ARCHICAD GDL Referenzhandbuch

GRAPHISOFT®
A NEMETSCHEK COMPANY

GRAPHISOFT®

Besuchen Sie die GRAPHISOFT Website bei *http://www.graphisoft.com* [https://www.graphisoft.com] für Informationen über ortsnahe Verkäufer und Verfügbarkeit der Produkte.

ARCHICAD GDL Referenzhandbuch

Copyright© 2019 by GRAPHISOFT, alle Rechte vorbehalten. Die Reproduktion, Änderung, Umschreibung oder Übersetzung ohne vorherige schriftliche Genehmigung ist strengstens verboten.

Warenzeichen

ARCHICAD® ist ein eingetragenes Warenzeichen von GRAPHISOFT. Alle anderen Warenzeichen sind Warenzeichen ihrer entsprechenden Eigentümer.

Einführung

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

Inhaltsverzeichnis

GDL Programmierungs-Grundlagen 1
Mit dem Übungshandbuch beginnen 1
Scripting 1
Wie das 3D-Bild erzeugt wird 8
Grundelemente der GDL-Syntax 11
Regeln der GDL Syntax 11
Anweisungen 11
Zeilen 11
Sprungmarke 11
Zeichen 12
Zeichenfolgen 12
Identifizierer 12
Variablen 13
Parameter 13
Einfache Typen 13
Derivierte Typen 14
Strukturierte Typen 14
In diesem Buch verwendete Konventionen 15
Transformationen des Koordinatensystems 17
2D-Transformationen 17
ADD2 17
MUL2 18
ROT2 18
3D-Transformationen 18
ADDX 18
ADDY 18
ADDZ 19
ADD 19
MULX 19
MULY 19

MULZ 19
MUL 20
ROTX 20
ROTY 20
ROTZ 20
ROT 20
XFORM 21
Verwaltung des Transformation-Stacks 21
DEL 21
DEL TOP 22
NTR 22
Dreidimensionale Elemente 24
Grundkörper 24
BLOCK 24
BRICK 24
CYLIND 25
SPHERE 25
ELLIPS 26
CONE 27
PRISM 27
PRISM_ 28
CPRISM_ 31
CPRISM_{2} 32
CPRISM_{3} 33
CPRISM_{4} 36
BPRISM_ 36
FPRISM_ 38
HPRISM_ 40
SPRISM_ 40
SPRISM_{2} 42
SPRISM_{3} 43
SPRISM_{4} 44
SLAB 45

SLAB_ 45
CSLAB_ 46
CWALL_ 46
BWALL_ 50
XWALL_ 52
XWALL_{2} 54
XWALL_{3} 55
BEAM 58
CROOF_ 58
CROOF_{2} 61
CROOF_{3} 62
CROOF_{4} 63
MESH 63
ARMC 65
ARME 66
ELBOW 67
Flächen-Gestalten in 3D 68
HOTSPOT 68
HOTLINE 68
HOTARC 69
LIN_ 69
RECT 69
POLY 69
POLY_ 70
PLANE 71
PLANE_ 71
CIRCLE 71
ARC 72
Körper aus Linienzügen 72
EXTRUDE 74
PYRAMID 77
REVOLVE 79
REVOLVE{2} 85

REVOLVE{3} 86
REVOLVE{4} 88
REVOLVE{5} 88
RULED 88
RULED{2} 88
RULEDSEGMENTED 92
RULEDSEGMENTED{2} 93
SWEEP 94
TUBE 97
TUBE{2} 101
TUBEA 103
COONS 106
COONS{2} 109
MASS 109
MASS{2} 112
POLYROOF 113
POLYROOF{2} 118
POLYROOF{3} 119
POLYROOF{4} 121
EXTRUDEDSHELL 121
EXTRUDEDSHELL{2} 122
EXTRUDEDSHELL{3} 124
REVOLVEDSHELL 124
REVOLVEDSHELL{2} 125
REVOLVEDSHELL{3} 127
REVOLVEDSHELLANGULAR 127
REVOLVEDSHELLANGULAR{2} 128
REVOLVEDSHELLANGULAR{3} 128
RULEDSHELL 129
RULEDSHELL{2} 131
RULEDSHELL{3} 134
Elemente zur Unterstützung der Photorealistik 134
LIGHT 134

PICTURE 139
3D-Textelemente 140
TEXT 140
RICHTEXT 141
Grundelemente 141
VERT 142
VERT{2} 143
TEVE 143
VECT 143
EDGE 143
PGON 144
PGON{2} 145
PGON{3} 145
PIPG 145
COOR 145
COOR{2} 147
COOR{3} 148
BODY 150
BASE 153
NURBS Primitiv-Elemente 154
NURBS-Außenflächen-Anschnitte 155
NURBS Geometrie Befehle 155
NURBSCURVE2D 156
NURBSCURVE3D 156
NURBSSURFACE 157
NURBS Mathematische Befehle 158
NURBSVERT 158
NURBSEDGE 158
NURBSTRIM 159
NURBSTRIMSINGULAR 159
NURBSFACE 160
NURBSFACE{2} 161
NURBSLUMP 161

NURBSBODY 162
Punktwolken 162
POINTCLOUD 162
Verschneiden im 3D-Raum 163
CUTPLANE 163
CUTPLANE{2} 163
CUTPLANE{3} 163
CUTPOLY 167
CUTPOLYA 169
CUTSHAPE 172
CUTFORM 172
CUTFORM{2} 174
Solid-Element-Befehle 174
GROUP - ENDGROUP 179
ADDGROUP 179
SUBGROUP 179
ISECTGROUP 179
ISECTLINES 180
PLACEGROUP 180
KILLGROUP 180
SWEEPGROUP 181
CREATEGROUPWITHMATERIAL 183
Binäres 3D 183
BINARY 183
2D Elemente 185
Zeichnungselemente 185
HOTSPOT2 185
HOTLINE2 185
HOTARC2 186
LINE2 186
RECT2 186
POLY2 187
POLY2_ 188

POLY2_A 189
POLY2_B 189
POLY2_B{2} 189
POLY2_B{3} 190
POLY2_B{4} 190
POLY2_B{5} 191
POLY2_B{6} 192
ARC2 192
CIRCLE2 192
SPLINE2 193
SPLINE2A 195
PICTURE2 196
PICTURE2{2} 196
Textelement 196
TEXT2 196
RICHTEXT2 197
Binäres 2D 197
FRAGMENT2 197
3D Projektionen in 2D 198
PROJECT2 198
PROJECT2{2} 198
PROJECT2{3} 201
PROJECT2{4} 203
Zeichnungen in der Liste 205
DRAWING2 206
DRAWING3 206
DRAWING3{2} 206
DRAWING3{3} 206
Bearbeitungsbefehle auf Hotspot-Basis 208
Statuscodes 216
Statuswert-Syntax 216
Zusätzliche Statuscodes 217
Vorgegebener erster Teil eines Polygonzuges: aktuelle Position und Tangente ist definiert 218

Segment, definiert durch den absoluten Endpunkt 218
Segment, definiert durch den relativen Endpunkt 219
Segment, definiert durch Längen- und Richtungsangabe 219
Tangentiales Segment, definiert durch Längenangabe 219
Angabe des Startpunktes 220
Schließen des Polygonzuges 220
Angabe der Tangente 220
Angabe des Mittelpunktes 221
Tangentialer Bogen zum Endpunkt 221
Tangentialer Bogen, definiert durch Radius und Winkel 222
Kreisbogen, definiert durch Mittelpunkt und Punkt auf der Kreislinie (letzter Radius) 222
Kreisbogen, definiert durch Mittelpunkt und Winkel 223
Geschlossener Kreis, definiert durch Mittelpunkt und Radius 223
Attribute 229
Anweisungen 229
Anweisungen für 3D- und 2D-Scripts 229
LET 229
RADIUS 229
RESOL 230
TOLER 231
PEN 232
LINE_PROPERTY 233
[SET] STYLE 233
Anweisungen nur für 3D-Scripts 233
MODEL 233
[SET] MATERIAL 234
[SET] BUILDING_MATERIAL 235
SECT_FILL 236
SECT_ATTRS 236
SECT_ATTRS{2} 236
SHADOW 237
Anweisungen nur für 2D-Scripts 238
DRAWINDEX 238

[SET] FILL 239
[SET] LINE_TYPE 239
Inline Attributdefinition 239
Materialien 240
DEFINE MATERIAL 240
DEFINE MATERIAL BASED_ON 242
DEFINE TEXTURE 243
Schraffuren 246
DEFINE FILL 246
DEFINE FILLA 249
DEFINE SYMBOL_FILL 252
DEFINE SOLID_FILL 253
DEFINE EMPTY_FILL 253
DEFINE LINEAR_GRADIENT_FILL 253
DEFINE RADIAL_GRADIENT_FILL 253
DEFINE TRANSLUCENT_FILL 253
DEFINE IMAGE_FILL 254
Linientypen 254
DEFINE LINE_TYPE 254
DEFINE SYMBOL_LINE 255
Text und Stile 255
DEFINE STYLE 255
DEFINE STYLE{2} 256
PARAGRAPH 257
TEXTBLOCK 258
TEXTBLOCK_ 259
Zusätzliche Daten 259
Externe Dateiabhängigkeit 260
FILE_DEPENDENCE 260
Nicht geometrische Scripts 262
Das Eigenschaften-Script 262
DATABASE_SET 262
DESCRIPTOR 263

REF DESCRIPTOR 263
COMPONENT 263
REF COMPONENT 264
BINARYPROP 264
SURFACE3D 264
VOLUME3D 264
POSITION 265
DRAWING 266
Das Parameter-Script 266
VALUES 266
VALUES{2} 267
PARAMETERS 268
LOCK 269
HIDEPARAMETER 269
Die Benutzeroberfläche (User Inteface Script) 269
UI_DIALOG 270
UI_PAGE 270
UI_CURRENT_PAGE 271
UI_BUTTON 271
UI_PICT_BUTTON 272
UI_SEPARATOR 272
UI_GROUPBOX 273
UI_PICT 273
UI_STYLE 273
UI_OUTFIELD 274
UI_INFIELD 274
UI_INFIELD{2} 275
UI_INFIELD{3} 275
UI_INFIELD{4} 275
UI_CUSTOM_POPUP_INFIELD 283
UI_CUSTOM_POPUP_INFIELD{2} 283
UI_RADIOBUTTON 286
UI_RADIOBUTTON{2} 286

UI_PICT_RADIOBUTTON 287
UI_PICT_RADIOBUTTON{2} 287
UI_PICT_PUSHCHECKBUTTON 287
UI_PICT_PUSHCHECKBUTTON{2} 287
UI_TEXTSTYLE_INFIELD 288
UI_TEXTSTYLE_INFIELD{2} 288
UI_LISTFIELD 289
UI_LISTITEM 290
UI_LISTITEM{2} 290
UI_CUSTOM_POPUP_LISTITEM 291
UI_CUSTOM_POPUP_LISTITEM{2} 292
UI_TOOLTIP 294
UI_COLORPICKER 295
UI_COLORPICKER{2} 295
UI_SLIDER 296
UI_SLIDER{2} 296
Das Vorwärts-Migration Script 296
SETMIGRATIONGUID 297
STORED_PAR_VALUE 298
DELETED_PAR_VALUE 298
Das Rückwärts-Migration Script 298
NEWPARAMETER 300
Ausdrücke und Funktionen 301
Ausdrücke (Expressions) 301
DICT 301
HASKEY 306
REMOVEKEY 306
DIM 307
VARDIM1 308
VARDIM2 308
PARVALUE_DESCRIPTION 311
Operatoren 311
Arithmetische Operatoren 311

Relationale Operatoren 312
Boolesche Operatoren 312
Funktionen 313
Arithmetische Funktionen 313
ABS 313
CEIL 313
INT 313
FRA 313
ROUND_INT 313
SGN 313
SQR 313
Winkelfunktionen 314
ACS 314
ASN 314
ATN 314
COS 314
SIN 314
TAN 314
PI 314
Transzendentale Funktionen 315
EXP 315
LGT 315
LOG 315
Boolesche Funktionen 315
NOT 315
Statistische Funktionen 315
MIN 315
MAX 315
RND 316
Bit-Funktionen 316
BITTEST 316
BITSET 316
Spezielle Funktionen 316

REQ 316
REQUEST 317
IND 317
APPLICATION_QUERY 318
LIBRARYGLOBAL 318
String-Funktionen 319
STR 319
STR{2} 319
SPLIT 323
STW 323
STRLEN 324
STRSTR 324
STRSUB 325
STRTOUPPER 325
STRTOLOWER 325
Befehle zur Programmsteuerung 327
Befehle zur Programmsteuerung 327
FOR - TO - NEXT 327
DO - WHILE 328
WHILE - ENDWHILE 328
REPEAT - UNTIL 329
IF - GOTO 330
IF - THEN - ELSE - ENDIF 331
GOTO 332
GOSUB 332
RETURN 332
END / EXIT 333
BREAKPOINT 333
Arbeiten mit dem internen Parameterspeicher 333
PUT 334
GET 334
USE 334
NSP 334

Makro-Objekte 337
CALL 337
Ausgabe in einem Popup-Fenster oder in einem Report-Fenster 339
PRINT 339
Datei Operationen 340
OPEN 340
INPUT 340
VARTYPE 341
OUTPUT 341
CLOSE 341
Verwenden von Deterministischen Add-Ons 341
INITADDONSCOPE 342
PREPAREFUNCTION 342
CALLFUNCTION 342
CLOSEADDONSCOPE 342
Verschiedenes 343
Globale Variablen 343
Script Kompatibilität 343
Allgemeine Umgebungsinformationen 344
Geschossinformationen 349
Animationsinformationen 350
Allgemeine Parameter von Elementen 352
Parameter von Objekt, Lampe, Tür, Fenster, Wandende Dachfenster 352
Parameter von Objekt, Lampe, Tür, Fenster, Wandende, Dachfenster, Fassadenzubehör, - nur für Listen und Etiketten verfügbar 353
Parameter von Objekten, Lampen und Fassadenzubehör - nur für Listen und Etiketten verfügbar 354
Öffnungsparameter - nur für Listen und Etiketten verfügbar 354
Parameter des Öffnungssysmbols 355
Parameter von Fenstern, Türen und Wandenden 359
Parameter von Fenstern und Türen - nur zum Auflisten und für Etiketten verfügbar 360
Parameter von Lampen - nur zum Auflisten und für Etiketten verfügbar 361
Marker-Parameter (Detail, Arbeitsblatt und Änderungswolken) 361
Bemaßungsparameter 362

Wandparameter, verfügbar für Fenster/Türen, Listen und Etiketten 363
Parameter von Wänden - nur zum Auflisten und für Etiketten verfügbar 366
Parameter von Stützen - nur zum Auflisten verfügbar 368
Trägerparameter - nur zur Auflistung verfügbar 374
Parameter von Decken - nur zum Auflisten verfügbar 379
Treppenkomponentenparameter 382
Allgemeine Treppenvariablen - verfügbar für Auswertungen und Etiketten 382
Allgemeine Auftrittvariablen - für Auflistung und Etiketten verfügbar 384
Allgemeine Steigungs-Variablen - verfügbar für Auflistung und Etiketten 385
Treppenstrukturvariablen - verfügbar für Auflistung und Etiketten 386
Modelldarstellungs-Variablen für Treppen 386
Treppen-2D-Variablen - nur für Grundrissdarstellung verfügbar 387
Treppenrastervariablen 388
Symbolvariablen der Lauflinie der Treppe 391
Symbolvariablen der Bruchlinien-Symbole 394
Beschreibungsvariablen von Treppenläufen und Podesten 395
2D-Variablen der Treppen-Abläufe 395
Treppenstruktur 2D-Variablen - Holm-Strukturen 397
Treppenstruktur 2D-Variablen - Massive Struktur 398
Allgemeine 2D zugehörige Variablen 402
Treppen-3D-Variablen - verfügbar nur für 3D-Darstellung (und Verbindungspunkte) 405
3D-Variablen für Setzstufe der Treppen 405
2D-3D Variablen der Trittstufe 406
Strukturvariablen der Treppe 409
Geländerkomponentenparameter 411
Allgemeine Geländervariablen - für Auswertungen und Etiketten verfügbar 411
Geländer 3D-Variablen 412
2D-Variablen beim Geländer 418
Dachparameter - verfügbar für Dachfenster, Listen und Etiketten 423
Parameter von Dächern - nur zum Auflisten verfügbar 425
Schraffurparameter - nur zum Auflisten verfügbar 426
Parameter der Freiflächenform - nur zum Auflisten verfügbar 427
Komponenten-Parameter des Fassadenwerkzeugs 429

Parameter von Fassaden - nur zum Auflisten und für Etiketten verfügbar 430
Profilparameter der Fassade 431
Allgemeine Profilvariablen der Fassade - nur für Listen und Etiketten verfügbar 431
3D-Variablen von Fassadenprofilen 432
Paneelvariablen der Fassade 434
Paneelparameter von Fassaden - nur zum Auflisten und für Etiketten verfügbar 434
Halterungsparameter von Fassaden - nur zum Auflisten und für Etiketten verfügbar 435
Zubehör-Parameter von Fassaden - nur zum Auflisten und für Etiketten verfügbar 435
Migrationsparameter - nur für die Migrations-Scripte verfügbar 435
Dachfenster-Parameter- nur für Etiketten und Listen verfügbar 436
Allgemeine Parameter von Schalen und Dachflächen - nur verfügbar für Listen und Etiketten 436
Parameter für Morphs - nur für Listen und Etiketten verfügbar 441
Benutzerdefinierbare globale Variablen 442
Beispielhafte Anwendung von Globalen Variablen 443
Veraltete Globale Variablen 443
Veraltete Globale Träger- / Stützenvariablen - nur für Listen und Etiketten verfügbar 444
Veraltete globale Etiketten-Variablen 444
Veraltete globale Variable der Fassadenprofile- verfügbar für nur Auflistung und Etiketten 446
Alte Globale Variablen 446
Festbenannte optionale Parameter 448
Parameter festgelegt von ARCHICAD 448
Parameter für D/W Attribute (verfügbar für Türen (D), Fenster (W), Etiketten und Listen) 449
Grundrissdarstellung 449
Richtung 449
Daten einer Polygonalwand 450
Position der Öffnung 450
Ankerdaten 450
Parameter für Wand-Attribute (verfügbar für Türe, Fenster, Etikett, Liste) 450
Grundrissdarstellung 450
Geometriedaten 451
Parameter für Stützen-Attribute (verfügbar für Etikett, Liste) 451
Grundrissdarstellung 452
Geometrische Daten 453

Parameter für Träger-Atribute (verfügbar für Etiketten, Listen) 453
Grundrissdarstellung 453
Geometrische Daten 454
Parameter für Dach-Attribute (verfügbar für Etikett, Liste) 455
Grundrissdarstellung 455
Tür/Fenster-Marker Attribute 455
Detail/Arbeitsblatt Marker Attribute 457
Zeichnungstitel-Attribute 457
Allgemein laufender Kontext 458
Raumparameter (verfügbar für Raumstempel) 459
Treppenbezogene Parameter 461
Unterstützte Subtypen von Lauf/Podest 461
Subtype für Setzstufen-Komponenten 461
Subtypen für 2D-Treppenkomponenten 461
Parameters, welche ARCHICAD liest und schreibt 461
Treppenrelevante Parameters 461
Subtyp für Struktur 461
Parameter von ARCHICAD ausgelesen 462
Objekte im Grundriss 462
Beschneiden von planaren Elementen im Grundriss (z.B. Dachfenster, Dachzubehörobjekte) 462
Tür/Fenster - Objekte 463
Eigene-Komponenten-Vorlage 464
Treppenrelevante Parameter 464
Subtyp der Struktur 464
Subtyp für Lauf / Podest: unterstützte Struktur 464
Subtypen für Treppenlauf-unterstützende Auskragungen / Podest-unterstützende Auskragungen 465
Geländerbezogene Parameter 465
Subtyp für Geländer-Paneel-Komponenten 465
Subtyp für Geländer-Gurt-Komponenten 465
Subtyp für Geländerpfosten-Komponenten 466
Subtyp für Geländer-Abschlusskomponente 467
Parameter für Fassaden 468
Fassaden-Parameter, welche ARCHICAD schreibt und liest 468

Fassadenprofil-Parameter 468
Fassaden-Parameter, welche ARCHICAD schreibt 469
Fassaden-Profil-Parameters 469
Parameter des Fassaden-Paneels 469
Parameter von Fassaden-Halterungen 470
Zubehörparameter von Fassaden 470
Veraltete Parameter von Fassaden-Profilen 471
Von ARCHICAD ausgelesene Fassaden-Parameter 472
Fassadenpaneel- und Fassadenprofil-Parameter 472
Parameter von Fassadenprofilen 473
Parameter von Fassaden-Paneelen 474
Parameter für Addons 475
Parameter des Dachfenster-Addons 475
Lochkanten-Schneidemanipulation 475
Parameter des Eckfenster-Addons 475
Grundparameter des Eckfenster-Objektes 475
Datenparameter von Wandschichten bei Eckfensterobjekten (verfügbar ab ARCHICAD 12) 476
Parameter des IFC Addons 476
Gemeinsame Grundparameter der Tür-und Fensterobjekte 476
Grundparameter von Türobjekten 478
Grundparameter von Fenster-Objekten 481
Grundparameter von Transportelementen 484
Grundparameter von Aufzugs-Objekten 484
Grundparameters von Treppenobjekten 485
Grundparameter von MEP-Elementen 487
Parameter für die Textverarbeitung 488
Parameter für Etiketten 490
Parameter, welche von ARCHICAD gelesen oder geschrieben werden 490
Parameter von ARCHICAD ausgelesen 492
Veraltete Parameter 493
Veraltete Träger- / Stützenparameter - nur für Auflistung und Etiketten verfügbar 493
Veraltete Raumstempel-Parameter 493
REQUEST Optionen 493

Request Parameter-Script Kompatibilität 494
Details der Requests 500
Profil-Requests 523
Veraltete Requests 526
Application Query Optionen 526
Dokumenteigenschaften 526
Blickrichtung 526
MEP-Systeme 527
Erfrage die MEP-Systeme 527
Abfrage der Domain 527
Abfrage Konturstift 528
Abfrage Schraffurstift 528
Abfrage Hintergrundstift 528
Abfrage Schraffurtyp 528
Abfrage Linentyp Achse 528
Abfrage Stift Achse 529
Abfrage System-Material 529
Abfrage Dämmmaterial 529
MEP Modeler 529
Ist verfügbar 529
MEP Verbindungstyp 530
Abfrage der Verbindungstypen 530
Abfrage des Stiles des Verbindungstypes 530
MEP Flexibles Segment 530
Start des Abschnitts 530
Füge Kontrollpunkt hinzu 531
Füge Richtungs- und Breiten-Vektor hinzu 531
Endverteiler 531
MEP Bögen 532
Start des Abschnitts 532
Parameter-Script 533
Erster Durchlauf ist im Gange 533
Tags und Kategorien 533

Abfrage Parameter-Ordnernamen 533
Abfrage der Parameternamen 534
Abfrage der Parameter 534
Bibliotheken-Manager 534
IES-Dateien 534
Benutzerbilddateien 534
GDL Style Guide 535
Einführung 535
Namensgebung-Konvention 535
Allgemeine Regeln 535
Variablennamen 536
Groß- und Kleinschreibung 537
Ausdrücke (Expressions) 537
Kontrollfluss-Ausdrücke 538
if - else - endif 538
for - next, do - while, while - endwhile, repeat - until 539
Subroutinen 540
Kommentare schreiben 541
Script-Kopf 541
Abschnittsbegrenzung 542
Scriptstruktur 544
Schlechte Lösung 545
Gute Lösung 546
Grundlegende Technische Standards 546
Einführung 546
Bibliothekselement-Format 547
Dateierweiterung 547
Identifizierung 547
Allgemeine Scriptingfragen 551
Numerische Typen - Präzision 551
Trigonometrische Funktionen 553
GDL-Warnungen 553
Hotspot und Hotline IDs 555

Zweck der hotspot / hotline / hotarc Identifikation 555
Problem von hotspots/hotlines der alten Schule 556
Korrektes Scripting für hotspot/hotline/hotarc 556
Editierbare Hotspots 556
Beispiel Editierbare hotspots - Schuh / Schuhregal 556
GDL-Ausführungs-Kontext 558
Werteaustausch zwischen Objekt und ARCHICAD 560
Informationsfluss von ARCHICAD zum Objekt 560
Globale Variablen 560
Festbenannte optionale Parameter 560
REQUESTs und APPLICATION QUERYs 561
Informationen, die aus einem Bibliothekselement kommen 561
Modelldarstellung (MVO), "Library Global" 561
Interne Modelldarstellungs-Optionen 561
Ansichtsoptionen mittels "Library Global" 561
Scripttyp spezifische Fragen 562
Master-Script 562
2D-Script 562
Ausführungs-Kontext 562
Allgemeine Empfehlung 563
Definieren von Eigenschaften von Linien und Schraffuren 563
3D-Script 564
Ausführungs-Kontext 564
Allgemeine Empfehlung 565
Modellieren transparenter Körper 565
Textur-Mapping 567
Bildelemente 573
Gruppenoperationen 575
Parameter-Script 575
Ausführungs-Kontext 575
Allgemeine Empfehlung 576
Schriftart Namen 577
Einstellungsbeschränkungen von Array-Parametern 577

