Einstellungen eingegeben (festgesetzte Parameter)</i>
LIGHT_BLUE	blauer Bestandteil der Lichtfarbe <i>wie im Dialogfenster für Lichtquellen-Einstellungen eingegeben (festgesetzte Parameter)</i>
LIGHT_INTENSITY	Lichtintensität <i>wie im Dialogfenster für Lichtquellen-Einstellungen eingegeben (festgesetzte Parameter)</i>

Bemaßungsparameter

LABEL_POSITION	Position der Bemaßung
<i>Datenfeld[3][2], das die Koordinaten der 3 Punkte enthält, die die Position der Bemaßung definieren</i>	
LABEL_CUSTOM_ARROW	Etikett Pfeil verwenden ja/nein
<i>1 wenn das Kontrollkästchen Etikett Pfeil verwenden aktiviert ist, sonst 0</i>	
LABEL_ARROW_PEN	Stift des Pfeils im Einstellungsdialogfeld
LABEL_ARROWHEAD_PEN	Stift der Pfeilspitze im Einstellungsdialogfeld
LABEL_FONT_NAME	Fontbezeichnung im Einstellungsdialogfeld
LABEL_TEXT_SIZE	Textgröße im Einstellungsdialogfeld
LABEL_TEXT_PEN	Stift des Texts im Einstellungsdialogfeld
LABEL_FONT_STYLE	Schriftschnitt im Einstellungsdialogfeld
<i>0-normal, 1-fett, 2-kursiv</i>	
LABEL_FRAME_ON	Bemaßungsrahmen ja/nein
<i>1 wenn der Bemaßungsrahmen aktiviert ist, sonst 0</i>	
LABEL_ANCHOR_POS	Position des Ankerpunkts
<i>0 - Mitte, 1 - Oben, 2 - Unten, entsprechend dem Dialogfenster Einstellungen</i>	
LABEL_ROTANGLE	Drehwinkel im Einstellungsdialogfeld
LABEL_TEXT_ALIGN	Textausrichtung im Einstellungsdialogfeld
<i>1: linksbündig, 2: zentriert, 3: rechtsbündig, 4: Blocksatz</i>	
LABEL_TEXT_LEADING	Zeilenabstandsfaktor im Einstellungsdialogfeld
LABEL_TEXT_WIDTH_FACT	Laufweitenfaktor im Einstellungsdialogfeld
LABEL_CHARSPACE_FACT	Zeichenabstandsfaktor im Einstellungsdialogfeld

Parameter von Wänden - nur für Türen/Fenster verfügbar

WALL_RESOL <i>nur im 3D effektiv</i>	J~	3D-Auflösung einer gekrümmten Wand
WALL_THICKNESS <i>im Falle von geneigten Wänden: Wandstärke an der Öffnungsachse (lokale z-Achse)</i>	C_	Wandstärke
WALL_START_THICKNESS		Wandstärke am Anfang
WALL_END_THICKNESS		Wandstärke am Ende
WALL_INCL <i>der Winkel zwischen den zwei geneigten Wandoberflächen - 0 für allgemeine gerade Wände</i>		Neigung der Wandoberflächen
WALL_HEIGHT	D_	Wandhöhe
WALL_MIN_HEIGHT		Minimale Höhe der Wand
WALL_MAX_HEIGHT		Maximale Höhe der Wand
WALL_MAT_A <i>im Falle von Öffnungen kann es von Öffnung zu Öffnung, die in derselben Wand platziert sind, verschieden sein</i>	G_	Material der Wand an der Seite gegenüber der Öffnungsseite
WALL_MAT_B <i>im Falle von Öffnungen kann es von Öffnung zu Öffnung, die in derselben Wand platziert sind, verschieden sein</i>	H_	Material der Wand an der Öffnungsseite
WALL_MAT_EDGE	I_	Material der Wandkanten
WALL_LINETYPE <i>wird nur auf den Konturen im Grundriß-Fenster verwendet</i>		Linientyp der Wand
WALL_FILL <i>Schraffurindex, erste Oberfläche mehrschichtiger Bauteile</i>	A~	Schraffurtyp der Wand
WALL_FILL_PEN	F_	Stift der Wandschraffur
WALL_COMPS_NAME <i>Name der Mehrschicht-Struktur, variiert von 1 bis 8, 0 wenn nur eine Schraffur benutzt wird</i>		Mehrschicht-Struktur der Wand
WALL_SKINS_PARAMS <i>Anordnung mit 12 Stützen: Schraffur, Dicke, (alter Konturstift), Stift der Schraffur, Stift des Schraffur-Hintergrundes, Stift der oberen Linie, Stift der unteren Linie, Typ der unteren Linie, Richtung der Schraffur - bis 8 Zeilen Kern-Status: 0 - nicht Teil, 1 - Teil, 3 - letzte Oberfläche des Kerns Richtung der Schraffur: 0 - global, 1 - lokal</i>		Parameter der Schichten einer zusammengesetzten Wand
WALL_SECT_PEN <i>Wird auf Konturen der geschnittenen Wandoberflächen sowohl im Grundriß-Fenster als auch im Schnitte/Ansichten-Fenster verwendet</i>	E_	Stift der Wandkonturen im Schnitt
WALL_VIEW_PEN <i>verwendet für alle Kanten im 3D-Fenster und für alle sichtbaren Kanten in den Schnitte/Ansichten-Fenstern</i>		Stift der Wandkonturen in der Ansicht
WALL_FBGD_PEN		Stift des Schraffurhintergrundes der Wand
WALL_DIRECTION <i>gerade Wände: die Orientierung der Konstruktionslinie,</i>		Ausrichtung der Wand
WALL_POSITION <i>die Position des Anfangspunktes der Wand im Verhältnis zum Projektursprung</i>		absolute Koordinaten der Wand

Parameter von Wänden - nur zum Auflisten und für Etiketten verfügbar

WALL_LENGTH_A	Länge der Wand an der Seite der Konstruktionslinie
WALL_LENGTH_B	Länge der Wand an der Seite gegenüber der Konstruktionslinie
WALL_CENTER_LENGTH	Länge der Wand in der Achse
WALL_AREA	Grundfläche der Wand
WALL_PERIMETER	Umfang der Wand
WALL_SURFACE_A	Oberfläche der Wand an der Seite der Konstruktionslinie
WALL_SURFACE_B	Oberfläche der Wand an der Seite gegenüber der Konstruktionslinie
WALL_GROSS_SURFACE_A	Brutto-Oberfläche der Wand an der Konstruktionsseite
WALL_GROSS_SURFACE_B	Brutto-Oberfläche der Wand an der Seite gegenüber der Konstruktionslinie
WALL_EDGE_SURF	Oberfläche der Wandkante
WALL_VOLUME	Volumen der Wand
WALL_GROSS_VOLUME	Brutto-Volumen der Wand
WALL_DOORS_NR	Anzahl der Türen in der Wand
WALL_WINDS_NR	Anzahl der Fenster in der Wand
WALL_HOLES_NR	Anzahl der leeren Öffnungen
WALL_DOORS_SURF	Oberfläche der Türen in der Wand
WALL_WINDS_SURF	Oberfläche der Fenster in der Wand
WALL_HOLES_SURF	Oberfläche der leeren Öffnungen in der Wand
WALL_HOLES_SURF_A	Analytische Oberfläche aller Öffnungen an der Konstruktionsseite
WALL_HOLES_VOLUME	Analytische Oberfläche der Öffnungen in der Wand
WALL_WINDS_WID	kombinierte Breite der Fenster in der Wand
WALL_DOORS_WID	kombinierte Breite der Türen in der Wand
WALL_COLUMNS_NR	Anzahl der Stützen in der Wand

Parameter von Stützen - nur zum Auflisten verfügbar

COLU_CORE	Kern-/Ummantelung-Beschreibungen <i>dient zur Kompatibilität: es hat eine Auswirkung nur im Descriptor-Script von .CPS-Dateien (Stützen.Beschreibungen)</i>
COLU_HEIGHT	Höhe der Stütze
COLU_MIN_HEIGHT	Minimale Höhe der Stütze
COLU_MAX_HEIGHT	Maximale Höhe der Stütze
COLU_VENEER_WIDTH	Dicke der Stützenummantelung
COLU_CORE_X	Breite des Kerns
COLU_CORE_Y	Tiefe des Kerns
COLU_DIM1	1. Maß der Stütze
COLU_DIM2	2. Maß der Stütze
COLU_MAT	Material der Stütze
Anmerkung: Durch die Wandanpassungsmethode wird das Material der Stütze mit den Materialien der Verbindungswände ersetzt	
COLU_LINETYPE	Linientyp der Säule <i>wird nur auf den Konturen im Grundriß-Fenster verwendet</i>
COLU_CORE_FILL	Schraffur des Stützenkernes
COLU_VENEER_FILL	Schraffur der Stützenummantelung
COLU_SECT_PEN	Stift der Stützenkonturen im Schnitt <i>verwendet auf den Konturen der geschnittenen Wandoberflächen im Grundriß-Fenster und im Schnitte/Ansichten-Fenster</i>
COLU_VIEW_PEN	Stift der sichtbaren Stütze <i>verwendet für alle Kanten im 3D-Fenster und für alle sichtbaren Kanten in den Schnitte/Ansichten-Fenstern</i>
COLU_CORE_FILL_PEN	Stift der Schraffur des Stützenkernes
COLU_CORE_FBGD_PEN	Stift des Schraffurhintergrundes des Stützenkernes
COLU_VENEER_FILL_PEN	Stift der Schraffur der Ummantelung
COLU_VENEER_FBGD_PEN	Stift des Schraffurhintergrundes der Stützenummantelung
COLU_PERIMETER	Umfang der Stütze
COLU_AREA	Fläche der Stütze
COLU_VOLUME	Volumen der Stütze
COLU_GROSS_VOLUME	Brutto-Volumen der Stütze
COLU_CORE_SURF	Oberfläche des Stützenkernes
COLU_CORE_GROSS_SURF	Brutto-Oberfläche des Stützenkernes

COLU_CORE_VOL	Oberfläche des Stützenkernes
COLU_CORE_GROSS_VOL	Brutto-Volumen der Stütze
COLU_VENEER_SURF	Oberfläche der Stützenummantelung
COLU_VENEER_GROSS_SURF	Brutto-Oberfläche der Stützenummantelung
COLU_VENEER_VOL	Volumen der Stützenummantelung
COLU_VENEER_GROSS_VOL	Brutto-Volumen der Stützenummantelung
COLU_CORE_TOP_SURF	Oberfläche der Oberseite des Stützenkernes
COLU_CORE_BOT_SURF	Oberfläche der Unterseite des Stützenkernes
COLU_VENEER_TOP_SURF	Oberfläche der Oberseite des Stützenmantels
COLU_VENEER_BOT_SURF	Oberfläche der Unterseite des Stützenmantels
COLU_CORE_GROSS_TOPBOT_SURF	Brutto-Oberfläche der Ober- und Unterseite des Stützenkernes
COLU_VENEER_GROSS_TOPBOT_SURF	Brutto-Oberfläche der Ober- und Unterseite des Stützenmantels

Unterzugparameter nur zur Auflistung verfügbar

BEAM_THICKNESS	Stärke des Unterzuges
BEAM_HEIGHT	Höhe der Unterzuges
BEAM_REFLINE_OFFSET	Abstand der Referenzachse zu den Achsen des Unterzugs
BEAM_PRIORITY	Indexnummer der 3D-Verschneidungspriorität
BEAM_MAT_RIGHT	Material des Unterzugs auf der rechten Seite der Referenzachse
BEAM_MAT_LEFT	Material des Unterzugs auf der linken Seite der Referenzachse
BEAM_MAT_TOP	Material des Unterzugs (oben)
BEAM_MAT_BOTTOM	Material des Unterzugs (unten)
BEAM_MAT_END	Material des Unterzugs an beiden Enden
BEAM_OUTLINE_LINETYPE	Linientyp der Unterzugskontur
BEAM_AXES_LINETYPE	Linientyp der Unterzugsachse
BEAM_FILL	Schraffurtyp des Unterzuges
BEAM_FILL_PEN	Stift der Unterzugsschraffur
BEAM_SECT_PEN	Stift der Kontur des Unterzugs im Schnitt
BEAM_FBGD_PEN	Stift des Schraffurhintergrund des Unterzugs
BEAM_DIRECTION	Richtung der Referenzachse des Unterzugs
BEAM_POSITION	absolute Koordinaten des Anfangspunkts der Unterzugsachsen
BEAM_LENGTH_RIGHT	Länge des Unterzugs auf der rechten Seite der Referenzachse
BEAM_LENGTH_LEFT	Länge des Unterzugs auf der linken Seite der Referenzachse
BEAM_RIGHT_SURF	Oberfläche des Unterzugs auf der rechten Seite der Referenzachse
BEAM_LEFT_SURF	Oberfläche des Unterzugs auf der linken Seite der Referenzachse
BEAM_TOP_SURF	Oberfläche der Oberseite des Unterzugs
BEAM_BOTTOM_SURF	Oberfläche der Unterseite des Unterzugs
BEAM_END_SURF	Oberfläche der Unterseite des Unterzugs
BEAM_VOLUME	Volumen des Unterzugs
BEAM_HOLES_NR	Anzahl der Durchbrüche im Unterzug
BEAM_HOLES_SURF	gesamte Oberfläche der Durchbrüche im Unterzug
BEAM_HOLE_EDGE_SURF	gesamte Oberfläche von Durchbruchskanten im Unterzug
BEAM_HOLES_VOLUME	gesamtes Volumen der Durchbrüche im Unterzug

Parameter von Decken - nur zum Auflisten verfügbar

SLAB_THICKNESS	Dicke der Decke
SLAB_MAT_TOP	Material der Oberseite der Decke
SLAB_MAT_EDGE	Material der Deckenkanten
SLAB_MAT_BOTT	Material der Unterseite der Decke
SLAB_LINETYPE	Linientyp der Decke
SLAB_FILL	Schraffur der Decke
<i>Schraffurindex - sein Wert ist negativ im Falle einer Mehrschicht-Aufbau</i>	
SLAB_FILL_PEN	Stift der Schraffur der Decke
SLAB_FBGD_PEN	Stift des Schraffurhintergrundes der Decke
SLAB_COMPS_NAME	Mehrschicht-Struktur der Decke
<i>Mehrschicht-Struktur der Decke</i>	
SLAB_SKINS_NUMBER	Anzahl der Schichten der zusammengesetzten Decke
<i>es liegt zwischen 1 und 8, es ist 0, falls eine einzige Schraffur verwendet wird</i>	
SLAB_SKINS_PARAMS	Parameter der Schichten einer zusammengesetzten Decke
<i>Anordnung mit 12 Stützen: Schraffur, Dicke, (alter Konturstift), Stift der Schraffur, Stift des Schraffur-Hintergrundes, Stift der oberen Linie, Stift der unteren Linie, Typ der unteren Linie, Richtung der Schraffur - bis 8 Zeilen Kern-Status: 0 - nicht Teil, 1 - Teil, 3 - letzte Oberfläche des Kerns Richtung der Schraffur: 0 - global, 1 - lokal</i>	
SLAB_SECT_PEN	Stift der Deckenkonturen im Schnitt
<i>auf Konturen der geschnittenen Wandoberflächen sowohl im Grundriß-Fenster als auch im Schnitte/Ansichten-Fenster verwendet</i>	
SLAB_VIEW_PEN	Stift der Decke
<i>verwendet für alle Kanten im 3D-Fenster und für alle sichtbaren Kanten in den Schnitte/Ansichten-Fenstern</i>	
SLAB_TOP_SURF	Oberfläche der Oberseite der Decke
SLAB_GROSS_TOP_SURF	Brutto-Oberfläche der Oberseite der Decke
SLAB_BOT_SURF	Oberfläche der Unterseite der Decke
SLAB_GROSS_BOT_SURF	Brutto-Oberfläche der Unterseite der Decke
SLAB_EDGE_SURF	Oberfläche der Deckenkanten
SLAB_GROSS_EDGE_SURF	Brutto-Oberfläche der Deckenkanten
SLAB_PERIMETER	Umfang der Decke
SLAB_VOLUME	Volumen der Decke
SLAB_GROSS_VOLUME	Brutto-Volumen der Decke
SLAB_SEGMENTS_NR	Anzahl der Segmenten der Decke
SLAB_HOLES_NR	Anzahl der Durchbrüche in der Decke
SLAB_HOLES_AREA	Fläche der Durchbrüche in der Decke
SLAB_HOLES_PRM	Umfang der Durchbrüche in der Decke

Parameter von Dächern - nur zum Auflisten verfügbar

ROOF_THICKNESS	Dicke des Daches
ROOF_ANGLE	Neigung des Daches
ROOF_MAT_TOP	Material der Oberfläche des Daches
ROOF_MAT_EDGE	Material der Dachkanten
ROOF_MAT_BOTT	Material der Unterseite des Daches
ROOF_LINETYPE	Linientyp des Daches
<i>wird nur auf den Konturen im Grundriß-Fenster verwendet</i>	
ROOF_FILL	Schraffur des Daches
<i>Schraffurindex - sein Wert ist negativ im Falle einer Mehrschicht-Aufbau</i>	
ROOF_FILL_PEN	Stift der Schraffur des Daches
ROOF_FBGD_PEN	Stift des Schraffurhintergrundes des Daches
ROOF_COMPS_NAME	Mehrschicht-Struktur des Daches
<i>Mehrschicht-Struktur der Decke</i>	
ROOF_SKINS_NUMBER	Anzahl der Schichten des zusammengesetzten Daches
<i>es liegt zwischen 1 und 8, es ist 0, falls eine einzige Schraffur verwendet wird</i>	
ROOF_SKINS_PARAMS	Parameter der Schichten des zusammengesetzten Daches
<i>Anordnung mit 12 Stützen: Schraffur, Dicke, (alter Konturstift), Stift der Schraffur, Stift des Schraffur-Hintergrundes, Stift der oberen Linie, Stift der unteren Linie, Typ der unteren Linie, Richtung der Schraffur - bis 8 Zeilen Kern-Status: 0 - nicht Teil, 1 - Teil, 3 - letzte Oberfläche des Kerns Richtung der Schraffur: 0 - global, 1 - lokal</i>	
ROOF_SECT_PEN	Stift der Dachkonturen im Schnitt
<i>auf Konturen der geschnittenen Wandoberflächen sowohl im Grundriß-Fenster als auch im Schnitte/Ansichten-Fenster verwendet</i>	
ROOF_VIEW_PEN	Stift des Daches in der Ansicht
<i>verwendet für alle Kanten im 3D-Fenster und für alle sichtbaren Kanten in den Schnitte/Ansichten-Fenstern</i>	
ROOF_BOTTOM_SURF	Oberfläche der Unterseite des Daches
ROOF_GROSS_BOTTOM_SURF	Brutto-Oberfläche der Unterseite des Daches
ROOF_TOP_SURF	Oberfläche der Oberseite des Daches
ROOF_GROSS_TOP_SURF	Brutto-Oberfläche der Oberseite des Daches
ROOF_EDGE_SURF	Oberfläche der Dachkante
ROOF_GROSS_EDGE_SURF	Brutto-Oberfläche der
ROOF_PERIMETER	Umfang des Daches
ROOF_VOLUME	Volumen des Daches
ROOF_GROSS_VOLUME	Brutto-Volumen des Daches
ROOF_SEGMENTS_NR	Anzahl von Segmenten des Daches
ROOF_HOLES_NR	Anzahl der Durchbrüche im Dach
ROOF_HOLES_AREA	Fläche der Durchbrüche im Dach
ROOF_HOLES_PRM	Umfang der Durchbrüche im Dach

Schraffurparameter- nur zum Auflisten verfügbar

FILL_LINETYPE	Linientyp der Schraffur
FILL_FILL	Schraffurtyp der Schraffur
FILL_FILL_PEN	Stift des Schraffurtyps der Schraffur
FILL_PEN	Stift der Schraffur
FILL_FBGD_PEN	Hintergrundstift der Schraffur
FILL_SURF	Bereich der Schraffur
FILL_PERIMETER	Umfang der Schraffur
FILL_SEGMENT_NR	Anzahl von Segmenten der Schraffur
FILL_HOLES_NR	Anzahl der Durchbrüche in der Schraffur
FILL_HOLES_PRM	Umfang der Durchbrüche in der Schraffur
FILL_HOLES_AREA	Fläche der Durchbrüche der Schraffur

Parameter der Freiflächenform - nur zum Auflisten verfügbar

MESH_TYPE	Typ der Freiflächenform
<i>1- geschlossener Körper 2 - Decke & Kante, 3 - nur Deckfläche</i>	
MESH_BASE_OFFSET	Versatz der Bodenfläche zum Basisniveau
MESH_USEREDGE_PEN	Stift der benutzerdefinierten Firsten der Freiflächenform
MESH_TRIEDGE_PEN	Stift der durch Triangulation gekrümmten Kanten der Freiflächenform
MESH_SECT_PEN	Stift der Konturen der Freiflächenform im Schnitt
<i>verwendet auf die Konturen der geschnittenen Wandoberflächen im Grundriss-Fenster und im Schnitte/Ansichten-Fenster</i>	
MESH_VIEW_PEN	Stift der Konturen in der Ansicht
<i>verwendet für alle Kanten im 3D-Fenster und für alle sichtbaren Kanten in den Schnitte/Ansichten-Fenstern</i>	
MESH_MAT_TOP	Material der Oberfläche der Freifläche
MESH_MAT_EDGE	Material der Freifläche
MESH_MAT_BOTT	Material der Unterseite Freifläche
MESH_LINETYPE	Linientyp der Freifläche
<i>wird nur auf den Konturen im Grundriß-Fenster verwendet</i>	
MESH_FILL	Schraffurtyp der Freifläche
MESH_FILL_PEN	Stift der Schraffur der Freifläche
MESH_FBGD_PEN	Stift des Schraffurhintergrund der Freifläche
MESH_BOTTOM_SURF	Oberfläche der Unterseite der Freifläche
MESH_TOP_SURF	Oberfläche der der Freifläche
MESH_EDGE_SURF	Oberfläche der Kante der Freifläche
MESH_PERIMETER	Umfang der Freifläche
MESH_VOLUME	Volumen der Freifläche
MESH_SEGMENTS_NR	Anzahl von Segmenten der Freifläche
MESH_HOLES_NR	Anzahl der Durchbrüche in der Freifläche
MESH_HOLES_AREA	Fläche der Durchbrüche in der Freifläche
MESH_HOLES_PRM	Umfang der Durchbrüche in der Freifläche

Benutzerdefinierbare globale Variablen

GLOB_USER_1	S_	
GLOB_USER_2	T_	
GLOB_USER_3	U_	
GLOB_USER_4	V_	
GLOB_USER_5	W_	
GLOB_USER_6	X_	
GLOB_USER_7	Y_	
GLOB_USER_8	Z_	
GLOB_USER_9	G~	
GLOB_USER_10	I~	benutzerdefinierbare globale Variablen von 1 bis 10 werden auf die grundeingestellte Nummer initialisiert
GLOB_USER_11		
GLOB_USER_12		
GLOB_USER_13		
GLOB_USER_14		
GLOB_USER_15		
GLOB_USER_16		
GLOB_USER_17		
GLOB_USER_18		
GLOB_USER_19		
GLOB_USER_20		benutzerdefinierbare globale Variablen von 11 bis 20 werden auf die grundeingestellte Nummer initialisiert

Beispiel zur Illustration der globalen Variablen GLOB_WORLD_ORIGO_...:

```
GLOB_WORLD_ORIGO_... globals:
ADD2 -GLOB_WORLD_ORIGO_OFFSET_X - SYMB_POS_X, -GLOB_WORLD_ORIGO_OFFSET_X -SYMB_POS_Y
LINE2 -0.1, 0.0, 0.1, 0.0
LINE2 0.0, -0.1, 0.0, 0.1
HOTSPOT2 0.0, 0.0, 1
TEXT2 0, 0, "( 0.00 ; 0.00 )"
TEXT2 0, 0.5, "World Origo"
DEL TOP
if ABS(GLOB_WORLD_ORIGO_OFFSET_X) > 0.01 OR ABS(GLOB_WORLD_ORIGO_OFFSET_Y) > 0.01 THEN
    ADD2 = SYMB_POS_X, - SYMB_POS_Y
    LINE2 -0.1, 0.0, 0.1, 0.0
    LINE2 0.0, -0.1, 0.0, 0.1
    HOTSPOT2 0.0, 0.0, 2
    TEXT2 0, 0, "(" + STR (GLOB_WORLD_ORIGO_OFFSET_X, 9, 4) + "; " + STR
(GLOB_WORLD_ORIGO_OFFSET_Y, 9, 4) + " )"
    TEXT2 0, 0.5, "Virtual Origo"
    DEL TOP
ENDIF
if ABS(GLOB_WORLD_ORIGO_OFFSET_X + SYMB_POS_X) > 0.01 OR ABS(GLOB_WORLD_ORIGO_OFFSET_Y +
SYMB_POS_Y) > 0.01 THEN
    LINE2 -0.1, 0.0, 0.1, 0.0
    LINE2 0.0, -0.1, 0.0, 0.1
    HOTSPOT2 0.0, 0.0, 3
    TEXT2 0, 0, "(" + STR (GLOB_WORLD_ORIGO_OFFSET_X + SYMB_POS_X, 9, 4) + "; " + STR
(GLOB_WORLD_ORIGO_OFFSET_Y + SYMB_POS_Y, 9, 4) + " )"
    TEXT2 0, 0.5, "Object Placement"
ENDIF
```

Alte Globale Variablen

Alte globale Variablenamen können benutzt werden, trotzdem empfehlen wir die Verwendung der Neuen. Alle alten Globalen beziehen sich auf eine neue Variable mit einem langen Namen.

A_	GLOB_SCALE	F~	WIDO_FRAME_THICKNESS
B_	GLOB_HISTORY_ELEV	G~	GLOB_USER_9
C_	WALL_THICKNESS	H~	WIDO_POSITION
D_	WALL_HEIGHT	I~	GLOB_USER_10
E_	WALL_SECT_PEN	J~	WALL_RESOL
F_	WALL_FILL_PEN	K~	GLOB_EYEPOS_X
G_	WALL_MAT_A	L~	GLOB_EYEPOS_Y
H_	WALL_MAT_B	M~	GLOB_EYEPOS_Z
I_	WALL_MAT_EDGE	N~	GLOB_TARGPOS_X
J_	GLOB_ELEVATION	O~	GLOB_TARGPOS_Y
K_	WIDO_SILL	P~	GLOB_TARGPOS_Z
L_	SYMB_VIEW_PEN	Q~	GLOB_CSTORY_ELEV
M_	SYMB_MAT	R~	GLOB_CSTORY_HEIGHT
N_	GLOB_FRAME_NR	S~	GLOB_CH_STORY_DIST
O_	GLOB_FIRST_FRAME	T~	GLOB_SCRIPT_TYPE
P_	GLOB_LAST_FRAME	U~	GLOB_NORTH_DIR
Q_	GLOB_HISTORY_HEIGHT	V~	SYMB_MIRRORED
R_	WIDO_ORIG_DIST	W~	SYMB_ROTANGLE
S_	GLOB_USER_1	X~	SYMB_POS_X
T_	GLOB_USER_2	Y~	SYMB_POS_Y
U_	GLOB_USER_3	Z~	SYMB_POS_Z
V_	GLOB_USER_4		
W_	GLOB_USER_5		
X_	GLOB_USER_6		
Y_	GLOB_USER_7		
Z_	GLOB_USER_8		
A~	WALL_FILL		
B~	WIDO_RIGHT_JAMB		
C~	WIDO_THRES_DEPTH		
D~	WIDO_HEAD_DEPTH		
E~	WIDO_REVEAL_SIDE		

SPEZIELLE FUNKTIONEN

REQ

REQ (parameter_string)

Diese Funktion ermittelt den laufenden Status des Programmes. Sein Parameter - die Frage - ist eine Zeichenfolge. Der GDL-Interpreter antwortet mit einem numerischen Wert. Wenn er die Frage nicht versteht ist die Antwort negativ.

Liste der aktuellen Fragen:

"GDL_version"

Versionsnummer des GDL-Compilers/Interpreters .

Achtung: ArchiCAD-Version und GDL-Version unterscheiden sich.

"Program"

Programmcode (z. B. 1: ArchiCAD).

"Serial_number"

Seriennummer des Schutzschlüssels.

"Model_size"

Größe der aktuellen 3D-Datenstruktur in Bytes.

"Red_of_material name"

"Green_of_material name"

"Blue_of_material name"

Definiert die gegebenen Farbkomponenten eines Materials in RGB-Werten zwischen 0 und 1.

"Red_of_pen index"

"Green_of_pen index"

"Blue_of_pen index"

Definiert die gegebenen Farbkomponenten eines Stiftes in RGB-Werten zwischen 0 und 1.

"Pen_of_RGB r g b"

Gibt den Index des Stiftes an, der der definierten Farbe am nächsten ist. r, g und b werden in Werten zwischen 0 und 1 definiert.

REQUEST

REQUEST (question_name, name | index, variable1 [, variable2,...])

Der erste Parameter stellt den Fragetext dar, der zweite den Gegenstand der Frage (falls es eine existiert) und kann entweder vom Text-oder vom numerischen Typ sein. (z.B. kann die Frage, "Rgb_of_material" und der Gegenstand der Materialname, oder "Rgb_of_pen" und sein Gegenstand der Index des Stiftes sein). Die anderen Parameter sind veränderliche Namen, in denen die Rückwerte (die Antworten) gespeichert werden.. Das Ergebnis der Funktion ist die Anzahl der Antworten (eine falsch gestellte Frage oder ein nicht existierender Name, ergibt einen 0 Wert).

dummy REQUEST (Name_of_Listed, , name)

Stellt den Namen des Programms in der bestimmten Variable, z.B. "ArchiCAD", etc dar.

Beispiel: Drucken des Programmnamens

```
n=REQUEST(Name_of_program, , program_name)
PRINT program_name
```

```
dummy REQUEST (Name_of_Listed, , name)
```

```
dummy REQUEST (Name_of_Listed, , name)
```

Nach der Ausführung dieser Funktionsaufrufe wird die Variable mein_name den Namen des Makros enthalten, während die Variable haupt_name den Namen des Hauptmakros enthält (falls es keines existiert, leerer Text).

REQUEST ("ID_of_main", "", id_string)

Für Bibliothekselemente, die im Grundriß platziert wurden, kehrt der Identifizierer-Bestand in dem Dialogfenster Werkzeugeinstellungen in der id_zeichenfolge Variable zurück (anderenfalls leere Zeichenfolge).

```
dummy REQUEST (Name_of_Listed, , name)
```

Ergibt den Namen des aktuellen Projektes in der bestimmten Variable.

REQUEST ("Story", "", index, story_name)

Ergibt den Index und Namen des aktuellen Geschosses in der index und story_name Variable.

REQUEST ("Home_story", "", index,
story_name)

Ergibt den Index und Namen des Referenzgeschosses in der index und story_name Variable.

REQUEST (STORY_INFO
, expr, nStories, index1, name1, elev1, height1 [
, index2, name2, ...])

Gibt die Geschossinformationen in den gegebenen Variablen zurück: Anzahl der Geschosse und Geschossindex, Name, Position, sukzessive Geschosshöhe. Wenn expr ein numerischer Ausdruck ist, kennzeichnet er einen Geschossindex: nur die Anzahl der Geschosse und die Informationen zu dem angegebenen Geschoss werden zurückgegeben. Ist expr ein Zeichenfolgeausdruck, bedeutet dies, dass Informationen zu allen Geschossen angefordert werden. Der Rückgabewert der Funktion ist die Anzahl der erfolgreich abgerufenen Werte.

Beispiel:

```

DIM t[]
  n = REQUEST ("STORY_INFO", "", nr, t)
  FOR i = 1 TO nr
    nr = STR ("%.0m", t [4 * (i - 1) + 1])
    name = t [4 * (i - 1) + 2]
    elevation = STR ("%m", t [4 * (i - 1) + 3])
    height = STR ("%m", t [4 * (i - 1) + 4])
    TEXT2 0, -i, nr + "," + name + "," + elevation + "," + height
  NEXT i

REQUEST ("Internal_id", "", id)
Ergibt die interne Id-Nummer des Bibliothekselementes in der id Variable.

REQUEST ("Linear_dimension", "",
  format_string)

REQUEST ("Angular_dimension", "",
  format_string)

REQUEST ("Angular_length_dimension", ""
  format_string)

REQUEST ("Radial_dimension", "",
  format_string)

REQUEST ("Level_dimension", "",
  format_string)

REQUEST ("Elevation_dimension", "",
  format_string)

REQUEST ("Window_door_dimension", "",
  format_string)

REQUEST ("Sill_height_dimension", "",
  format_string)

REQUEST ("Area_dimension", ""
  format_string)

REQUEST ("Calculation_length_unit", "",
  format_string)

REQUEST ("Calculation_area_unit", "",
  format_string)

```

```
REQUEST ("Calculation_volume_unit", "",  
         format_string)
```

```
REQUEST ("Calculation_angle_unit", "",  
         format_string)
```

Durch diese REQUESTs ermitteln Sie mit die Bemaßungsformate im Grundeinstellungen/Bemaßungen Menü. Diese REQUESTs ergeben einen Formattext, der als erster Parameter in der STR ()- Funktion verwendet werden kann.

Beispiel:

```
format = ""  
num = 60.55  
REQUEST ("Angular_dimension", "",format)  
! "%.2dd"  
TEXT2 0, 0, STR (format, num)!60.55
```

```
REQUEST ("Clean_intersections", "", state)
```

Ergibt den Status der Reinzeichnung-Funktion aus dem Menü (1, wenn eingeschaltet, 0, wenn ausgeschaltet).

```
REQUEST ("Zone_category", "", name, code)
```

Für Räume, ergibt den Namen und den Code der aktuellen Raum-Kategorie.

```
REQUEST ("Zone_relations", "", category_name, code, name, number [ ,category_name2,  
         code2, name2, number2])
```

Ergibt den Namen und Code der Raumkategorie, den Raumnamen und die Nummer des Raumes, in dem das Bibliothekselement mit diesem Funktionsaufruf plaziert wird. Für Türen und Fenster können maximum zwei Räume bestimmt werden. Ergibt die Anzahl der erfolgreich eingelesenen Werte (0, wenn das Bibliothekselement nicht innerhalb eines Raumes ist).

```
REQUEST ("Zone_colus_area", "", area)
```

Ergibt den gesamten Bereich der im aktuellen Raum plazierten Stützen in der Bereich-Variable. Es hat nur für Raumstempel eine Auswirkung . Nur bei Raumstempeln wirksam.

```
REQUEST ("Custom_auto_label", "", name)
```

Ergibt den Namen der benutzerdefinierten, automatischen Sprungmarke des Bibliothekselementes oder der leeren Zeichenfolge, falls es nicht existiert, in der Namensvariable.

```
REQUEST ("Rgb_of_material", name, r, g, b)
```

```
REQUEST ("Rgb_of_pen", penindex, r, g, b)
```

```
REQUEST ("Pen_of_RGB", "r g b", penindex)
```

Wie die REQ()-Funktion, ergibt der Wert die r, g, b Bestandteile des Materials und Stiftes oder den Index des Stiftes (nur in einem einzigen Aufruf) mit den angegebenen r, g, b Werten.

```
REQUEST ("Height_of_style", name,  
         height [, descent, leading])
```

Ergibt in den gegebenen Variablen der Gesamthöhe des gemessenen Stils in Millimetern (Höhe in Meter entspricht der Höhe / 1000 * GLOB_SCALE); die Unterlänge (Abstand in Millimetern von der Text-Grundlinie bis zur Unterlängelinie) und den Zeilenabstand (Abstand in Millimetern von der Unterlängelinie bis zur Oberlängelinie).

REQUEST ("Style_info", name, fontname [, size, anchor, face_or_slant])

Ergibt in den gegebenen Variablen Informationen zu dem zuvor definierten Stil (siehe Stilparameter bei der Beschreibung der Anweisung DEFINE STYLE. *"DEFINE STYLE" auf Seite 176* Kann in Makros hilfreich sein zum Erfassen von Informationen zu dem im Haupt-Script definierten Stil.

REQUEST ("Name_of_material", index, name)

Ergibt den Namen des durch index identifizierten Materials in der bestimmten Variable.

REQUEST ("Name_of_fill", index, name)

Ergibt den Namen der durch Index identifizierten Schraffur im Variablenamen.

REQUEST ("Name_of_line_type", index, name)

Ergibt den Durch den Index angegebenen Liniennamen in der gegebenen Variablen.

REQUEST ("Name_of_style", index, name)

Ergibt den Namen der durch Index identifizierten Linie in der bestimmten Variable.

Ist der index < 0, bezieht er sich auf ein Material, eine Schraffur, einen Linientyp oder Stil, die im GDL-Script definiert werden oder auf die MASTER_GDL-Datei. Ein REQUEST-Aufruf mit dem index=0 stellt den Namen des grundeingestellten Materials oder Linientypes in der Variable wieder her. (Leere Zeichenfolge für Schraffur und Stil.)

Der Ergebniswert der Funktion ist die Anzahl der erfolgreich abgerufenen Werte, 1 wenn keine Eigenschaftsdaten gefunden wurden oder 0 im Fall eines Fehlers, wenn der Index ungültig ist.

REQUEST ("window_door_show_dim", " ", show)

Ergibt 1 in der Anzeige der Variable, wenn unter Optionen/Reinzeichnungseinstellungen/Türen & Fenster die Option "Mit Bemaßungen anzeigen" gewählt ist, andernfalls ist es Null. Es kann zum Verbergen/Anzeigen von benutzerdefinierten Bemaßungen verwendet werden, entsprechend den aktuellen Reinzeichnungseinstellungen.

REQUEST ("name_of_listed", " ", name)

Ergibt in der Namensvariable den Namen des Bibliothekselementes, das verbunden mit demjenigen Bibliothekselement vom Eigenschaftentyp ist, das diesen Fragetext enthält. Für Elemente (Wände, Decken, etc.) ist der Name eine leere Zeichenfolge.

REQUEST ("window_door_zone_relev", " ", out_direction)

Dieser Fragetext ist lediglich für Türen und Fenster wirksam. Verwenden Sie diesen Fragetext als Ergänzung zum Fragetext "ZONE_RELATIONS".Stellt 1 in der out_direction variable wieder her, falls sich die Richtung der Tür-/Fensteröffnung in dem ersten Raum befindet, der durch den Fragetext "ZONE_RELATIONS" identifiziert wird, der Wert ist 2 falls die Öffnungsrichtung in den zweiten Raum zeigt. Dieser Wert wandelt auch dann in 2, falls nur ein Raum vorhanden ist und die Öffnungsrichtung nach außen zeigt.

REQUEST ("matching_properties", type, name1, name2, ...)

Wenn `typ = 1`, werden in den vorgegebenen Variablen individuell assoziierte Namen von Beschreibungs-Bibliothekselementen übergeben, sonst Namen von Beschreibungs-Bibliothekselementen, die durch Kriterien verbunden sind. Wenn sie in einem assoziativen Etikett verwendet wird, übergibt die Funktion die Beschreibungen der Elemente, mit denen das Etikett assoziiert ist.

REQUEST (`extension_name`, `parameter_string`, `variable1`, `variable2`, ...)

Falls die Frage mit keiner der aufgelisteten Fragen übereinstimmt, versucht die **REQUEST** () Funktion diese als erweiterungsspezifischer Name zu verwenden. Befindet sich diese Erweiterung im Erweiterungen-Ordner, so wird diese zum Anschaffen sämtlicher Werte für alle genau angegebenen Variablenamen verwendet. Der Parameter -Text wird durch die Erweiterung interpretiert.

REQUEST ("**Constr_Fills_display**", "", `optionVal`)

Gibt in der bestimmten Variable den Wert der Reinzeichnung der Konstruktionsschraffuren zurück, wie er unter Optionen/Reinzeichnungseinstellungen/Konstruktionsschraffuren eingestellt wurde). Mögliche Statuswerte:

- 1 - Leer
- 2 - Keine Schraffuren
- 4 - Vollinie
- 5 - Bitmap-Muster
- 6 - Vektorielle Schraffur

Der Rückgabewert der Funktion ist die Anzahl der erfolgreich abgerufenen Werte bzw. 0, wenn ein Fehler aufgetreten ist.

REQUEST ("**Working_length_unit**", "", `format_string`)

REQUEST ("**Working_angle_unit**", "", `format_string`)

Durch diese Fragetexte können Sie sich mit den verschiedenen Formaten der Einheiten im Projekt vertraut machen, wie sie unter Optionen/Grundeinstellungen/Einheiten im Projekt eingestellt wurden. Es wird ein Formattext zurückgegeben, der als erster Parameter in der **STR** () Funktion verwendet werden kann. Die Fragetexte funktionieren nur wenn der Parameter oder die Scripts der Benutzeroberfläche interpretiert werden.

IND (`TEXTURE`, `name_string`)
IND (`FILL`, `name_string`)
IND (`LINE_TYPE`, `name_string`)
IND (`STYLE`, `name_string`)
IND (`TEXTURE`, `name_string`)

Über diese Funktion wird der aktuelle Index von Materialien, Schraffuren, Linientypen oder Stilattributen wiederhergestellt. Die Hauptverwendung der Ergebniszahl ist ihre Transformation zu einem Makro, das dasselbe Attribut wie das Aufrufmakro benötigt. Das Ergebnis ist für vorübergehende Definitionen negativ, für globale Definitionen positiv.

Siehe "Attribute" auf Seite 129 des ArchiCAD 9 Referenzbuches.

Siehe auch "Inline Attributdefinition" auf Seite 162.

REQUEST ("ASSOCLP_PARVALUE", expr, name_or_index, type, flags, dim1, dim2, p_values)

Stellt die Information in den gegebenen Variablen über den Bibliothekselementparameter, mit dem das Bibliothekselement mit dieser Frage verbunden ist, wieder her. Es kann bei Eigenschaftenobjekten, Etiketten und Markerobjekten verwendet werden.

Der Rückgabewert der Funktion ist die Anzahl der erfolgreich abgerufenen Werte, 0 wenn die spezifizierte Parameter nicht vorhanden ist oder ein Fehler auftrat.