User Interface Script 577
Ausführungs-Kontext 577
Allgemeine Empfehlung 578
Vorschau-Steuerbilder 578
Handhabung der Tab-Seiten 579
Vorschaubild-Auswahl mit dynamischen Elementen 582
Transparente UI-Grafiken 583
Fontgrößen im UI 584
Das Vorwärts-Migration Script 585
Ausführungs-Kontext 585
Allgemeine Empfehlung 585
Das Rückwärts-Migration Script 586
Ausführungs-Kontext 586
Allgemeine Empfehlung 587
Migrationstabelle 588
Das Schreiben von Makros 589
Makro Rückgabeparameter 589
Erweitertes "parameters all" 590
Schnellerer Makroaufruf 590
Beispiel Makroaufruf 590
Probleme bei der Rückwärtskonvertierung 590
Geschwindigkeits-Betrachtungen 591
Windows-Macintosh Kompatibilität 592
7.2 Wechseln der Plattform mit Binär-Bibliotheken 592
Bilder und HDPI-Unterstützung in GDL 592
Türen und Fenster 593
GDL Programmierungs-Grundlagen 593
Positionierung 594
Erstellung von Bibliothekselementen des Türen/Fenster-Typs 596
Rechteckige Türen/Fenster in geraden Wänden 597
3D-bezogene Aufgaben 600
Nicht rechteckige Türen/Fenster in geraden Wänden 600
WALLHOLE 600

WALLNICHE 603
Rechteckige Türen/Fenster in gekrümmten Wänden 604
Nicht rechteckige Türen/Fenster in gekrümmten Wänden 606
2D-bezogene Aufgaben 609
Individuelle Wandöffnung schneiden 609
WALLHOLE2 609
WALLHOLE2{2} 610
Das Wandpolygon erweitern 611
WALLBLOCK2 611
WALLBLOCK2{2} 611
WALLLINE2 611
WALLARC2 611
Grafische Erzeugung von GDL-Objekten im Grundriss 612
Befehle 612
Häufige Befehle 612
Reservierte Befehle 615
Befehle für den 3D-Bereich 616
Befehle für den 2D-Bereich 621
Befehle für den 2D- und 3D-Bereich 623
Nicht geometrische Scripts 623
Eigenschaften-Script 623
Parameter-Script 624
Interface-Script 625
Vorwärts- und Rückwärts-Migration Scripts 626
GDL Data I/O Add-On 626
Beschreibung der Datenbank 626
Öffnen einer Datenbank 626
Lesen der Werte aus der Datenbank 628
Eingeben von Werten in die Datenbank 628
Datenbank schließen 629
GDL Datetime Add-On 629
Kanal öffnen 630
Lesen der Information 631

Schließen des Kanals 631
GDL File Manager I/O Add-On 632
Spezifikation des Ordners 632
Datei/Ordnernamen erhalten 632
Beenden Ordner Scanning 633
GDL Text I/O Add-On 633
Datei öffnen 634
Lesen der Werte 635
Schreiben der Werte 635
Schließen einer Datei 636
Eigenschaften GDL Add-On 637
Öffnet die Eigenschaften-Datenbank 637
Schliesst die Eigenschaften-Datenbank 638
Eingabe in die Eigenschaften-Datenbank 638
Ausgabe in die Eigenschaften-Datenbank 641
GDL XML Erweiterung 641
Das XML-Dokument öffnen 643
Das XML-Dokument lesen 644
Das XML-Dokument ändern 648
Polygon-Operationserweiterung 652
Einen Kanal öffnen 653
Container-Management 653
CreateContainer 653
DeleteContainer 653
EmptyContainer 653
SetSourceContainer 653
SetDestinationContainer 653
Polygon / Polylinien-Management 653
Array 654
Dictionary 654
Store 655
StorePolyline 656
StoreDictPolygon 656

StoreDictPolyline 657
Dispose 657
Polygon / Polylinien-Operations-Einstellungen 657
HalfPlaneParams 657
OffsetParams 657
MultipleEdgeOffsetParams 658
PolylineOffsetVectors 658
Polygon / Polylinien-Operationen 658
+ - / 658
ClipPolyline 659
CopyPolygon 659
Regularize 659
PolyCut 659
OffsetEdge 660
OffsetMultipleEdges 660
OffsetPolyline 660
OffsetPolylineWithVectors 660
ResizeContour 660
CentreOfGravity 661
Ergibt die resultierenden Polygone / Polylinien 661
Array 661
GetSourcePolygons, GetSourcePolylines 661
GetDestinationPolygons, GetDestinationPolylines 661
GetVertices, GetPolylineVertices 661
GetContourEnds 661
GetInhEdgeInfos, GetPolylineInhEdgeInfos 662
Dictionary 662
GetSourceDictPolygon, GetSourceDictPolyline 662
GetDestinationDictPolygon, GetDestinationDictPolyline 662
Schließen des Kanals 662
Autotext-Führer 663
Projectinfo Schlüsselworte 663
Allgemeines 664

Layout Autotexte 664
Zeichnungs-Autotexte 665
Referenztyp Autotexte 665
Markertyp Atotexte 665
Änderungsbedingte Autotexte 666
Layout Revisions-bezogene Autotexte 666
Neue GDL Features in ARCHICAD 23 666
Neuer Datentyp: dictionary 666
Neue Funktionen im PolyOperations-Addon 667
Neues Werkzeug: Öffnung 668
Aktualisierte Werkzeuge: Träger und Stütze 668
Neue Eigenschaften-Requests 669
Diagnosemodus 669
Befehls-Updates und erweiterte Versionen 670
IFC4 670
Neues Quellformat mit dem LP_XMLConverter 671
Index 672
Syntax-Liste aller GDL-Befehle 672

GDL Programmierungs-Grundlagen

GDL (Geometric 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 im Grundriss darstellen. Diese Elemente werden als **Bibliothekselemente** bezeichnet.

Mit dem Übungshandbuch beginnen

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-Scripte 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.

Mehrere Bücher und Materialien wurden zur GDL-Programmierung und zur Entwicklung von Objektbibliotheken veröffentlicht.

- *"Einführung in die Objekterstellung mit ARCHICAD"* ist eine hervorragende Anleitung für Einsteiger.
- Der e-Guide *"Creating GDL Objects"* bietet einen einführenden Überblick über die Methoden der GDL-Objekterstellung.
- Das *"GDL Kochbuch"* von David Nicholson Cole ist ein beliebtes Kursbuch für Einsteiger und fortgeschrittene GDL-Programmierer.
- Eine aktuelle Quelle zum Vertiefen von GDL ist das *"GDL Handbook"* von Andrew Watson, geeignet sowohl für Anfänger als auch erfahrene Anwender.
- *"GDL Advanced Technical Standards"* enthält die offiziellen GRAPHISOFT Standards für professionelle Bibliotheksentwickler; Dieses Dokument kann von GRAPHISOFT's Website nach einer Registrierung heruntergeladen werden: *https://www.graphisoft.com/support/developer/*. Als Richtlinie für grundlegende GDL-Programmierung dient „Grundlegende Technische Standards" dieses Referenzhandbuch.

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.

Die Befehle des **Masterscriptes** 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 FRAGMENT2 Befehl aufgerufen werden. Ist das 2D-Script leer, werden die binären 2D-Daten zur Darstellung des Bibliothekselementes im Grundriss verwendet.

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 Abschnitten **Bestandteile** und **Beschreibungen** festgelegten binären **Beschreibungsdaten** können durch den Befehl BINARYPROP aufgerufen werden. Ist das Beschreibungsscript und das Master-Script leer, dann werden die binären Beschreibungsdaten zur Erzeugung der Listen benutzt.

Im **Benutzeroberfläche-Script (User Interface Script)** kann eine graphische aufbereitete Eingabemaske der Parameter 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.

Im **Vorwärts-Migration Script** kann man die Konvertierungslogik definieren, welche platzierte Instanzen von älteren Elementen konvertieren kann.

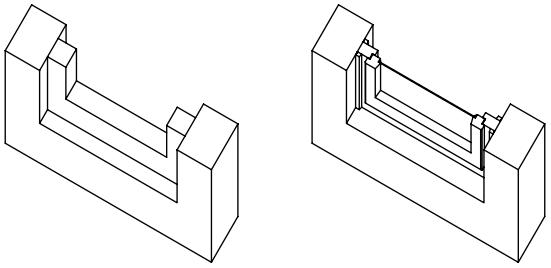
Im **Rückwärts-Migration Script** kann man die Rückwärts-Konvertierung zu einer älteren Version eines Elementes definieren.

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

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

Analysieren, Zerlegen und Vereinfachen

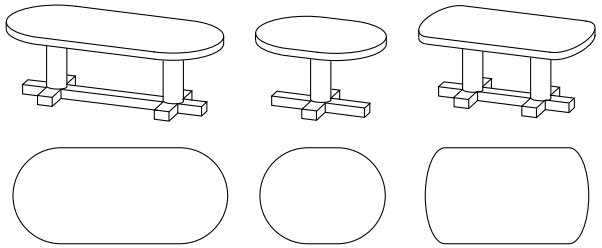
Wie komplex sie auch immer sein mögen, die meisten Objekte, die Sie erstellen wollen, lassen sich in einfache geometrische Grundkörper zerlegen. Beginnen Sie mit einer kurzen Analyse und definieren Sie die geometrischen Einzelkörper, aus denen das zu erstellende Objekt besteht. Ist das Objekt aufgeteilt, 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 Projektphase 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 textlichen Parametern, die die Dimensionen bestimmen.

Die Anzahl der Werte ist von Befehl zu Befehl 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:

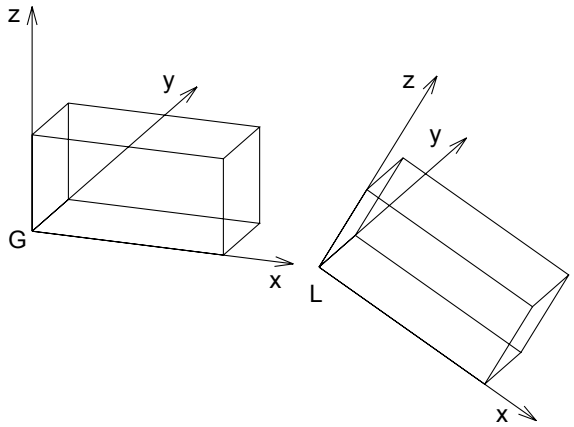
`BLOCK`, `CYLIND`, `SPHERE`, `PRISM`

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:

Meist verwendete Transformationen

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

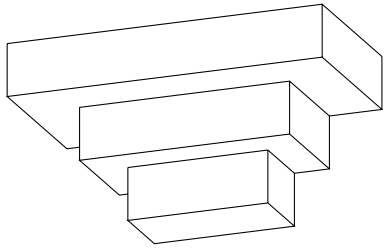
Für das 2D-Symbol:

Meist verwendete Transformationen

`MUL2`

Beispiel:

```
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, SPLINE2A

Befehle zur Programmsteuerung

FOR - TO - NEXT

DO - WHILE, WHILE - ENDWHILE

REPEAT - UNTIL

IF - THEN - ELSE - ENDIF

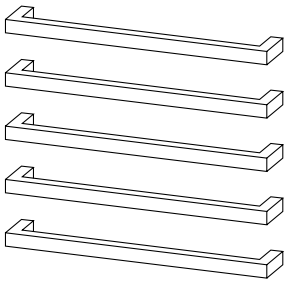
GOTO, GOSUB

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,
            -0.45, -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 Handbuches. 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 ihres Projektes, das Sie auf dem Bildschirm sehen, besteht aus geometrischen Grundelementen. Diese sind im Arbeitsspeicher Ihres Rechners in binärer Form gespeichert. Die 3D-Engine erzeugt die 3D-Darstellung aus den Elementen im 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 ± 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.

Transformationen des Koordinatensystems

Alle geometrischen Elemente in GDL sind fest an ihr eigenes, lokales 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 der Programmteil, welcher 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 Einsetzposition des Objektes verschoben, um die weiteren GDL-Befehle aus dem Script des Bibliothekselementes abarbeiten 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 hat, wird das ganze 3D- Binär-Modell in dem freien Arbeitsspeicher gespeichert, wo Sie 3D-Projektionen, photorealistische Darstellungen oder Sonnenstudien erzeugen können.

ARCHICAD enthält einen Precompiler und einen Interpreter für GDL. Die Übersetzung von GDL-Scripten 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 im 3D-Bereich des Bibliothekselementes in Binär-Form abgelegt. Der BINARY-Befehl bezieht sich auf diese Datenstrukturen.

Der Ablauf der GDL-Script Analyse

Anwender haben keinen Einfluss darauf, 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 dem Start eines Listenerzeugungsprozesses und nach dem Laden der aktiven Bibliothek ausgeführt.

Scripte, deren Namen mit "MASTEREND_GDL" beginnen, werden ausgeführt, wenn die aktive Bibliothek geändert wird (Bibliothek laden, ein Projekt öffnen, neues Projekt erstellen, Beenden).

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

MASTER_GDL und MASTEREND_GDL Scripte können Attribute- Definitionen, Initialisierungen der GLOB_USER Variablen, 3D-Befehle (welche nur in 3D wirksam sind), Werteliste-Definitionen (siehe VALUES) und erweiterungsspezifische GDL-Befehle beinhalten. Die in diesen Scripten definierten Attribute werden den Attributen im aktuellen Projekt hinzugefügt (Attribute mit den selben Namen werden nicht ersetzt, während Attribute, die in GDL erzeugt 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 Anführungszeichen. 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 oder textlichen 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 Operanden 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, auf die sich ein Befehl vor oder nach dieser Zeile beziehen kann. Eine Sprungmarke ist eine ganze Zahl oder ein konstanter Text zwischen Anführungszeichen, gefolgt von einem Doppelpunkt (:). Bei textlichen Sprungmarken ist die Groß-/ Kleinschreibung zu beachten. 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) ~ ! : , ; . + - * / ^ = < > <= >= # ( ) [ ] { } \ @ & |
(senkrechter Strich) " ' ` ´ ‟ ” ’ ‛ <Zeilenende>
```

ZEICHENFOLGEN

Zeichenfolgen mit Unicode-Zeichen (Texte/ Strings) sind beliebige Reihen von Zeichen, die sich zwischen Anführungszeichen (",',`,´) befinden oder Zeichen ohne Anführungszeichen, die in dem Script nicht als Identifizierer mit einem gegebenen Wert (Makro-Aufruf, Attributname, Dateiname) fungieren. Zeichenfolgen ohne Anführungszeichen werden in Großbuchstaben konvertiert, daher sind Anführungszeichen empfohlen. Eine Zeichenfolge darf maximal 255 Zeichen beinhalten.

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

\\	'\' Zeichen selbst
\n	neue Zeile
\t	Tabulator
neue Zeile	Fortsetzen mit dem GDL-Text in der nächsten Zeile ohne neue Zeile
\andere	nicht zulässig, erzeugt Fehlermeldung

Beispiel:

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

IDENTIFIZIERER

Identifizierer sind spezielle ASCII Zeichenfolgen:

- sie dürfen nicht mehr als 255 Zeichen beinhalten ;
- sie beginnen mit einem Buchstaben des Alphabets oder einem '_' oder '~' Zeichen;
- sie bestehen aus ASCII 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 Variablennamen werden von dem GDL ausführenden Programm selbst festgelegt; alle anderen Identifizierer können als Variablennamen verwendet werden.

Variablen

Ein GDL-Programm kann numerische Variablen und Textvariablen (definiert 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. Lokale Variablen werden beim Aufruf von Makros zwischengespeichert. Ist ein Makro abgearbeitet, so werden die zwischengespeicherten Werte wiederhergestellt.

Globale Variablen haben reservierte Namen *(für die Liste der globalen Variablen siehe „Globale Variablen“)*. Diese werden beim Aufruf von Makros nicht zwischengespeichert und dienen dem Programmierer dazu spezielle Ergebnisse aus der Modellierung festzuhalten oder Rückgabewerte von Makros zu simulieren. Die Werte globaler Variablen sind für alle GDL-Texte verfügbar, wie z. B. der Zeichnungsmaßstab. Den benutzerdefinierten GLOB_USER Variablen können in beliebigen Scripts Werte zugewiesen werden, die dann aber erst in den nachfolgend abgearbeiteten Scripts verfügbar sind. Wenn Sie sicherstellen wollen, dass das gewünschte Script zuerst ausgeführt wird, setzen sie die Werte dieser Variablen in einem MASTER_GDL-Bibliothekselement. Alle Elemente werden diese durch die Master-GDL-Datei gesetzten Werte immer als erstes auslesen, es sei denn ihre eigenen Scripte (aufrufendes Objekt oder aufgerufenes Makro) modifizieren diese Werte. Es gibt zwischen den verschiedenen Interpretations-Instanzen keinen benutzerdefinierten Globalen Datenaustausch. Die übrigen 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 Bibliothekselementes aufgelistet sind, werden Parameter genannt. Parameter Identifizierer dürfen 31 Zeichen nicht überschreiten. Die maximale Anzahl der Parameter darf nicht mehr als 1024 sein. 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 konstante Zahlen, numerische Variablen oder Parameter, oder Funktionen, die numerische Werte und deren beliebige Kombinationen aus der Operation zurückgeben. Numerische Ausdrücke können Ganzzahlen oder reelle Zahlen sein. Ganzzahl Ausdrücke sind konstante "ganze Zahlen", numerische Variablen oder Parameter, oder Funktionen, die Ganzzahlwerte und deren beliebige Kombinationen aus der Operation zurückgeben. Reelle Ausdrücke sind konstante Fliesskommazahlen, numerische Variablen oder Parameter, oder Funktionen, die reelle Werte und deren beliebige Kombinationen (oder Ganzzahl Ausdrücke) aus der Operation zurückgeben. Ein numerischer Ausdruck (eine Ganzzahl oder reelle Zahl) wird während der Ausführung ermittelt und ist nur von den konstanten Zahlen, Variablen, Parametern und den Operationen abhängig, die diese miteinander kombinieren. Reelle 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 bei der Überprüfung des GDL-Scriptes (Vergleich von reellen Zahlen oder reellen Zahlen und Ganzzahlen, die relationale Operatoren '=' oder '<>', oder boolesche Operatoren AND, OR, EXOR; IF oder GOTO Anweisungen mit reellen Sprungmarken Ausdrücken verwenden).

String Ausdrücke sind konstante Texte, Zeichenfolgen-Variablen oder Parameter, oder Funktionen, die Strings und deren beliebige Kombinationen aus der Operation zurückgeben.

Derivierte Typen

Variablen und Parameter können auch Arrays (Datenfelder) sein, und Parameter können einfache Wertelisten sein.

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

Wertelisten sind Listen möglicher Werte für numerische oder Textparameter. 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.

Strukturierte Typen

Variablen und Parameter können auch Dictionaries sein. *Kompatibilität: eingeführt in ARCHICAD 23.*

Dictionaries sind eine hierarchische Sammlung von Schlüssel- und Wertepaaren. Schlüssel können andere Werte vom Typ Dictionary, Array, Ganzzahl, String oder Fließkomma enthalten.

Schlüssel gelten als Identifizierer („Identifizierer“) - es gelten dieselben Syntaxregeln, mit der Ausnahme, dass das Zeichen '~' nicht zulässig ist.

An den folgenden Stellen dürfen keine Dictionary-Schlüssel verwendet werden (auch wenn es sich um einfache Typen handelt):

- as `FOR - TO - NEXT` Schleifenvariable.
- als `HOTSPOT2` oder `HOTSPOT` bearbeiteter oder angezeigter Parameter.
- als UI_... Eingabeparameter, wobei der Eingabeparameter als Ausdruck angegeben wird. (`UI_INFIELD{2}`, `UI_INFIELD{3}`, `UI_CUSTOM_POPUP_INFIELD{2}`, `UI_RADIOBUTTON`, `UI_PICT_RADIOBUTTON`, `UI_PICT_PUSHCHECKBUTTON`,

UI_TEXTSTYLE_INFIELD, UI_LISTITEM{2}, UI_CUSTOM_POPUP_LISTITEM{2}, UI_COLORPICKER{2}, UI_SLIDER{2})

- als UI_... Eingabeparameter, bei dem der Eingabeparameter als String angegeben wird, wenn der String einen Parameter vom Typ Dictionary bildet.
- als VALUES oder VALUES{2} Parameter - Values-Listen können damit nicht generiert werden.
- als Rückgabewerte von REQUEST -Befehlen - ausschließlich das Root-Level eines Dictionnaires ist erlaubt in Requests, welche dies unterstützen.
- als Rückgabewerten von APPLICATION_QUERY, SPLIT, INPUT, LIBRARYGLOBAL oder CALLFUNCTION.
- als zürückgegebener extra_accuracy_string eines STR{2}.
- als RETURNED_PARAMETERS eines CALL - nur zurückgegebene Dictionnaires können nur auf dem Rootlevel eines Dictionnaires gespeichert werden.

In diesem Buch verwendete Konventionen

[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

|

Ausschlüsse oder Beziehungen zwischen Parametern eines Befehls

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, Skalieren, 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. Jede Verschiebung, Drehung oder Streckung des Koordinatensystems entlang oder um seine Achsen heißt Transformation. Alle Transformationen werden nacheinander in einem Datenspeicher (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

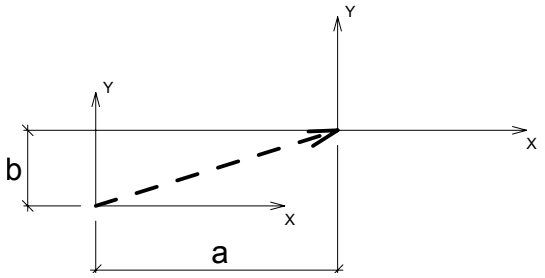
Diese sind die Äquivalenten im 2D-Raum der 3DTransformationen `ADD`, `MUL` und `ROTZ`.

ADD2

`ADD2` `x, y`

Beispiel:

```
ADD2 a, b
```



MUL2

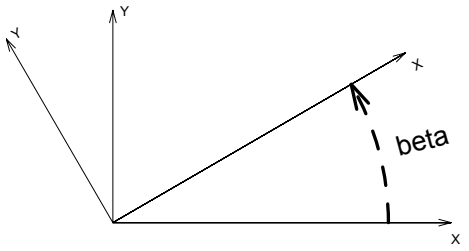
MUL2 x, y

ROT2

ROT2 alpha

Beispiel:

```
ROT2 beta
```



3D-Transformationen

ADDX

ADDX dx

ADDY

ADDY dy

ADDZ

ADDZ dz

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

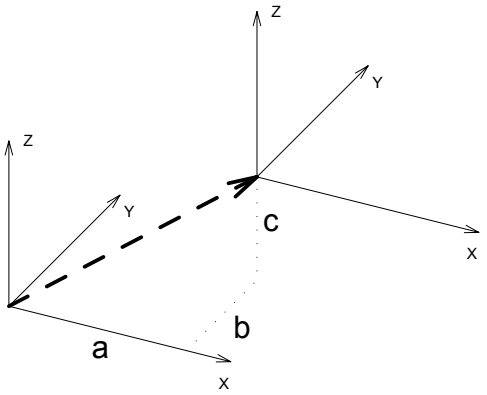
ADD

ADD dx, dy, dz

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

Beispiel:

```
ADD a, b, c
```



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

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 alphax

ROTY

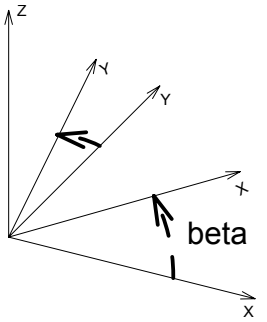
ROTY alphay

ROTZ

ROTZ alphaz

Dreht das lokale Koordinatensystem um einen Winkel alphax, alphay oder alphaz. Die Drehung um die angegebene Achse erfolgt entgegen dem Uhrzeigersinn.

Beispiel:




```
ROTZ beta
```

ROT

ROT x, y, z, alpha

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

XFORM

```
XFORM newx_x, newy_x, newz_x, offset_x,
      newx_y, newy_y, newz_y, offset_y,
      newx_z, newy_z, newz_z, offset_z
```

Definiert eine komplette Transformationsmatrix. Es wird hauptsächlich in automatisch generiertem GDL-Code verwendet. Der Befehl erzeugt nur einen Eintrag im Stack.

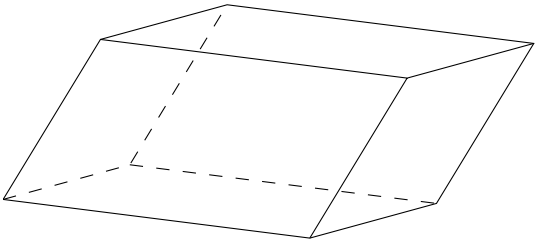
x' = newx_x * x + newy_x * y + newz_x * z + offset_x

y' = newx_y * x + newy_y * y + newz_y * z + offset_y

z' = newx_z * x + newy_z * y + newz_z * z + offset_z

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

```
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 begin_with festgelegten Eintrag. Die Nummerierung im Stack beginnt mit 1. Falls der Parameter begin_with nicht definiert wird und n einen negativen Wert hat, erfolgt die Löschung in umgekehrter Reihenfolge 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`

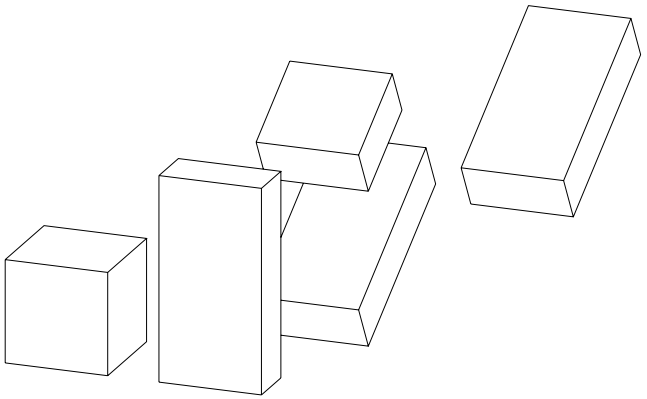
Löscht alle aktuellen Transformationen im aktuellen Script.

NTR

`NTR` `()`

Funktion, die die Anzahl der aktuell auf dem Transformationsstack befindlichen Einträge zurückgibt. Die Funktion benötigt kein Argument.

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 und 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 und Solid Elemente Befehlen.

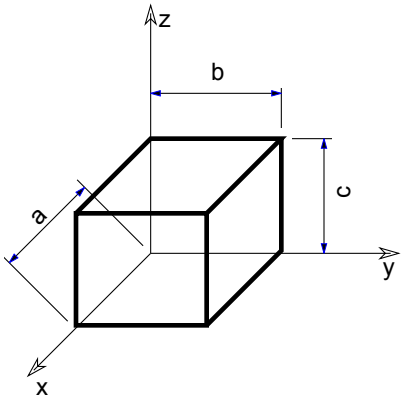
GRUNDKÖRPER

BLOCK

```
BLOCK a, b, c
```

BRICK

```
BRICK a, b, c
```



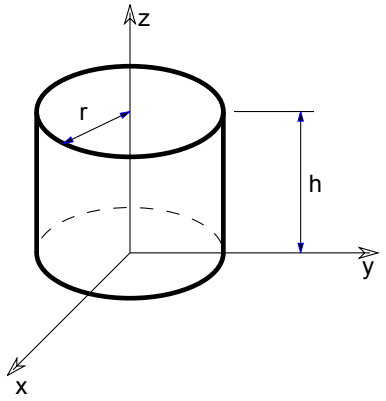
Der erste Eckpunkt eines Blocks liegt im lokalen Nullpunkt. Die Kanten mit den Längen a, b und c sind entlang der x-, y- und z-Achse ausgerichtet. Nullwerte erzeugen Sonderformen eines Quaders (Rechteck, Linie).

Einschränkung der Parameter:

```
a >= 0, b >= 0, c >= 0
a + b + c > 0
```

CYLIND

CYLIND h, r



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

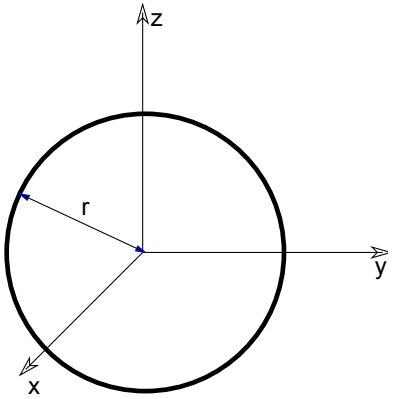
Ist h = 0, dann wird ein Kreis in der x-y-Ebene erzeugt.

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.



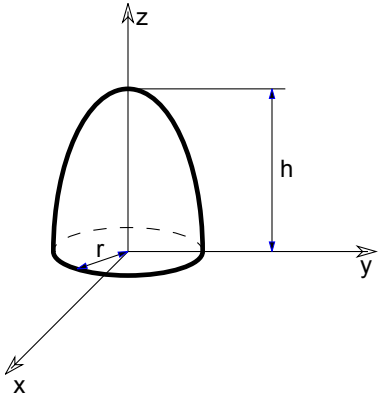
ELLIPS

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.

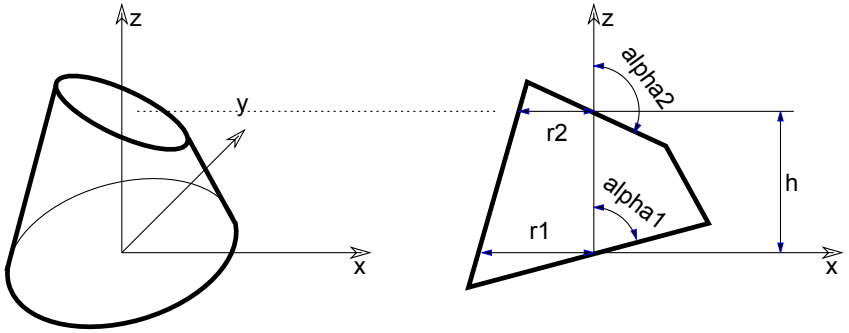
Beispiel: Halbkugel

```
ELLIPS h, r
```



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 die Radien unten und oben und h die Höhe der Mittelachse (auf der x-y-Achse).

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° und 0 < alpha2 < 180°
```

Beispiel: spitzer Kegel

```
CONE h, r, 0, 90, 90
```

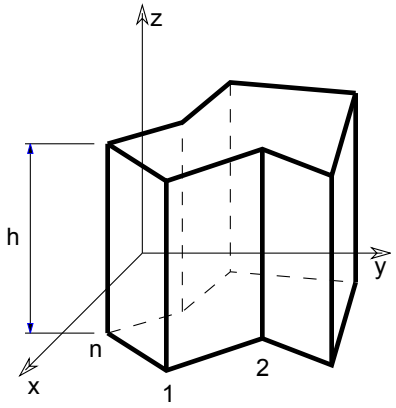
PRISM

PRISM n, h, x1, y1, ..., xn, yn

Senkrechtes Prisma, dessen Grundfläche ein Polygon in der x-y-Ebene ist (vgl. die Parameter von POLY und POLY_). 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 >= 3
```



PRISM_

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

Ähnlich der PRISM Anweisung. Jedoch können beliebige Kanten und Seitenflächen sichtbar oder unsichtbar gesetzt werden.

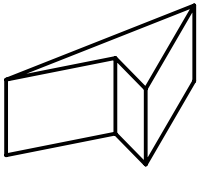
Einschränkung der Parameter:

```
n >= 3
```

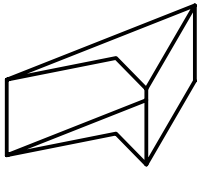
si: Statuscode, zum Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen. Sie können mit speziellen Statuscodes auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

Siehe Statuscodes für Details.

Beispiel 1: Massivkörper und Hohlkörper

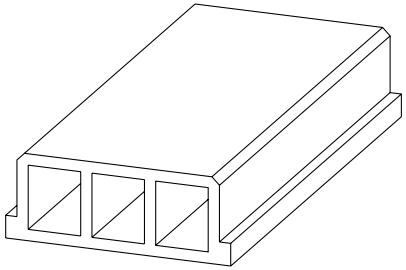


```
PRISM_ 4,1,
    0,0,15,
    1,1,15,
    2,0,15,
    1,3,15
```



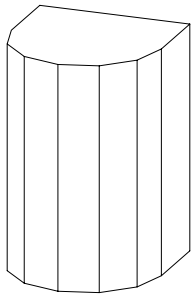
```
PRISM_ 4,1,
    0,0,7,
    1,1,5,
    2,0,15,
    1,3,15
```

Beispiel 2: Löcher im Polygon



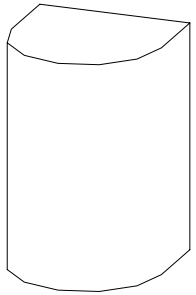
```
ROTX 90
PRISM_ 26, 1.2,
    0.3,   0,    15,
    0.3,   0.06, 15,
    0.27,  0.06, 15,
    0.27,  0.21, 15,
    0.25,  0.23, 15,
    -0.25, 0.23, 15,
    -0.27, 0.21, 15,
    -0.27, 0.06, 15,
    -0.3,  0.06, 15,
    -0.3,  0,    15,
    0.3,   0,    -1,    !Ende der Kontur
    0.10,  0.03, 15,
    0.24,  0.03, 15,
    0.24,  0.2,  15,
    0.10,  0.2,  15,
    0.10,  0.03, -1,    !Ende der ersten Öffnung
    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,    !Ende der zweiten Öffnung
    -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     !Ende der dritten Öffnung
```

Beispiel 3: Gekrümmte Oberflächen



j7 = 0

```
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
```



j7 = 1

```
R=1
H=3
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:

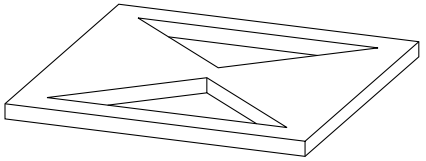
```
n >= 3
```

Siehe auch „Materialien“.

si: Statuscode, zum Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen. Sie können mit speziellen Statuscodes auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

Siehe Statuscodes für Details.

Beispiel: Das Material referenziert auf ein vordefiniertes Material durch Name, Index und Globale Variable



```
CPRISM_ "Mtl-Iron", 0, SYMB_MAT,
        13, 0.2,
        0, 0, 15,
        2, 0, 15,
        2, 2, 15,
        0, 2, 15,
        0, 0, -1,            !Ende der Kontur
        0.2, 0.2, 15,
        1.8, 0.2, 15,
        1.0, 0.9, 15,
        0.2, 0.2, -1,        !Ende der 1. Öffnung
        0.2, 1.8, 15,
        1.8, 1.8, 15,
        1.0, 1.1, 15,
        0.2, 1.8, -1         !Ende der 2. Öffnung
```

CPRISM_{2}

```
CPRISM_{2} top_material, bottom_material, side_material,
        n, h,
        x1, y1, alpha1, s1, mat1,
        ...
        xn, yn, alphan, sn, matn
```

CPRISM_{2}ist eine Erweiterung des Befehles CPRISM_ mit der Möglichkeit, verschiedene Winkel und Materialien für jede Seite des Prismas zu definieren.

Die Definition des Seitenwinkels ist ähnlich der Definition des CROOF_-Befehls.

alphai: Der Winkel zwischen der Stirnseite, die zu der i. Kante des Prismas gehört, und der Fläche, die senkrecht zur Unterseite ist.

mati: Materialindex zur Steuerung des Materials der Seitenfläche.

CPRISM_{3}

```
CPRISM_{3} top_material, bottom_material, side_material, mask,
           n, h,
           x1, y1, alpha1, s1, mat1,
           ...
           xn, yn, alphan, sn, matn
```

CPRISM_{3} ist eine Erweiterung von CPRISM_{2} mit der Möglichkeit das globale Verhalten des erzeugten Prismas zu steuern.

mask: kontrolliert das globale Verhalten des Prismas.

`mask = j1 + 2*j2 + 4*j3 + 8*j4`, hierbei kann j jeweils 0 oder 1 sein.

`j1`: Linienausschaltung Oberkante.

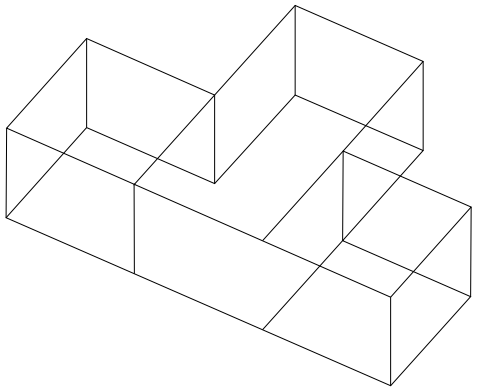
`j2`: Linienausschaltung der Unterkante.

`j3`: Linienausschaltung der Seitenkante.

`j4`: Seitenkante und Oberfläche sind in gebogenen Bereichen des Profils geglättet. *Kompatibilität: eingeführt in ARCHICAD 21.*

Beispiel 1:

```
PEN 1
mat = IND (MATERIAL, "Metal-Aluminium")
FOR i=1 TO 4 STEP 1
  IF i = 1 THEN mask = 1+2+4
  IF i = 2 THEN mask = 1
  IF i = 3 THEN mask = 2
  IF i = 4 THEN mask = 4
  CPRISM_{3} mat, mat, mat, mask,
      5, 1,
      0, 0, 0, 15, mat,
      1, 0, 0, 15, mat,
      1, 1, 0, 15, mat,
      0, 1, 0, 15, mat,
      0, 0, 0, -1, mat
  BODY -1
  DEL TOP
  IF i = 1 THEN ADDY 1
  IF i = 2 THEN ADDX -1
  IF i = 3 THEN ADDX 1
NEXT i
```



Beispiel 2:

```
PEN 1
mat = IND (MATERIAL, "Metal-Aluminium")
!visible side segment edges
mask = 1 + 2 + 4
_secondStat = 15
CPRISM_{3} mat, mat, mat, mask,
 6,  1,
 0,  0,   0, 15,   mat,
 1,  0,   0, _secondStat, mat,
 0.5, 0.5, 0, 900,   mat,
 1,   1,   0, 3015,   mat,
 0,   1,   0, 15,   mat,
 0,   0,   0, -1,   mat
!smooth edges using first node status copy
mask = 1 + 2 + 4
_secondStat = 15 + 64
CPRISM_{3} mat, mat, mat, mask,
 6,  1,
 0,  0,   0, 15,   mat,
 1,  0,   0, _secondStat, mat,
 0.5, 0.5, 0, 900,   mat,
 1,   1,   0, 3015,   mat,
 0,   1,   0, 15,   mat,
 0,   0,   0, -1,   mat
!smooth edges using mask, first edge is not smooth
mask = 1 + 2 + 4 + 8
_secondStat = 15
CPRISM_{3} mat, mat, mat, mask,
 6,  1,
 0,  0,   0, 15,   mat,
 1,  0,   0, _secondStat, mat,
 0.5, 0.5, 0, 900,   mat,
 1,   1,   0, 3015,   mat,
 0,   1,   0, 15,   mat,
 0,   0,   0, -1,   mat
```

CPRISM_{4}

```
CPRISM_{4} top_material, bottom_material, side_material, mask,
           n, h,
           x1, y1, alpha1, s1, mat1,
           ...
           xn, yn, alphan, sn, matn
```

CPRISM_{4} ist eine Erweiterung von CPRISM_{3} mit der Möglichkeit einer Inline-Materialdefinition, was bedeutet, dass im GDL-Script lokal definierte Materialien ebenfalls verwendet werden können neben den Materialien, welche in globalen Materialdefinitionen definiert wurden.

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.

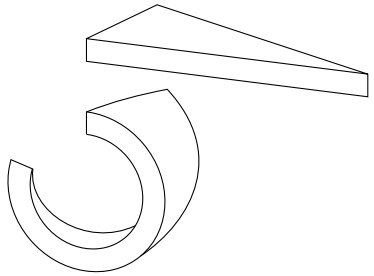
Siehe BWALL_ für Details.

si: Statuscode, zum Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen. Sie können mit speziellen Statuscodes auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

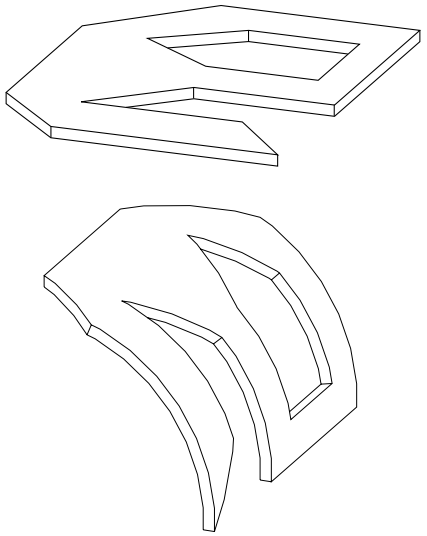
Siehe Statuscodes für Details.

Beispiel: Gebogene Prismen mit den korrespondierenden geraden Prismen

```
BPRISM_ "Glass - Blue", "Glass - Blue", "Glass - Blue",
        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 hill_material, angle und hill_height. Der Parameter thickness entspricht dem Parameter h des PRISM_-Befehls. Eine Anhöhe wird der Deckfläche des Prismas hinzugefügt.

hill_material: das Seitenmaterial der Anhöhe

angle: der Neigungswinkel der Seitenkanten der Anhöhe.

Einschränkung: 0 <= angle < 90.

Ist winkel= 0, bilden die Seitenkanten der Anhöhe, aus einem orthogonalen Blickpunkt gesehen, einen Viertelkreis mit einem durch die RESOL-Anweisung bestimmten Glättungsgrad. (siehe die Befehle: RADIUS, RESOL und TOLER).

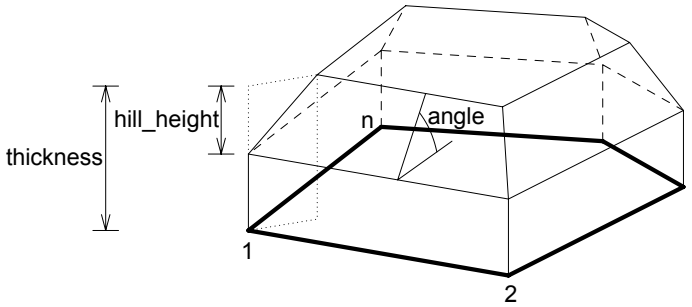
hill_height: Die Höhe der Anhöhe. Beachten sie, dass der Parameter thickness die totale Höhe von Prisma bestimmt.

si: Statuscode, zum Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen. Sie können mit speziellen Statuscodes auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

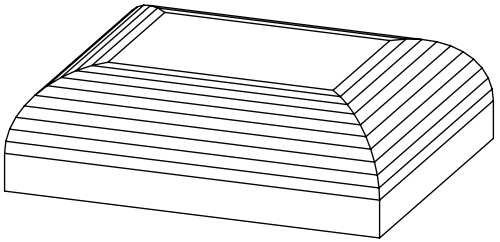
Einschränkung der Parameter:

```
n >= 3, hill_height < thickness
```

Siehe Statuscodes für Details.

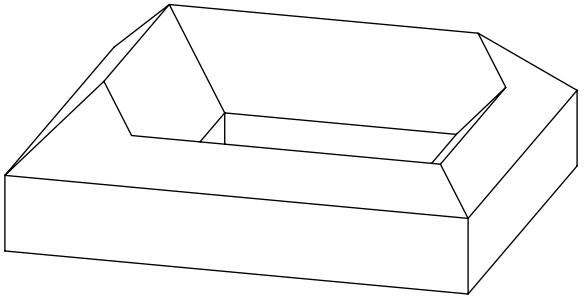


Beispiel 1: Prisma mit abgerundeter Oberkante



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

Beispiel 2: Prisma mit abgeschrägter Oberkante



```
FPRISM_ "Roof Tile", "Brick-Red", "Brick-White",
        "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_mat, bottom_mat, side_mat,
        hill_mat,
        n, thickness, angle, hill_height, status,
        x1, y1, s1,
        ...
        xn, yn, sn
```

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

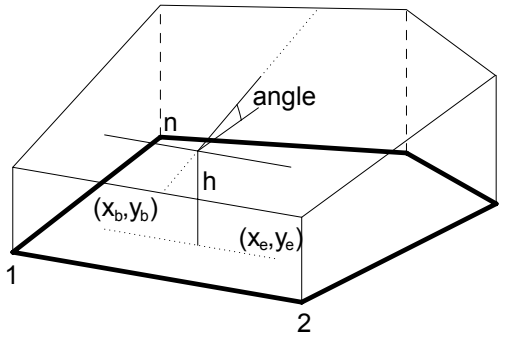
status: steuert die Sichtbarkeit der Hügelkanten:

- 0: alle Hügelkanten sind sichtbar (Darstellung wie 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, das obere Polygon gegen die x-y-Ebene zu neigen. Die Definition der oberen Ebene ist ähnlich der Ebenendefinition des CROOF_-Befehls. Die Höhe des Prismas wird durch die Referenzlinie bestimmt. Obere und untere Polygonfläche dürfen sich nicht berühren oder schneiden.



xb, yb, xe, ye: Start- und Endkoordinaten der Referenzlinie (Richtungsvektor),

angle: Der Drehwinkel des oberen Deckpolygons um die definierte Referenzlinie in Grad (entgegen dem Uhrzeigersinn),

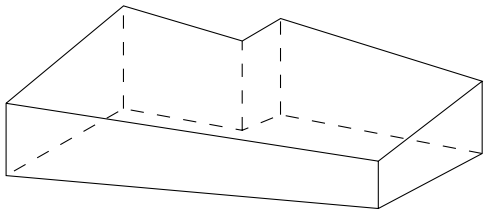
si: Statuscode, zum Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen. Sie können mit speziellen Statuscodes auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

Siehe Statuscodes für 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
```



SPRISM_{2}

```
SPRISM_{2} top_material, bottom_material, side_material,
        n,
        xtb, ytb, xte, yte, topz, tangle,
        xbb, ybb, xbe, ybe, bottomz, bangle,
        x1, y1, s1, mat1,
        ...
        xn, yn, sn, matn
```

Erweiterung des Befehls SPRISM_ mit der Option, dass das obere und untere Polygon nicht mit der x-y-Ebene parallel sind. Die Definition der Ebenen ist ähnlich der Ebenendefinition des CROOF_-Befehls. Die Ober- und Unterseite des Prismas werden durch die Referenzlinie bestimmt. Obere und untere Polygonfläche dürfen sich nicht berühren oder schneiden.

xtb, ytb, xte, yte: Start- und Endkoordinaten der Referenzlinie (Richtungsvektor) des oberen Deckpolygons,

topz: Höhenlage (in Z) der Referenzlinie des oberen Deckpolygons,

tangle: Der Drehwinkel des oberen Deckpolygons um die definierte Referenzlinie in Grad (entgegen dem Uhrzeigersinn),

xbb, ybb, xbe, ybe: Start- und Endkoordinaten der Referenzlinie (Richtungsvektor) des unteren Deckpolygons,

bottomz: Höhenlage (in Z) der Referenzlinie des unteren Bodenpolygons,

bangle: Der Drehwinkel des unteren Deckpolygons um die definierte Referenzlinie in Grad (entgegen dem Uhrzeigersinn),

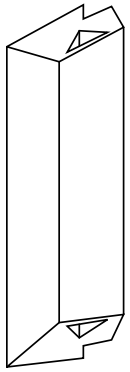
si: Statuscode, zum Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen. Sie können mit speziellen Statuscodes auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

Siehe Statuscodes für Details.

mati: Materialindex zur Steuerung des Materials der Seitenfläche.

Beispiel:

```
SPRISM_{2} 'Grass', 'Earth', 'Earth',
        11,
        0, 0, 11, 0, 30, -30.0,
        0, 0, 0, 11, 2, 30.0,
        0,  0, 15, IND (MATERIAL, 'C10'),
        10, 1, 15, IND (MATERIAL, 'C11'),
        11, 6, 15, IND (MATERIAL, 'C12'),
        5,  7, 15, IND (MATERIAL, 'C13'),
        4,  5, 15, IND (MATERIAL, 'C14'),
        1,  6, 15, IND (MATERIAL, 'C10'),
        0,  0, -1, IND (MATERIAL, 'C15'),
        9,  2, 15, IND (MATERIAL, 'C15'),
        10, 5, 15, IND (MATERIAL, 'C15'),
        6,  4, 15, IND (MATERIAL, 'C15'),
        9,  2, -1, IND (MATERIAL, 'C15')
```



SPRISM_{3}

```
SPRISM_{3} top_material, bottom_material, side_material, mask,
        n,
        xtb, ytb, xte, yte, topz, tangle,
        xbb, ybb, xbe, ybe, bottomz, bangle,
        x1, y1, s1, mat1,
        ...
        xn, yn, sn, matn
```

Erweiterung von SPRISM_{2} mit der Möglichkeit das globale Verhalten des erzeugten Prismas zu steuern.

mask: kontrolliert das globale Verhalten des Prismas.

mask = j_1 + 2*j_2 + 4*j_3 + 8*j_4, hierbei kann j jeweils 0 oder 1 sein.

j_1: Linienausschaltung Oberkante.