Ausdruck: Objekt des Fragetextes, zugeordneter Bibliothekselementname oder Indexausdruck

name_or_index: ergibt den Index des Parameternamens, anhängig vom vorherigen Ausdruckstyp (stellt den Index wieder her, wenn ein Parametername angegeben wird und umgekehrt)

type: Parametertyp, mögliche Werte

- 1: Boolesche
- 2: ganzzahlig
- 3: reale Zahl
- 4: Zeichenfolge
- 5: Länge
- 6: Winkel
- 7: Linie
- 8: Material
- 9: Schraffur
- 10: Stiftfarbe
- 11: Lichtschalter
- 12: rgb Farbe
- 13: Lichtintensität
- 14: Trennzeichen
- 15: Titel

flags:

flage = j1 + 2 * j2 + 64 * j7 + 128 * j8,

wobei ji 0 oder 1 sein kann

- j1 (1): child
- j2 (2): fett
- j7 (64): inaktiv
- j8 (128): ausgeblendet

dim1, dim2: gibt die Parameterdimensionen zurück: Beide 0 wenn einfach, dim1 > 0, dim2 = 0 bei eindimensionalen Arrays, beide > 0 bei zweidimensionalen Arrays. dim1 ist die Anzahl der Reihen, dim2 die Anzahl der Spalten

p_values: ergibt den Parameterwert eines Werte-Arrays. Die Array-Elemente werden sukzessive ausgelesen, Zeile für Zeile als eindimensionaler Array, unabhängig von den zum Speichern der Variablen angegebenen Dimensionen. Wenn die Variable

kein dynamischer Array ist, werden so viele Elemente gespeichert, wie Platz für Elemente vorhanden ist (bei einer einfachen Variablen also nur eins, das erste Element). Ist "values" ein zweidimensionaler dynamischer Array, werden alle Elemente in der ersten Reihe gespeichert.

REQUEST ("ASSOCLP_NAME", "", name)

Ergibt in der gegebenen Variablen den Namen des Bibliothekselement, das dem Etikett- bzw. Marker-Objekt zugewiesen wurde. Für Elemente (Wände, Decken, etc.) ist der Name eine leere Zeichenfolge.

REQUEST ("ASSOCEL_PROPERTIES ", parameter_string, nr_data, data)

Ergibt in den gegebenen Variablen eigene Eigenschaftsdaten oder die Elementeigenschaften, die dem Bibliothekselement, das diese Anforderung enthält, zugeordnet sind (in Etiketten und zugeordneten Marker-Objekten). Der Ausgabewert der Funktion ist die Anzahl der erfolgreich abgerufenen Werte, 0 wenn keine Eigenschaftsdaten gefunden wurden oder ein Fehler auftrat. Die Funktion kann während des Auflistungsprozesses in Eigenschaftsobjekten nicht verwendet werden.

parameter_string: Eine Kombination von Kurzbeschreibungen, durch Kommas getrennt, für die angeforderten Felder der Eigenschafts-Datensätze. Die Datensätze werden entsprechend angeordnet. Mögliche Werte:

ISCOMP
DBSETNAME
KEYCODE
KEYNAME
CODE
NAME
FULLNAME
QUANTITY
TOTQUANTITY
UNITCODE
UNITNAME
UNITFORMATSTR
PROPOBJNAME

nr_data: gibt die Anzahl der Datenelemente zurück

data: gibt die Eigenschaftsdaten zurück, Datensätze sortiert nach den in der Parameter-Zeichenfolge angegebenen Feldern. Die Werte werden sukzessive Zeile für Zeile zurückgegeben als eindimensionaler Array mit den angeforderten Datensatzfeldern, unabhängig von den zum Speichern der Variablen angegebenen Dimensionen. Wenn die Variable kein dynamischer Array ist, werden so viele Elemente gespeichert, wie Platz für Elemente vorhanden ist (bei einer einfachen Variablen also nur eins, das erste Element). Ist "values" ein zweidimensionaler dynamischer Array, werden alle Elemente in der ersten Reihe gespeichert.

Beispiel:

DIM DATA []


```

n = REQUEST ("ASSOCEL_PROPERTIES", "iscomp, code, name", nr, data)
      IF nr = 0 THEN
        TEXT2 0, 0, "No properties"
      ELSE
        j = 0
        FOR i = 1 TO nr
          IF (i) MOD 3 = 0 THEN
            TEXT2 0, -j, DATA [i] ! name
            j = j + 1
          ENDIF
        NEXT i
      ENDIF

```

```

REQUEST ("REFERENCE_LEVEL_DATA", "",
  name1, elev1, name2, elev2, name3, elev3)

```

Gibt in den gegebenen Variablen die Namen und Positionen der Referenzhöhen an entsprechend der Einstellung im Dialogfenster Einstellungen/Arbeitseinheit/Referenzhöhen.

Der Rückgabewert der Funktion ist die Anzahl der erfolgreich abgerufenen Werte bzw. 0, wenn ein Fehler aufgetreten ist.

```

REQUEST ("ANCESTRY_INFO", expr, name [,
  guid, parent_name1, parent_guid1,
  ...,
  parent_namen, parent_guidn)

```

Informationen zum "Erbe" bei einem Bibliothekselement

Wenn *expr* = 0, wird in den gegebenen Variablen der Name und die globale Unic ID-Nr. des Bibliothekselements, das diese Anforderungsfunktion enthält, zurückgegeben. Optional gibt die Funktion die Namen und die globalen Unic ID-Nummern der übergeordneten Elemente des Bibliothekselements (parent_namei, parent_guidi) zurück. Wenn die Vorlagen für die übergeordneten Elemente nicht geladen sind, sind ihre Namen leere Zeichenfolgen.

Wenn *expr* = 1, werden Informationen zu dem Bibliothekselement zurückgegeben, das durch die Vorlage, die diese Funktion enthält, ersetzt wurde. Wenn die Vorlage in diesem Fall nicht wirklich ersetzt wird, werden keine Werte zurückgegeben.

Der Rückgabewert der Anforderung ist die Anzahl der erfolgreich abgerufenen Werte.

Beispiel:

```

DIM strings[]
n = REQUEST ("ANCESTRY_INFO", 1, name, guid, strings)
IF n > 2 THEN
  ! data of replaced library part
  TEXT2 0, -1, "replacing: " + name + ' ' + guid
  ! parents
  l = -2

```

```
FOR i = 1 to n - 2 STEP 2
    TEXT2 0, 1, strings [i]
    l = l - 1
NEXT i
ENDIF
```

```
REQUEST ( 'TEXTBLOCK_INFO',
    textblock_name, width, height)
```

Ergibt die gegebenen Variablen der Maße in x und y Richtung eines vorher definierten TEXTBLOCK. Die Maße werden abhängig vom Parameterwert `fixed_height` des TEXTBLOCKs in mm oder m im Modellraum angegeben (Millimeters bei 1, Meter im Modellraum bei 0). War die Breite `width` 0, ergibt die Anfrage die kalkulierte Breite und Höhe, war die Breite `width` in der TEXTBLOCK Definition angegeben, wird die sich auf diese Breite beziehende kalkulierte Höhe ausgegeben.

```
REQUEST{2} ("Material_info", name_or_index,
    param_name, value_or_values)
```

```
REQUEST{2} ("Material_info", name_or_index,
    extra_param_name,
    value_or_values)
```

Stellt die Informationen in der gegebenen Variablen eines Parameters eines spezifizierten Materials wieder her (oder Zusatzparameter, siehe *“Zusätzliche Daten” auf Seite 180*). RGB Informationen werden in drei unterschiedlichen Variablen wiederhergestellt, Texturinformationen werden in folgenden Variablen wiederhergestellt: `file_name`, `width`, `height`, `mask`, `rotation_angle` bezogen auf die Texturdefinition. Alle anderen Parameterinformationen werden in einzelnen Variablen wiederhergestellt. Mögliche Parameternamen, die sich auf Parameter der Materialdefinition beziehen:

```
gs_mat_surface_rgb (surface R, G, B [0.0..1.0])
gs_mat_ambient (ambient coefficient [0.0..1.0])
gs_mat_diffuse (diffuse coefficient [0.0..1.0])
gs_mat_specular (specular coefficient [0.0..1.0])
gs_mat_transparent (transparent coefficient [0.0..1.0])
gs_mat_shining (shininess [0.0..100.0])
gs_mat_transp_att (transparency attenuation [0.0..4.0])
gs_mat_specular_rgb (specular color R, G, B [0.0..1.0])
gs_mat_emission_rgb (emission color R, G, B [0.0..1.0])
gs_mat_emission_att (emission attenuation [0.0..65.5])
gs_mat_fill_ind (fill index)
gs_mat_fillcolor_ind (fill color index)
gs_mat_texture (texture index)
```

Beispiel:

```
REQUEST{2} ("Material_info",
    "Brick-Face", "gs_mat_ambient", a)
```

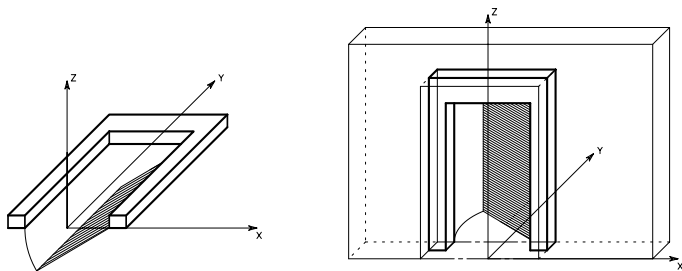
```
REQUEST{2} ("Material_info", 1,
  "gs_mat_surface_rgb", r, g, b)
REQUEST{2} ("Material_info",
  "Brick-Face", "gs_mat_texture",
  file_name, w, h, mask, alpha)
REQUEST{2} ("Material_info",
  "My-Material", "my_extra_parameter", e)
```

TÜREN UND FENSTER

In diesem Abschnitt werden die unterschiedlichen besonderen Optionen für die Erstellung von Fenster/Tür Bibliothekselementen behandelt.

GDL Programmierungs-Grundlagen

Nachdem Türen/Fenster in eine Wand eingefügt wurden, wird das Koordinatensystem dieser Bibliothekselemente gedreht, so dass die x-y-Ebene vertikal ist und die z-Achse senkrecht zur Wand steht. Der Ursprung befindet sich in der Mitte der Wandöffnung auf der Außenseite. Dadurch können Türen/Fenster durch Elemente in der x-y-Ebene leicht dargestellt werden. Siehe Illustrationen unten.



Wegen des besonderen Verhaltens dieser Bibliothekselemente wird das 2D-Symbol durch eine integrierte Projektion erstellt, die für Anwender normalerweise nicht zugänglich ist (Aufsicht im Drahtmodell). Das Symbol und die 3D-Form sind mit dem Ursprung der Türen/Fenster durch die untere (y) Mitte (x) des Rahmens verbunden. Die Position auf der z-Achse wurde nicht festgelegt, damit Fenster/Türen auf der z-Achse auch außerhalb der Wand liegen können.

Bezüglich dieser Regeln ermöglichen einige Hinweise die Konstruktion von einwandfrei funktionierenden Türen/Fenstern, :

- Konstruieren Sie Türen/Fenster im Grundrissfenster und visualisieren Sie es so, als ob Sie es von innen durch die Wand sehen würden, in die es eingesetzt wird.
- Nehmen Sie die Projektursprungsebene als Außenseite der Wand an.
- Elemente wie Fensterrahmen, die innerhalb der Wand liegen sollen, sollten über dem Höhennullpunkt liegen.
- Ein Türblatt, das sich nach außen öffnen soll, soll unterhalb des Nullpunktes liegen.

Erstellung von Bibliothekselementen des Türen/Fenster-Typs

Bei der Erstellung von Bibliothekselementen des Türen/Fenster-Typs stehen mehrere Optionen zur Verfügung, die verschiedene Probleme darstellen:

- Erstellung von rechteckigen Türen/Fenstern in geraden Wänden
- Erstellung von nicht rechteckigen Türen/Fenstern in geraden Wänden
- Erstellung von rechteckigen Türen/Fenstern in gekrümmten Wänden
- Erstellung von nicht rechteckigen Türen/Fenstern in gekrümmten Wänden

Rechteckige Türen/Fenster in geraden Wänden

Das ist die leichteste und direkteste Methode der Erstellung von Türen und Fenstern. Die Verwendung von einfachen GDL-Befehlen, wie PRISM_ oder RECT wird empfohlen.

Möchten Sie die Oberflächenmaterialien von Türen/Fenstern denen der Wand anpassen, soll die Bodenfläche der Elemente der Außenseite und die Deckfläche der Innenseite der Wand angepaßt sein. Sie können dies aus Ihren Scripts unter Verwendung der globalen Variablen G_, H_ und I_ einstellen, die die Materialien der Wand, in der die Türen/Fenster platziert werden, angeben. Im 2D-Script können die globalen Variablen E_, F_ und A nützlich sein, da die Stiftnummer der Wandkontur und -Schraffur, sowie die Indexnummer der Schraffur der Wand im Grundriß, in der die Türen/Fenster platziert werden, durch diese Variablen bestimmt werden. Im Falle von Mehrschichtwänden müssen Sie die entsprechenden globalen Variablen verwenden.

Siehe *“Verschiedenes” auf Seite 225* für mehr Details.

Die Bibliothek beinhaltet eine Gruppe von Türen/Fenster-Makros. Diese GDL-Scripts enthalten allgemeine Bauelemente, die durch viele Türen/Fenster in der Bibliothek verwendet werden. Es gibt Makros, die zur Generierung von allgemeinbenutzten Rahmen, Panelen und vielen anderen Typen der Türen/Fenster-Elemente benutzt werden. Öffnen Sie einige Bibliothekselemente vom Türen-/Fenster-Typ, um die Art der durch sie aufgerufenen Makros und den Typ der durch diese Makros generierten Elemente zu betrachten.

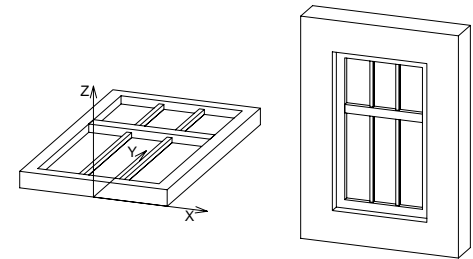
Beispiel:

```

A=0.9: B=1.5: C=0.1: D=0.08
E=0.08: F=0.9: G=0.03: H=3
PRISM 10,C,
  -A/2, 0, 15, A/2, 0, 15,
  A/2, B, 15, -A/2, B, 15,
  -A/2, 0, -1,
  -A/2+D, D, 15, A/2-D, D, 15,
  A/2-D, B-D, 15, -A/2+D, B-D, 15,
  -A/2+D, D, -1
ADD -A/2+D, F, 0
BRICK A-2*D, E, C
ADD -G/2, -F+D, C/2
GOSUB 1
ADDZ -G
GOSUB 1
DEL 2
MATERIAL "Glass"
ADD0, -F+D, C/2
RECT A-2*D, F-D
ADDY F-D+E
RECT A-2*D, B-F-E-D
END

1: FOR I=1 TO H-1
  ADDX (A-2*D)/3
  BLOCK G, F-D, G
  ADDY F+E-D
  BLOCK G, B-F-D-E, G
  DEL 1
NEXT I
DEL H-1
RETURN

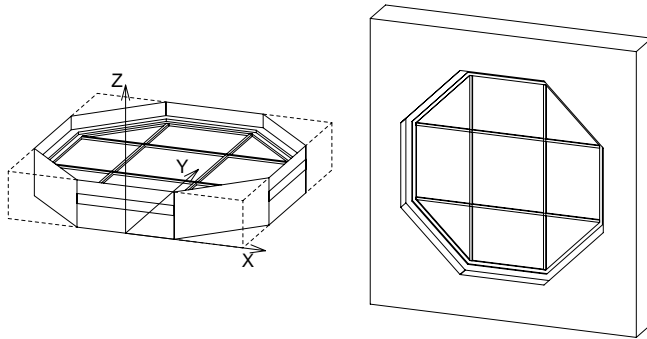
```



Nicht rechteckige Türen/Fenster in geraden Wänden

Bei der Arbeit mit Türen/Fenstern ist es wichtig zu wissen, daß ArchiCAD immer eine rechteckige Öffnung in der Wand ausschneidet. Die Größe dieser Öffnung wird durch die Parameter A und B des Bibliothekselementes vom Türen/Fenster-Typ bestimmt. Wenn die Türen/Fenster nicht rechteckig von der Höhenlage her sind, werden diese die ausgeschnittene rechteckige Öffnung nicht vollkommen ausfüllen. Eine Lösung dieses Problems ist, die Befehle WALLHOLE oder WALLNICHE zu benutzen, um die rechteckige Form festzulegen, die in die Wand geschnitten werden soll. Es gibt zwei Lösungen zu diesem Problem:

- Das 3D-Script soll Elemente beinhalten, die jene Wandelemente generieren, die die Öffnung zwischen dem Körper von Türen/Fenstern und den Kanten des rechteckigen Wandausschnittes ausfüllen. In diesem Falle muß die Sichtbarkeit der Kanten von diesen Füllungen besonders berücksichtigt werden.



- Mit dem Befehl WALLHOLE- oder WALLNICHE können Sie einen Polygonkörper bestimmen, der in der Wand auszuschneiden ist, wo die Türen/Fenster platziert werden sollen..

WALLHOLE

```
WALLHOLE n, status,
        x1, y1, mask1,
        ...
        xn, yn, maskn
        [, x, y, z]
```

n: Anzahl der Eckpunkte des Basispolygonzuges.

Status:

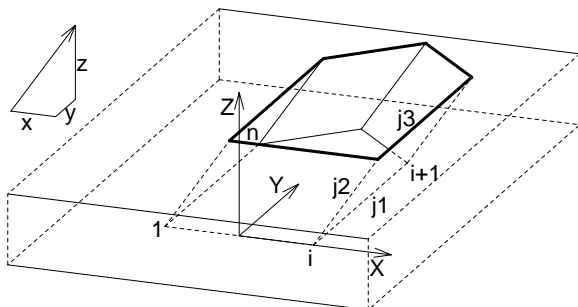
- 1: benutzen Sie die eigenen Attribute des Körpers für die generierten Polygone und Kanten
- 2: generierte Ausschnittspolygone werden als normale verwaltet

xi, yi: Koordinaten des Querschnittspolygons

maski: gleicht der CUTPOLYA-Anweisung:

$$\text{maski} = j1 + 2 * j2 + 4 * j3 + 64 * j7$$

x, y, z: optionaler Richtungsvektor (z-Achse von Türen/Fenstern wird grundeingestellt)

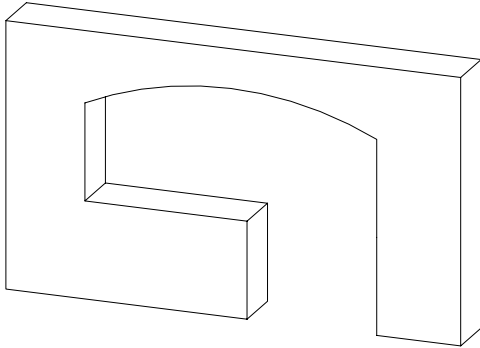


Dieser Befehl kann im 3D-Script von Türen/Fenstern zum Ausschneiden benutzerdefinierter Öffnung(en) in der Wand benutzt werden, wo sie platziert werden. Während der 3D-Generierung der aktuellen Wand wird das 3D-Script von allen ihren Türen/Fenstern ohne Generierung eines Modells interpretiert werden, um die WALLHOLE-Befehle zu sammeln. Falls diese vorhanden sind, wird das Programm die aktuelle Wand unter Verwendung einer unendlichen Röhre mit dem im Script definierten, polygonalen Querschnitt und Richtung ausschneiden. Es gibt eine Vielzahl von WALLHOLE-s für Türen/Fenster, so dass selbst für ein(e) Tür/Fenster mehrere Öffnungen auch überschneidend angelegt werden können. Wird mindestens ein WALLHOLE-Befehl in einem 3D-Script von Türen/Fenstern interpretiert, wird das Programm keine rechteckige Öffnung für diese Türen/Fenster generieren.

Anmerkung: Der 3D-Anschlag wird für benutzerdefinierte Öffnungen nicht automatisch generiert, Sie sollten dies aus dem Script generieren. Die auf diese Weise bestimmte Öffnung wird nur im 3D sichtbar sein, weil WALLHOLE-Befehle keine Auswirkung im 2D haben. Eine 2D-Darstellung kann, wenn nötig, beschrieben werden (verwendet mit nicht aktivem Umrahmen im Grundriß).

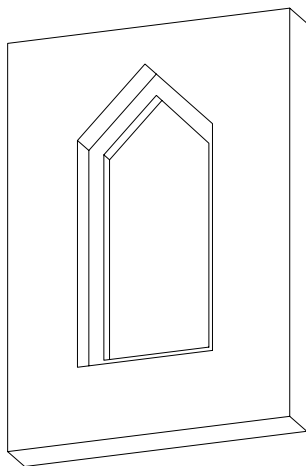
Die Verwendung von konvexen polygonalen Querschnitte wird empfohlen; werden konkave Polygone benutzt, kann dies zu eigenartigen Schattierungen und photorealistischen Darstellungen oder Ausschnittsfehlern führen. Konvexe Polygone können kombiniert werden, um konkave zu erstellen.

Beispiele:



```

RESOL 72
L1=2.7: L2=1.2: H1=2.1: H2=0.3: H3=0.9
R=( (L1/2)^2+H2^2)/(2*H2)
A=ATN((L1/2)/(R-H2))
WALLHOLE 5,1,
    -L1/2,H3,15,
    L1/2,H3,15,
    L1/2,H1-H2,13,
    0,H1-R,915,
    0,2*A,4015
WALLHOLE 4,1,
    L1/2-L2,0,15,
    L1/2,0,15,
    L1/2,H3,15,
    L1/2-L2,H3,15
    
```

```

WALLHOLE    5,1,
            -0.45, 0,    15,
            0.45, 0,    15,
            0.45, 1.5,  15,
            0,    1.95, 15,
            -0.45, 1.5,  15

```

```

PRISM       12,    0.1,
            -0.45, 0,    15,
            0.45, 0,    15,
            0.45, 1.5,  15,
            0,    1.95, 15,
            -0.45, 1.5,  15,
            -0.45, 0,    -1,
            -0.35, 0.1,  15,
            0.35, 0.1,  15,
            0.35, 1.45, 15,
            0,    1.80, 15,
            -0.35, 1.44, 15,
            -0.35, 0.1,  -1

```

WALLNICHE

WALLNICHE n, method, status,
 rx, ry, rz, d,
 x1, y1, mask1,
 ...
 xn, yn, maskn

Entspricht der Definition CUTFORM.

method: Steuert die Form des Schnittkörpers

1: prismenförmig

2: pyramidenförmig

3: keilförmiger Schnittkörper. Die Richtung der Deckkante des Keils ist parallel zur Y-Achse und ihre Position liegt in rx, ry, rz (ry wird ignoriert.)

status: Steuert den Umfang des Schnittkörpers und die Behandlung der generierten Schnittpolygone.

status = j1 + 2*j2 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8

j1: verwendet die Attribute des Körpers der generierten Polygone und Kanten

j2: generierte Schnittpolygone werden als normale Polygone behandelt

j4, j5: definiert das Limit des Schnitts:

j4 = 0 und j5 = 0: endlicher Schnitt

j4 = 0 und j5 = 1: halb-unendlicher Schnitt

j4 = 1 und j5 = 1: unendlicher Schnitt

j6: generiert eine Boole'sche Schnittmenge mit dem Schnittkörper statt eines Boole'schen Unterschieds. (kann nur mit dem Befehl CUTFORM verwendet werden)

j7 : Kanten, generiert durch den Körperboden werden unsichtbar

j8 : Kanten, generiert durch die Körperspitze werden unsichtbar

rx, ry, rz: definiert die Ausrichtung des Schnitts, wenn der Schnitt prismenförmig ist, oder die Spitze der Pyramide, wenn die Schnittmethode pyramidenförmig ist.

d: definiert den Abstand entlang rx, ry, rz bis zum Ende des Schnitts. Wenn der Schnitt unendlich ist, hat dieser Parameter keine Auswirkung. Handelt es sich um einen endlichen Schnitt, liegt der Beginn des Schnittkörpers am lokalen Koordinatensystem und der Körper endet in einem Abstand von d entlang der von rx, ry, rz definierten Richtung

Ist der Schnitt halb-endlich, liegt der Beginn des Schnittkörpers in einem Abstand von d entlang der von rx , ry , rz definierten Richtung, und die Ausrichtung des halb-unendlichen Schnitts liegt in der Gegenrichtung zu der von rx , ry , rz definierten Richtung.

mask: Definiert die Sichtbarkeit der Kanten des Schnittkörpers:

$$\text{mask}_i = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 64*j_7$$

j1: Das Polygon erzeugt eine sichtbare Kante beim Eintritt in den geschnittenen Körper

j2: Die Längskante der Schnittform ist sichtbar

j3: Das Polygon erzeugt eine sichtbare Kante beim Verlassen des geschnittenen Körpers

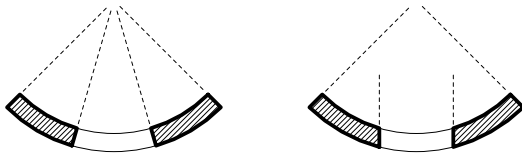
j4: Die untere Kante der Schnittform ist sichtbar

j5: Die obere Kante der Schnittform ist sichtbar

j7: Steuert die vom Blickpunkt abhängige Sichtbarkeit der Längskante

Rechteckige Türen/Fenster in gekrümmten Wänden

Werden Türen/Fenster in gekrümmten Wänden angeordnet, können die Seiten der Öffnung variieren, wie in der Abbildung unten dargestellt.



Die Öffnung in der Wand an der linken Seite wird erstellt, wenn das Programm die Öffnung für die Türen/Fenster automatisch ausschneidet. In diesem Falle werden die Seiten in radialer Richtung gestellt. Im Bild rechts wird die Öffnung durch den WALLHOLE-Befehl im 3D-Script des Objektes vom Türen-/Fenster-Typ ausgeschnitten. Das Objekt selbst muss unter Berücksichtigung dieser Faktoren beschrieben werden.

Außerdem muss berücksichtigt werden, ob die in der gekrümmten Wand platzierten Türen/Fenster gerade oder gekrümmt sind.



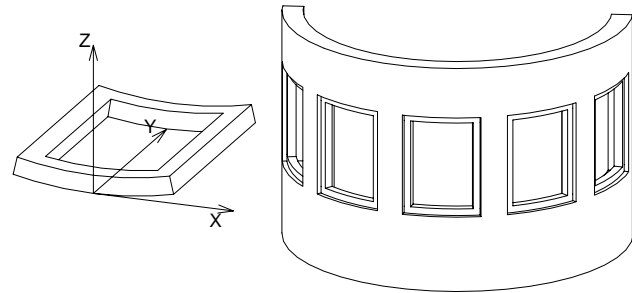
Im Falle gerader Türen/Fenster, wie links oben, sind die Dicke und Breite des Objektes von der Dicke der Wand abhängig, weil das Objekt über einer bestimmten Größe nicht mehr in der Wand platziert würde. Werden gekrümmte Türen/Fenster benutzt, tritt dieses Problem nicht auf.

Beispiel:

```

RESOL 72
ROTX -90
MULY -1
C= 0.12: Z=(360*A)/(2*R_*PI)
Y= (360*C)/(2*R_*PI)
A1= 270+Z/2: A2=270-Z/2
GOSUB 1
ADDZ B
MULZ -1
GOSUB 1
DEL 2
ADDZ C
GOSUB 2
MULX -1
GOSUB 2
END
1:
PRISM_ 9, C,
  COS(A2)*R_, SIN(A2)*R_+R_, 11,
  COS(A2+Y)*R_, SIN(A2+Y)*R_+R_, 13,
  0, R_, 900,
  0, Z-2*Y, 4009,
  COS(A1)*R_, SIN(A1)*R_+R_, 11,
  COS(A1)*(R_-0.1), SIN(A1)*(R_-0.1)+R_, 11,
  COS(A1-Y)*(R_-0.1), SIN(A1-Y)*(R_-0.1)+R_, 13,
  0, -(Z-2*Y), 4009,
  COS(A2)*(R_-0.1), SIN(A2)*(R_-0.1)+R_, 11
RETURN
2:
PRISM_ 4, B-2*C,
  COS(A2)*R_, SIN(A2)*R_+R_, 10,
  COS(A2+Y)*R_, SIN(A2+Y)*R_+R_, 15,
  COS(A2+Y)*(R_-0.1), SIN(A2+Y)*(R_-0.1)+R_, 10,
  COS(A2)*(R_-0.1), SIN(A2)*(R_-0.1)+R_, 10
RETURN

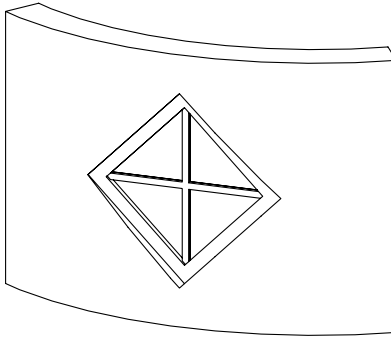
```



Nicht rechteckige Türen/Fenster in gerümmten Wänden

Die allgemeinen Richtlinien für rechteckige Türen/Fenster in gekrümmten Wänden gelten auch hier.

Beispiel:



```

C=0.1: D=0.025
Z=A/2-SQR(2)*C: Y=A/2-SQR(2)*C-D
ADDY A/2
WALLHOLE 4, 1,
    0, -A/2, 15,
    A/2, 0, 15,
    0, A/2, 15,
    -A/2, 0, 15
PRISM_ 10, 0.1,
    0, -A/2, 15,
    A/2, 0, 15,
    0, A/2, 15,
    -A/2, 0, 15,
    0, -A/2, -1,
    0, -Z, 15,
    Z, 0, 15,
    0, Z, 15,
    -Z, 0, 15,
    0, -Z, -1
ADDZ 0.02
GOSUB 1
ADDZ 0.03
GOSUB 1
ADDY -Z
  
```

```

SET MATERIAL "Glass"
ROTZ 45
RECT SQR(2)*Z, SQR(2)*Z
END
1:
PRISM_ 16, 0.03,
      0, -Z, 15,
      D, -Y, 15,
      D, -D, 15,
      Y, -D, 15,
      Z, 0, 15,
      Z, D, 15,
      D, D, 15,
      D, Y, 15,
      0, Z, 15,
      -D, Y, 15,
      -D, D, 15,
      -Y, D, 15,
      -Z, 0, 15,
      -Y, -D, 15,
      -D, -D, 15,
      -D, -Y, 15
RETURN

```

GRAFISCHE ERZEUGUNG VON GDL-OBJEKTEN IM GRUNDRIß

Wenn Sie den Grundriß als GDL-Script oder Bibliothekselement abspeichern, wird es auf die folgenden GDL-Elemente hinauslaufen. Sie können diese GDL-Scripts als Muster für Ihre eigenen Bibliothekselemente verwenden.

BEFEHLE

Häufige Befehle

Operatoren, Funktionen

FOR NEXT
DO, WHILE, ENDWHILE
REPEAT UNTIL
IF, THEN, ELSE, ENDIF
GOTO
GOSUB
RETURN
END
EXIT
PUT
GET
USE
NSP
CALL, PARAMETERS
PRINT
OPEN
INPUT
VARTYPE
OUTPUT
CLOSE
DIM
BREAKPOINT

Befehle für den 3D-Bereich

ADDX, ADDY, ADDZ
ADD
MULX, MUY, MULZ
MUL
ROTX, ROTY, ROTZ
ROT

Reservierte Befehle

Die Schlüsselwörter in dieser Liste sind aus Kompatibilitätsgründen reserviert oder wurden nicht publiziert.

BAS
BOX
FILTER
GDLBIN
LIN
LINE
NOD
NODE
ORIGO
PARS
PLOTMAKER
PLOTTER
RECT_
SFLINE
TET
TETRA
TRI
WALL_
VOCA_
UI_OK
UI_CANCEL

XFORM

HOTSPOT
LIN_
RECT
POLY, POLY_
PLANE, PLANE_
CIRCLE

- Bogen

BINARY

BLOCK, BRICK

CUTPLANE

CYLIND

CUTSHAPE

SPHERE

CUTPOLY

ELLIPS

CUTPOLYA

CONE

CUTEND

PRISM, PRISM_, CPRISM_, BPRISM_, FPRISM_,
SPRISM_

DEFINE MATERIAL

SLAB, SLAB_, CSLAB_

DEFINE TEXTURE

CWALL_, BWALL_, XWALL_

[SET] MATERIAL

WALLHOLE

SHADOW

BEAM

Modell

CROOF_

ARMC

SECT_FILL

ARME

ELBOW

EXTRUDE

PYRAMID

REVOLVE

RULED

SWEEP

TUBE, TUBEA

COONS

MESH

MASS

LIGHT

PICTURE

TEXT

VERT, TEVE

VECT

EDGE

PGON, PIPG

COOR

BODY

BASE

Befehle für den 2D-Bereich

ADD2
 MUL2
 ROT2

 HOTSPOT2
 LINE2
 RECT2
 POLY2, POLY2_, POLY2_A, POLY2_B
 ARC2
 CIRCLE2
 SPLINE2, SPLINE2A

 PICTURE2
 TEXT2
 FRAGMENT2

 PROJECT2

 DEFINE FILL
 DEFINE FILLA
 DEFINE LINE_TYPE

 [SET] FILL
 [SET] LINE_TYPE
 DRAWINDEX
 DRAWING2
 DRAWING3

Befehle für den 2D- und 3D-Bereich

DEL
 [LET]
 Radius
 RESOL
 TOLER
 PEN
 DEFINE STYLE
 [SET] STYLE

Nicht geometrische Scripts

Eigenschaften-Script

DATABASE_SET
 DESCRIPTOR
 COMPONENT
 REF
 SURFACE3D
 VOLUME3D

 POSITION

 WALLS
 COLUMNS
 BEAMS
 DOORS
 WINDOWS
 OBJECTS
 PITCHED_ROOFS
 HIP_ROOFS
 LIGHTS
 HATCHES
 ROOMS
 MESHES

 DRAWING
 BINARYPROP

Parameter-Text

VALUES

PARAMETERS

LOCK

Interface-Script

UI_DIALOG

UI_PAGE

UI_BUTTON

UI_PREV

UI_NEXT

UI_GROUPBOX

UI_SEPARATOR

UI_PICT

UI_STYLE

UI_OUTFIELD

UI_INFIELD

Alphabetische Liste der aktuellen GDL-Befehle

A

ABS (x) ergibt den absoluten Wert von x (ganzzahlig, falls x ganzzahlig, sonst real).

ACS (x) ergibt den Bogen-Cosinus von x. ($-1.0 \leq x \leq 1.0$; $0^\circ \leq \text{ASN}(x) \leq 180^\circ$).

ADD dx, dy, dz

ADD2 x, y

ADDGROUP (g_expr1, g_expr2)

ADDX dx

ADDY dy

ADDZ dz

AND (oder &) UND-Verknüpfung 6. Priorität

ARC r, alpha, beta

ARC2 x, y, r, alpha, beta

ARMC r1, r2, l, h, d, alpha

ARME l, r1, r2, h, d

ASN (x) ergibt den Bogen-Sinus von x. ($-1.0 \leq x \leq 1.0$; $-90^\circ \leq \text{ASN}(x) \leq 90^\circ$).

ATN (x) ergibt die Bogen-Tangens von x. ($-90^\circ \leq \text{ATN}(x) \leq 90^\circ$).

B

BASE

BEAM left_material, right_material, vertical_material, top_material, bottom_material,
height, x1, x2, x3, x4,
y1, y2, y3, y4, t,
mask1, mask2, mask3, mask4

BINARY mode [, section]

BINARYPROP

BITSET (x, b [, expr])

BITTEST (x, b)

BLOCK a, b, c

BODY status

BPRISM _top_material, bottom_material, side_material,
n, h, radius, x1, y1, s1, ... xn, yn, sn

BREAKPOINT expression

BRICK a, b, c

BWALL _left_material, right_material, side_material,
height, x1, x2, x3, x4, t, radius,
mask1, mask2, mask3, mask4,
n,
x_start1, y_low1, x_end1, y_high1, frame_shown1,
...
x_startn, y_lown, x_endn, y_highn, frame_shownn,
m,
a1, b1, c1, d1,
...
am, bm, cm, dm

C

CALL macro_name **PARAMETERS ALL**

CALL macro_name_string [,parameter_list]

CALL macro_name_string **PARAMETERS** [name1=value1 , ... namen=valuen]

CEIL (x) Ergibt den kleinsten Ganzzahlwert, der nicht
kleiner ist als x (immer ganzzahlig). (e.g., CEIL(1.23) = 2; CEIL (-1.9) = -1).

CIRCLE r

CIRCLE2 x, y, r 117

CLOSE Kanal

COMPONENT name, quantity, unit [, proportional_with, code, keycode, unitcode]

CONE h, r1, r2, alpha1, alpha2

COONS *n*, *m*, *mask*,
x11, *y11*, *z11*, ... *x1n*, *y1n*, *z1n*,
x21, *y21*, *z21*, ... *x2n*, *y2n*, *z2n*,
x31, *y31*, *z31*, ... *x3m*, *y3m*, *z3m*,
x41, *y41*, *z41*, ... *x4m*, *y4m*, *z4m*94

COOR *wrap*, *vert1*, *vert2*, *vert3*, *vert4*

COS (*x*) ergibt den Cosinus von *x*.

CPRISM *top_material*, *bottom_material*, *side_material*,
n, *h*, *x1*, *y1*, *s1*, ... *xn*, *yn*, *sn*

CROOF *top_material*, *bottom_material*, *side_material*,
n, *xb*, *yb*, *xe*, *ye*, *height*, *angle*, *thickness*,
x1, *y1*, *alpha1*, *s1*,
...,
xn, *yn*, *alphan*, *sn*

CSLAB *top_material*, *bottom_material*, *side_material*,
n, *h*, *x1*, *y1*, *z1*, *s1*, ... *xn*, *yn*, *zn*, *sn*

CUTFORM *n*, *method*, *status*,
rx, *ry*, *rz*, *d*,
x1, *y1*, *mask1*,
...
xn, *yn*, *maskn*

CUTPLANE [*x*, *y*, *z* [, *side*]]
[statement1
...
statementn]
CUTEND

CUTPLANE *angle*
[statement1
...
statementn]
CUTEND

```

CUTPOLY n,
    x1, y1, ... xn, yn
    [, x, y, z]
    [statement1
    statement2
    ...
    statementn]
CUTEND

CUTPOLYA n, status, d,
    x1, y1, mask1, ... xn, yn, maskn [,
    x, y, z]
    [statement1
    statement2
    ...
    statementn]

CUTSHAPE d
    [statement1
    statement2
    ...
    statementn]

CWALL _left_material, right_material, side_material,
    height, x1, x2, x3, x4, t,
    mask1, mask2, mask3, mask4,
    n,
    x_start1, y_low1, x_end1, y_high1, frame_shown1,
    ...
    x_startn, y_lown, x_endn, y_highn, frame_shownn,
    m,
    a1, b1, c1, d1,
    ...
    am, bm, cm, dm

CYLIND h, r

```

D

```
DATABASE_SET set_name [descriptor_name, component_name, unit_name, key_name, criteria_name,  
    list_set_name]  
DEFINE EMPTY_FILL name [FILLTYPES_MASK fill_types]  
DEFINE FILL name [FILLTYPES_MASK fill_types,] pattern1, pattern2, pattern3, pattern4,  
    pattern5, pattern6, pattern7, pattern8,  
    spacing, angle, n,  
    frequency1, direction1, offset_x1, offset_y1, m1,  
    length11, ... length1m,  
    ...  
    frequencyn, directionn, offset_xn,  
    lengthn1, ... lengthnm  
DEFINE FILL parameters [ADDITIONAL_DATA name1 = value1, name2 = value2, ...]  
DEFINE FILL_A parameters [ADDITIONAL_DATA name1 = value1, name2 = value2, ...]  
DEFINE FILL_A name [FILLTYPES_MASK fill_types,] pattern1, pattern2, pattern3, pattern4,  
    pattern5, pattern6, pattern7, pattern8, spacing_x, spacing_y, angle, n, frequency1,  
    directional_offset1, direction1,  
    offset_x1, offset_y1, m1, length11,  
    ...  
    length1m, ... frequencyn,  
    directional_offsetn, directionn,  
    offset_xn, offset_yn, mn,  
    lengthn1, ... lengthnm  
DEFINE LINE_TYPE name spacing, n,  
    length1, ... lengthn  
DEFINE LINE_TYPE parameters [ADDITIONAL_DATA name1 = value1, name2 = value2, ...]  
DEFINE MATERIAL name BASED_ON orig_name PARAMETERS name1 = expr1 [,  
    ...][ADDITIONAL_DATA name1 = expr1 [, ...]]  
DEFINE MATERIAL name type, parameter1,  
    parameter2, ... parametern  
DEFINE MATERIAL parameters [ADDITIONAL_DATA name1 = value1, name2 = value2, ...]  
DEFINE SOLID_FILL name [FILLTYPES_MASK fill_types]  
DEFINE STYLE name font_family, size, anchor, face_code
```



```

DEFINE STYLE{2} name font_family, size, face_code

DEFINE SYMBOL_FILL name [FILLTYPES_MASK fill_types,]
    pat1, pat2, pat3, pat4, pat5, pat6, pat7, pat8,
    spacingx1, spacingy1,
    spacingx2, spacingy2, angle,
    scaling1, scaling2, macro_name PARAMETERS [name1 = value1, ... namen = valuen]

DEFINE SYMBOL_FILL parameters [ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE SYMBOL_LINE name dash, gap, macro_name PARAMETERS [name1 = value1, ... namen = valuen]

DEFINE SYMBOL_LINE parameters [ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE TEXTURE name expression, x, y, mask, angle

DEL n [, begin_with]

DEL TOP

DESCRIPTOR name [,code, keycode]

DIM var1[dim_1], var2[dim_1][dim_2], var3[ ],
    var4[ ][ ], var5[dim_1][ ],
    var5[ ][dim_2]

DO
    [statement1
    statement2
    ...
    statementn]

DRAWINDEX number

DRAWING2 [expression]

DRAWING3 projection_code, angle, method

DRAWING3{2} projection_code, angle, method [,backgroundColor, origoX, origoY,
    fillldirection]

DRAWING

```

E

EDGE vert1, vert2, pgon1, pgon2, status

ELBOW r1, alpha, r2

ELLIPS h, r

END

ENDGROUP

EXIT

EXOR (or @) Logical exclusive or precedence 8

EXP (x) Ergibt die zehnte Potenz von e (e = 2.7182818).

EXTRUDE n, dx, dy, dz, mask, x1, y1, s1,
..., xn, yn, sn

F

FILE_DEPENDENCE "name1" [, "name2", ...]