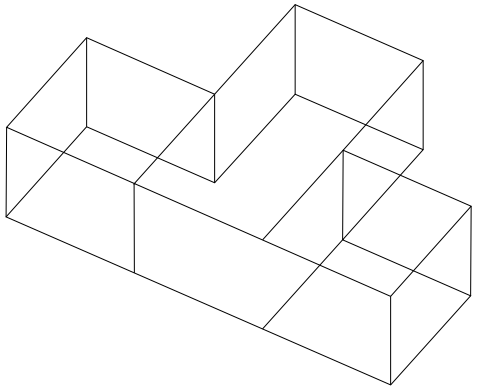
j_2: Linienausschaltung der Unterkante.

j_3: Linienausschaltung der Seitenkante.

j_4: Seitenkante und Oberfläche sind in gebogenen Bereichen des Profils geglättet. *Kompatibilität: eingeführt in ARCHICAD 21.*

Beispiel:

```
PEN 1
mat = IND (MATERIAL, "Metal-Aluminium")
FOR i=1 TO 4 STEP 1
  IF i = 1 THEN mask = 1+2+4
  IF i = 2 THEN mask = 1
  IF i = 3 THEN mask = 2
  IF i = 4 THEN mask = 4
  SPRISM_{3} mat, mat, mat, mask,
      5,
      0, 0, 1, 0, 1, 0,
      0, 0,  1,  0, 0, 0,
      0, 0, 15, mat,
      1, 0, 15, mat,
      1, 1, 15, mat,
      0, 1, 15, mat,
      0, 0, -1, mat

  BODY -1
  DEL TOP
  IF i = 1 THEN ADDY 1
  IF i = 2 THEN ADDX -1
  IF i = 3 THEN ADDX 1
NEXT i
```



SPRISM_{4}

```
SPRISM_{4} top_material, bottom_material, side_material, mask,
        n,
        xtb, ytb, xte, yte, topz, tangle,
        xbb, ybb, xbe, ybe, bottomz, bangle,
        x1, y1, s1, mat1,
        ...
        xn, yn, sn, matn
```

SPRISM_{4} ist eine Erweiterung von SPRISM_{3} mit der Möglichkeit einer Inline-Materialdefinition, was bedeutet, dass im GDL-Script lokal definierte Materialien ebenfalls verwendet werden können neben den Materialien, welche in globalen Materialdefinitionen definiert wurden.

SLAB

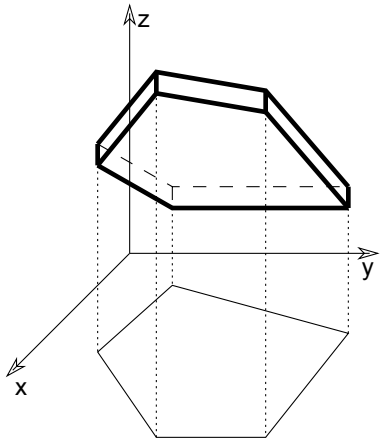
SLAB n, h, x1, y1, z1, ..., xn, yn, zn

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 >= 3
```



SLAB_

SLAB_ n, h, x1, y1, z1, s1, ..., xn, yn, zn, sn

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

si: Statuscode, zum Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen. Sie können mit speziellen Statuscodes auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

Siehe Statuscodes für 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.

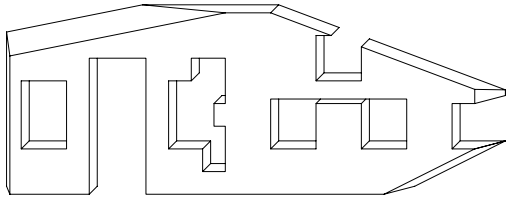
si: Statuscode, zum Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen. Sie können mit speziellen Statuscodes auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

Siehe Statuscodes für 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 die Stirnseite. (Die linke und die rechte Seite der Wand sind parallel zur x-Achse.)



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.

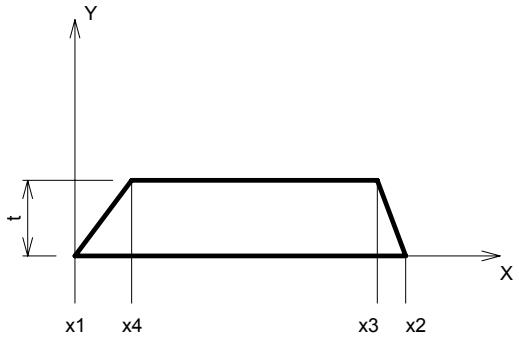
x1, x2, x3, x4: Die in die x-y-Ebene projizierten Eckpunkte der Wand (siehe Abb. unten). Ist die Wand freistehend (keine Verschneidung), so sind x1 = x4 = 0 der Wandanfang und x2 = x3 = Länge der Wand bzw. Wandende.

t: Die Wandstärke.

`t < 0:` der Wandkörper steht rechts der x-Achse,

`t > 0:` der Wandkörper steht links der x-Achse,

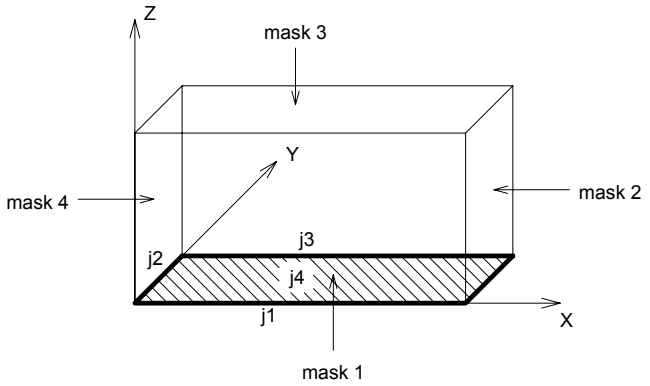
`t = 0:` die Wand wird als Polygon und die Wandöffnungen als Rahmen dargestellt.



mask1, mask2, mask3, mask4: Bestimmen die Sichtbarkeit der Kanten und Oberflächen.

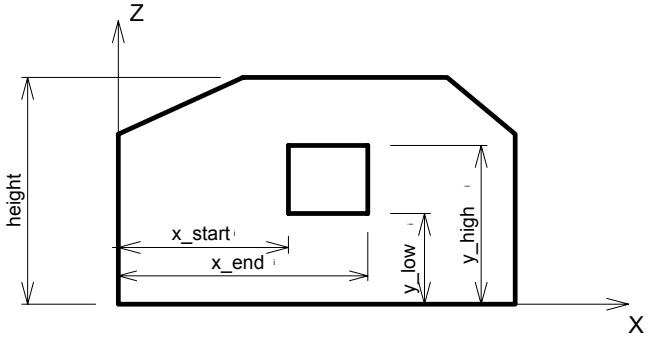
`mask1, mask2, mask3, mask4 =` $j_1 + 2*j_2 + 4*j_3 + 8*j_4$, hierbei kann j jeweils 0 oder 1 sein.

Die Bits j1, j2, j3 geben an, ob die Kanten des Seitenpolygons sichtbar sind (1) oder nicht (0). Das Bit j4 gibt an, ob die Kanten des aus dem Schnitt entstandenen Seitenpolygons sichtbar sind (1) oder nicht(0).



n: Anzahl der Öffnungen in einer Wand.

x_starti, y_lowi, x_endi, y_highi: Koordinaten der Öffnungen (siehe Abb. unten).



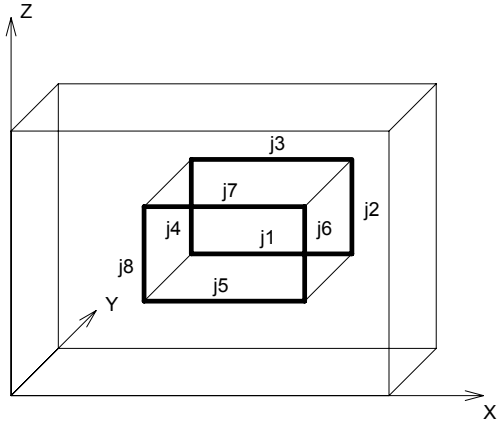
frame_showni:

1: wenn die Konturen einer Öffnung sichtbar sind,

0: wenn die Konturen einer Öffnung unsichtbar sind,

< 0: mit Hilfe negativer Werte kann die Sichtbarkeit jeder Kante einer Öffnung kontrolliert werden: frame_showni = - (1*j1 + *j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8) wo j1, j2 … j8 kann entweder 0 oder 1 sein. Die Nummern von j1 bis j4 prüfen die Sichtbarkeit

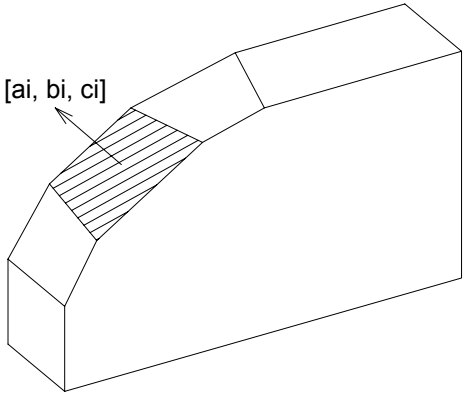
der Kanten der Öffnung auf der linken Seite der Wandoberfläche, während j5 bis j8 auf die Kanten auf der rechten Seite wirken, wie Sie es in der Abbildung unten sehen.



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

`m:` Anzahl der Schnittflächen.

`ai, bi, ci, di:` Koeffizienten der Gleichungen [ai*x + bi*y + ci*z = di], die Schnittebenen angeben. Die Teile der Wand, die auf der positiven Seite der Schnittebene liegen (ai*x + bi*y + ci*z > di), werden geschnitten und entfernt.



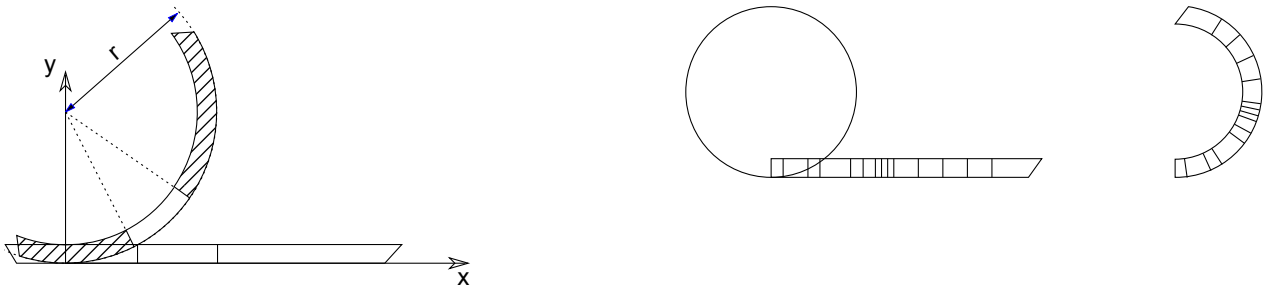
BWALL_

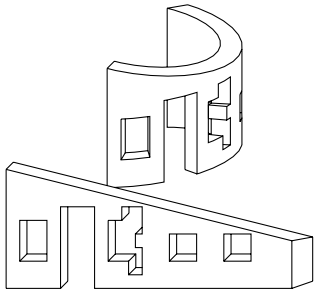
```
BWALL_ left_material, right_material, side_material,
       height, x1, x2, x3, x4, t, radius,
       mask1, mask2, mask3, mask4,
       n,
       x_start1, y_low1, x_end1, y_high1, frame_shown1,
       ...
       x_startn, y_lown, x_endn, y_highn, frame_shownn,
       m,
       a1, b1, c1, d1,
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

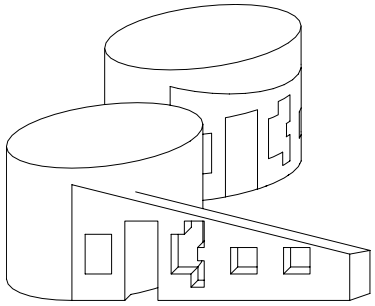
Gleichmäßig gekrümmte Wand mit derselben Datenstruktur wie die mit dem CWALL_-Befehl erzeugte gerade Wand. Der einzige zusätzliche Parameter ist die Angabe des Radius. Die Ableitung erfolgt aus dem entsprechenden CWALL_-Element durch Krümmen der x-z-Ebene zu einem zu dieser Ebene tangentialen Zylinder. 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, RESOL und TOLER).

Siehe CWALL_ für mehr Details.

Beispiel 1: BWALL_ und die entsprechende CWALL_.

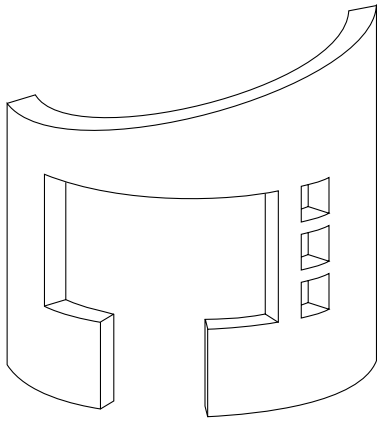






Beispiel 2:

```
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
```



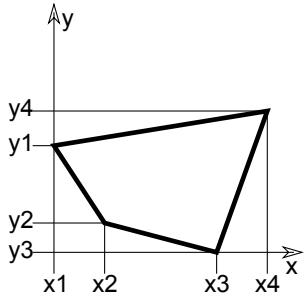
XWALL_

```
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
```

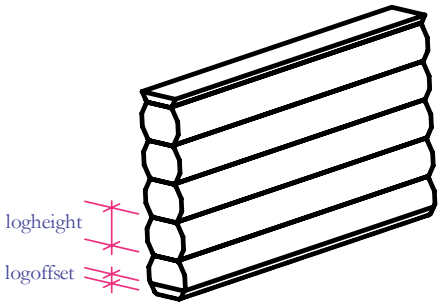
Erweiterte Wanddefinition auf der Basis der gleichen Datenstruktur wie das Element XWALL_.

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

y1, y2, y3, y4: die projizierten Endpunkte der Wand, die auf 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`, hierbei kann j jeweils 0 oder 1 sein.

`j1:` rechtes Seitenmaterial auf horizontale Kanten anwenden

`j2:` linkes Seitenmaterial auf horizontale Kanten anwenden

`j3:` mit halbem Block beginnen

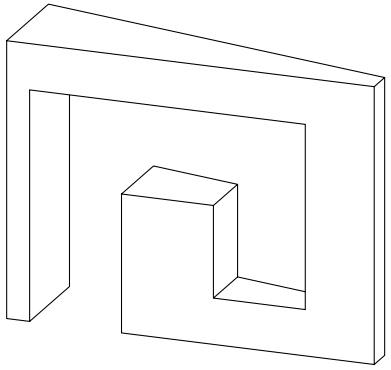
`j6:` Textur an Wandkanten ausrichten

`j7:` doppelter Radius auf der gekrümmten Seite

`j8:` quadratischer Block auf der rechten Seite

`j9:` quadratischer Block auf der linken Seite

Beispiel:



```
XWALL_ "Surf-White", "Surf-White", "Surf-White", "Surf-White",
        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_.

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

`frame_showni:`

`1:` wenn die Konturen einer Öffnung sichtbar sind,

`0:` wenn die Konturen einer Öffnung unsichtbar sind,

< 0: mit Hilfe negativer Werte kann die Sichtbarkeit jeder Kante einer Öffnung kontrolliert werden: frame_showni = -(1*j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8 + 256*j9 + 512*j10), wo j1, j2... j10 kann entweder 0 oder 1 sein. Es gibt zwei zusätzliche Werte zur Steuerung der Oberflächenmaterialien. 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 Stirnseitenmaterial der Wand. Der Wert j10 steuert die Form der Trennlinie zwischen dem Durchbruchsmaterial der oberen und unteren Polygone bei einem Durchbruch in einer gebogenen Wand. Wenn j10 den Wert 1 hat, ist die Trennlinie gerade, andernfalls gebogen.

XWALL_{3}

```
XWALL_{3} 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
```

XWALL_{3} ist eine Erweiterung des Befehls XWALL_{2} mit der Möglichkeit, alle Kanten der Blockhauswand zu verbergen.

status: steuert das Verhalten einer aus Blöcken zusammengesetzten Wand.

status = $j_1 + 2*j_2 + 4*j_3 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9 + 512*j_{10}$, hierbei kann j jeweils 0 oder 1 sein.

j_1: rechtes Seitenmaterial auf horizontale Kanten anwenden

j_2: linkes Seitenmaterial auf horizontale Kanten anwenden

j_3: mit halbem Block beginnen

j_6: Textur an Wandkanten ausrichten

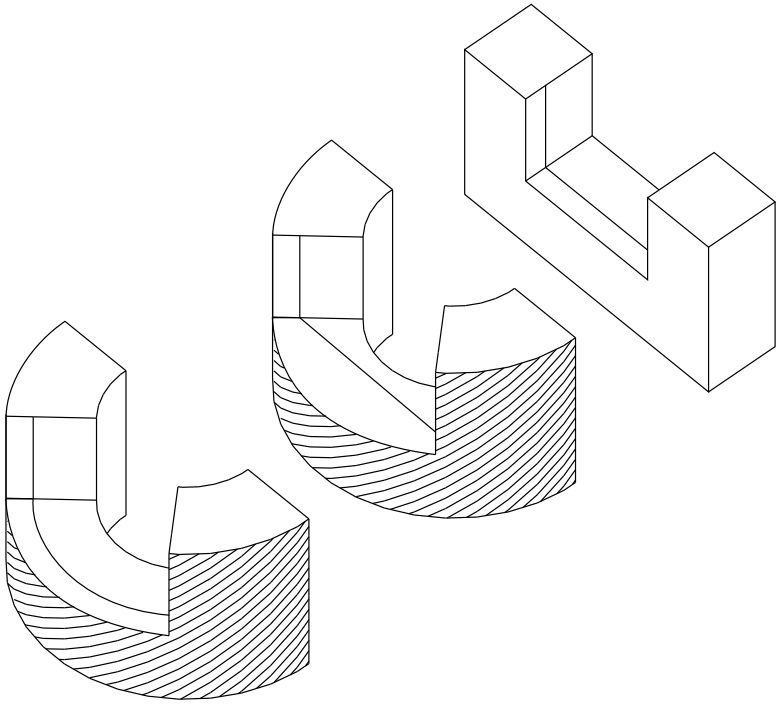
j_7: doppelter Radius auf der gekrümmten Seite

j_8: quadratischer Block auf der rechten Seite

j_9: quadratischer Block auf der linken Seite

`j10`: Alle Kanten einer Blockhauswand ausblenden.

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
DEL 1
ADDX 2
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),
        0,
        0
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
```

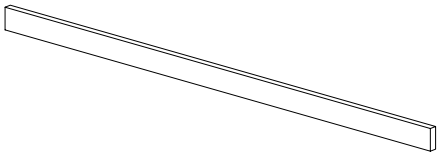
BEAM

```
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
```

Definition eines Trägers. Die Parameter sind ähnlich denen des XWALL_-Elements.

top_material, bottom_material: Materialien der Deckfläche und der Bodenfläche.

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
```

Geneigte Dachfläche mit beliebig geneigten Firsten.

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

n: die Anzahl der Eckpunkte des Dachpolygons.

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

height: die Höhe des Dachunterseite an der Referenzlinie.

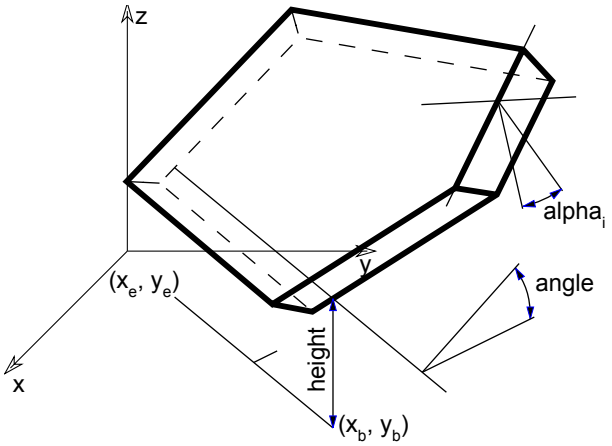
angle: der Drehwinkel der Dachfläche um die angegebene Referenzlinie in Grad (entgegen dem Uhrzeigersinn).

thickness: die Dachstärke senkrecht zur Dachfläche gemessen.

xi, yi: die Koordinaten der Eckpunkte des unteren Dachflächenpolygons.

alphai: Der Winkel zwischen der Stirnseite, die zu der i. Kante des Daches gehört, und der Ebene, die senkrecht zur Dachfläche ist -90 < alphai < 90. Wenn man bei einem richtig orientierten Dachpolygon in Richtung der jeweiligen Kante blickt, ist der Drehwinkel entgegen dem Uhrzeigersinn positiv. Die Kanten des Dachpolygons sind dann richtig orientiert, wenn die Eckpunkte der Kontur von oben betrachtet dem Uhrzeigersinn entgegen definiert sind, die Öffnungen jedoch im Uhrzeigersinn.

si: Statuscode, zum Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen. Sie können mit speziellen Statuscodes auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

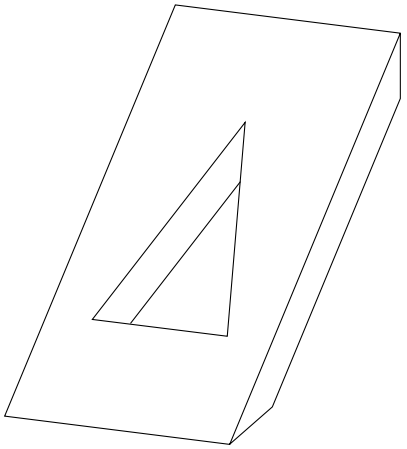


Siehe Statuscodes für Details.

Einschränkung der Parameter:

```
n >= 3
```

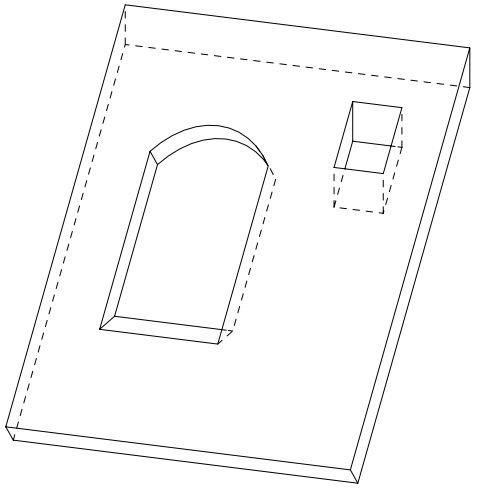
Beispiel 1:



```
CROOF_ 1, 1, 1, ! MATERIAL
        9,
        0, 0,
        1, 0, ! Referenzlinie (xb,yb)(xe,ye)
        0.0,  ! Höhe
        -30,  ! Winkel
        2.5,  ! Dicke
        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
```

Beispiel 2:

```
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
```



CROOF_{2}

```
CROOF_{2} top_material, bottom_material, side_material,
        n, xb, yb, xe, ye, height, angle, thickness,
        x1, y1, alpha1, s1, mat1,
        ...
        xn, yn, alphan, sn, matn
```

Erweiterung des Befehles CROOF_mit der Möglichkeit, verschiedene Winkel und Materialien für die Seiten zu definieren.

mati: Materialindex zur Steuerung des Materials der Seitenfläche.

CROOF_{3}

```
CROOF_{3} top_material, bottom_material, side_material, mask,
          n, xb, yb, xe, ye, height, angle, thickness,
          x1, y1, alpha1, s1, mat1,
          ...
          xn, yn, alphan, sn, matn
```

Erweiterung von CROOF_{2} mit der Möglichkeit das globale Verhalten des erzeugten Daches zu steuern.

mask: kontrolliert das globale Verhalten des erzeugten Daches.

mask = j_1 + 2*j_2 + 4*j_3 + 8*j_4, hierbei kann j jeweils 0 oder 1 sein.

j_1: Linienausschaltung Oberkante.

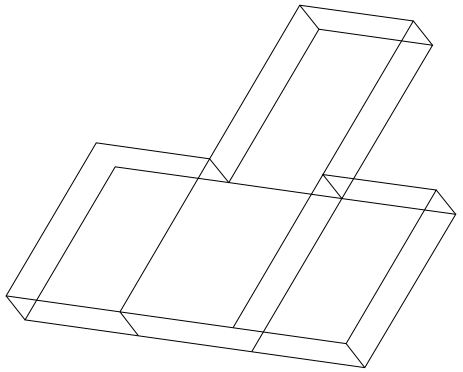
j_2: Linienausschaltung der Unterkante.

j_3: Linienausschaltung der Seitenkante.

j_4: Seitenkante und Oberfläche sind in gebogenen Bereichen des Profils geglättet. *Kompatibilität: eingeführt in ARCHICAD 21.*

Beispiel:

```
PEN 1
mat = IND (MATERIAL, "Metal-Aluminium")
FOR i=1 TO 4 STEP 1
  IF i = 1 THEN mask = 1+2+4
  IF i = 2 THEN mask = 1
  IF i = 3 THEN mask = 2
  IF i = 4 THEN mask = 4
  CROOF_{3} mat, mat, mat, mask,
      5,  0, 1,  2,  1, 3,   -45,  0.3,
      0,  0,  0,  15, mat,
      1,  0,  0,  15, mat,
      1,  1,  0,  15, mat,
      0,  1,  0,  15, mat,
      0,  0,  0,  -1, mat
  BODY -1
  DEL TOP
  IF i = 1 THEN ADD 0,1,1
  IF i = 2 THEN ADDX -1
  IF i = 3 THEN ADDX 1
NEXT i
```



CROOF_{4}

```
CROOF_{4} top_material, bottom_material, side_material, mask,
          n, xb, yb, xe, ye, height, angle, thickness,
          x1, y1, alpha1, s1, mat1,
          ...
          xn, yn, alphan, sn, matn
```

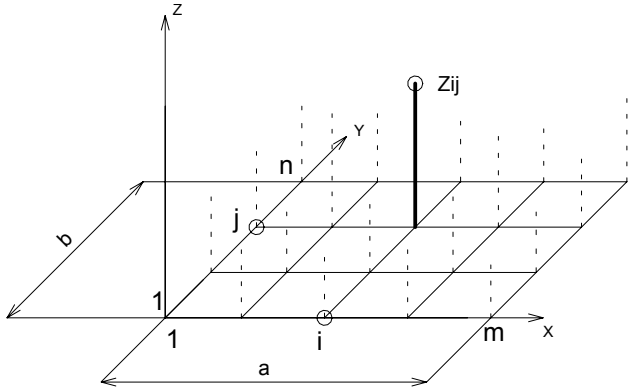
CROOF_{4} ist eine Erweiterung vonCROOF_{3} mit der Möglichkeit einer Inline-Materialdefinition, was bedeutet, dass im GDL-Script lokal definierte Materialien ebenfalls verwendet werden können neben den Materialien, welche in globalen Materialdefinitionen definiert wurden.

MESH

```
MESH a, b, m, n, mask,
       z11, z12, ..., z1m,
       z21, z22, ..., z2m,
       ...
       zn1, zn2, ..., 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. zij ist die Höhe eines Scheitelpunktes.