FOR variable_name = initial_value TO end_value [STEP step_value]

FPRISM top_material, bottom_material, side_material, hill_material,
n, thickness, angle, hill_height,
x1, y1, s1,
...
xn, yn, sn

FRA (x) Ergibt den Bruchzahlenwert von x (e.g., FRA(1.23) = 0.23, FRA(-1.23) = 0.77).

FRAGMENT2 fragment_index,
use_current_attributes_flag

FRAGMENT2 ALL, use_current_attributes_flag

G

GET (n)

GOSUB label

GOTO label

GROUP "name"

H

HIDEPARAMETER name1 [, name2, ..., namen]

HOTARC2 x, y, r, startangle, endangle

HOTLINE2 x1, y1, x2, y2

HOTSPOT x, y, z [, unID [, paramReference, flags] [, displayParam]]

HOTSPOT x, y, z [, unID [, paramReference, flags] [, displayParam]]

FPRISM top_material, bottom_material, side_
material, hill_material,
n, thickness, angle, hill_height,
x1, y1, s1,
...
xn, yn, sn

I

IF condition **GOSUB** label

IF condition **GOTO** label

IF condition **THEN** label

IND (TEXTURE, name_string)

IND (TEXTURE, name_string)

IND (TEXTURE, name_string)

IND (TEXTURE, name_string)

IND (TEXTURE, name_string)

INPUT (channel, recordID, fieldID, variable1 [, variable2,...])

INT (x) Ergibt den ganzzahligen Teil von x (immer ganzzahlig). (e.g., INT(1.23) = 1, INT(-1.23) = -2).

ISECTGROUP (g_expr1, g_expr2)

ISECTLINES (g_expr1, g_expr2)

K

KILLGROUP g_expr

L

[LET] varnam = n

LGT (x) ergibt den Basis 10 Logarithmus von x.

LIGHT red, green, blue, shadow,
radius, alpha, beta, angle_falloff,
distance1, distance2,
distance_falloff [ADDITIONAL_DATA name1 = value1,
name2 = value2, ...]

LIN_ x1, y1, z1, x2, y2, z2

LINE_PROPERTY expr

LINE2 x1, y1, x2, y2

LOCK name1 [, name2, ..., namen]

LOG (x) ergibt den natürlichen Logarithmus von x.

M

MASS top_material, bottom_material, side_material,
n, m, mask, h,
x1, y1, z1, s1,
...
xn, yn, zn, sn,
xn+1, yn+1, zn+1, sn+1,
...
xn+m, yn+m, zn+m, sn+m

MAX (x1,x2, ... xn) Returns the largest of an unlimited number of arguments.

MESH a, b, m, n, mask,
 z11, z12, ... z1m,
 z21, z22, ... z2m,
 ...
 zn1, zn2, ... znm

MIN (x1,x2, ... xn) ergibt das kleinste einer unbegrenzten Anzahl von Argumenten.

MODEL SOLID

MODEL SURFACE

MODEL WIRE

MUL mx, my, mz

MUL2 x, y

MULX mx

MULY my

MULZ mz

N

NEXT variable_name

NOT (x) Ergibt falsch (=0.0), wenn x wahr ist (<0.0) und wahr (=1.0), wenn x falsch (=0.0) ist. (Logische Verneinung).

NSP

NTR

O

OPEN (filter, filename, parameter_string)

OR (oder |) einschließlich logischer ODER-Verknüpfung, 7. Priorität

OUTPUT (ch, recordID, fieldID, var1, var2...)

OUTPUT channel, recordID, fieldID, expression1 [, expression2, ...]

P

PARAGRAPH name alignment, firstline_indent,
left_indent, right_indent, line_spacing [,
tab_size1, ...]
[PEN index]
[[SET] STYLE style1]
[[SET] MATERIAL index]
'string1'
'string2'
...
'string n'
[PEN index]
[[SET] STYLE style2]
[[SET] MATERIAL index]
'string1'
'string2'
...
'string n'
...

PARAMETERS name1 = expression1 [,
name2 = expression2, ...,
namen = expressionn]

PEN n

PGON n, vect, status, edge1, edge2, ... edgen

PI ergibt die Ludolph Konstante. (p = 3.1415926...).

PICTURE expression, a, b, mask

PICTURE expression, a, b, mask

PICTURE2{2} expression, a, b, mask

PIPG expression, a, b, mask, n, vect,
status,
edge1, edge2, ... edgen

PLACEGROUP g_expr

PLANE n, x1, y1, z1, ... xn, yn, zn

```

PLANE_ n, x1, y1, z1, s1, ... xn, yn, zn, sn
POLY n, x1, y1, ... xn, yn
POLY_ n, x1, y1, s1, ... xn, yn, sn61
PRISM n, h, x1, y1, ... xn, yn
PRISM_ n, h, x1, y1, s1, ... xn, yn, sn
POLY2_A n, frame_fill, fill_pen,
    x1, y1, s1, ..., xn, yn, sn
POLY2_B n, frame_fill, fill_pen,
    fill_background_pen,
    x1, y1, s1, ..., xn, yn, sn
POLY2_B{2} n, frame_fill, fill_pen,
    fill_background_pen,
    fillOrigoX, fillOrigoY,
    fillAngle,
    x1, y1, s1, ..., xn, yn, sn
POSITION position_keyword
PRINT expression [, expression, ...]
PRISM n, h, x1, y1, ... xn, yn
PRISM_ n, h, x1, y1, s1, ... xn, yn, sn
PROJECT2 projection_code, angle, method
PROJECT2{2} projection_code, angle,method [,backgroundColor, fillOrigoX,
    fillOrigoY, fillldirection]
PUT expression [ , expression, ...]
PYRAMID n, h, mask, x1, y1, s1, ... xn, yn, sn

```

R

```
RADIUS radius_min, radius_max
RECT a, b
RECT2 x1, y1, x2, y2 114
REF COMPONENT code [, keycode [, numeric_expression]]
REF DESCRIPTOR code [, keycode]
CUTSHAPE d
    [statement1
    statement2
    ...
    statementn]
REQ (parameter_string)
REQ (parameter_string)
REQUEST ("ANCESTRY_INFO", expr, name [,
    guid, parent_name1, parent_guid1,
    ...,
    parent_namen, parent_guidn)
REQUEST ("Angular_dimension", "",
    format_string)
REQUEST ("Angular_length_dimension", ""
    format_string)
REQUEST ("Area_dimension", ""
    format_string)
REQUEST ("ASSOCEL_PROPERTIES ", parameter_string, nr_data, data)
REQUEST ("ASSOCLP_PARVALUE", expr, name_or_index, type, flags, dim1, dim2, values)
REQUEST ("Calculation_angle_unit", "",
    format_string)
REQUEST ("Calculation_area_unit", "",
    format_string)
REQUEST ("Calculation_length_unit", "",
    format_string)
```



```

REQUEST ("Calculation_volume_unit", "",
        format_string)
REQUEST ("Constr_Fills_display", "", optionVal)
REQUEST ("Elevation_dimension", "",
        format_string)
REQUEST ("Height_of_style", name,
        height [, descent, leading])
REQUEST ("Level_dimension", "",
        format_string)
REQUEST ("matching_properties", type, name1, name2, ...)
dummy REQUEST (Name_of_Listed, , name)
REQUEST ("Radial_dimension", "",
        format_string)
REQUEST ("REFERENCE_LEVEL_DATA", "",
        name1, elev1, name2, elev2, name3, elev3)
REQUEST ("Sill_height_dimension", "",
        format_string)
REQUEST ("Style_info", name, fontname [, size, anchor, face_or_slant])
REQUEST ("Window_door_dimension", "",
        format_string)
REQUEST ("window_door_zone_relev", " ", out_direction)
REQUEST ("Working_angle_unit", "", format_string)
REQUEST ("Working_length_unit", "", format_string)
REQUEST ("Zone_relations", "", category_name, code, name, number [, category_name2,
        code2, name2, number2])
REQUEST ('TEXTBLOCK_INFO',
        textblock_name, width, height)
REQUEST (extension_name, parameter_string, variable1, variable2, ...)
REQUEST (question_name, name | index, variable1 [, variable2,...])
REQUEST (question_name, name | index, variable1 [, variable2,...])

```

```
REQUEST{2} ("Material_info", name_or_index,  
            extra_param_name,  
            value_or_values)  
  
REQUEST{2} ("Material_info", name_or_index,  
            param_name, value_or_values)  
  
RESOL n  
  
RETURN  
  
REVOLVE n, alpha, mask, x1, y1, s1, ... xn, yn, sn  
  
RIGHTTEXT x, y,  
            height, 0, textblock_name  
  
RIGHTTEXT2 x, y, textblock_name  
  
RND (x) Ergibt einen Zufallswert zwischen  
        0.0 und x (x >0.0).  
  
ROT x, y, z, alpha  
  
ROT2 alpha  
  
ROTX alphax  
  
ROTY alphay  
  
ROTZ alphaz  
  
ROUND_INT (x)  
  
RULED n, mask,  
        u1, v1, s1, ... un, vn, sn,  
        x1, y1, z1, ... xn, yn, zn  
  
RULED{2} n, mask,  
        u1, v1, s1, ... un, vn, sn,  
        x1, y1, z1, ... xn, yn, zn
```

S

```
[SET] FILL index  
[SET] FILL name_string  
[SET] LINE_TYPE index  
[SET] LINE_TYPE name_string
```

[SET] MATERIAL index
[SET] MATERIAL name_string
[SET] STYLE index
[SET] STYLE name_string
SECT_FILL fill, fill_background_pen,
 fill_pen, contour_pen
SGN (x) Ergibt +1 wenn x positiv ist, -1 wenn x negativ ist, ansonsten 0.
SHADOW keyword_1[, keyword_2]
SIN (x) Returns the sine of x.
SLAB n, h, x1, y1, z1, ... xn, yn, zn
SLAB_ n, h, x1, y1, z1, s1, ... xn, yn, zn, sn
SPHERE r
SPLINE2 n, status, x1, y1,
 angle1, ..., xn, yn, anglen
SPLINE2A n, status, x1, y1, angle1, length_previous1, length_next1,
 ...
 xn, yn, anglen, length_previousn,
 length_nextn 119
SPLIT (string, format, variable1 [, variable2, ..., variablen])
SPRISM_ top_material, bottom_material, side_material,
 n, xb, yb, xe, ye, h, angle,
 x1, y1, s1, ... xn, yn, sn
SQR (x) ergibt die Quadratwurzel von x (immer real).
STR (numeric_expression, length, fractions)
STR (format_string, numeric_expression)
STR{2} (format_string, numeric_expression [, extra_accuracy_string])
STRLEN (string_expression)
STRSTR (string_expression1, string_expression2)
STRSUB (string_expression, start_position, characters_number)

STW (string_expression)

SUBGROUP (g_expr1, g_expr2)

SURFACE3D ()

SWEEP n, m, alpha, scale, mask,
 u1, v1, s1, ... un, vn, sn,
 x1, y1, z1, ... xm, ym, zm

SWEEPGROUP (g_expr, x, y, z)

T

TAN (x) ergibt den Tangens von x.

TEVE x, y, z, u, v

TEXT d, 0, expression

TEXT2 x, y, expression

TEXTBLOCK name width, anchor, angle, width_factor, charspace_factor, fixed_height,
 'string_expr1' [, 'string_expr2', ...]

TOLER d

TUBE n, m, mask,
 u1, w1, s1,
 ...
 un, wn, sn,
 x1, y1, z1, angle1,
 ...
 xm, ym, zm, anglem

TUBEA n, m, mask,
 u1, w1, s1,
 ...
 un, wn, sn,
 x1, y1, z1,
 ...
 xm, ym, zm

U

UI_BUTTON type, text, x, y, width, height
UI_DIALOG title [, size_x, size_y]
UI_GROUPBOX text, x, y, width, height
UI_INFIELD "name", x, y, width, height [,
 version_flag, picture_name,
 images_number,
 rows_number, cell_x, cell_y,
 image_x, image_y,
 expression_image1, text1,
 ...,
 expression_imagen, textn]
UI_OUTFIELD expression,x,y,width,height
UI_PAGE page_number
UI_PICT expression, x, y [,width, height]
UI_SEPARATOR x1, y1, x2, y2
UI_STYLE fontsize, face_code
USE (n)

V

VALUES "fillparam_name" [**FILLTYPES_MASK** fill_types,] value_definition1 [, value_definition2,
 ...]
VARDIM1 (expr)
VARDIM2 (expr)
VARTYPE (expression)
VERT x, y, z
VERT x, y, z
VOLUME3D ()

W

```
WALLHOLE n, status,  
    x1, y1, mask1,  
    ...  
    xn, yn, maskn  
    [, x, y, z]
```

```
CUTFORM n, method, status,  
    rx, ry, rz, d,  
    x1, y1, mask1,  
    ...  
    xn, yn, maskn
```

```
WHILE condition DO  
    [statement1  
    statement2  
    ...  
    statementn]
```

X

```

XFORM a11, a12, a13, a14,
        a21, a22, a23, a24,
        a31, a32, a33, a34

XWALL _left_material, right_material, vertical_material, horizontal_material,
        height, x1, x2, x3, x4,
        y1, y2, y3, y4,
        t, radius,
        log_height, log_offset,
        mask1, mask2, mask3, mask4,
        n,
        x_start1, y_low1, x_end1, y_high1,
        frame_shown1,
        ...
        x_startn, y_lown, x_endn, y_highn,
        frame_shownn,
        m,
        a1, b1, c1, d1,
        ...
        am, bm, cm, dm,
        status

XWALL_{2} left_material, right_material, vertical_material, horizontal_material,
        height, x1, x2, x3, x4,
        y1, y2, y3, y4,
        t, radius,
        log_height, log_offset,
        mask1, mask2, mask3, mask4,
        n,
        x_start1, y_low1, x_end1, y_high1,
        sill_depth1, frame_shown1,
        ...
        x_startn, y_lown, x_endn, y_highn,
        sill_depthn, frame_shownn,
        m,
        a1, b1, c1, d1,
        ...
        am, bm, cm, dm,
        status

```

Konventionen für Parameternamen

Wegen der Subtype Hierarchie erben die untergeordneten Bibliothekselemente automatisch alle Parameter der übergeordneten Elemente. (Mehr über Subtypes und Parameter finden Sie im ArchiCAD Benutzerhandbuch). Die Parameter werden durch ihren Namen angegeben, geerbte und Original-Parameter können daher den gleichen Namen haben. Der Bibliothekenersteller ist dafür zuständig, durch Verwendung beschreibender Parameternamen und Präfixe mit abgekürzten Bibliothekselementnamen Konflikte zu vermeiden.

Für Handler-Parameter und benutzerdefinierte Parameter hat Graphisoft in seinen Bibliotheken bestimmte Namenskonventionen für Parameter eingeführt.

Anmerkung: Handlers erweitern Bibliothekselemente um zusätzliche Funktionen (z. B. Durchbrüche in Wänden für Türen und Fenster). Parameternamen mit dem Präfix **ac_** sind für spezielle, den ArchiCAD-Handlers zugeordnete Parameter reserviert (z. B. **ac_corner_window**). Die vollständige Liste dieser Namen finden Sie in den Standard ArchiCAD Bibliotheks-Subtypvorlagen.

Standard Graphisoft-Parameternamen sind mit dem Präfix **gs_** gekennzeichnet (z. B. **gs_frame_pen**). Bitte überprüfen Sie zur Referenz alle AC-Bibliothekselemente. Verwenden Sie diese Parameter in Ihren GDL-Skripts, um eine umfassende Kompatibilität mit Graphisoft-Bibliotheken sicherzustellen.

FM_ ist reserviert für ArchiFM (z. B. **FM_Type**) und **HVAC_** ist HVAC für ArchiCAD-Parameter zugeordnet (z. B. **HVAC_Manufacturer**).

GDL DATA I/O ADD-ON

Das Add-On “GDL Data In/Out” ermöglicht den Zugang zu einer einfachen Datenbank durch GDL-Befehle.

Im Übrigen entspricht dieses Add-On dem “Text GDL In/Out” Add-On.

Beschreibung der Datenbank

Die Datenbank ist eine Textdatei, in der die Eintragungen in separaten Zeilen gespeichert werden. Die Datenbank kann mit einem einzigen Schlüssel abgefragt und verändert werden. Der Schlüssel und die anderen Elemente werden durch ein Zeichen (im Befehl OPEN festgelegt) getrennt.

Die Zeilenlänge muss nicht identisch sein und selbst die Zahl der Spalten kann differieren.

Ist die Datenbank zum Schreiben geöffnet, sollte neben der Datenbankdatei ausreichend Platz für deren Duplizierung vorhanden sein.

Öffnen und Schließen einer Datenbank sind zeitaufwendig, so dass wiederholtes Öffnen und Schließen vermieden werden sollte.

Große Datenbanken (mit mehr als 100.000 Eintragungen) sollten nach Schlüsselwerten sortiert werden.

Eine Datenbank kann durch dieses Add-On über die GDL-Befehle OPEN, INPUT, OUTPUT und CLOSE, geöffnet, abgefragt, modifiziert und geschlossen werden.

Öffnen einer Datenbank

```
channel = OPEN (filter, filename, paramstring)
filter: interner Name des Add-Ons, hier "DATA"
filename: Name der zu öffnenden Datenbank
```


paramstring: Add-On-spezifische Parameter, enthält Trennzeichen und Parameter zum Öffnen der Datei

Öffnet die Datei. Soll eine Datenbankdatei zur Modifikation geöffnet werden, die nicht existiert, wird eine neue Datei erstellt. Soll eine Datenbankdatei zum Lesen geöffnet werden, die nicht existiert, erscheint eine Fehlermeldung.

Sein Rückwert ist eine positive ganze Zahl, die die spezifische Datenbank identifiziert. Dieser Wert wird die zukünftige Referenznummer der Datenbanken.

Ist die Datenbank bereits vor einem Öffnungsbefehl geöffnet, erstellt dieser nur eine Kanalnummer.

Die Parameterzeichenfolge kann folgendes enthalten:

SEPARATOR: nach dem Befehl in einfachen Anführungszeichen (") können Sie ein Zeichen angeben, dass in der Textdatei für die Trennung von Spalten verwandt werden soll (beim Schreiben und Lesen). Ein Sonderfall ist das Tabulatorzeichen ('\t').

MODE: auf den Befehl muss die Art der Öffnung folgen.

Es gibt drei Öffnungsweisen:

- RO (read only (nur lesen))
- WA (read, modify (lesen, modifizieren))
- WO (read, modify) öffnet die Datenbank, falls vorhanden.

DIALOG: der Parameter 'Dateiname' wird als Datei-Identifikator genutzt, andernfalls ist es ein vollständiger Pfad. Der Datei-Identifikator ist eine einfache Zeichenfolge, die sich während eines standard 'Open/Save as' Dialogs über das Add-On auf eine vorhandene Datei bezieht. Dieser Dialog wird durch das Add-On gespeichert und nicht erneut abgefragt, es sei denn, die Datei ist nicht verfügbar. Ist die Öffnungsmethode read only, startet das Add-On einen Dialog um das Dokument auszuwählen. Andernfalls startet das Add-On einen Warndialog mit den Optionen 'Erstellen' und 'Browse':

- Browse: sucht eine vorhandene Datei (Öffnungsdialog)
- Erstellen: erstellt eine neue Datei (Dialog speichern als).

Setzen Sie immer ein Komma (,) zwischen SEPARATOR, MODE und DIALOG.

LIBRARY: wird der Befehl LIBRARY im Parameterstring angegeben, muss sich die Datei in der geladenen Datei befinden.

Wenn Sie nicht existierende Befehle verwenden, die Trennzeichen falsch eingegeben wurden oder der Parameterstring leer ist, wird die Erweiterung folgende Standardeinstellungen benutzen:

SEPARATOR = '\t', MODE = RO.

Beispiel:

```
ch1 = OPEN ("DATA", "file1",
           "SEPARATOR=';', MODE = RO", DIALOG)
ch2 = OPEN ("DATA", "file2", "")
ch3 = OPEN ("DATA", "newfile",
           "SEPARATOR = '\t', MODE = WA")
```

Lesen der Werte aus der Datenbank

```
INPUT (channel, recordID, fieldID, var1  
      [, var2, ...])
```

recordID: Befehlswert (numerisch oder String)

fieldID: die Spaltenzahl der gegebenen

Eintragung (kleinste Zahl : 1, bezieht sich auf das Element nach dem Befehlswert)

var1,...: Variablen für die Annahme der gelesenen Eintragungselemente

Fragt die Datenbank aufgrund der Befehlswerte ab..

Finden den Eintrag, liest das Element des Eintrags ab der angegebenen Spalte und listet die gelesenen Werte der Parametersequenz entsprechend auf.

In der Parameterliste muss mindestens ein Wert angegeben sein. Die Werte können numerisch oder Zeichenfolgen sein, unabhängig von dem für sie definierten Parametertyp. Der Ergebniswert ist die Anzahl der erfolgreich gelesenen Werte.

Gibt es mehr Parameter als Werte, werden die Parameter und korrespondierenden Werte auf 0 gesetzt. Bei leeren Spalten (z.B. wenn zwischen den Trennzeichen nichts eingetragen wurde) werden die Parameter auf 0 gesetzt.

Findet es keine Eintragungen, wird (-1) ausgegeben.