Mask-Werte



mask:

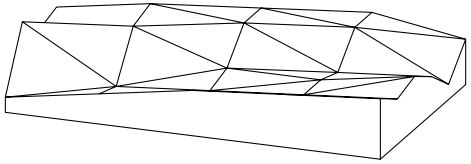
mask = j_1 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7, hierbei kann j jeweils 0 oder 1 sein.

`j1`: Grundfläche ist vorhanden.
`j3`: Seitenflächen sind vorhanden.
`j5`: Konturen der Grund- und Seitenflächen sind sichtbar.
`j6`: Konturen der Deckfläche sind sichtbar.
`j7`: Innenkonturen der Deckfläche sind sichtbar, die Oberfläche ist in Segmente aufgeteilt.

Einschränkung der Parameter:

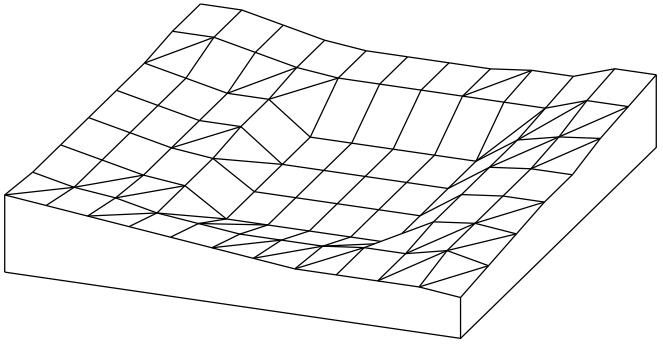
```
m >= 2, n >= 2
```

Beispiel 1:



```
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
```

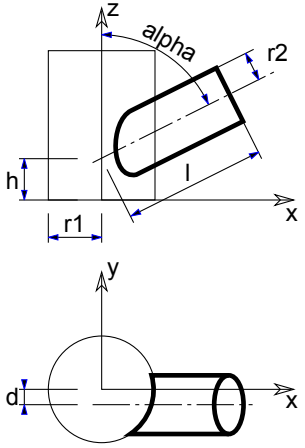
Beispiel 2:



```
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



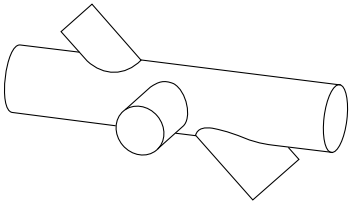
Teil einer Röhre, die an einer anderen Röhre ansetzt; die Verschneidungskurven werden berechnet und dargestellt. Alpha wird in Grad angegeben.

Einschränkung der Parameter:

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

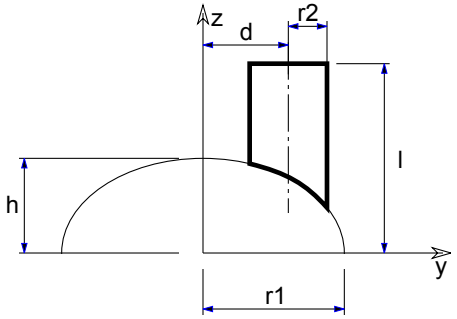
Beispiel:

```
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



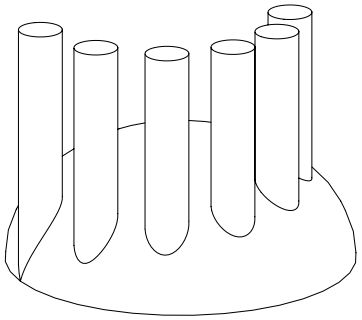
Teil einer Röhre, die an einem Ellipsoid parallel zur z-Achse ansetzt; die Verschneidungskurven werden berechnet und dargestellt.

Einschränkung der Parameter:

```
r1 >= r2+d
l >= h*sqrt(1-(r2-d)2/r12)
```

Beispiel:

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



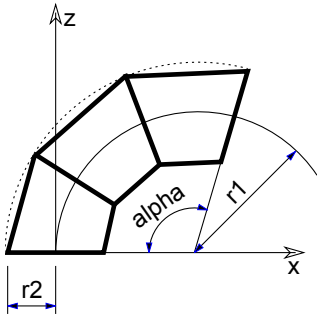
ELBOW

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. Alpha wird in Grad angegeben.

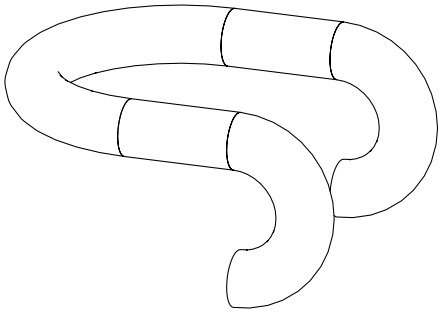
Einschränkung der Parameter:

```
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 Abschnitt vorgestellten planaren Zeichnungselemente können in 3D-Scripts verwendet werden. Mit Hilfe dieser können Sie Punkte, Linien, Bögen, Kreisbögen und planare Polygone im 3D-Raum darstellen.

HOTSPOT

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

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

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

paramReference: Parameter, der durch diesen Fixpunkt mit der Parameter-Bearbeitungsmethode auf Basis grafische veränderbarer.

displayParam: Parameter, der auf der Infopalette ausgegeben wird, wenn der Fixpunkt mithilfe der Parameter-Bearbeitungsmethode grafischer Hotspots bearbeitet wird. Es können auch Elemente von Arrays übergeben werden.

customDescription: benutzerdefinierte Beschreibung des in der Infopalette angezeigten Parameters Wenn Sie diese Option verwenden, muss displayParam ebenfalls gesetzt sein (verwenden Sie paramReference als Standardwert).

Siehe Bearbeitungsbefehle auf Hotspot-Basis über die Verwendung von HOTSPOT.

HOTLINE

HOTLINE x1, y1, z1, x2, y2, z2, unID

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

HOTARC

HOTARC r, alpha, beta, unID

Ein Fangbogen in der x-y Ebene mit seinem Mittelpunkt im Ursprung von Winkel alpha zum Winkel beta, mit dem Radius r.
alpha und beta werden in Grad angegeben.

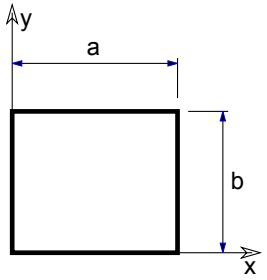
LIN_

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



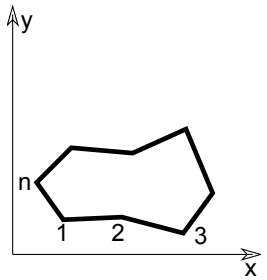
Ein Rechteck in der x-y Ebene mit den Seiten a und b.

Einschränkung der Parameter:

```
a >= 0, b >= 0
```

POLY

POLY n, x1, y1, ..., xn, yn



Ein Polygon mit n Konturlinien in der x-y Ebene. Die Koordinaten der i Eckpunkte sind (xi, yi, 0).

Einschränkung der Parameter:

```
n >= 3
```

POLY_

POLY_ `n, x1, y1, s1, ..., xn, yn, sn`

Gleicht dem einfachen POLY-Befehl, jedoch kann die Sichtbarkeit jeder beliebigen Kante unterdrückt werden.

si: Statuscode, zum Steuern der Sichtbarkeit von Polygonkanten und Seitenflächen. Sie können mit speziellen Statuscodes auch Durchbrüche definieren sowie Segmente und Bögen in der Polylinie erstellen.

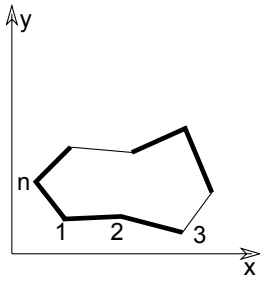
`si = 0:` die Kante ausgehend vom Punkt (xi, yi) wird unsichtbar.

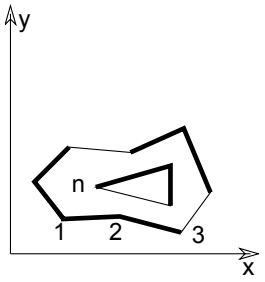
`si = 1:` die Kante wird dargestellt.

`si = -1:` wird zur Definition der Löcher benutzt.

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

Siehe „Zusätzliche Statuscodes" für Details.





Einschränkung der Parameter:

```
n >= 3
```

PLANE

PLANE n, x1, y1, z1, ..., xn, yn, zn

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

Einschränkung der Parameter:

```
n >= 3
```

PLANE_

PLANE_ n, x1, y1, z1, s1, ..., xn, yn, zn, sn

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.

Siehe „Zusätzliche Statuscodes“.

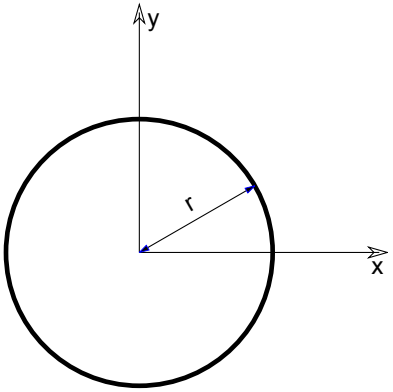
Einschränkung der Parameter:

```
n >= 3
```

CIRCLE

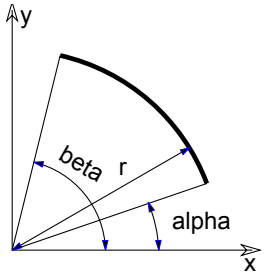
CIRCLE r

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



ARC

ARC `r, alpha, beta`



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

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 jeden Befehl ihre spezifischen Eigenschaften. Genauere Angaben dazu finden Sie bei dem jeweiligen Befehlstyp.

mask:

mask = $j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7$, hierbei kann j jeweils 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.

j_1: Grundfläche.
j_2: Deckfläche.
j_3: 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 Konturenanzeige berücksichtigt.

-1: nur für den EXTRUDE-Befehl: 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 oder die Definition von Durchbrüchen beim EXTRUDE-Befehl (s.o.).

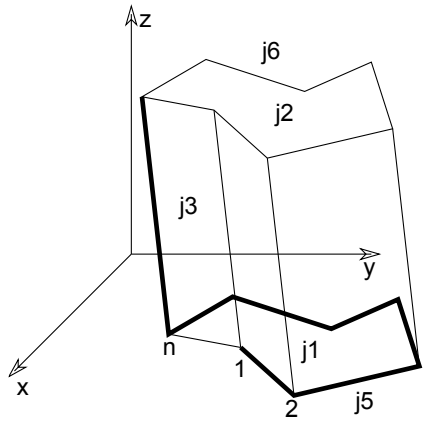
Siehe „Zusätzliche Statuscodes“ für Details.

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- undSLAB-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.

$mask = j_1 + 2*j_2 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8$, hierbei kann j jeweils 0 oder 1 sein.

j_1: Grundfläche ist vorhanden.

j_2: Deckfläche ist vorhanden.

j_3: Die das Prisma schließende Seitenfläche ist vorhanden.

j_5: Konturen der Grundfläche sind sichtbar.

j_6: Konturen der Deckfläche sind sichtbar.

j_7: Schnittkonturen sind sichtbar, die Oberfläche ist segmentiert,

j_8: Schnittkonturen sind scharfkantig, Glättung der Oberfläche endet hier in OpenGL und im Rendering.

si: 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, sowie Durchbrüche in der Fläche definieren.

0: Seitenkanten, ausgehend vom Punkt sind sichtbar.

1: Seitenkanten, ausgehend vom Punkt werden nur für die Konturenanzeige verwendet.

`-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.

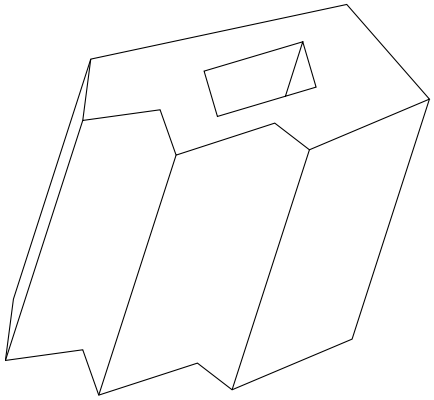
Siehe „Zusätzliche Statuscodes“ für Details.

Einschränkung der Parameter:

```
n > 2
```

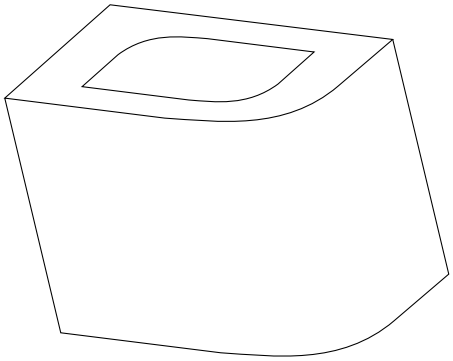
Beispiel 1:

```
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
```



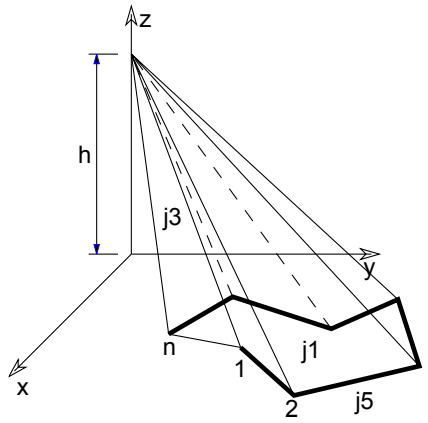
Beispiel 2:

```
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, mask, x1, y1, s1, ..., xn, yn, sn



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).

`mask = j1 + 4*j3 + 16*j5`, hierbei kann j jeweils 0 oder 1 sein.

`j1:` Grundfläche ist vorhanden.

`j3:` Die das Prisma schließende Seitenfläche ist vorhanden.

`j5:` Konturen der Grundfläche sind sichtbar.

si: Status der seitlichen Kanten.

`0:` Seitenkanten, vom Gelenkpunkt ausgehend, sind alle sichtbar.

`1:` Seitenkanten, vom Gelenkpunkt ausgehend, werden nur für die Konturenanzeige verwendet.

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

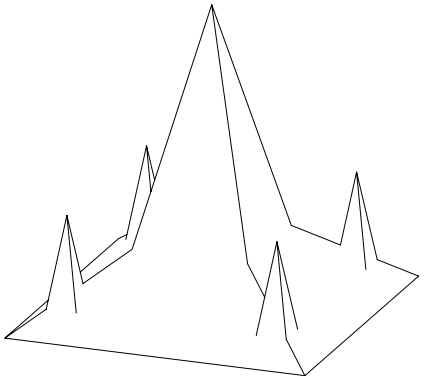
Siehe „Zusätzliche Statuscodes“ für Details.

Einschränkung der Parameter:

```
h > 0 and n > 2
```

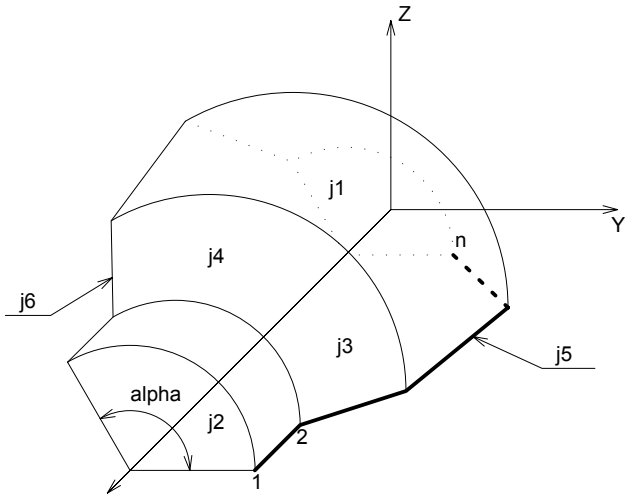
Beispiel:

```
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
for i = 1 to 4          ! 4 Spitzen
    ADD -1.4, -1.4, 0
    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
    DEL 1
    ROTZ 90
next i
del 4
```



REVOLVE

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



Körper, der durch Rotation eines in der x-y-Ebene liegenden Linienzuges um die x-Achse erzeugt wird. Die Polylinie des Profils kann keine Löcher enthalten.

n: Anzahl der Eckpunkte des Polygonzuges.

alpha: Drehwinkel. Angabe erfolgt in Grad.

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

mask = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9, hierbei kann j jeweils 0 oder 1 sein.

j_1: Grundfläche ist vorhanden,
j_2: Schlussfläche ist vorhanden,
j_3: Grundabschlussfläche (in der Profil-Ebene) ist vorhanden,
j_4: Endabschlussfläche (in der Drehkörper-Ebene) ist vorhanden,
j_5: Konturen der Basisfläche (bei x2, y2, z2) sind sichtbar,
j_6: Konturen der Schlussfläche (bei xm-1, ym-1, zm-1) sind sichtbar,
j_7: Schnittkonturen sind sichtbar, die Oberfläche ist segmentiert,
j_8: Linienausschaltung der horizontalen Kanten,
j_9: vertikale Kante in der Linien-Elimination.

si: Status der Begrenzungsbögen.

0: Kreisbögen parallel zu y-z-Ebene, vom Eckpunkt ausgehend sind sichtbar.

1: Kreisbögen parallel zu y-z-Ebene, vom Eckpunkt ausgehend werden nur für die Konturenanzeige verwendet.

2: Wenn man ARCHICAD - oder Z-Buffer-Rendering-Engine verwendet und diese auf "Glatte Oberflächen" stellt, definiert die, zu diesem Punkt gehörige, parallel zu y-z-Ebene liegende Kante eine Unterbrechung. Diese Lösung entspricht der Definition von zusätzlichen Punkten. Die Berechnung wird vom Compiler ausgeführt. Mit anderen photorealistischen Renderern hat si=2 denselben Effekt, als würde man si=0 verwenden.

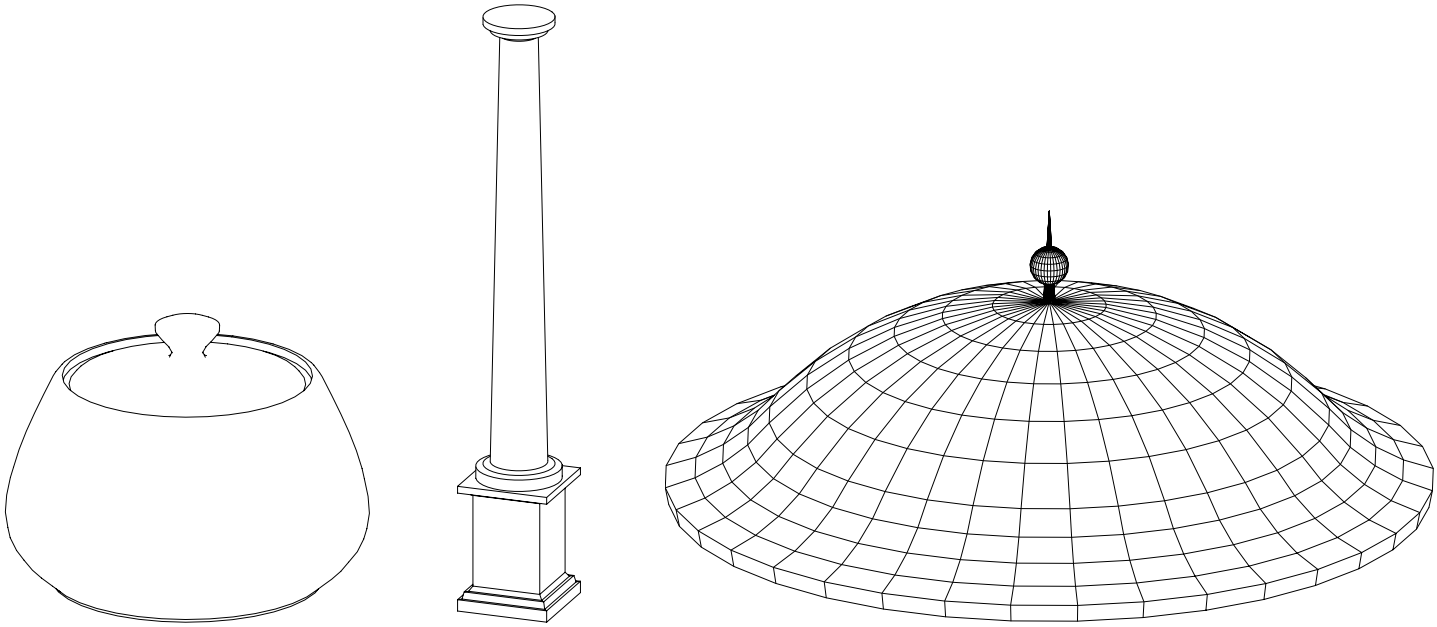
Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie.

Siehe „Zusätzliche Statuscodes“ für Details.

Einschränkung der Parameter:

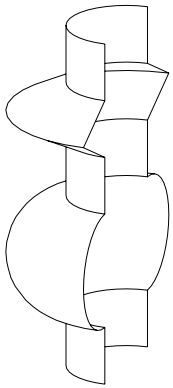
```
n >= 2
yi >= 0.0
yi = 0.0 und yi+1 = 0. (d.h. der y-Wert zwei benachbarter Eckpunkte)
dürfen nicht gleichzeitig Null sein).
```

Beispiel 1:



```
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
```

Beispiel 2:



Lösung ohne 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
```

das gleiche Ergebnis mit 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
```

REVOLVE{2}

REVOLVE{2} n, alphaOffset, alpha, mask, sideMat,
x1, y1, s1, mat1, ..., xn, yn, sn, matn

Erweiterte Version von REVOLVE. Das Profilpolygon wird immer geschlossen und kann Löcher enthalten. Startwinkel und Flächenmaterialien sind kontrollierbar.

alphaOffset: Startwert des Drehwinkels.

alpha: Winkellänge des Drehwinkels in Grad, kann negativ sein.

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

`mask = 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8 + 256*j9`, hierbei kann j jeweils 0 oder 1 sein.