Beispiel:

```
nr = INPUT (ch1, "key1", 1, v1, v2, v3)  
! Eingabe von drei Werten ab der ersten Spalte  
! (nach dem Befehl) der Eintragung mit dem  
! key "key1"  
PRINT nr, v1, v2, v3
```

Eingeben von Werten in die Datenbank

OUTPUT channel, recordID, fieldID, expr1 [, expr2, ...]

recordID: Befehlswert (numerisch oder String)

fieldID: Fahne: 0 angeben (oder <= 0) um eine Eintragung zu löschen oder 1 angeben 1 (oder > 0) um eine Eintragung zu erstellen oder zu ändern

expr1,...: neue Elementwerte der gefundenen oder neuer Eintrag, beim Löschen werden diese Werte ignoriert

Bei der Erstellung oder Modifizierung eines Eintrags wird die Eintragung dem angegebenen Befehlswert entsprechend eingestellt. Die Eintragung wird die angegebenen Werte in der gleichen Reihenfolge enthalten, wie im Befehl angegeben. Die Werte können numerisch oder Zeichenfolgen sein. Es muß mindestens eine Bedingung vorhanden sein.

Beim Löschen wird die zum angegebenen Befehlswert gehörende Eintragung aus der Datenbank entfernt. Die Ausdruckswerte werden ignoriert, trotzdem sollte mindestens einer angegeben werden.

Beispiel:

```
string = "Date: 19.01.1996"
```

```
a = 1.5
```

```
OUTPUT ch2, "keyA", 1, "New record"
```

```
OUTPUT ch2, "keyA", 1, "Modified record"
```

```
OUTPUT ch2, "keyA", 0, 0 ! löscht die Eintragung
```

```
OUTPUT ch2, "keyB", 1, a, string
```

Datenbank schließen

CLOSE Kanal

channel: channel value

Schließen Sie die Datenbank identifiziert durch den Kanalwert.

GDL DATETIME ADD-ON

Mit Hilfe der DateTime-Erweiterung können Sie aktuelles Datum und aktuelle Zeit, die auf Ihrem Rechner eingestellt sind, in verschiedenen Formaten erhalten.

Das Add-On funktioniert wie die GDL Datei-Operationen. Sie müssen einen Kanal öffnen, die Information lesen und den Kanal schließen. Dieses Add-On ist auch über den Befehl REQUEST GDL erreichbar, in welchem Fall die Sequenz der Befehle OPEN, INPUT und CLOSE intern aufgerufen wird. Es ist die einfachste Methode, die Datum/Zeit Information durch eine einzelne GDL-Befehllinie zu erreichen:

```
REQUEST ("DateTime", Format, datetimestring)
```

Der zweite Parameter der REQUEST-Funktion stimmt mit der im OPEN-Funktion paramstring Parameter beschrieben, überein.

Kanal öffnen

```
channel = OPEN (Filter, Dateiname, ParameterString)
```

```
filter: interner Name des Add-Ons, hier "DateTime"
```

```
filename: unbenutzt (um Systemdatum und Zeit zu erhalten, braucht keine Datei geöffnet zu werden)
```

```
paramstring: Add-on spezifischer Parameter, enthält das gewünschte Ausgabeformat von Datum und Zeit
```

Sein Rückwert ist eine positive ganze Zahl, die den geöffneten Kanal identifiziert. Dieser Wert wird die zukünftige Referenznummer des Kanals. Die Paramzeichenfolge kann Spezifizierer und andere Zeichensätze beinhalten.

Die Spezifizierer werden durch Datum- und Zeitwerten wie folgt ersetzt:

%a	abgekürzter Wochentag-Name
%A	abgekürzter Wochentag-Name
%b	abgekürzter Monatsname
%B	abgekürzter Monatsname
%c	Datum und Zeit in der Form: 01:35:56 PM Wednesday, March 27, 1996
%d	Tag des Monats als eine Dezimalzahl (01-31)
%H	Stunde (24-Stunden Uhr), als Dezimalzahl (00-23)
%I	Stunde (12-Stunden Uhr), als Dezimalzahl (01-12)
%j	Tag des Jahres, als Dezimalzahl (001-366)
%m	Monat, als Dezimalzahl (01-12)
%M	Minute, als Dezimalzahl (00-59)
%P	AM/PM Bezeichnung für eine 12-Stunden Uhr
%S	Sekunde, als Dezimalzahl (00-61)
%U	Wochennummer des Jahres (mit Sonntag als erster Tag der ersten Woche), als Dezimalzahl
%w	Wochentag, als Dezimalzahl (0 (Sonntag)-6 (Samstag))
%W	Wochennummer des Jahres (mit Monat als erster Tag der ersten Woche), als Dezimalzahl(00-53)
%x	Datum in der Form von Wednesday, March 27, 1998
%X	Zeit in der Form 01:35:56 PM
%y	Jahr ohne Jahrhundert, als eine Dezimalzahl (00-99)
%Y	y Jahr mit Jahrhundert, als eine Dezimalzahl
%Z	GDL ignoriert diesen Spezifizierer. Laut dem Standard, druckt es die Zeit-Zone aus, falls es bestimmt werden kann
%%	der % Zeichensatz

Beispiel:

```
dstr = ""
ch = OPEN ("DateTime", "", "%w/%m/%d/%Y, %H:%M%P")
n = INPUT (ch, "", "", dstr)
CLOSE (Kanal)
PRINT dstr !it prints 3/03/27/1996, 14:36 PM
```

Lesen der Information

```
n = INPUT (channel, "", "", datetimestr)
```

channel: channel value

datetimestr: Wert des Typs Zeichenfolge

Es liest einen Wert vom Typ Zeichenfolge ein, der Datum und/oder Zeit in dem, durch die ÖFFNEN-Sequenz angegebenen Format repräsentiert. Der zweite und dritte Parameter wird nicht benutzt (sie können leere Zeichenfolgen oder auch 0-s sein).

Der Rückwert ist die Anzahl der erfolgreich eingelesenen Werte, in diesem Fall ist sie 1.

Schließen des Kanals

```
CLOSE Kanal
```

Schließt den Kanal, der von dem Kanalwert identifiziert wird.

GDL FILE MANAGER I/O ADD-ON

Der "GDL File Manager In-Out" Add-On ermöglicht Ihnen, die Dateien und Unterdateien eines Ordners GDL Scripts enthalten.

Definieren Sie den Ordner den Sie prüfen möchten mit dem Befehl OPEN.

Übernehmen Sie über den Befehl INPUT den erste/nächste Datei/Ordner Namen in den definierten Ordner.

Beenden Sie die Ordnerprüfung mit dem Befehl CLOSE.

Spezifikation des Ordners

```
channel = OPEN (Filter, Dateiname, ParameterString)
```

Kanal: Ordner ID

filter: interner Name des Add-Ons, hier "FileMan"

filename : - Name des zu scannenden Ordners (OS abhängiger Pfad) - Ordner ID String (im DIALOG Modus - weiteres später)

paramstring : spezifischer Parameter des Add-ons.

Die Parameter in paramString müssen durch Kommas getrennt werden (,).

1. Parameter: DATEIEN/ORDNER

Was möchten Sie suchen?

2. Parameter (optional): DIALOG

Zeigt an, dass der Ordner durch eine Ordner ID Zeichenfolge angegeben wird, und nicht durch einen Verzeichnispfad.

Wenn dies der Fall ist (und jedesmal wenn der entsprechende Dateipfad ungültig zu sein scheint) wird eine Dialogbox angezeigt, in die der Anwender den ID String - Dateipfad eingibt, der dann gespeichert wird.

Zum Beispiel die Meldung

```
Ordner = OPEN( "FileMan", "c:\")
```

Öffnet das Grundverzeichnis der Festplatte C (auf PC) zum Datei-Scanning.

Datei/Ordernamen erhalten

```
n = INPUT (channel, recordID, fieldID, var1 [,
           var2, ...])
```

channel: Der vom Befehl OPEN zurückgegebene Kanal
 recordID : 0 (für spätere Entwicklungen reserviert)
 fieldID : 0 (für spätere Entwicklungen reserviert)
 var1, ... : Variable(n) für den Empfang der Datei/Ordernamen
 n : Anzahl der erfolgreich ausgefüllten Variablen

Zum Beispiel die Meldung

```
n = INPUT (folder, 0, 0, fileName)
```

Holt den nächsten Namen vom angegebenen Ordner und gibt 1 zurück
 Existieren nicht mehr Dateien/Unterdateien wird die Variable n auf null gesetzt.

Beenden Ordner Scanning

CLOSE (Kanal)
 Schließen Sie den Ordner identifiziert durch den Kanalwert.

Beispiel: Listing eines einzelnen Ordners

Die folgende Befehlskette (z. B. wie das 2D Script Sektion eines Objektes) führt die Dateien der Ordner spezifiziert nach Favoriten. Beim ersten Gebrauch gibt der Anwender einen bestehenden Ordner zur Identifizierung an. Später wird die Eigene Dateien Id dieser Ordner sein.

```
topFolder = offen ( "FileMan", "MyFavouriteFolder", "files, dialog" )
y = 0
n = INPUT (folder, 0, 0, fileName)
while n = 1 do
  text2 0, y, fileName
  y = y - 0.6
  n = INPUT (folder, 0, 0, fileName)
endwhile
CLOSE (Kanal)
```

GDL TEXT I/O ADD-ON

Über die nachstehenden Schlüsselwörter können Sie äußere Dateien zwecks Lesens und Schreibens öffnen, sowie Werte aus /in GDL-Skripts einfügen und einholen.

Dieses Add-On interpretiert die Zeichenfolgen in der Parameterliste der Befehle OPEN, INPUT, OUTPUT des GDL-Skripts.

Es zeigt an, dass neben ArchiCAD ein Ordner namens "ArchiCAD Data Folder" für anwenderspezifische Dateien existiert.

(Der Name dieses Verzeichnisses befindet sich in der Add-On Ressource Fork und kann deshalb lokalisiert werden.) Existiert kein Ordner dieses Namens, wird er vom Add-On erstellt. Der Ordner kann Unterordner enthalten, die von der Erweiterung auf vorhandene Dateien überprüft wird. Es kann Textdateien lesen und schreiben.

Datei öffnen

`channel = OPEN (Filter, Dateiname, ParameterString)`

`filter`: interner Name des Add-Ons, hier "Text"

`Dateiname`: Name der zu öffnenden Datei

`Parameterzeichenfolge`: Add-On-spezifische Parameter, enthält Trennzeichen und Parameter zum Öffnen der Datei

Öffnet die Datei. Existiert die Datei nicht, in der Sie schreiben möchten, wird die Datei erstellt. Existiert die Datei nicht, die Sie lesen möchten, erscheint eine Fehlermeldung.

Sein Rückwert ist eine positive ganze Zahl, die die spezifische Datei identifiziert. Dieser Wert wird die zukünftige Referenznummer der Dateien.

Die Parameterzeichenfolge kann folgendes enthalten:

SEPARATOR: nach dem Befehl zwischen einzelnen Anführungszeichen (") können Sie ein Zeichen zur Trennung von Spalten für die Benutzung in der Textdatei angeben (zum Schreiben und).

Sonderfälle sind die Zeichen Tabulator ("t") und neue Zeile ("n").

MODE: auf diesen Befehl muss die Art der Öffnung folgen. Es gibt nur drei Öffnungsmodi:

RO (nur lesen)

WA (nur schreiben, ans Dateiende angehängt)

WO (nur schreiben, überschreiben) die vorher in der Datei gespeicherten Daten gehen verloren!

Eine Datei kann nicht gleichzeitig zum Lesen und Schreiben geöffnet werden.

DIALOG: ist dieser Befehl vorhanden, wird ein Dialogfenster erscheinen, in das Sie einen Dateinamen eingeben können.

FULLPATH: ist dieser Befehl vorhanden, wird der Dateiname als vollständiger Dateipfad interpretiert.

LIBRARY: ist dieser Befehl vorhanden, muss sich die Datei in der geladenen Bibliothek befinden.

Fügen Sie zwischen den Befehlen immer ein Komma (,) ein.

Wenn Sie nicht existierende Befehle verwenden, die Trennzeichen falsch eingegeben wurden oder der Parameterstring leer ist, wird die Erweiterung folgende Standardeinstellungen benutzen:

Beispiel:

```
ch1 = OPEN ("TEXT", "file1", "SEPARATOR = ';', MODE = RO")
ch2 = OPEN ("TEXT", "file2", "")
ch3 = OPEN ("TEXT", "file3", "SEPARATOR = '\n', MODE = WO")
```

Lesen der Werte

```
INPUT (channel, recordID, fieldID,
      var1 [, var2, ...])
```

channel: channel value

recordID: Zeilennummer (numerisch oder String)

fieldID: Spaltennummer in der angegebenen Zeile

var1,...: Variablen für die Annahme der gelesenen Eintragungselemente

Schreibt ebenso viele Werte in die Datei ein, -die durch den Kanal-Wert der angegebenen Ausgangsposition wiedererkannt werden-, wie es definierte Bedingungen gibt. In der Parameterliste muss mindestens ein Wert angegeben sein. Diese Funktion fügt die roten Werte in die Parameter wie angewiesen ein. Diese Werte können vom numerischen- oder vom Zeichentyp sein, unabhängig von dem gespeicherten Parametertyp.

Der Rückwert ist die Anzahl der erfolgreich eingelesenen Werte, in diesem Fall ist sie -1.

Sowohl die Zeilen- als auch die Spaltennummern müssen positive ganze Zahlen sein, andernfalls erhalten sie eine Fehlermeldung.

Sind die Zeilen- oder Spaltennummern falsch, wird die Eingabe nicht ausgeführt. (n = 0)

Können Zeile und Spalte identifiziert werden, sollten von der angegebenen Startposition aus so viele Werte eingegeben werden, wie Parameter angegeben sind. Sollten mehr Parameter als Werte vorhanden sein, werden die Parameter ohne korrespondierende Werte auf 0 gesetzt.

Bei leeren Spalten (z.B. wenn zwischen den Trennzeichen nichts eingetragen wurde) werden die Parameter auf 0 gesetzt.

Beispiel:

```
nr = INPUT (ch1, 1, 1, v1, v2, v3) ! ! Eingabe von drei Werten ab der ersten Spalte der ersten Zeile.
```

```
PRINT nr, v1, v2, v3
```

Schreiben der Werte

OUTPUT channel, recordID, fieldID, expr1 [, expr2, ...]

channel: channel value

recordID: falls positiv, folgt eine neue Zeile auf die Ausgabewerte

fieldID: keine Funktion, der Wert wird nicht benutzt

expr1: auszugebende Werte

Schreibt ebenso viele Werte in die Datei ein, -die durch den Kanal-Wert der angegebenen Position wiedererkannt werden-, wie es definierte Bedingungen gibt. Es muß mindestens eine Bedingung vorhanden sein. Die Typen der Ausgabewerte stimmen mit denen der Ausdrücke überein.

Bei einer Texterweiterung wird der OUTPUT die Datei mit den gegebenen Ausdrücken entweder überschreiben oder zum Ende der Datei hinzufügen (hängt von der Öffnungsweise ab). Dies geschieht der Reihe nach unter Verwendung der bei der Dateioffnung definierten Trennzeichen. In diesem Fall wird die gegebene Position nicht interpretiert.

Die recordID wird zur Angabe der neuen Zeilen in der Ausgabe eingesetzt.

Ist die recordID positiv, folgt eine neue Zeile auf die Ausgabewerte, andernfalls folgt ein Trennzeichen auf den letzten Wert.

Beispiel:

```
string = "Date: 19.01.1996"
```

```
a = 1.5
```

```
OUTPUT ch2, 1, 0, string ! Zeichenfolge, gefolgt von einer neuen Zeile
```

```
OUTPUT ch2, 0, 0, a, a + 1, a + 2! Trennzeichen nach einer + 2 ! ohne neue Zeile.
```

Schließen einer Datei

CLOSE Kanal

channel: channel value

Schließen Sie die Datenbank identifiziert durch den Kanalwert.

Beispiel:

Ein GDL-Objekt, dass den Inhalt von der Datei "f1" nur in die Dateien "f2" und "f3" kopiert, aber alle in "f1" tabellierten Werte in den Dateien "f2" und "f3" in separate Zeilen einfügt.

```
ch1 = open ("TEXT", "f1", "mode = ro")
ch2 = open ("TEXT", "f2", "separator = '\n', mode = wo")
ch3 = open ("TEXT", "f3", "separator = '\n', mode = wo")
i = 1
1:
n = input (ch1, i, 1, var1, var2, var3, var4)
if n <> -1 then
output ch2, 1, 0, var1, var2, var3, var4
output ch3, 1, 0, var1, var2, var3, var4
i = i + 1
goto 1
else
goto 2
endif
2:
close ch1
close ch2
close ch3
END
```

EIGENSCHAFTEN GDL ADD-ON

Dieses Add-On erstellt eine ArchiCAD Eigenschafts-Datenbasis, die von GDL-Scripts aus aufgerufen werden kann. Sie können die Datenbasistabellen öffnen und die Inhalte abfragen, genau wie mit SQL. Sie können einzelne Datensätze und Mehrfach-Datensätze (Listen) abfragen. Sie können die Datenbasis nicht ändern und keine Datensätze hinzufügen.

Eine ausführliche Beschreibung der Eigenschafts-Datenbasis finden Sie im ArchiCAD Berechnungshandbuch. ["Berechnungshandbuch" in ArchiCAD Hilfe](#)

OPEN

Syntax: OPEN ("PROP", "Datenbasis-Set-Name", ["Datenbasis-Dateien"])

<Datenbasis-Set-Name> ist ein beliebiger Name, der ein Set von Datenbank-Dateien in aufeinander folgenden OPEN-Aufrufen kennzeichnet.

<Datenbasis-Dateien> ist eine Liste von Textdateien, die Teil der Eigenschafts-Datenbasis sind. Dieser Parameter ist optional, falls Sie den einzulesenden Dateien zuvor einen <Datenbasis-Set-Name> zugeordnet haben. Die Reihenfolge der Dateien ist fest: <Schlüsseldatei>, <Komponentendatei>, <Eigenschaftsdatei>, <Einheitendatei>. Sie brauchen keine vollständigen Pfade anzugeben, da ArchiCAD diese Dateien für Sie in den aktiven Bibliotheken sucht. Wenn Sie lange Dateinamen verwenden, setzen Sie die Namen in Anführungszeichen (‘ oder ’).

Rückgabewert: Kanalnummer

Öffnet einen Kommunikationskanal zu den angegebenen Datenbasisdateien. Der Inhalt der Datenbasisdateien wird für einen schnelleren Zugriff in den Speicher eingelesen. So lange die Eigenschafts-Datenbasis geöffnet ist, sind die Änderungen über dieses Add-On nicht verfügbar. Normalerweise stellt dies jedoch kein Problem dar.

Beispiele:

1.

```
channel = OPEN ("PROP", "Muster", "'AC 8_KEY.txt', 'AC 8_COMP.txt', 'AC 8_DESC.txt', 'AC 8_UNIT.txt'")
```

Mit diesem Befehl wird eine Datenbank geöffnet, die aus den beiden oben angegebenen Dateien besteht (den Dateien der ArchiCAD 7.0 Eigenschafts-Datenbasis) und mit dem Namen "Muster" versehen. Beachten Sie, dass innerhalb des dritten Parameters ein anderes Anführungszeichen verwendet werden muss (Sie können " und ‘ verwenden).

2.

```
channel = OPEN ("PROP", "Muster", "")
```

Dieser Befehl kann nach dem expliziten Öffnen der Datenbasisdateien verwendet werden (wie in Beispiel 1), jedoch vor dem Schließen. Somit können Sie den expliziten Befehl an einer Stelle im Master_GDL Script verwenden und die Kurzversion an späterer Stelle.

CLOSE

Syntax: CLOSE (Kanalnummer)

Rückgabewert: keiner

Schließt den zuvor geöffneten Kommunikationskanal.

INPUT

Syntax: INPUT (Kanalnummer, "Abfragetyp", "Feldliste", Variable1[, ...])

<Parameter: > ist eine gültige Nummer eines Kommunikationskanals, die von einem vorangegangenen Befehl OPEN angegeben wurde.

<Abfragetyp> gibt die auszuführende Abfrage an. Das Add-On kennt die folgenden Schlüsselwörter:

Single-record queries: KEY, <Keycode> – fragt den Datensatz aus der Schlüssel-Datenbasis ab, wobei <Keycode> der Wert des Keycode-Attributs ist.

Gültige Felder: KEYCODE, KEYNAME

UNIT, <Einheitencode> – fragt den Datensatz aus der Einheiten-Datenbasis ab, wobei <Einheitencode> der Wert des Einheitencode-Attributs ist.

Gültige Felder: UNITCODE, UNITNAME, UNITFORMATSTR

COMP, <Keycode>, <Code> – fragt den Datensatz aus der Einheiten-Datenbasis ab, wobei <Keycode> der Wert des Keycode-Attributs ist und <Code> der Wert des Komponentencode-Attributs.

Gültige Felder: KEYCODE, KEYNAME, CODE, NAME, QUANTITY, QUANTITYSTR, UNITCODE, UNITNAME, UNITFORMATSTR

DESC, <Keycode>, <Code> – fragt den Datensatz aus der Einheiten-Datenbasis ab, wobei <Keycode> der Wert des Keycode-Attributs ist und <Code> der Wert des Eigenschaftscode-Attributs.

Gültige Felder: KEYCODE, KEYNAME, CODE, NAME, NUMOFLINES, FULLNAME

Listing queries: KEYLIST – listet alle Datensätze in der Schlüssel-Datenbasis auf

Gültige Felder: KEYCODE, KEYNAME

UNITLIST – listet alle Datensätze in der Einheiten-Datenbasis auf

Gültige Felder: UNITCODE, UNITNAME, UNITFORMATSTR

COMPLIST[, <Keycode>] – listet alle Datensätze in der Komponenten-Datenbasis auf oder, wenn der <Keycode> angegeben ist, nur die Datensätze mit dem angegebenen <Keycode>

Gültige Felder: KEYCODE, KEYNAME, CODE, NAME, QUANTITY, QUANTITYSTR, UNITCODE, UNITNAME, UNITFORMATSTR

DESLIST[, <Keycode>] – listet alle Datensätze in der Eigenschafts-Datenbasis auf oder, wenn der <Keycode> angegeben ist, nur die Datensätze mit dem angegebenen <Keycode>

Gültige Felder: KEYCODE, KEYNAME, CODE, NAME, NUMOFLINES, FULLNAME

COMPDESLIST[, <Keycode>] – listet alle Datensätze in der Komponenten-Datenbasis und der Eigenschafts-Datenbasis auf oder, wenn der <keycode> angegeben ist, nur die Datensätze mit dem angegebenen <keycode>.

Gültige Felder: ISCOMP, KEYCODE, KEYNAME, CODE, NAME, QUANTITY, QUANTITYSTR, UNITCODE, UNITNAME, UNITFORMATSTR, NUMOFLINES, FULLNAME

Verwenden Sie diese Abfrage mit großer Sorgfalt! Wenn eines der Felder in der Datenbasis nicht gültig ist (z. B. FULLNAME in der Komponenten-Datenbasis), wird es in der Ergebnisliste einfach weggelassen (darüber sollten Sie sich im Klaren sein).

<Feldliste> listet die Datenbankattribute auf, deren Werte in der Ausgabe angezeigt werden sollen. Wenn die Ausgabe eine Liste ist, wird sie in der hier angegebenen Reihenfolge der Felder sortiert.

Die folgenden Felder können verwendet werden:

KEYCODE – Keycode-Attribut.

Typ: String

Verwendung in Abfragen: KEY, COMP, DESC, KEYLIST, COMPLIST, DESCLIST, COMPDESCLIST

KEYNAME – Keyname-Attribut.

Typ: String

Verwendung in Abfragen: KEY, COMP, DESC, KEYLIST, COMPLIST, DESCLIST, COMPDESCLIST

UNITCODE – Einheitencode-Attribut

Typ: String

Verwendung in Abfragen: UNIT, COMP, UNITLIST, COMPLIST, COMPDESCLIST

UNITNAME – Einheitenname-Attribut

Typ: String

Verwendung in Abfragen: UNIT, COMP, UNITLIST, COMPLIST, COMPDESCLIST

UNITFORMATSTR – GDL-Format-String der Einheit

Typ: String

Verwendung in Abfragen: UNIT, COMP, UNITLIST, COMPLIST, COMPDESCLIST

CODE – Komponenten- oder Eigenschaftscode-Attribute (abhängig von der Abfrage)

Typ: String

Verwendung in Abfragen: COMP, DESC, COMPLIST, DESCLIST, COMPDESCLIST

NAME – Name der Komponente oder die erste Zeile eines Eigenschafts-Datensatzes

Typ: String

Verwendung in Abfragen: COMP, DESC, COMPLIST, DESCLIST, COMPDESCLIST

QUANTITY – Menge einer Komponente als Anzahl (für Berechnungen)

Typ: Anzahl

Verwendung in Abfragen: COMP, COMPLIST, COMPDESCLIST

QUANTITYSTR – Menge einer Komponente im String-Format

Typ: String

Verwendung in Abfragen: COMP, COMPLIST, COMPDESCLIST

NUMOFLINES – Anzahl der Zeilen in einem Eigenschafts-Datensatz

Typ: Anzahl

Verwendung in Abfragen: DESC, DESCLIST

FULLNAME – der gesamte Eigenschafts-Datensatz

Typ: String(s)

Verwendung in Abfragen: DESC, DESCLIST

ISCOMP – gibt an, ob der nächste Datensatz eine Komponente oder eine Eigenschaft ist

Typ: Anzahl (1 bei Komponente, 0 bei Eigenschaft)

Verwendung in Abfragen: COMPDESCLIST

<Variablen> enthält nach Abschluss der Abfrage das Ergebnis. Sie können mehrere Variablen auflisten, wenn Sie genau wissen, wie viele Sie benötigen (z. B. mit Einzelabfragen), oder Sie können einen dynamischen Array angeben. Die Datensätze werden sequenziell aufgelistet.

Beispiele:

1.

INPUT (Kanal, "KEY, 001", "KEYNAME", Keyname)

Dies ist eine einfache Abfrage: Der Name des Schlüssels mit dem Code '001' wird in die Variable "Keyname" eingetragen.

2. < >

INPUT (Kanal, "DESC, 004, 10", "NUMOFLINES, FULLNAME", desc_txt)

Der Eigenschafts-Datensatz mit dem Keycode '004' und dem Code '10' wird verarbeitet; die Anzahl der Zeilen des Beschreibungstextes und der Text selbst werden im Array desc_txt eingetragen. Das Ergebnis lautet:

desc_txt[1] = <AnzahlZeilen> (Anzahl)

desc_txt[2] = <ErsteBeschreibungszeile> (String)

...

desc_txt[<AnzahlZeilen+1>] = <Letzte Beschreibungszeile

3.

INPUT (Kanal, "COMPLIST", "NAME, KEYNAME, QUANTITY", comp_list)

Erstellt eine Komponentenliste, sortiert nach dem Namensfeld, anschließend nach Keyname und zuletzt nach dem Mengenfild; das Ergebnis wird in den Array comp_list eingetragen. Das Ergebnis lautet:

complist[1] = <Name1> (String)

complist[2] = <Keyname1> (String)

complist[3] = <Menge1> (Anzahl)

complist[4] = <Name2> (String)

... etc.

4.

INPUT (Kanal, "COMPDESCLIST, 005", "ISCOMP, KEYNAME, NAME, QUANTITY", x_list)

Erstellt eine allgemeine Komponenten- und Eigenschaftsliste; dies bedeutet, dass Datensätze aus beiden Tabellen aufgelistet werden, wobei der <Keycode> '005' lautet. Die Ausgabe lautet:

x_list[1] = 0 (Anzahl, 0 -> bedeutet Eigenschaft)

x_list[2] = <Name1> (String -> Eigenschaften haben kein Feld <Keyname>, daher weggelassen)

x_list[3] = 0 (Anzahl, Eigenschaften haben kein Mengenfild)

...


```
x_list[(n*2)-1] = 1 (Anzahl -> es wurden n-1 Eigenschaften aufgelistet; jetzt folgen die Komponenten)
x_list[n*2] = <Keyname_n> (String)
... etc.
```

OUTPUT

Dieser Befehl ist in diesem Add-On nicht implementiert, da Eigenschafts-Datenbasen schreibgeschützt sind.

GDL XML ERWEITERUNG

Diese Erweiterung ermöglicht das Lesen, Schreiben und Bearbeiten von XML-Dateien. Sie implementiert eine Untergruppe der DOM-Schnittstelle (Document Object Model). XML ist eine Textdatei, die Markierungen verwendet zur Strukturierung von Daten zu einem hierarchischen System, vergleichbar mit HTML. Ein XML-Dokument kann über eine hierarchische Baumstruktur modelliert werden, deren Knoten die Daten des Dokuments enthalten. Die Erweiterung kennt die folgenden Knotentypen:

- *Element*: der Inhalt zwischen einer Start-Markierung und eine Ende-Markierung im Dokument oder, bei einem leeren Element, eine "Leere-Element"-Markierung. Elemente haben einen Namen und können Attribute haben; normalerweise haben Elemente auch einen Inhalt, dies ist jedoch nicht unbedingt erforderlich. Das bedeutet, dass Elementtyp-Knoten untergeordnete Knoten haben können. Attribute werden in einer Attributliste gespeichert, in der jedes Attribut einen anderen Namen und einen Textwert hat.
- *Text*: eine Folge von Zeichen. Text kann keine untergeordneten Knoten haben.
- *Kommentar*: Text zwischen den Kommentargrenzzeichen: `<!--` der Kommentar selbst `-->`. Im Kommentartext müssen jedem `'` Zeichen ein vom Zeichen `'` zu unterscheidendes Zeichen folgen. Das bedeutet auch, dass folgendes falsch ist: `<!-- Kommentar --->`. Knotenpunkte des Kommentartyps können keine Unterknotenpunkte haben.
- *CDATA-Abschnitt*: Text zwischen den CDATA-Abschnittsbegrenzern: `<![CDATA[` der eigentliche Text `]]>`. In einem CDATA-Abschnitt brauchen (und dürfen) Zeichen, die in einem XML-Dokument eine spezielle Bedeutung haben, nicht mit Escape-Zeichen maskiert werden. Die einzige erkannte Markierung ist das schließende Anführungszeichen `']]>`. Knoten des Typs CDATA-Abschnitt haben keine untergeordneten Knoten.
- *Einheiten-Referenz*: Referenz auf eine vordefinierte Einheit. Ein solcher Knoten kann eine schreibgeschützte Teil-Baumstruktur haben; diese Teil-Baumstruktur gibt den Wert der referenzierten Einheit an. Bei der Syntaxanalyse ("Parsing") des Dokuments kann ausgewählt werden, dass Einheiten-Referenzen in Textknoten umgewandelt werden sollen.

Auf der obersten Ebene muss genau ein Knoten des Typs Element vorhanden sein ("Root"), und es können mehrere Knoten des Typs Kommentar vorhanden sein. Der Knoten des Typs Dokument der DM-Schnittstelle ist über die Schnittstelle der Erweiterung nicht verfügbar.

Für jeden Knoten in der Baumstruktur sind ein Name und ein Wert-String zugeordnet, dessen Bedeutung vom Typ des Knotens abhängt:

	Name:	Wert:
Element:	Name der Markierung	"" (Leerer String)

Text:	"#text"	der Textinhalt des Knotens
Kommentar:	"#comment"	der Textinhalt des Knotens
CDATA-Abschnitt:	"#cdata-section"	der Textinhalt des Knotens
Einheiten-Referenz:	Name der referenzierten Einheit	"" (Leerer String)

Für jeden Knotentyp definiert die Erweiterung String-Schlüsselwörter, die in bestimmten Anweisungen an die Erweiterung übergeben werden können:

Element:	ELEM
Text:	TXT
Kommentar:	CMT
CDATA-Abschnitt:	CDATA
Einheiten-Referenz:	EREF

Das Ergebnis des Fehlercodes eines Befehls OPEN, INPUT oder OUTPUT kann über die Anweisung GetLastError des Befehls INPUT abgerufen werden.

Das XML-Dokument öffnen

Der Befehl OPEN:

`channel = OPEN (filter, filename, parameter_string)`

filter: Dateierweiterung. Diese Erweiterung sollte 'XML' lauten.

filename: Name und Pfad der zu öffnenden (bzw. zu erstellenden) Datei oder eine ID-Nr., wenn die Datei über ein Dialogfenster geöffnet und die Position der Datei vom Benutzer angegeben wird.

parameter_string: Eine Folge von Zeichenmarkierungen, die den Modus für das Öffnen angeben:

- 'r': Öffnen im schreibgeschützten Modus. Im Allgemeinen kann nur der Befehl INPUT verwendet werden.
- 'e': Einheiten-Referenzen in der Baumstruktur werden nicht in Textknoten umgesetzt. Ohne diese Markierung gibt es keine Einheiten-Referenzen in der Dokumentstruktur.
- 'v': Beim Lesen und beim Schreiben wird eine Gültigkeitsprüfung durchgeführt. Wenn ein DTD im Dokument vorhanden ist, muss die Dokumentstruktur damit übereinstimmen. Ohne diese Markierung kann ein korrekt strukturiertes, aber ungültiges Dokument ohne Fehlermeldung eingelesen und geschrieben werden.

- **'n'**: Erstellt eine neue Datei. Wenn die Datei bereits vorhanden ist, schlägt das Öffnen fehl. (Nach dem Befehl OPEN muss die Anweisung CreateDocument die erste ausgeführte Anweisung sein.)
- **'w'**: Falls die Datei vorhanden ist, wird sie mit einem leeren Dokument überschrieben. Ist sie nicht vorhanden, wird eine neue Datei erstellt. (Nach dem Befehl OPEN muss die Anweisung CreateDocument die erste ausgeführte Anweisung sein.)
- **'d'**: Die Datei wird vom Benutzer über ein Dialogfenster angefordert. Bei späteren Ausführungen wird sie mit der ID-Nr verknüpft, der im Dateinamens-Parameter des Befehls OPEN angegeben wurde. (Falls die ID-Nr bereits einer Datei zugeordnet ist, wird dem Benutzer kein Dialogfenster angezeigt)
- **'f'**: Der Dateinamens-Parameter enthält einen vollständigen Pfad.
- **'l'**: Die Datei befindet sich in den geladenen Bibliothekselementen.
channel: Wird in nachfolgenden E/A-Befehlen zur Angabe der Verbindung verwendet.

Wenn Sie eine vorhandene XML-Datei zum Ändern öffnen wollen, dürfen keine der Markierungen 'r', 'n' und 'w' im Parameter-String gesetzt sein. Es darf nur eine der Markierungen 'd', 'f' und 'l' gesetzt sein. Ist keine dieser Markierungen gesetzt, wird der Dateiname als Pfad relativ zu dem Dokument-Ordner des Benutzers interpretiert.

Das XML-Dokument ändern

DOM ist ein objektorientiertes Modell, das nicht direkt in eine BASIC-ähnliche Sprache wie GDL umgesetzt werden kann. Zur Darstellung der Knoten in der Hierarchie definieren wir Positionbeschreibungen. Bei der Navigation durch die Knoten des Baums müssen wir zunächst eine neue Positionsbeschreibung aus der Erweiterung anfordern. Ursprünglich verweist eine neue Beschreibung auf das Root-Element. Die Beschreibung ist tatsächlich eine 32-Bit ID-Nummer, deren Wert für das GDL-Script nicht von Bedeutung ist. Die Position, auf die sie verweist, kann beim Wechsel von einem Knoten im Baum zu einem anderen geändert werden. Der Befehl INPUT:

```
<B>INPUT</B> (ch, recordID, fieldID, var1, var2...)
```

ch: Der vom Befehl OPEN zurückgegebene Kanal

recordID: Name der Anweisung plus Parameter

fieldID: normalerweise eine Positionsbeschreibung var1, var2,...: optionale Liste von Variablen, die Ergebnisdaten empfangen.

INPUT-Anweisungen:

1 . **GetLastError**: Ruft das Ergebnis der letzten Operation ab

recordID: "GetLastError"

fieldID: ignoriert

Ergebniswerte:

var1: error code / ok

var2: Der Erläuterungstext zum Fehler / ok

- 2 . **NewPositionDesc:** Anforderung einer neuen Positionsbeschreibung

recordID: "NewPositionDesc"

fieldID: ignoriert

Ergebniswert: var1: die neue Positionsbeschreibung (bezieht sich auf die Wurzel)

- 3 . **CopyPositionDesc:** Anforderung einer neuen Positionsbeschreibung, deren Startknoten aus einer anderen Beschreibung geholt wird.

recordID: "CopyPositionDesc"

fieldID: eine vorhandene Positionsbeschreibung

return value: var1: Die neue Positionsbeschreibung (verweist ursprünglich auf die Stelle, auf die die Beschreibung in fieldID verweist)

- 4 . **ReturnPositionDesc:** Wenn eine Positionsbeschreibung nicht mehr erforderlich ist.

recordID: "CopyPositionDesc"

fieldID: die Positionsbeschreibung

Verwenden Sie diese Anweisung, wenn eine über die Anweisungen NewPositionDesc oder CopyPositionDesc abgerufene Positionsbeschreibung nicht mehr verwendet wird.

- 5 . **MoveToNode:** Ändert die Position einer Beschreibung. (und übernimmt die Daten des neuen Knotenpunktes)

Diese Anweisung kann für die Navigation im Hierarchybaum verwandt werden.

recordID: "MoveToNode searchmode nodename nodetype nodenumber"

fieldID: Positionsbeschreibung

Suchmodus (oder Bewegungsmodus): der Parameter des Knotenpunktnamens muss einen Pfad enthalten, der ein Element oder einen Eigenschaftsreferenzknotenpunkt im xml-Dokument angibt.

Der Pfad ist relativ zu dem im fieldID angegebenen Knoten. Das Begrenzungszeichen ist '!' (das ansonsten ein gültiges Zeichen in einem Elementnamen ist, daher funktioniert diese Angabe

nicht in allen Fällen). Der String '..' im Pfad kennzeichnet einen Schritt in Richtung

des übergeordneten Knotens. Der Startknoten kann sich von einem Element- oder

Einheiten-Referenzknoten unterscheiden; in diesem Fall muss der Pfad

mit '..' beginnen für einen Schritt zurück. Wenn mehrere Elementknoten

mit dem gleichen Namen auf der gleichen Ebene vorhanden sind, wird

der erste Knoten ausgewählt.

Für die folgenden Verschiebemodi dürfen die weiteren Parameter nicht angegeben sein:

ToParent: Wechselt zu dem übergeordneten Knoten des in fieldID angegebenen Knotens.

ToNextSibling: Wechselt zum nächsten Knoten auf der gleichen Ebene.

ToPrevSibling: Wechselt zum vorigen Knoten auf der gleichen Ebene.

ToFirstChild: Wechselt zum ersten Abkömmling des fieldID-Knotens.

ToLastChild: Wechselt zum letzten Abkömmling des fieldID-Knotens.

Für die folgenden Suchmodi können die weiteren Parameter angegeben sein; falls sie nicht angegeben sind, gelten entsprechende Standardwerte:

FromNextSibling: Die Suche beginnt mit dem nächsten Knoten auf der gleichen Ebene und wird in Vorwärtsrichtung fortgesetzt.

FromPrevSibling: Die Suche beginnt mit dem Knoten vor fieldID und wird rückwärts auf der gleichen Ebene fortgesetzt.

FromFirstChild: Die Suche beginnt mit dem ersten Abkömmling des fieldID-Knotens und wird in Vorwärtsrichtung fortgesetzt.

FromLastChild: Die Suche beginnt mit dem letzten Abkömmling des fieldID-Knotens und wird in Rückwärtsrichtung fortgesetzt.

nodename: Die Suche berücksichtigt nur die Knoten, deren Name oder Wert dem Knotennamen entspricht. Die Zeichen * und ? im Knotennamen werden als Universalzeichen betrachtet. Für Knoten der Typen Element und Einheiten-Referenz wird der Name verglichen, während für Knoten der Typen Text, Kommentar und CDATA-Abschnitt der Wert verglichen wird. Standardwert: *

nodetype: Die Suche berücksichtigt nur Knoten, deren Typ für den Knotentyp zulässig ist. Das Zeichen * bedeutet, dass alle Typen zulässig sind. Andernfalls können die Schlüsselwörter des Typs mit dem Zeichen + kombiniert werden, um den Knotentyp zu bilden (es muss sich um ein Wort ohne Leerzeichen handeln, z. B. TXT+CDATA.) Der Standardwert lautet *

nodenumber: Wenn mehrere Knoten übereinstimmen, gibt diese Anweisung die Anzahl der gesuchten Knoten in der Sequenz der übereinstimmenden Knoten an. (Beginnend mit 1) Standardwert: 1

Rückgabewerte:

var1: Name des Knotens

var2: Wert des Knotens

var3: Schlüsselwörter des Knotens eingeben

Beispiel:

Wir wollen auf derselben Ebene zum 2. Knotenpunkt zurückgehen, die ein Element oder eine Eigenschaftsreferenz ist, und dessen Namen mit K beginnt:

```
INPUT (ch, "MoveToNode FromPrevSibling K* ELEM+EREF 2", posDesc, name, val, type)
```

**6 . GetNodeData: **Ruft die Daten eines angegebenen Knotens ab.

recordID: "GetNodeData"

fieldID: die Positionsbeschreibung

Rückgabewerte:

var1: Name des Knotens

var2: Wert des Knotens

var3: Schlüsselworts des Knotens eingeben

**7 . NumberofChildNodes: **Gibt die Anzahl der untergeordneten Knoten eines angegebenen Knotens an

recordID: "NumberofChildNodes Knotentyp Knotenname"

fieldID: Positionsbeschreibung

Mit den folgenden optionalen Parametern kann das Set der berücksichtigten untergeordneten Knoten eingegrenzt werden:

nodetype: Die zulässigen Knotentypen sind in der Anweisung MoveToNode definiert

nodename: Zulässige Namen oder Werte der Knoten entsprechend der Definition in der Anweisung MoveToNode

Rückgabewerte:

var1: Anzahl der untergeordneten Knoten

**8 . NumberofAttributes: **Gibt die Anzahl der Attribute eines Elementknotens zurück.

"NumberofAttributes attrname"

fieldID: Positionsbeschreibung (muss auf einen Elementknoten verweisen)

attrname: Mit dieser Angabe kann das Set der berücksichtigten Attribute eingegrenzt werden, da nur die Attribute gezählt werden, deren Namen (und nicht die Werte)

attrname entsprechen. In attrname werden die Zeichen * und ? als Universalzeichen betrachtet.