`j3`: Grundabschlussfläche (in der Profil-Ebene) ist vorhanden,
`j4`: Endabschlussfläche (in der Drehkörper-Ebene) ist vorhanden,
`j5`: Konturen der Basisfläche (bei x2, y2, z2) sind sichtbar,
`j6`: Konturen der Schlussfläche (bei xm-1, ym-1, zm-1) sind sichtbar,
`j7`: Schnittkonturen sind sichtbar, die Oberfläche ist segmentiert,
`j8`: Linienausschaltung der horizontalen Kanten,
`j9`: vertikale Kante in der Linien-Elimination.

sideMat: Material der schließenden Oberflächen

mati: Material der Fläche, erstellt von der i-Kante

REVOLVE{3}

```
REVOLVE{3} n, alphaOffset, alpha, betaOffset, beta, mask, sideMat,
        x1, y1, s1, mat1, ..., xn, yn, sn, matn
```

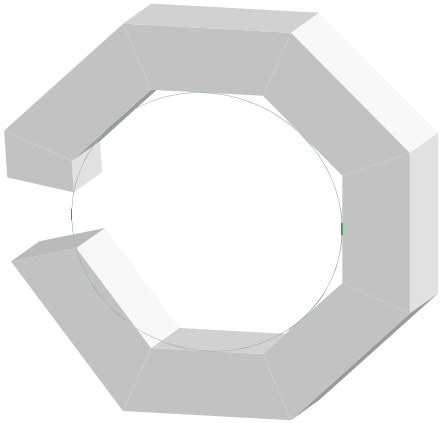
REVOLVE{3} ist eine Erweiterung des REVOLVE{2} - Befehls mit der Möglichkeit 2 Fang-Positionen zu definieren. Während des Rotationsvorganges ist der Pfad eines jeden Punktes der Basis-Polyline ein kreisförmiger Bogen , welcher annäherungsweise von einer Polylinie dargestellt wird. Bei REVOLVE{3} können zwei Fang-Positionen definiert werden, wobei die Polylinie mit den Kreis exakt übereinstimmt. Bei REVOLVE{2} liegen die beiden Fang-Positionen am Start- und Endpunkt der Rotation . Bei REVOLVE{3} liegen die Endpunkte nicht notwendigerweise auf dem Kreis, aber sie schneiden die Endflächen

betaOffset: Winkel, welcher die Lage der ersten Fangposition definiert. Der definierte Winkel muss sich nicht innerhalb des Bereiches der Rotation befinden.

beta: Winkel, welcher die Lage der zweiten Fangposition definiert, relativ zur ersten Fangposition. Darf negativ sein. Der definierte Winkel muss sich nicht innerhalb des Bereiches der Rotation befinden.

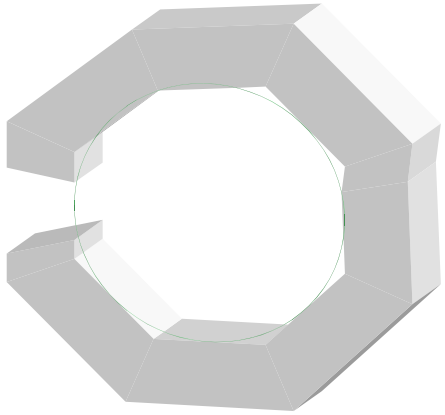
Beispiel:

REVOLVE{2} Fang-Position am Ende



```
resol 8
revolve{2} 4,
  10, 335,    ! alphaOffset, alpha
  444, 2,
  0, 4, 2, 2,
  3, 4, 2, 2,
  3, 6, 2, 2,
  0, 6, 2, 2
! reference circle
resol 72
revolve{2} 4,
  0, 360,    ! alphaOffset, alpha
  444, 0,
  -0.01, 3.99, 2, 0,
  0, 3.99, 2, 0,
  0, 4, 2, 0,
  -0.01, 4, 2, 0
```

REVOLVE{3} benutzerdefinierte Fangposition



```
resol 8
revolve{3} 4,
  10, 335,    ! alphaOffset, alpha
  67.5, 100,  ! betaOffset, beta
  444, 2,
  0, 4, 2, 2,
  3, 4, 2, 2,
  3, 6, 2, 2,
  0, 6, 2, 2
! reference circle
resol 72
revolve{2} 4,
  0, 360,    ! alphaOffset, alpha
  444, 0,
  -0.01, 3.99, 2, 0,
  0, 3.99, 2, 0,
  0, 4, 2, 0,
  -0.01, 4, 2, 0
```

REVOLVE{4}

```
REVOLVE{4} n, alphaOffset, alpha, betaOffset, beta, mask, sideMat,
        x1, y1, s1, mat1, ..., xn, yn, sn, matn
```

REVOLVE{4} ist eine Erweiterung des REVOLVE{3} - Befehls mit der Möglichkeit, alle Kanten unsichtbar zu machen.

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

mask = $4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9 + 512*j_{10} + 1024*j_{11}$, hierbei kann j jeweils 0 oder 1 sein.

j_3: Grundabschlussfläche (in der Profil-Ebene) ist vorhanden,

j_4: Endabschlussfläche (in der Drehkörper-Ebene) ist vorhanden,

j_5: Konturen der Basisfläche (bei x2, y2, z2) sind sichtbar,

j_6: Konturen der Schlussfläche (bei xm-1, ym-1, zm-1) sind sichtbar,

j_7: Schnittkonturen sind sichtbar, die Oberfläche ist segmentiert,

j_8: Linienausschaltung der horizontalen Kanten,

j_9: Linienausschaltung der vertikalen Kanten,

j_{10}: alle Kanten des Rotationskörpers unsichtbar machen.

j_{11}: Seitenkante und Oberfläche sind in gebogenen Bereichen des Profils geglättet. *Kompatibilität: eingeführt in ARCHICAD 21.*

REVOLVE{5}

```
REVOLVE{5}n, alphaOffset, alpha, betaOffset, beta, mask, sideMat,
        x1, y1, s1, mat1, ..., xn, yn, sn, matn
```

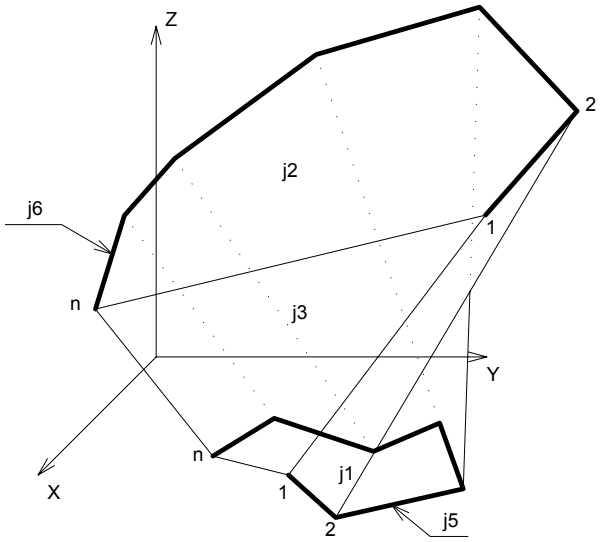
REVOLVE{5} ist eine Erweiterung des REVOLVE{4} Möglichkeit einer Inline-Materialdefinition, was bedeutet, dass im GDL-Script lokal definierte Materialien ebenfalls verwendet werden können neben den Materialien, welche in globalen Materialdefinitionen definiert wurden.

RULED

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

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. Die ebene Kurven-Polylinie kann keine Löcher enthalten. 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.

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

`xi, yi, zi:` Koordinaten der Eckpunkte des zweiten Polygones.

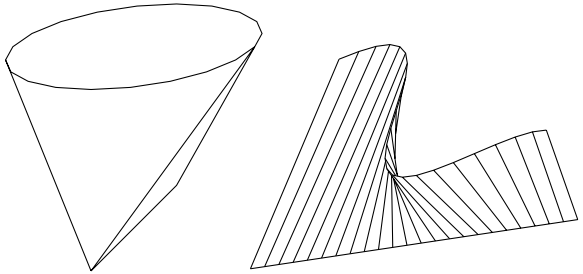
`mask:` Kontrolliert die Existenz des Grund-, Deck- und abschließenden Seitenpolygons und die Sichtbarkeit der Konturen der beiden Hauptpolygone. Das Seitenpolygon verbindet den ersten und den letzten Eckpunkt der beiden Hauptpolygone, falls mindestens eines davon nicht geschlossen ist.
`mask = j1 + 2*j2 + 4*j3 + 16*j5 + 32*j6 + 64*j7`, hierbei kann j jeweils 0 oder 1 sein.

`j1:` Grundfläche ist vorhanden.
`j2:` Deckfläche ist vorhanden (nicht wirksam, falls die Deckfläche keine Ebene ist).
`j3:` Fläche des Schlusspolygons ist vorhanden (ein Viereck in einer Ebene oder zwei Dreiecke).
`j5:` Konturen des Polygons der x-y-Ebene sind sichtbar.
`j6:` Konturen des zweiten Polygons sind sichtbar.
`j7:` Konturen der Oberflächen sind sichtbar, die Oberfläche ist in Segmente aufgeteilt.

`si:` Status der seitlichen Kanten.
`0:` Seitenkanten, vom Gelenkpunkt ausgehend, sind alle sichtbar.
`1:` Seitenkanten, vom Gelenkpunkt ausgehend, werden nur für die Konturenanzeige verwendet.

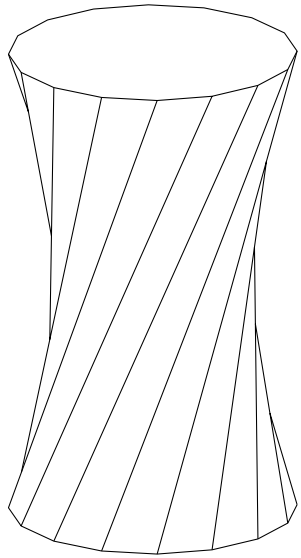
Einschränkung der Parameter:

```
n > 1
```



Beispiel:

```
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, 10,
        cos(135)*R,   sin(135)*R,   10,
        cos(157.5)*R, sin(157.5)*R, 10,
        cos(180)*R,   sin(180)*R,   10,
        cos(202.5)*R, sin(202.5)*R, 10,
        cos(225)*R,   sin(225)*R,   10,
        cos(247.5)*R, sin(247.5)*R, 10,
        cos(270)*R,   sin(270)*R,   10,
        cos(292.5)*R, sin(292.5)*R, 10,
        cos(315)*R,   sin(315)*R,   10,
        cos(337.5)*R, sin(337.5)*R, 10,
        cos(360)*R,   sin(360)*R,   10,
        cos(22.5)*R,  sin(22.5)*R,  10,
        cos(45)*R,    sin(45)*R,    10,
        cos(67.5)*R,  sin(67.5)*R,  10,
        cos(90)*R,    sin(90)*R,    10
```



RULEDSEGMENTED

```
RULEDSEGMENTED n, mask,
        x11, y11, z11, s1,..., x1n, y1n, z1n, sn,
        x21, y21, z21, ..., x2n, y2n, z2n
```

Kompatibilität: eingeführt in ARCHICAD 21.

RULEDSEGMENTED erzeugt eine Oberfläche basierend auf zwei willkürlich-geformten Polylinien im 3D-Raum. Die beiden Polylinien müssen aus der gleichen Anzahl an Vertices bestehen. Er erzeugt eine Folge von doppelt gekrümmte Flächen, ähnlich wie RULED, aber mit geringerer Beschränkung der eingegebene Polylinien und mit einer Unterteilung von besserer Qualität.

Zusammengehörige Vertices der beiden Profile sind mit geraden Linien verbunden. Zusammengehörige Paare von Schrägsegmenten der Profile sind durch eine doppelt gekrümmte Fläche (mathematisch: hyperbolisches Paraboloid) verbunden, wobei die Segmentierung in beiden Richtungen erfolgt, was zu deutlich glatteren Renderings und Schnittdarstellungen führt.

Bedingungen der Polylinien der Profile:

- beide sind 3D-Polylinien und brauchen nicht koplanar zu sein
- jedes kann geschlossen sein, aber darf keine Löcher enthalten
- 178/5000 jedes kann identische Vertices enthalten, sogar mehrere aufeinanderfolgende, die zu einer fächerförmigen Oberfläche führen.
- wenn eine Profil-Polylinie geschlossen und koplanar ist, kann ein geschlossenes Polygon erzeugt werden

n: Anzahl der Eckpunkte in jedem Polygon.

x1i, y1i, z1i: 3D Positionen von Vertices auf der ersten Profil-Polylinie.

x2i, y2i, z2i: 3D Positionen von Vertices auf der zweiten Profil-Polylinie.

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 mindestens eines davon nicht geschlossen ist.

$mask = j_1 + 2*j_2 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7$, hierbei kann j jeweils 0 oder 1 sein.

j_1: Grundfläche ist vorhanden (nicht von Bedeutung, wenn die erste Polylinie nicht koplanar ist und j3 nicht gesetzt ist),

j_2: Deckfläche ist vorhanden (nicht von Bedeutung, wenn die zweite Polylinie nicht koplanar ist und j3 nicht gesetzt ist),

j_3: Seitenschließfläche ist vorhanden (Fläche auf zusätzlichen Segmenten zwischen den letzten und ersten Knotenpunkten),

j_5: Kanten auf der ersten Profil-Polylinie sind sichtbar,

j_6: Kanten auf der zweiten Profil-Polylinie sind sichtbar,

j_7: Konturen der Oberflächen sind sichtbar, die Oberfläche ist in Segmente aufgeteilt.

si: Status der Erzeugenden (Seitenkanten zwischen einem Knoten auf der ersten Profil-Polylinie und dem entsprechenden Knoten auf der zweiten Polylinie).

`0:` Erzeugende ist sichtbar,
`1:` Erzeugende wird zum Anzeigen der Kontur verwendet,
`2:` Erzeugende ist sichtbar und erzeugt einen Bruch beim 3D-Rendern.

Einschränkung der Parameter:

```
n > 1
```

Beispiel:

```
RULEDSEGMENTED 4, 16+32,
               0, 0, 0, 2,
               1, 0, 0, 2,
               1, 1, 0, 2,
               1, 1, 1, 2,
               0, 0, 1,
               0, 1, 1,
               0, 1, 2,
               1, 2, 2
```

RULEDSEGMENTED{2}

```
RULEDSEGMENTED{2} top_material, bottom_material,
        n, mask, textureMode,
        x11, y11, z11, s1, mat1..., x1n, y1n, z1n, sn, matn,
        x21, y21, z21, ..., x2n, y2n, z2n
```

Kompatibilität: eingeführt in ARCHICAD 23.

RULEDSEGMENTED{2} ist eine Erweiterung von RULEDSEGMENTED mit der Möglichkeit, die Oberflächenattribute der erzeugten Oberflächen im Segmentdetail zu steuern und eine nutzerdefinierte Texturprojektion anzulegen.

Zusätzliche Parameter:

`top_material:` Oberflächenattributeindex der unteren Oberfläche (wenn die erste Polylinie koplanar ist und j1 + j3 gesetzt sind).

`bottom_material:` Oberflächenattributeindex der oberen Oberfläche (wenn die zweite Polylinie koplanar ist und j2 + j3 gesetzt sind).

`textureMode:` Textur-Projektionsmodus
`0:` automatisch, optimiert für gewölbte Flächen, wie bei RULEDSEGMENTED.
`1:` benutzerdefiniert, definiert durch COOR.

`mati:` Oberflächenattributeindex des erzeugten Flächensegments i.

Beispiel:

```
_topMatIndex = 22
_bottomMatIndex = 34
_segmentMatIndex_1 = 55
_segmentMatIndex_2 = 44

RULEDSEGMENTED{2} _topMatIndex, _bottomMatIndex,
                  4, 1+2+16+32, 0,
                  0, 0, 0, 2, _segmentMatIndex_1,
                  1, 0, 0, 2, _segmentMatIndex_2,
                  1, 1, 0, 2, _segmentMatIndex_1,
                  0, 1, 0, 2, _segmentMatIndex_2,
                  1, 0, 1,
                  1, 1, 1,
                  0, 1, 1,
               0, 0, 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 Grundpolygones 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. Trifft diese Bedingung nicht zu, wird der Pfad entlang der z-Achse mit dem Startpunkt in der x-y-Ebene erzeugt.

Die 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 in seiner eigenen Ebene. Drehung erfolgt an jedem Eckpunkt des Pfades.

scale: Skalierungsfaktor der Grundfläche. Skaliert an jedem Eckpunkt des Pfades.

ui, vi: Koordinaten der Eckpunkte des Grundpolygons.

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

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

mask = j_1 + 2*j_2 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7, hierbei kann j jeweils 0 oder 1 sein.

j_1: Grundfläche ist vorhanden.

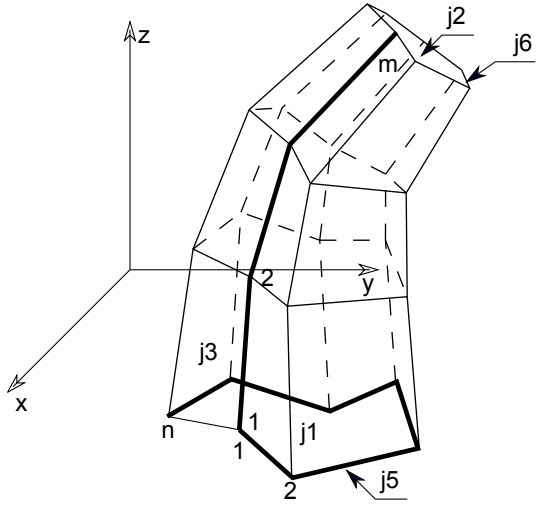
j_2: Deckfläche ist vorhanden.

j_3: Die Seitenfläche ist vorhanden.

j_5: Konturen der Grundfläche sind sichtbar.

j_6: Konturen der Deckfläche sind sichtbar.

j_7: Schnittkonturen sind sichtbar, die Oberfläche ist segmentiert.



si: Status der seitlichen Kanten.

0: Seitenkanten, vom Gelenkpunkt ausgehend, sind alle sichtbar.

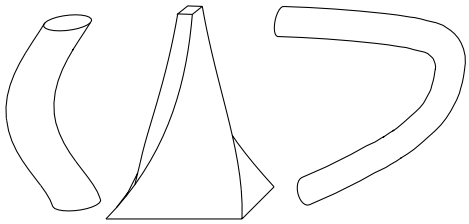
1: Seitenkanten, vom Gelenkpunkt ausgehend, werden nur für die Konturenanzeige verwendet.

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

Siehe „Zusätzliche Statuscodes“ für Details.

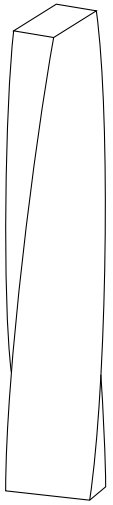
Einschränkung der Parameter:

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



Beispiel:

```
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, mask,
        u1, w1, s1,
        ...
        un, wn, sn,
        x1, y1, z1, angle1,
        ...
        xm, ym, zm, anglem
```

Körper, der durch Führen eines Polygonzuges entlang eines räumlichen Polygonzuges ohne Verzerren der Schnittflächen erzeugt wird. Die inneren Verbindungsflächen können in der u-w-Ebene des lokal an jedem Knotenpunkt erzeugten u-v-w Koordinatensystems gedreht werden.

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

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

U axis: 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. Dann wird die W-Achse aus dem vorhergehenden Punkt verwendet, um eine horizontale Ausrichtung zu definieren.

Das Schnittflächenpolygon der Röhre, in der Mitte der Pfad-Segmente gemessen, ist mit dem Basispolygon immer identisch (u1, w1, ..., un, wn).

Die Schnittpolygone in den Verbindungspunkten werden in der winkelhalbierenden Ebene der Verbindungssegmente plaziert. Das Basispolygon muss geschlossen sein.

n: Anzahl der Eckpunkte des Basispolygonzuges.

m: Anzahl der Eckpunkte des Pfad-Polygons.

ui, wi: Koordinaten der Eckpunkte des Grundpolygons.

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

anglei: Drehwinkel der Schnittflächen.

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

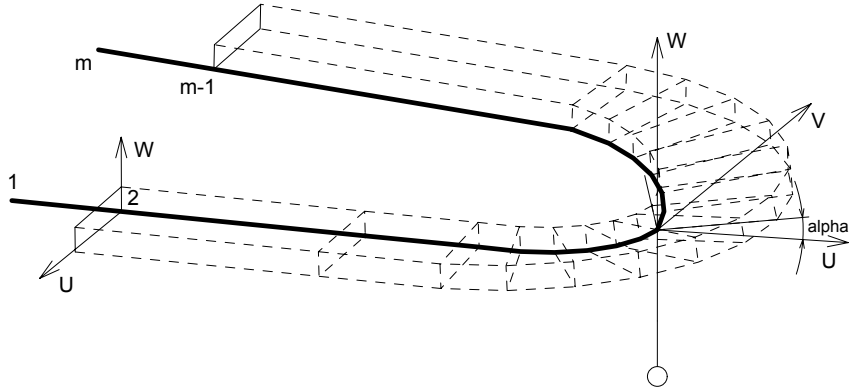
$mask = j_1 + 2*j_2 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8 + 512*j_{10} + 1024*j_{11} + 2048*j_{12} + 4096*j_{13}$, hierbei kann j jeweils 0 oder 1 sein.

j_1: Grundfläche ist vorhanden.

j_2: Schlussfläche ist vorhanden.

j_5: Konturen der Basisfläche (bei x2, y2, z2) sind sichtbar.

`j6:` Konturen der Schlussfläche (bei xm-1, ym-1, zm-1) sind sichtbar.
`j7:` Schnittkonturen sind sichtbar, die Oberfläche ist segmentiert,
`j8:` Schnittkonturen sind scharfkantig, Glättung der Oberfläche endet hier in OpenGL und im Rendering,
`j10:` Basiskanten sind an der Linieneliminierung beteiligt (*Kompatibilität: eingeführt in ARCHICAD 23.*),
`j11:` Endkanten sind an der Linieneliminierung beteiligt (*Kompatibilität: eingeführt in ARCHICAD 23.*),
`j12:` Längskanten (welche Kreuzungspunkte verbinden) sind an der Linienbeseitigung beteiligt (*Kompatibilität: eingeführt in ARCHICAD 23.*),
`j13:` Kanten von Kreuzungspunkten sind an der Linienbeseitigung beteiligt(*Kompatibilität: eingeführt in ARCHICAD 23.*).



`si:` Status der seitlichen Kanten.
`0:` Seitenkanten, vom Gelenkpunkt ausgehend, sind alle sichtbar.
`1:` Seitenkanten, vom Gelenkpunkt ausgehend, werden nur für die Konturenanzeige verwendet.
`2:` Wenn man ARCHICAD - oder Z-Buffer-Rendering-Engine verwendet und diese auf "Glatte Oberflächen" stellt, definiert die, zu diesem Punkt gehörige, parallel zu y-z-Ebene liegende Kante eine Unterbrechung. Diese Lösung entspricht der Definition von zusätzlichen Punkten. Die Berechnung wird vom Compiler ausgeführt. Mit anderen photorealistischen Renderern hat si=2 denselben Effekt, als würde man si=0 verwenden.

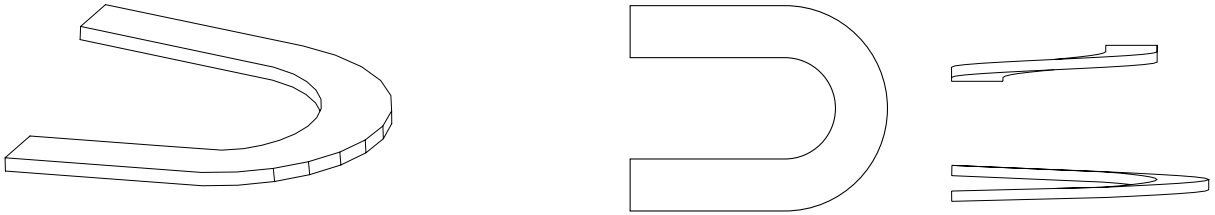
Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie.

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 an den Röhrenenden ist die gleiche, als wäre das Polygon an den benachbarten Endpunkten erzeugt worden, wenn die Röhre in die entsprechende Richtung fortgesetzt worden wäre.

Einschränkung der Parameter:

```
n > 2 and m > 3
```

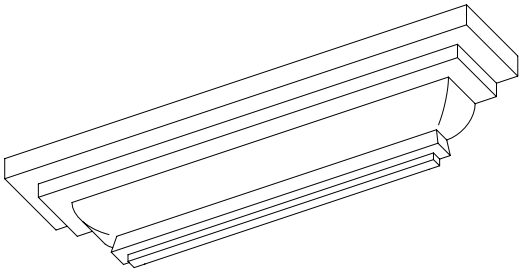
Beispiel 1:



```
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
```

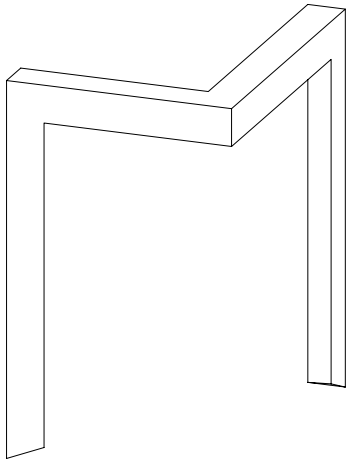
Beispiel 2:

```
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
```



Beispiel 3:

```
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
```



TUBE{2}

```
TUBE{2} top_material, bottom_material, cut_material,
        n, m, mask,
        u1, w1, s1, mat1,
        ...
        un, wn, sn, matn,
        x1, y1, z1, angle1,
        ...
        xm, ym, zm, anglem
```

Kompatibilität: eingeführt in ARCHICAD 21. Erweiterte Version vonTUBE:

- Löcher können innerhalb des Konturbasis-Polygons definiert werden
- Einzelne Flächen Attribute für obere und untere Polygone und Schnittbereiche-
- Individuelles Flächenattribut für Seitenpolygone, die zur gleichen Basispolygonkante gehören

V axis, W axis, U axis: gleiche Bedeutung wie in TUBE.

top_material: Oberfläche des schließenden Polygons.

bottom_material: Oberfläche des Ausgangspolygons.

cut_material: Oberfläche der Schnittflächen.

n, m, ui, wi: gleiche Bedeutung wie in TUBE.

xi, yi, zi, anglei: gleiche Bedeutung wie in TUBE. Pfad kann keine Bögen enthalten (Segmentierung erfolgt manuell).

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

mask = j_1 + 2*j_2 + 16*j_5 + 32*j_6 + 256*j_9 + 512*j_{10} + 1024*j_{11} + 2048*j_{12} + 4096*j_{13}, hierbei kann j jeweils 0 oder 1 sein.

j_1: Grundfläche ist vorhanden.

j_2: Schlussfläche ist vorhanden.

j_5: Konturen der Basisfläche (bei x2, y2, z2) sind sichtbar.

j_6: Konturen der Schlussfläche (bei xm-1, ym-1, zm-1) sind sichtbar.

j_9: Seitenkante und Oberfläche sind in gebogenen Bereichen des Profils geglättet,

j_{10}: Basiskanten sind an der Linieneliminierung beteiligt (*Kompatibilität: eingeführt in ARCHICAD 23.*),

j_{11}: Endkanten sind an der Linieneliminierung beteiligt (*Kompatibilität: eingeführt in ARCHICAD 23.*),

j_{12}: Längskanten (welche Kreuzungspunkte verbinden) sind an der Linieneliminierung beteiligt (*Kompatibilität: eingeführt in ARCHICAD 23.*),

j_{13}: Kanten von Kreuzungspunkten sind an der Linieneliminierung beteiligt (*Kompatibilität: eingeführt in ARCHICAD 23.*).

si: Status der seitlichen Kanten.

-1: Zeigt den letzten Knoten eines Lochs innerhalb des Basis-Polygons (duplizierter erster Knoten des Lochs) oder den Schließknoten des äußeren Polygons im Falle eines Basis-Polygons, das Löcher enthält. Der matn-Parameter wird in diesen duplizierten Knoten mit Status -1 ignoriert,

0, 1, 2: gleiche Bedeutung wie in TUBE.

mati: Einzelne Oberfläche der seitlichen Polygone, die zu der Kante gehören, die von ui, wi Knoten des Basis-Polygons ausgeht.

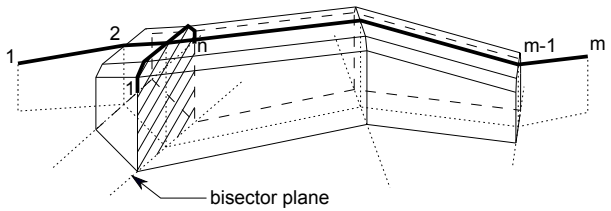
Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie. Solche Polygonränder werden bei der Verarbeitung automatisch segmentiert.

Beispiel:

```
matEnds1 = 12
matEnds2 = 24
matCut = 15
matOuter = 10
matInner = 13

TUBE{2} matEnds1, matEnds2, matCut,
          10, 4, 1 + 2 + 16 + 32,

          ! Außenkontur
          -0.01,  0.01,  0, matOuter,
          -0.01, -0.01,  0, matOuter,
           0.01, -0.01,  0, matOuter,
           0.01,  0.01,  0, matOuter,
          -0.01,  0.01, -1, matOuter,

          ! Lochkontur
          -0.008,  0.008,  0, matInner,
          -0.008, -0.008,  0, matInner,
           0.008, -0.008,  0, matInner,
           0.008,  0.008,  0, matInner,
          -0.008,  0.008, -1, matInner,

          ! Pfad
          0, 0, -1, 45,
          0, 0,  0, 45,
          0, 0,  1, 45,
          0, 0,  2, 45
```

TUBEA

```
TUBEA n, m, mask,
        u1, w1, s1,
        ...
        un, wn, sn,
        x1, y1, z1,
        ...
        xm, ym, zm
```



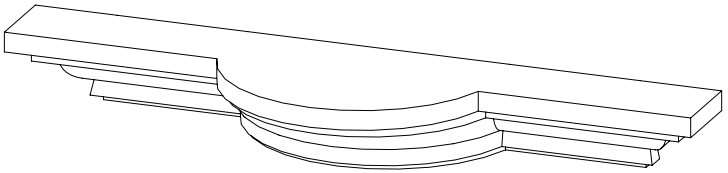
TUBEA ist ein Körper, der durch Führen eines Polygonzuges entlang eines räumlichen Polygonzuges mit einem anderen Algorithmus, als der TUBE-Anweisung erzeugt wird.

Das Schnittpolygon wird in jedem Verbindungspunkt der Pfad-Kurve generiert, es gleicht dem Basispolygon (u1, w1, ..., un, wn) und wird in der winkelhalbierenden Ebene der Projektionen der Verbindungssegmente zu der lokalen x-y-Ebene platziert. Das Basispolygon kann offen sein. In diesem Fall werden die Schnittpolygone generiert, um die lokale x-y-Ebene zu erreichen, wie bei REVOLVE-Oberflächen.

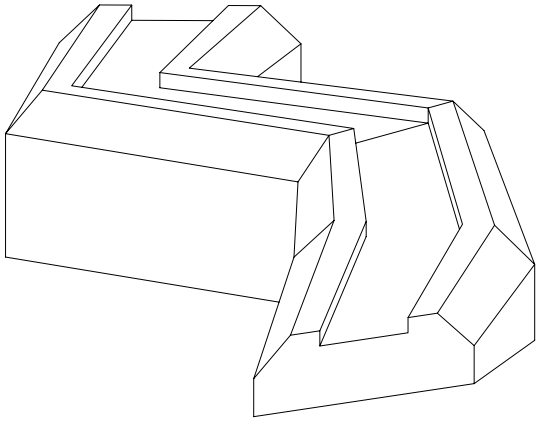
Die Schnittfläche der Röhre in der Mitte der Pfad-Segmente gemessen, kann von der des Basispolygons abweichen.

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

Siehe „Zusätzliche Statuscodes“ für Details.



Beispiel:



```
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, 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
```

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

mask:

mask = $4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7$, hierbei kann j jeweils 0 oder 1 sein.

j_3: Kontur der 1. Grenzlinie (x1, y1, z1) ist sichtbar (nur wirksam, wenn j7 eingestellt ist),

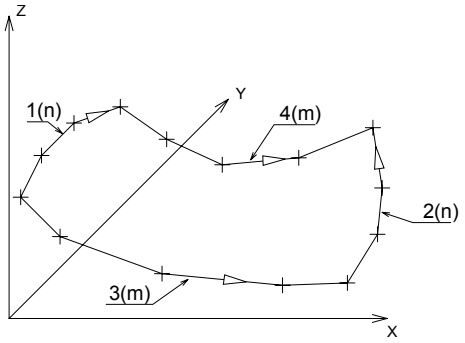
j_4: Kontur der 2. Grenzlinie (x2, y2, z2) ist sichtbar (nur wirksam, wenn j7 eingestellt ist),

j_5: Kontur der 3. Grenzlinie (x3, y3, z3) ist sichtbar (nur wirksam, wenn j7 eingestellt ist),

j_6: Kontur der 4. Grenzlinie (x4, y4, z4) ist sichtbar (nur wirksam, wenn j7 eingestellt ist),

j_7: Konturen der Oberflächen sind sichtbar, die Oberfläche ist in Segmente aufgeteilt.

Falls die Kanten auf der Oberfläche unsichtbar sind (Bit j7 ist auf Null gesetzt), werden alle Randkanten sichtbar, und die Bits j3-j6 werden unwirksam. Verwenden Sie COONS{2}, um die Sichtbarkeit der Begrenzungskante unabhängig von der Sichtbarkeit der Oberflächenkante zu definieren.



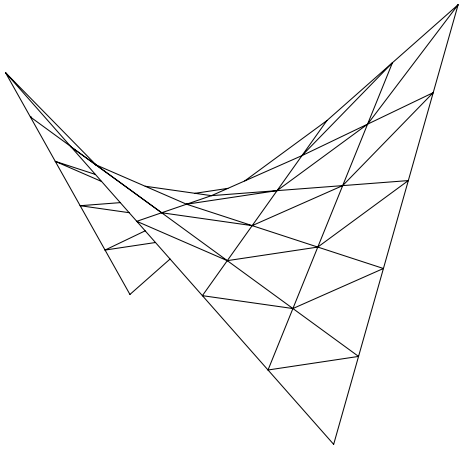
Die Ausrichtung der Begrenzungen ist obligatorisch: die Kurven 1 und 2 müssen von Kurve 3 nach 4 verlaufen, und die Kurven 3 und 4 müssen von Kurve 1 nach 2 verlaufen. Die Eckkoordinaten müssen in den jeweiligen Kurven die gleichen sein.

Einschränkung der Parameter:

```
n > 1, m > 1
```

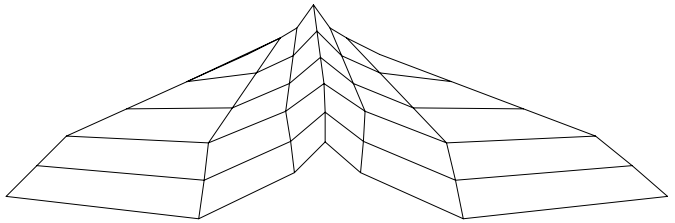
Beispiel 1:

```
COONS 6, 6, 4+8+16+32+64,
        ! 1. Begrenzung, n=6
        0, 0, 5,
        1, 0, 4,
        2, 0, 3,
        3, 0, 2,
        4, 0, 1,
        5, 0, 0,
        ! 2. Begrenzung, n=6
        0, 5, 0,
        1, 5, 1,
        2, 5, 2,
        3, 5, 3,
        4, 5, 4,
        5, 5, 5,
        ! 3. Begrenzung, m=6
        0, 0, 5,
        0, 1, 4,
        0, 2, 3,
        0, 3, 2,
        0, 4, 1,
        0, 5, 0,
        ! 4. Begrenzung, m=6
        5, 0, 0,
        5, 1, 1,
        5, 2, 2,
        5, 3, 3,
        5, 4, 4,
        5, 5, 5
```



Beispiel 2:

```
COONS 7, 6, 4+8+16+32+64,
        ! 1. Begrenzung, 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,
        ! 2. Begrenzung, 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,
        ! 3. Begrenzung, m=6
        1, 2, 0,
        2, 4, -0.5,
        3, 6, -1,
        4, 8, -1.5,
        5, 9, -1.8,
        6, 10, -2,
        ! 4. Begrenzung, m=6
        1, -2, 0,
        2, -4, -0.5,
        3, -6, -1,
        4, -8, -1.5,
        5, -9, -1.8,
        6, -10, -2
```



COONS{2}

```
COONS{2} 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
```

COONS {2} ist eine Erweiterung von COONS mit der Möglichkeit, die Sichtbarkeit von Flächen- und Begrenzungskanten unabhängig voneinander einzustellen.

mask:

mask = 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7, hierbei kann j jeweils 0 oder 1 sein.

j_3: Kontur der 1. Grenzlinie (x1, y1, z1) ist sichtbar.

j_4: Kontur der 2. Grenzlinie (x2, y2, z2) ist sichtbar.

j_5: Kontur der 3. Grenzlinie (x3, y3, z3) ist sichtbar.

j_6: Kontur der 4. Grenzlinie (x4, y4, z4) ist sichtbar.

j_7: Konturen der Oberflächen sind sichtbar, die Oberfläche ist in Segmente aufgeteilt.

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 der Deck-, Boden-, und Seitenmaterialien.

n: die Anzahl der Punkte des Massenpolygons.

m: die Anzahl der Punkte auf den Graten.

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

xi, yi, zi: die Koordinaten der Punkte.

mask:

`mask = j1 + 4*j3 + 16*j5 + 32*j6 + 64*j7 + 128*j8`, hierbei kann j jeweils 0 oder 1 sein.

`j1:` Grundfläche ist vorhanden.

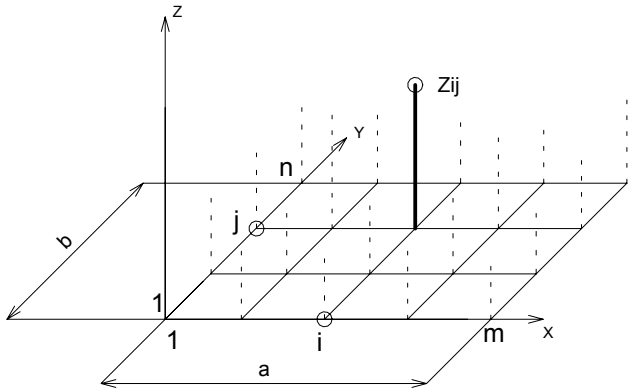
`j3:` Seitenflächen sind vorhanden.

`j5:` Konturen der Grund- und Seitenflächen sind sichtbar.

`j6:` Triangulationskanten sind sichtbar,

`j7:` Triangulationskanten sind sichtbar, obere Fläche ist nicht geglättet.

`j8:` Alle Graten sind scharfkantig, aber die Oberfläche ist glatt.



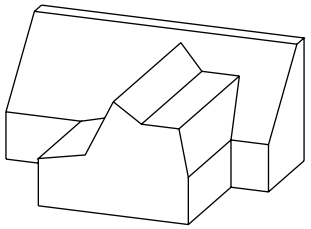
`si:` gleicht dem PRISM_-Befehl. Zusätzliche Statuscodes ermöglichen das Erstellen von Segmenten und Bögen in der planaren Polylinie. *Siehe „Zusätzliche Statuscodes“ für Details.*

Einschränkung der Parameter:

`n >= 3, m >= 0`

Beispiel:

```
MASS "Surf-White", "Surf-White", "Surf-White",
        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
```



MASS{2}

```
MASS{2} 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
```

Erweiterung von MASS mit einem zusätzlichen Mask-Bit und der Möglichkeit alle oberen Kanten des MASS-Körpers unsichtbar zu machen.

mask:

mask = $j_1 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9 + 512*j_{10}$, hierbei kann j jeweils 0 oder 1 sein.

j_1: Grundfläche ist vorhanden.

j_3: Seitenflächen sind vorhanden.

j_5: Konturen der Grund- und Seitenflächen sind sichtbar.

j_6: Konturen der Deckfläche sind sichtbar.

j_7: Innenkonturen der Deckfläche sind sichtbar, die Oberfläche ist in Segmente aufgeteilt.

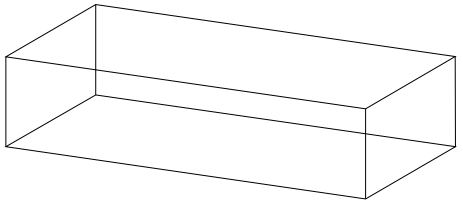
j_8: Alle Graten sind scharfkantig, aber die Oberfläche ist glatt.

j_9: Die Kanten sind an der Linieneleminierung beteiligt.

j_{10}: alle oberen Kanten sind unsichtbar.

Beispiel:

```
PEN 1
mat = IND (MATERIAL, "Metal-Aluminium")
FOR i=1 TO 2 STEP 1
  MASS{2} mat, mat, mat,
    5,  0,  1+4+16+32+64+256,   -1,
    0,  0,  0,   15,
    2,  0,  0,   15,
    2,  2,  0,   15,
    0,  2,  0,   15,
    0,  0,  0,   -1
  BODY -1
  ADDX 2
NEXT i
```



POLYROOF

```
POLYROOF defaultMat, k, m, n,
        offset, thickness, applyContourInsidePivot,
        z_1, ..., z_k,
        pivotX_1, pivotY_1, pivotMask_1,
        roofAngle_11, gableOverhang_11, topMat_11, bottomMat_11,
        ...
        roofAngle_1k, gableOverhang_1k, topMat_1k, bottomMat_1k,
        ...
        pivotX_m, pivotY_m, pivotMask_m,
        roofAngle_m1, gableOverhang_m1, topMat_m1, bottomMat_m1,
        ...
        roofAngle_mk, gableOverhang_mk, topMat_mk, bottomMat_mk,
        contourX_1, contourY_1, contourMask_1, edgeTrim_1, edgeAngle_1, edgeMat_1,
        ...
        contourX_n, contourY_n, contourMask_n, edgeTrim_n, edgeAngle_n, edgeMat_n
```

Der Befehl erzeugt ein mögliches multi-level Dach, bei welchem die Geometrie durch viele Parameter kontrolliert wird, im wesentlichen die Dachneigungswinkel und 2 Polygone: ein Zwischen-Polygon und ein Kontur-Polygon. Am Zwischen-Polygon ist das Dach im Dachneigungswinkel angeschlossen. Es steigt an bis es entweder die Höhe des nächsten Levels erreicht hat oder bis zu der Stelle, an welcher die

Dachflächen zusammenstoßen. Es fällt außerdem nach unten ab, bis es das Kontur-Polygon erreicht, welches Dachteile außerhalb des Polygons abschneidet. Kontur-Polygon kann außerdem dazu verwendet werden, um Öffnungen in das Dach zu schneiden.

defaultMat: der numerische Index des "inneren" Materials des Daches. Diese Material ist an Giebeln und angeschnittenen Oberflächen sichtbar, z.B. wenn das Dach durch einen Schnittbefehl beschnitten ist.

k: Anzahl der Levels

m: Anzahl der Eckpunkte des Zwischenpolygonzuges

n: Anzahl der Eckpunkte des Basispolygonzuges.

offset: ein Offset für die Stärke des Daches

thickness: Stärke des Daches

applyContourInsidePivot: falls Wert 0 ist, wird das äußere Kontur-Polygon ausschließlich außerhalb des Zwischenpolygons angewandt. Falls auf Wert 1 gesetzt, wird das äußere Kontur-Polygon sowohl oberhalb als auch unterhalb der Zwischenpolygon-Ebene angewandt. Die Einstellung 0 kann dazu verwendet werden, um zu verhindern, dass das Konturpolygon Giebel beschneidet, welche nach außen geneigt sind.

z_i: die Z-Koordinate eines Levels

pivotX_i, pivotY_i: Koordinaten der Eckpunkte des Zwischenpolygonzuges.

pivotMask_i:

`0:` kennzeichnet einen normalen Eckpunkt,

`-1:` kennzeichnet das Ende des derzeitigen Subpolygons (äußeren Kontur einer Öffnung). Die Werte für solch einen Eckpunkt müssen eine Kopie der Werte für den ersten Eckpunkt des Subpolygons sein. Ein Polygon muss immer mit einem Maskwert von -1 geschlossen werden, auch wenn es keine Öffnungen gibt.

roofAngle_i: Neigungswinkel einer Zwischenpolygonkante auf einem gegebenen Level. Falls der Winkel >= 90 ist, wird der entsprechende Teil des Daches zu einem Giebel.

gableOverhang_i: an den Seiten eines Giebels kann das Dach über den unteren Bereich seiner Grundfläche hinausragen. Das entsprechende Maß kann mit Hilfe dieses Parameters kontrolliert werden, welcher sich nur auf die Giebel auswirkt (roofAngle >= 90), welche mindestens auf dem 2. Level des Daches befinden.

topMat_i, bottomMat_i: der numerische Indexwert der Materialien der Ober- und Unterseite des Daches.

contourX_i, contourY_i: Koordinaten der Eckpunkte des Kontur-Polygonzuges

contourMask_i:

`0:` kennzeichnet einen normalen Eckpunkt,

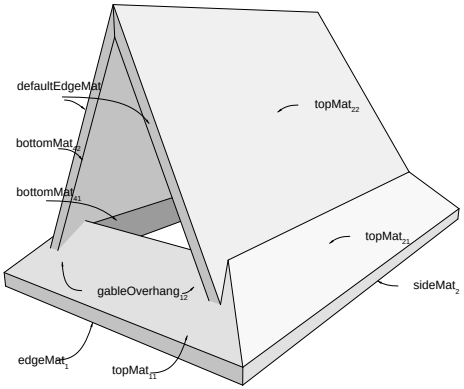
`-1:` kennzeichnet das Ende des derzeitigen Subpolygons (äußeren Kontur einer Öffnung). Die Werte für solch einen Eckpunkt müssen eine Kopie der Werte für den ersten Eckpunkt des Subpolygons sein. Ein Polygon muss immer mit einem Maskwert von -1 geschlossen werden, auch wenn es keine Öffnungen gibt.

`edgeTrim_i:` Legt fest, wie die Kante des Konturpolygons getrimmt ist. Mögliche Statuswerte:

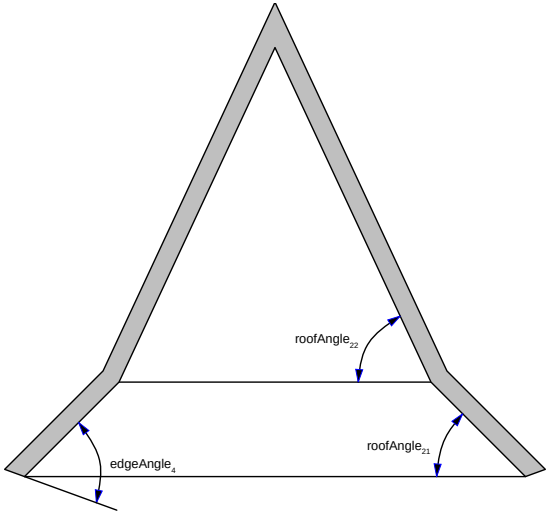
`0:` vertikal
`1:` rechtwinklig zur Dachoberfläche
`2:` horizontal
`3:` benutzerdefinierter Winkel in Bezug zur Dachoberfläche

`edgeAngle_i:` der benutzerdefinierte Winkel in Bezug zur Dachoberfläche. Hat nur dann einen Effekt, wenn edgeTrim den Wert 3 besitzt (benutzerdefinierter Winkel in Bezug zur Dachoberfläche).

`edgeMat_i:` der numerische Indexwert der Materialien an der Kante des Daches, an welcher das Konturpolygon die Dachfläche beschneidet:



Figur 1: Materialien



Figur 2: Winkel

Beispiel:

```
POLYROOF "Paint-01",
        2, 5, 5,
        0, 0.2, 0,
        ! Start der Z_Werte
        2.7,
        3.2,
        ! Start des Zwischenpolygons
        2, 8, 0,
        45, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        90, 0.5, ind(material, "Paint-01"), ind(material, "Paint-01"),
        2, 3, 0,
        45, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        65, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        10, 3, 0,
        45, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        65, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        10, 8, 0,
        45, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        65, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        2, 8, -1,
        45, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        90, 0.5, ind(material, "Paint-01"), ind(material, "Paint-01"),
        ! Start des Kontur-Polygons
        1.5, 8.5, 0, 0, 0, ind(material, "Paint-01"),
        1.5, 2.5, 0, 0, 0, ind(material, "Paint-01"),
        10.5, 2.5, 0, 0, 0, ind(material, "Paint-01"),
        10.5, 8.5, 0, 0, 0, ind(material, "Paint-01"),
        1.5, 8.5, -1, 0, 0, ind(material, "Paint-01")
```

Ergebnis: siehe Figur 1

POLYROOF{2}