Rückgabewerte:

var1: Anzahl der Attribute

9 . **GetAttribute:** ****Gibt die Daten eines Attributs eines Elementknotens zurück

"GetAttribute attrname attrnumber"

fieldID: Positionsbeschreibung (muss auf einen Elementknoten verweisen)

optionale Parameter:

attrname: Gibt den Namen des Attributs an. Die Zeichen * und ? werden als Universalzeichen betrachtet. Standardwert: *

attrnumber: Wenn mehrere Attribute der Angabe attrname entsprechen, wählt attrnumber das Attribut in der Sequenz der übereinstimmenden Attribute aus. Das Zählen beginnt von 1. Standardwert: 1

Rückgabewerte:

var1: Wert des Attributs

var2: Name des Attributs

10 . **Validate:** ****Gültigkeit des Dokuments überprüfen.

Bei einer Anweisung zur Änderung des Dokuments wird die Gültigkeit nicht geprüft. Sie wird beim Zurückschreiben der Datei auf die Platte geprüft, wenn die Markierung 'v' in dem Öffnungsmodus-String gesetzt wurde. Eine Gültigkeitsprüfung kann über die Anweisung Validate jederzeit erzwungen werden, sie kann jedoch viel Zeit und Speicher in Anspruch nehmen und sollte daher nicht nach jeder Änderung durchgeführt werden.

recordID: "Validate"

fieldID: ignoriert

Das XML-Dokument ändern

OUTPUT (ch, recordID, fieldID, var1, var2...)

ch: Der vom Befehl OPEN zurückgegebene Kanal

recordID: Name der Anweisung plus Parameter

fieldID: Normalerweise eine Positionsbeschreibung

var1, var2,...: Zusätzliche Eingabedaten

OUTPUT-Anweisungen:

Die meisten OUTPUT-Anweisungen sind für Dateien, die im schreibgeschützten Modus geöffnet wurden, ungültig.

1 **"CreateDocument "**

recordID: "CreateDocument"

fieldID: ignoriert

var1: Name des Dokumentes Dies ist gleichzeitig der Markierungsname des Root-Elements.

CreateDocument ist nur zulässig, wenn die Datei im Modus "Neue Datei" oder "Überschreiben" geöffnet wurde. In diesen Modi muss diese Anweisung zuerst ausgeführt werden, um das XML-Dokument zu erstellen.

**2 . NewElement: **Fügt einen neuen Elementknoten in das Dokument ein
recordID: "NewElement insertpos"

fieldID: Eine Positionsbeschreibung, relativ zu der der neue Knoten eingefügt wird.

var1: Name des neuen Elements (Element-Markierungsname)

insertpos kann folgendes sein::

AsNextSibling: Das neue Element wird nach der in fieldID angegebenen Position eingefügt

AsPrevSibling: Das neue Element wird vor der in fieldID angegebenen Position eingefügt

AsFirstChild: Das neue Element wird als erstes untergeordnetes Element des in fieldID (muss ein Elementknoten sein) angegebenen Knotens eingefügt

AsLastChild: Das neue Element wird als letztes untergeordnetes Element des in fieldID (muss ein Elementknoten sein) angegebenen Knotens eingefügt

**3 . NewText: **Fügt einen Textknoten in das Dokument ein
recordID: NewText insertpos"

fieldID: Positionsbeschreibung

var1: Einzufügender Text

Siehe auch die Anweisung NewElement.

**4 . NewComment: **Fügt einen neuen Kommentarknoten in das Dokument
recordID: "NewComment insertpos"

fieldID: Positionsbeschreibung

var1: Text des einzufügenden Kommentars

Siehe auch die Anweisung NewElement.

**5 . NewCDATASection: **Fügt einen neuen Knoten des Typs CDATA-Abschnitt in das Dokument ein
recordID: "NewCDATASection insertpos"

fieldID: Positionsbeschreibung

var1: Text des einzufügenden CDATA-Abschnitts

Siehe auch die Anweisung NewElement.

**6 . Copy: **Erstellt eine Kopie einer Teil-Baumstruktur des Dokuments unter einem bestimmten Knoten
recordID: "Copy insertpos"

fieldID: Eine Positionsbeschreibung, relativ zu der die Teil-Baumstruktur eingefügt wird

var1: Positionsbeschreibung mit der Angabe der zu kopierenden Teil-Baumstruktur

insertpos: wie bei der Anweisung NewElement

Die kopierte Teil-Baumstruktur bleibt unverändert. Positionsbeschreibungen, die auf einen Knoten in der kopierten Teil-Baumstruktur verweisen, verweisen nach dem Kopieren auf den gleichen Knoten.

**7 . Move: **Verschiebt eine Teil-Baumstruktur in dem Dokument an eine andere Position

recordID: "Move insertpos"

fieldID: Eine Positionsbeschreibung, relativ zu der die Teil-Baumstruktur eingefügt wird

var1: Positionsbeschreibung mit der Angabe der zu verschiebenden Teil-Baumstruktur

insertpos: wie bei der Anweisung NewElement

Die ursprüngliche Teil-Baumstruktur wird gelöscht. Positionsbeschreibungen, die auf einen Knoten in der zu verschiebenden Teil-Baumstruktur verweisen, verweisen nach dem Verschieben auf die neue Position der Teil-Baumstruktur.

**8 . Delete: **Löscht einen Knoten und seine Teil-Baumstruktur aus dem Dokument

recordID: "Delete"

fieldID: Positionsbeschreibung mit der Angabe des zu löschenden Knotens

Alle Positionsbeschreibungen, die auf einen Knoten in der gelöschten Teil-Baumstruktur verweisen, werden ungültig.

**9 . SetNodeValue: **Ändert den Wert eines Knotens

recordID: "SetNodeValue"

fieldID: Positionsbeschreibung, muss sich auf einen Knoten des Typs Text, Kommentar oder CDATA-Abschnitt beziehen

var1: Neuer Textwert des Knotens

**10 . SetAttribute: **Ändert ein Attribut eines Elementknotens oder erstellt

ein neues

recordID: "SetAttribute"

fieldID: Positionsbeschreibung, muss auf einen Elementknoten verweisen

var1: Name des Attributs

var2: Textwert des Attributs

Wenn das Element bereits ein Attribut mit diesem Namen hat, wird sein Wert geändert; andernfalls wird der Liste der Attribute des Elements ein neues

Attribut hinzugefügt.

**11 . RemoveAttribute: **Entfernt ein Attribut aus einem Elementknoten

recordID: "RemoveAttribute"

fieldID: Positionsbeschreibung, muss auf einen Elementknoten verweisen

var1: Name des zu entfernenden Attributs

**12 . Flush: **Schreibt das aktuelle Dokument in die Datei zurück

recordID: "Flush"

fieldID: ignoriert

Wenn die Datei im Modus "Validate" geöffnet worden war, wird nur ein gültiges Dokument gesichert.

**13 . ChangeFileName: **Ordnet dem aktuellen Dokument eine andere Datei zu

recordID: "ChangeFileName"

fieldID: Erstellt einen neuen Pfad

var1: Gibt an, wie fieldID interpretiert werden soll. wenn var1 ein leerer String ist, enthält fieldID einen Pfad relativ zu dem Ordner der Benutzerdokumente. 'd' bedeutet, dass die Position der Datei über ein Dialogfenster von dem Benutzern abgerufen wird (siehe die Markierung des Befehls OPEN *"OPEN" auf Seite 303*). 'l' bedeutet, dass die Datei aus den geladenen Bibliotheken verwendet wird. 'f' bedeutet, dass fieldID einen vollständigen Pfad enthält.

Diese Anweisung kann auch aufgerufen werden, wenn die Datei im schreibgeschützten Modus aufgerufen wurde. In diesem Fall verliert das Dokument nach der Ausführung das Schreibschutzattribut; es kann dann geändert und an der neuen Dateiposition gesichert werden.

Fehlercodes und Nachrichten:

- 0: "Ok"
- 1: "Add-on Initialisierung fehlgeschlagen"
- 2: "Nicht genügend Speicher"
- 3: "Falscher Parameter String"
- 4: "Dateidialog Fehler"
- 5: "Datei existiert nicht"
- 6: "XML Parse Fehler"
- 7: "Dateibearbeitungs-Fehler"
- 8: "Datei existiert bereits"
- 9: "Der Kanal ist nicht offen"
- 10: "Syntax Fehler"
- 11: "Fehler beim Öffnen"
- 12: "Ungültige Positionsbeschreibung"
- 13: "Ungültiger Knotenpunkt für diese Art Bearbeitung"
- 14: "Keinen solchen Knotenpunkt gefunden"
- 15: "Interner Fehler"
- 16: "Parameter Fehler"
- 17: "Kein solches Attribut gefunden"
- 18: "Ungültiges XML-Dokument"
- 19: "Unbehandelte Ausnahme"
- 20: "Nur-lesen Dokument"
- 21: "Dokument erstellen nicht erlaubt"
- 22: "Dokument erstellen fehlgeschlagen"

- 23: "Knotenpunkt Wert einstellen fehlgeschlagen"
- 24: "Verschieben nicht erlaubt"
- 25: "Löschen nicht erlaubt"
- 26: "AttributeSet nicht erlaubt"
- 27: "Dateiformat Fehler"
- 28: "Einfügen (oder kopieren) nicht erlaubt"
- 29: "Knotenpunkt Erstellen Fehler"
- 30: "Falscher String"
- 31: "Ungültiger Name"

INDEX

2D-Daten sichern 13
2D-Script 13
3D-Daten sichern 14
3D-Text 14

A

ABS 199
absoluter Nullpunkt 20
ACS 200
ADD 29
ADD2 27
ADDDGROUP 112, 116
ADDITIONAL_DATA 180
ADDX 28
ADDY 28
ADDZ 28
AND 198
Anführungszeichen in GDL Scripts 24
Anweisungen 23
ARC2 125
ArchiCAD 14
 Elementlisten in ~ 132, 185
ArchiFM 14
ARMC 60
ARME 61
ASN 200
ATN 200
Attribute
 definieren ~ 21
Ausrufezeichen in GDL Scripts 23

B

BackgroundColor 131
BEAM 55
Bedingungen 212
Befehle für den Einstieg 15
Befehle zur Programmsteuerung 18
Beschreibungen 14
 ~ bezug 184
 ~ Definition 184
Bestandteile 14
 ~ bezug 185
 ~ Definition 184

Bezug auf binäre Daten 185
Bibliothekselemente 13
Bildelement Definition 94, 129
Binäre Beschreibungen Daten 14
BINARY 14, 119
BINARYPROP 14, 185
Bitmapmuster
 170
BITSET 202
BITTEST 201
Block 33
Block definition 33
Block Wandparameter 52
BODY 102
Bogendefinition 125
Boole'scher Unterschied 113
BPRISM_ 41
BREAKPOINT 216
BRICK 33
BWALL_ 49

C

CALL 220
CEIL 199
CIRCLE2 125
CLOSE 223
COMPONENT 184, 185
CONE 36
COONS 87
COOR 100
COS 200
CPRISM_ 40
CSLAB_ 47
CUSTOM 187
CUTFORM 110
CUTPLANE 104
CUTPOLY 106
CUTPOLYA 108
CUTSHAPE 110
CWALL_ 47
CYLIND 33

D

Dachfläche definition
 geneigte ~ 56
Darstellungsform Drahtmodell 157
Das Benutzeroberfläche-Script 14
das lokale Koordinatensystem bewegen 28
das lokale Koordinatensystem skalieren 29
DATABASE_SET 183
Datei öffnen 222
Datei Operationen 222
DEFINE FILL 180
DEFINE FILL_A 180
DEFINE FILLA 171
DEFINE LINE_TYPE 175, 180
DEFINE MATERIAL 162, 180
DEFINE STYLE 176
DEFINE SYMBOL_FILL 174, 180
DEFINE SYMBOL_LINE 176, 180
Definition der Datenbank 183
Definition des Bruchpunktes im Script 216
Definition des Koordinatensystems
 lokale ~ 100
Definition des Schnittpolygons 106, 108
Definition eines Bild-Polygons 100
Definition eines Punktes im 3D 98
 ~ mit Texturursprung 98
Definition von Linien 121
Definition von Rechtecken 122
Definition der Darstellungsreihenfolge 161
DEL 31
DEL TOP 32
DESCRIPTOR 184
dezimaler Logarithmus 200
DIALOG 298
DIM 195
DO 212
Doppelpunkte in GDL Scripts 23
Drahtmodell 114
Drahtmodellbasis 114
DRAWINDEX 161
DRAWING 186
drawing
 ~ bezug im 2D-Script 186

DRAWING2 132
DRAWING3 133
DRAWING3{2} 129, 133
Drehen des Koordinatensystems 30
Drucken 221

E

Eckige Klammern in GDL Scripts 26
EDGE 99
Einfache Formen 15
Einstellung der Darstellungsform 157
Einstellung der Stiftfarbe 156
Einstellung des Schattenwurfs 160
Einstellung des Schraffurmusters
~ für Schnitte/Ansichten-Fenster 159
~ in 2D-Ansichten 161
ELBOW 61
Elementtyp in der Liste ändern 185
ELLIPS 34
ELSE 214
END 23, 216
Ende der Scriptdefinition 216
ENDGROUP 116
ENDIF 214
ENDWHILE 212
Erstellen von Freiflächen 90
EXIT 23, 216
EXOR 198
EXP 200
EXTRUDE 67
extrudierte Prismenerstellung prism generation 67

F

FILL 161, 168, 221
FILLA 171
FOR 211
fortgeschrittene Befehle und Funktionen 18
FPRISM_ 42
FRA 199
FRAGMENT2 13, 130
FULLPATH 298
Funktionsaufruf 202

G

GDL-Programmieren für Experten 19
gekrümmte Wand Definition 49
gekrümmtes Prisma Definition 41

geometrische Grundelemente 19
GET 216
Gitternetz Definition
gleichseitige ~ 59
Glättungsdefinition zylindrischer Elemente
~ durch Annäherung 156
~ durch Auflösung 155
~ durch Radius 154
globale Variablen 21, 25
globale Variablen von Benutzern 25
GOSUB 23, 214, 215
GOTO 23, 214, 215
Grundelemente der GDL-Syntax 23
Gruppieren 116

H

Halbiertes Ellipsoid Definition 34
Hauptkoordinatensystem 20
HIDEPARAMETER 189
HOTSPOT 135
HOTSPOT2 121, 135
HPRISM_ 44
Hybrid 114

I

Identifizierer 24
IF 214
in GDL Scripts verfügbare Zeichen 23
IND 202, 248
INPUT 222
INT 199

K

Kanten 97
Kantendefinition 99
Kegelstumpf Definition 36
KILLGROUP 117
Kommas in GDL-Scripts 23
Kommentare 14
komplexe Transformationsmatrix 31
Körper 97
Körper erstellt durch Zeichnen eines Polygonszuges
~ der auf einem Polygon der x-y-Ebene und einem
frei im Raum definierten Polygon auf-
baut 76
~ eines Pfades im Raum 78, 81, 85
Körperdefinition mit Grundelementen 102

Kreisdefinition 125
Kreuzungspunkte 97
Kugel Definition 34

L

LET 153
LGT 200
LIBRARY 298
Lichtquellendefinition 92
Light 92
LINE_TYPE 162, 175, 221
LINE2 121
Linientyp
~ Definition 175
~ einstellung 162
Linientypen 175
LOCK 189
LOG 201
lokale Variablen 24
lokales Koordinatensystem 20
loops 211
Ludolph Konstante 200

M

Macro-Objekte 220
~ Definition 220
Makro-Objekte 19
~ für Türen/Fenster 254
Maskenwerte 142
Maskierungsregeln
~ für Freiflächen 59
~ für Prismen 141
MASS 90
Massivbasis 114
Massivkörper 114, 157
MASTER_GDL 21, 25, 162, 186
MASTEREND_GDL 21
Master-Script 13
MATERIAL 158, 162, 221
Materialdefinition 163
Materialeinstellung 158
MAX 201
MESH 59
MIN 201
MOD 198
MODE 298
MODEL SURFACE 115

Modell 157, 221

MUL 29

MUL2 28

MULX 29

MULY 29

MULZ 29

N

Natürlicher Logarithmus 201

NEXT 211

nicht rechteckige Türen/Fenster

~ in gekrümmten Wänden 263

~ in geraden Wänden 256

NOT 201

NSP 217

NTR 32

Numerische Ausdrücke 25

O

Oberfläche 114, 157

Oberfläche der dreidimensionalen Form des Objektes 185

Oberflächenbasis 114

Oberflächenstruktur generieren 87

oder 198

OPEN 222

OUTPUT 223

P

Parameter 14, 25, 188, 220

~ in GDL-Scripts 19

~ speicher 216

Änderung ~ werte in GDL 188

Derivierte Typen 25

Einfache Typen 25

Schützen von ~ Werten 189

Parameter-Text 14

PEN 221

PGON 99

PI 200

PICTURE 14

PICTURE2 14, 128

PIPG 100

PLACEGROUP 117

planare Polylinien

geschlossener Kreis, definiert durch Mittelpunkt und Radius 149

Segment, definiert durch relativen Endpunkt 144

POLY2 122

POLY2_ 123

POLY2_A 124

POLY2_B 124

Polygondefinition 99, 122

erweiterte ~ 124

Polygonflächen 97

POSITION 185

PRISM 36

PRISM_ 37

Prismendefinition 36

~ mit Anhöhe 42

~ mit nicht-parallelem oberen Polygon 44

erweiterte ~ 40

erweiterte geneigte ~ 46

extrudierte allgemeine ~ 67

geneigte ~ 46

Programmierungssprache 13

PROJECT2 130

PROJECT2{2} 131

PROJEKT2{2} 131

Projektion eines 3D-Scripts in ein 2D-Symbol 131

prompt 26

PYRAMID 70

Pyramide-Definition 70

R

Radius 154, 221

RANGE 187

rechteckige Türen/Fenster

~ in geraden Wänden 254

RECT2 122

REF 184

Reihen für Parameter 25

REQ 202, 243

REQUEST 202, 244

RESOL 221

RETURN 215

RND 201

RO 298

Röhre Definition

~ beginnen von einer anderen Röhre 60

~ starten vom Ellipsoid 61

gebogene ~ 61

ROT 30

ROT2 28

Rotationskörper-Definition 72

ROTX 30

ROTY 30

ROTZ 30

RULED 75

RULED{2} 76

S

Schließen einer Datei 223

Schnittform definition 110

Schraffurdefinition

einfache ~ 169

Schreiben von Argumenten 221

Script-Typen 13

SECT_FILL 159

SECTGROUP 112, 116

SECTLINES 113, 116

SEPARATOR 298

158, 162

SET_FILL 161

SET_LINE_TYPE 162

SET_MATERIAL 158

SET_STYLE 157

SGN 199

SHADOW 160, 221

SIN 200

spezielle Zeichen 26

SPHERE 34

SPLINE2 126

SPLINE2_A 127

Spline-Definition 126

~des Bézier-Typs 128

Splitten 207

SPRISM_ 44

Sprungmarken 23

SQR 199

STEP 187, 211

STR 202

STR{2} 202

string expressions

~ aus numerischen Ausdrücken erstellen 202

Formattext 204

Länge von ~ 208

splitting ~ 207

Stellung ~ ineinander 208

Sub-Zeichenfolge 209

width of ~ 208

STRLEN 208

STRSTR 208

STRSUB 209
STW 208
STYLE 157, 176, 221
SUBGROUP 112, 116
subroutines 214
SURFACE3D 185
SWEEP 78
SWEEPGROUP 113, 119

T

TAN 200
TEVE 98
TEXT 95
TEXT2 129
Text-Definition
 ~ im 2D-Raum 129
 ~ im 3D-Raum 95
Texterweiterung 290
Textstil
 ~ einstellung 157
Texturdefinition 166
TEXTURE 166
THEN 214
TO 211
TOLER 156, 221
Transformationen
 ~ löschen 31
Transformationen des Koordinatensystems 27
 einfache ~ 16
 Weiterführende 17
TUBE 81
TUBEA 85

U

UI_BUTTON 190
UI_DIALOG 189
UI_GROUPBOX 190
UI_INFIELD 191
UI_NEXT 190
UI_OUTFIELD 191
UI_PAGE 189
UI_PREV 190
UI_SEPARATOR 190
UI_STYLE 191
Unterprogramm 215
Unterzug Definition 55
UNTIL 213
USE 217

V

VARDIM1 196
VARDIM2 196
variable 26
Variablen 24
VARTYPE 222
VECT 98
Vektor 97
 ~ Definition 98
Vektorschraffur 170
VERT 98
Volumen der dreidimensionalen Form des Objektes 185
Vorschaubild 14

W

WA 298

WALLHOLE 256
WALLNICHE 260
Wand Definition 47
 erweiterte ~ 52
 gekrümmte ~ 49
Wandabschluß 229
weiterführende Befehle 16
Werte 187
Werte aus Datei importieren 222
Werte im Parameterspeicher ablegen 216
Werte in eine Datei exportieren 223
Wertelisten 21, 25
Wertzuweisung 153
WHILE 212
WIRE 157
WO 298

X

XFORM 30
XWALL_{2} 54

Z

Zähler neu setzen
 ~ für einfache Körper 104
Zeichenfolgen 24
Zeichnung
 ~ definitionen für Elementlisten 132
Zeilen in Scripts 23
Zweidimensionales Symbol
 Türen/Fenster erstellen~ 253
Zylinder definition 34