```
POLYROOF{2} defaultMat, k, m, n,
        offset, thickness, totalThickness, applyContourInsidePivot,
        z_1, ..., z_k,
        pivotX_1, pivotY_1, pivotMask_1,
        roofAngle_11, gableOverhang_11, topMat_11, bottomMat_11,
        ...
        roofAngle_1k, gableOverhang_1k, topMat_1k, bottomMat_1k,
        ...
        pivotX_m, pivotY_m, pivotMask_m,
        roofAngle_m1, gableOverhang_m1, topMat_m1, bottomMat_m1,
        ...
        roofAngle_mk, gableOverhang_mk, topMat_mk, bottomMat_mk,
        contourX_1, contourY_1, contourMask_1, edgeTrim_1, edgeAngle_1, edgeMat_1,
        ...
        contourX_n, contourY_n, contourMask_n, edgeTrim_n, edgeAngle_n, edgeMat_n
```

POLYROOF{2} ist eine Erweiterung von POLYROOF mit der Möglichkeit, die Gesamtstärke des Daches zu definieren. Dieser Parameter sollte zusammen mit dem Offset und der Stärke berücksichtigt werden, wenn die Erzeugung eines Dachausschnitts erwünscht ist. In diesem Fall sollte die Stärke und der Offset auf den Wert der Stärke des Ausschnitts, beziehungsweise den Abstand zwischen der oberen Ebene des Ausschnitts und dem kompletten Dach gesetzt werden.

totalThickness: die Gesamtstärke des Daches.

POLYROOF{3}

```
POLYROOF{3} defaultMat, mask, k, m, n,
          offset, thickness, totalThickness, applyContourInsidePivot,
          z_1, ..., z_k,
          pivotX_1, pivotY_1, pivotMask_1,
          roofAngle_11, gableOverhang_11, topMat_11, bottomMat_11,
          ...
          roofAngle_1k, gableOverhang_1k, topMat_1k, bottomMat_1k,
          ...
          pivotX_m, pivotY_m, pivotMask_m,
          roofAngle_m1, gableOverhang_m1, topMat_m1, bottomMat_m1,
          ...
          roofAngle_mk, gableOverhang_mk, topMat_mk, bottomMat_mk,
          contourX_1, contourY_1, contourMask_1, edgeTrim_1, edgeAngle_1, edgeMat_1,
          ...
          contourX_n, contourY_n, contourMask_n, edgeTrim_n, edgeAngle_n, edgeMat_n
```

POLYROOF{3} ist eine Erweiterung von POLYROOF{2} mit der Möglichkeit, das globale Verhalten des erzeugten Daches zu kontrollieren.

mask: kontrolliert das globale Verhalten des erzeugten Daches.

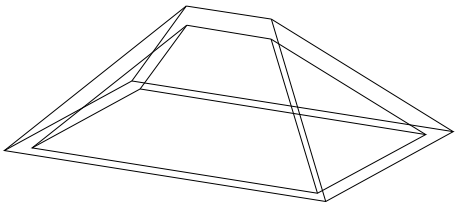
`mask = j1 + 2*j2`, hierbei kann j jeweils 0 oder 1 sein.

`j1`: Kanten sind an der Linieneleminierung beteiligt.

`j2`: Macht alle Kanten unsichtbar.

Beispiel:

```
pen 1
mat = IND (MATERIAL, "Metal-Aluminium")
a = -0.4242640691048 : b =  4.424264068326
c =  6.424264068326
POLYROOF{3} mat,1, 2,  5,  5,
  0, 0.3, 0.3,  1, 0, 1,
  a, b, 0, 45, 0, mat, mat, 90, 0, mat, mat,
  a, a, 0, 45, 0, mat, mat, 90, 0, mat, mat,
  c, a, 0, 45, 0, mat, mat, 90, 0, mat, mat,
  c, b, 0, 45, 0, mat, mat, 90, 0, mat, mat,
  a, b, -1,45, 0, mat, mat, 90,  0, mat, mat,
  -0.8, -0.8,  0,  2,  0, mat,
  6.8, -0.8,  0,  2,  0, mat,
  6.8,  4.8,  0,  2,  0, mat,
  -0.8,  4.8,  0,  2,  0, mat,
  -0.8, -0.8, -1,  2,  0, mat

a = 0.1514718617904 : b = 3.848528136652
c = 5.848528136652 : q = 0.5757359305057
w = 5.424264067936 :  e = 3.424264056692
POLYROOF{3} mat,1, 1,  5,  5,
  0,   0.3,  0.3,  1, 0.5757359312847,
  a, b, 0, 45,  0, mat, mat,
  a, a, 0, 45,  0, mat, mat,
  c, a, 0, 45,  0, mat, mat,
  c, b, 0, 45,  0, mat, mat,
  a, b, -1, 45,  0, mat, mat,
  q, q,   0,  0,  0, mat,
  w, q,   0,  0,  0, mat,
  w, e,   0,  0,  0, mat,
  q, e,   0,  0,  0, mat,
  q, q,  -1,  0,  0, mat
```



POLYROOF{4}

```
POLYROOF{4} defaultMat, mask, k, m, n,
          offset, thickness, totalThickness, applyContourInsidePivot,
          z_1, ..., z_k,
          pivotX_1, pivotY_1, pivotMask_1,
          roofAngle_11, gableOverhang_11, topMat_11, bottomMat_11,
          ...
          roofAngle_1k, gableOverhang_1k, topMat_1k, bottomMat_1k,
          ...
          pivotX_m, pivotY_m, pivotMask_m,
          roofAngle_m1, gableOverhang_m1, topMat_m1, bottomMat_m1,
          ...
          roofAngle_mk, gableOverhang_mk, topMat_mk, bottomMat_mk,
          contourX_1, contourY_1, contourMask_1, edgeTrim_1, edgeAngle_1, edgeMat_1,
          ...
          contourX_n, contourY_n, contourMask_n, edgeTrim_n, edgeAngle_n, edgeMat_n
```

POLYROOF{4} ist eine Erweiterung von POLYROOF{3} mit der Möglichkeit einer Inline-Materialdefinition, was bedeutet, dass im GDL-Script lokal definierte Materialien ebenfalls verwendet werden können neben den Materialien, welche in globalen Materialdefinitionen definiert wurden.

EXTRUDEDSHELL

```
EXTRUDEDSHELL topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4,
        defaultMat,
        n, offset, thickness, flipped, trimmingBody,
        x_tb, y_tb, x_te, y_te, topz, tangle,
        x_bb, y_bb, x_be, y_be, bottomz, bangle,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThicakenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n
```

Oberfläche, welche zunächst durch das Extrudieren einer Polylinie erzeugt wird, welcher anschließend eine Wandstärke hinzugefügt wird.

topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4: Materialien auf der Oberseite, der Unterseite und den 4 Seiten des Objektes.

defaultMat: der numerische Index des "inneren" Materials des Objektes. Diese Material ist an angeschnittenen Oberflächen sichtbar, z.B. wenn das Objekt durch einen Schnittbefehl beschnitten ist.

n: Anzahl der Basispolygon-Eckpunkte.

offset: ein Offset für die Stärke der Schale. Darf nicht negativ sein.

thickness: Stärke der Schale

flipped:
1: wenn die Schale gespiegelt werden soll,
0: andernfalls.

trimmingBody:
1: wenn die Schale zu Verschneidungszwecken geschlossen werden muss,
0: andernfalls.

x_tb, y_tb, x_te, y_te, topz, tangle: Legt die obere Abschlussfläche der Extrusion fest. Die Bedeutung der Variablen ist die selbe wie beim Befehl SPRISM_{2}.

x_bb, y_bb, x_be, y_be, bottomz, bangle: Legt die untere Abschlussfläche der Extrusion fest. Die Bedeutung der Variablen ist die selbe wie beim Befehl SPRISM_{2}.

preThickenTran_i: eine Transformation, welche vor der Erzeugung der Wandstärke ausgeführt wird. Siehe auch XFORM bezüglich der Bedeutung der Parameter.

x_i, y_i, s_i: X und Y Koordinaten, Statuswerte für die Polylinie des Grundpolygons. Siehe EXTRUDE für Details. Die Sichtbarkeit der Seiten kann nicht mit Hilfe der Statuswerte beeinflusst werden.

EXTRUDEDSHELL{2}

```
EXTRUDEDSHELL{2} topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4,
        defaultMat,
        n, status, offset, thickness, flipped, trimmingBody,
        x_tb, y_tb, x_te, y_te, topz, tangle,
        x_bb, y_bb, x_be, y_be, bottomz, bangle,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThicakenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n
```

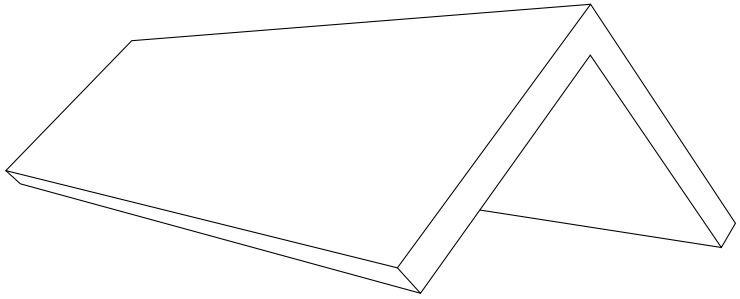
EXTRUDEDSHELL{2} ist eine Erweiterung von EXTRUDEDSHELL mit der Möglichkeit, die Kanten zwischen der originalen und verdickten Oberfläche unsichtbar zu machen.

status: Status-Werte:

`status = j1`, hierbei kann j jeweils 0 oder 1 sein.

`j1`: Macht die Kanten zwischen der originalen und verdickten Oberfläche unsichtbar.

Beispiel:



```
EXTRUDEDSHELL "Paint-02", "Surf-Stucco Yellow",
        "Surf-Stucco Yellow", "Surf-Stucco Yellow", "Surf-Stucco Yellow",
        "Surf-Stucco Yellow", "Surf-Stucco Yellow",
        3,   0.00,   0.30,      0, 0,
        ! 2 slant planes
        0.00,   0.00,   0.00,   1.00,   0.00,   0.00,
        0.00,   0.00,   0.00,   1.00, -10.00,   0.00,
        ! transformation matrix
        0.00,   0.00,   1.00,   0.00,
        1.00,   0.00,   0.00,   0.00,
        0.00,   1.00,   0.00,   0.00,
        ! profile polyline
        2.00,   0.00,     15,
        0.00,   2.00,     15,
        -2.00,   0.00,     15
```

EXTRUDEDSHELL{3}

```
EXTRUDEDSHELL{3} topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4,
        defaultMat,
        n, status, offset, thickness, flipped, trimmingBody,
        x_tb, y_tb, x_te, y_te, topz, tangle,
        x_bb, y_bb, x_be, y_be, bottomz, bangle,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThicakenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n
```

EXTRUDEDSHELL{3} ist eine Erweiterung von EXTRUDEDSHELL{2} mit der Möglichkeit einer Inline-Materialdefinition, was bedeutet, dass im GDL-Script lokal definierte Materialien ebenfalls verwendet werden können neben den Materialien, welche in globalen Materialdefinitionen definiert wurden.

REVOLVEDSHELL

```
REVOLVEDSHELL topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4,
        defaultMat,
        n, offset, thickness, flipped, trimmingBody, alphaOffset, alpha,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n
```

Körper, der durch Rotation eines in der x-y-Ebene liegenden Linienzuges um die x-Achse erzeugt wird, zu welchem anschließend eine Wandstärke hinzugefügt wird.

topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4: Materialien auf der Oberseite, der Unterseite und den 4 Seiten des Objektes.

defaultMat: der numerische Index des "inneren" Materials des Objektes. Diese Material ist an angeschnittenen Oberflächen sichtbar, z.B. wenn das Objekt durch einen Schnittbefehl beschnitten ist.

n: Anzahl der Basispolygon-Eckpunkte.

offset: ein Offset für die Stärke der Schale. Darf nicht negativ sein.

thickness: Stärke der Schale

flipped:
1: wenn die Schale gespiegelt werden soll,
0: andernfalls.

trimmingBody:
1: wenn die Schale zu Verschneidungszwecken geschlossen werden muss,
0: andernfalls.

alphaOffset: Startwert des Drehwinkels

alpha: Winkellänge des Drehwinkels in Grad, kann negativ sein.

preThickenTran_i: eine Transformation, welche vor der Erzeugung der Wandstärke ausgeführt wird. Siehe auch XFORM bezüglich der Bedeutung der Parameter.

x_i, y_i, s_i: X und Y Koordinaten, Statuswerte für die Polylinie des Grundpolygons. Siehe EXTRUDE für Details. Die Sichtbarkeit der Seiten kann nicht mit Hilfe der Statuswerte beeinflusst werden.

REVOLVEDSHELL{2}

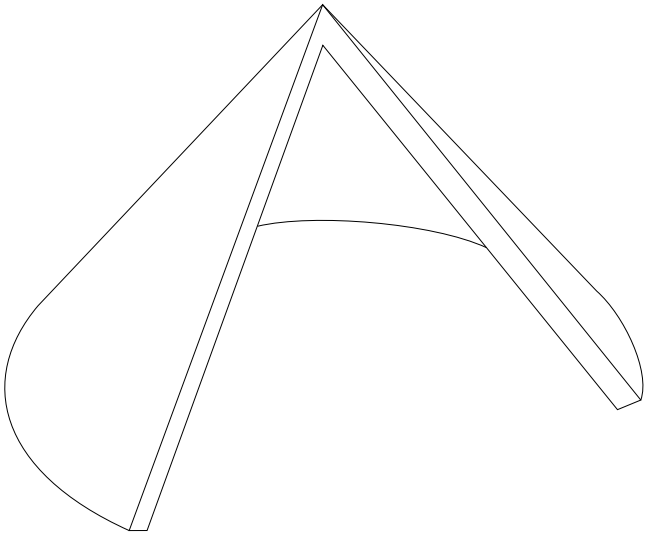
```
REVOLVEDSHELL{2} topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4,
        defaultMat,
        n, status, offset, thickness, flipped, trimmingBody, alphaOffset, alpha,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n
```

REVOLVEDSHELL{2} ist eine Erweiterung von REVOLVEDSHELL mit der Möglichkeit, Kanten von Oberflächen und Kanten zwischen der originalen und verdickten Oberfläche unsichtbar zu machen.

status: Status-Werte:
status = j_1 + 2*j_2, hierbei kann j jeweils 0 oder 1 sein.
j_1: Macht die Kanten zwischen der originalen und verdickten Oberfläche unsichtbar.
j_2: Macht die Kanten auf Oberflächen unsichtbar.

Beispiel:

```
REVOLVEDSHELL "Paint-02", "Surf-Stucco Yellow",
        "Surf-Stucco Yellow", "Surf-Stucco Yellow", "Surf-Stucco Yellow",
        "Surf-Stucco Yellow", "Surf-Stucco Yellow",
        2,   0.00,   0.30,      0,  0, 0.00, 270.00,
        ! transformation matrix
        0.00,   0.00,  -1.00,   0.00,
        0.00,   1.00,   0.00,   0.00,
        1.00,   0.00,   0.00,   0.00,
        ! profile polyline
        4.00,   0.00,      2,
        0.00,   4.00,      2
```



REVOLVEDSHELL{3}

```
REVOLVEDSHELL{3} topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4,
        defaultMat,
        n, status, offset, thickness, flipped, trimmingBody, alphaOffset, alpha,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n
```

REVOLVEDSHELL{3} ist eine Erweiterung von REVOLVEDSHELL{2} mit der Möglichkeit einer Inline-Materialdefinition, was bedeutet, dass im GDL-Script lokal definierte Materialien ebenfalls verwendet werden können neben den Materialien, welche in globalen Materialdefinitionen definiert wurden.

REVOLVEDSHELLANGULAR

```
REVOLVEDSHELLANGULAR topMat, bottomMat,
        sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
        n, offset, thickness, flipped, trimmingBody, alphaOffset, alpha,
        segmentationType, nOfSegmente,
        preThickenTran_11, preThickenTran_12, preThickenTran_13,
        preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23,
        preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33,
        preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n
```

Eine eckige Variante des Befehls REVOLVEDSHELL. Die Parameter sind die gleichen, wobei es zusätzlich folgende Parameter gibt:

segmentationType: Muss entweder 1 oder 2 sein.

1: bedeutet, dass 360 Grad aufgesplittet werden in nOfSegments Segmente.

2: bedeutet, dass der aktuelle Rotationswinkel (durch den Parameter alpha bestimmt)) in nOfSegmente aufgesplittet wird.

nOfSegments: Anzahl der Segmente, siehe auch segmentationType (vorhergehend).

REVOLVEDSHELLANGULAR{2}

```
REVOLVEDSHELLANGULAR{2} topMat, bottomMat,
        sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
        n, status, offset, thickness, flipped, trimmingBody, alphaOffset, alpha,
        segmentationType, nOfSegments,
        preThickenTran_11, preThickenTran_12, preThickenTran_13,
        preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23,
        preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33,
        preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n
```

REVOLVEDSHELLANGULAR{2} ist eine Erweiterung von REVOLVEDSHELLANGULAR mit der Möglichkeit, Kanten von Oberflächen und Kanten zwischen der originalen und verdickten Oberfläche unsichtbar zu machen.

status: Status-Werte:

status = j_1 + 2*j_2, hierbei kann j jeweils 0 oder 1 sein.

j_1: Macht die Kanten zwischen der originalen und verdickten Oberfläche unsichtbar.

j_2: Macht die Kanten auf Oberflächen unsichtbar.

REVOLVEDSHELLANGULAR{3}

```
REVOLVEDSHELLANGULAR{3} topMat, bottomMat,
        sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
        n, status, offset, thickness, flipped, trimmingBody, alphaOffset, alpha,
        segmentationType, nOfSegments,
        preThickenTran_11, preThickenTran_12, preThickenTran_13,
        preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23,
        preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33,
        preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n
```

REVOLVEDSHELLANGULAR{3} ist eine Erweiterung von REVOLVEDSHELLANGULAR{2} mit der Möglichkeit einer Inline-Materialdefinition, was bedeutet, dass im GDL-Script lokal definierte Materialien ebenfalls verwendet werden können neben den Materialien, welche in globalen Materialdefinitionen definiert wurden.

RULEDSHELL

```
RULEDSHELL topMat, bottomMat,
        sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
        n, m, g,
        offset, thickness, flipped, trimmingBody,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        firstpolyX_1, firstpolyY_1, firstpolyS_1,
        ...
        firstpolyX_n, firstpolyY_n, firstpolyS_n,
        secondpolyX_1, secondpolyY_1, secondpolyS_1,
        ...
        secondpolyX_m, secondpolyY_m, secondpolyS_m,
        profile2Tran_11, profile2Tran_12, profile2Tran_13, profile2Tran_14
        profile2Tran_21, profile2Tran_22, profile2Tran_23, profile2Tran_24
        profile2Tran_31, profile2Tran_32, profile2Tran_33, profile2Tran_34
        generatrixFirstIndex_1, generatrixSecondIndex_1,
        ...
        generatrixFirstIndex_g, generatrixSecondIndex_g
```

Oberfläche, die aus der Verbindung von zwei Polylinien erzeugt wird.

topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4: Materialien auf der Oberseite, der Unterseite und den 4 Seiten des Objektes.

defaultMat: der numerische Index des "inneren" Materials des Objektes. Diese Material ist an angeschnittenen Oberflächen sichtbar, z.B. wenn das Objekt durch einen Schnittbefehl beschnitten ist.

n: Anzahl der Polygon-Eckpunkte des ersten Profils

m: Anzahl der Polygon-Eckpunkte des zweiten Profils

g: Anzahl der Erzeugenden

offset: ein Offset für die Stärke der Schale. Darf nicht negativ sein.

thickness: Stärke der Schale

flipped:
1: wenn die Schale gespiegelt werden soll,
0: andernfalls.

preThickenTran: eine Transformation, welche vor der Erzeugung der Wandstärke ausgeführt wird. Siehe auch XFORM bezüglich der Bedeutung der Parameter.

trimmingBody:
1: wenn die Schale zu Verschneidungszwecken geschlossen werden muss,
0: andernfalls.

firstpolyX, firstpolyY, firstpolyS: X und Y Koordinaten, Statuswerte für die Polylinie des ersten Polygons. Siehe REVOLVE für Details.

secondpolyX, secondpolyY, secondpolyS: X und Y Koordinaten, Statuswerte für die Polylinie des zweiten Polygons. Siehe REVOLVE für Details.

profile2Tran: eine Transformation, welche auf das zweite Profil angewendet wird. Verwenden Sie diese Transformation, um das zweite Profil relativ zum ersten zu positionieren. Siehe auch XFORM bezüglich der Bedeutung der Parameter.

generatrixFirstIndex, generatrixSecondIndex: Paare von Indizes, eine von der ersten Polylinie und einer von der zweiten Polylinie. Die Eckpunkte mit den gleichen Indizes werden mit einer Linie verbunden.

RULEDSHELL{2}

```
RULEDSHELL{2} topMat, bottomMat,
        sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
        n, m, g, status,
        offset, thickness, flipped, trimmingBody,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        firstpolyX_1, firstpolyY_1, firstpolyS_1,
        ...
        firstpolyX_n, firstpolyY_n, firstpolyS_n,
        secondpolyX_1, secondpolyY_1, secondpolyS_1,
        ...
        secondpolyX_m, secondpolyY_m, secondpolyS_m,
        profile2Tran_11, profile2Tran_12, profile2Tran_13, profile2Tran_14
        profile2Tran_21, profile2Tran_22, profile2Tran_23, profile2Tran_24
        profile2Tran_31, profile2Tran_32, profile2Tran_33, profile2Tran_34
        generatrixFirstIndex_1, generatrixSecondIndex_1,
        ...
        generatrixFirstIndex_g, generatrixSecondIndex_g
```

RULEDSHELL{2} ist eine Erweiterung von RULEDSHELL mit der Möglichkeit, Kanten von Oberflächen und Kanten zwischen der originalen und verdickten Oberfläche unsichtbar zu machen.

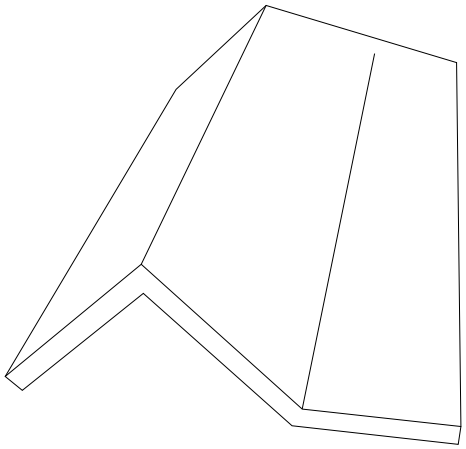
status: Status-Werte:

status = j_1 + 2*j_2, hierbei kann j jeweils 0 oder 1 sein.

j_1: Macht die Kanten zwischen der originalen und verdickten Oberfläche unsichtbar.

j_2: Macht die Kanten auf Oberflächen unsichtbar.

Beispiel:



```
RULEDSHELL "Paint-14", "Paint-14",
        "Paint-14", "Paint-14", "Paint-14", "Paint-14", "Paint-14",
        4,      3,      3,
        0.00,   0.30,      0, 0,
        ! transformation matrix
        1.00,   0.00,   0.00,   0.00,
        0.00,   0.00,  -1.00,   0.00,
        0.00,   1.00,   0.00,   0.00,
        ! profile 1 polyline
        0.00,   0.00,      2,
        2.00,   2.00,      2,
        4.00,   0.00,      2,
        6.00,   0.00,      2,
        ! profile 2 polyline
        0.00,   0.00,      2,
        2.00,   2.00,      2,
        6.00,   1.00,      2,
        ! transformation matrix
        1.00,   0.00,   0.00,   0.00,
        0.00,   1.00,   0.00,   0.00,
        0.00,   0.00,   1.00, -10.00,
        ! generatrices
        1,      1,
        2,      2,
        4,      3
```

RULEDSHELL{3}

```
RULEDSHELL{3} topMat, bottomMat,
        sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
        n, m, g, status,
        offset, thickness, flipped, trimmingBody,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        firstpolyX_1, firstpolyY_1, firstpolyS_1,
        ...
        firstpolyX_n, firstpolyY_n, firstpolyS_n,
        secondpolyX_1, secondpolyY_1, secondpolyS_1,
        ...
        secondpolyX_m, secondpolyY_m, secondpolyS_m,
        profile2Tran_11, profile2Tran_12, profile2Tran_13, profile2Tran_14
        profile2Tran_21, profile2Tran_22, profile2Tran_23, profile2Tran_24
        profile2Tran_31, profile2Tran_32, profile2Tran_33, profile2Tran_34
        generatrixFirstIndex_1, generatrixSecondIndex_1,
        ...
        generatrixFirstIndex_g, generatrixSecondIndex_g
```

RULEDSHELL{3} ist eine Erweiterung von RULEDSHELL{2} mit der Möglichkeit einer Inline-Materialdefinition, was bedeutet, dass im GDL-Script lokal definierte Materialien ebenfalls verwendet werden können neben den Materialien, welche in globalen Materialdefinitionen definiert wurden.

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 [RGB-Farbanteile: red, green, blue] 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 tangierenden, 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

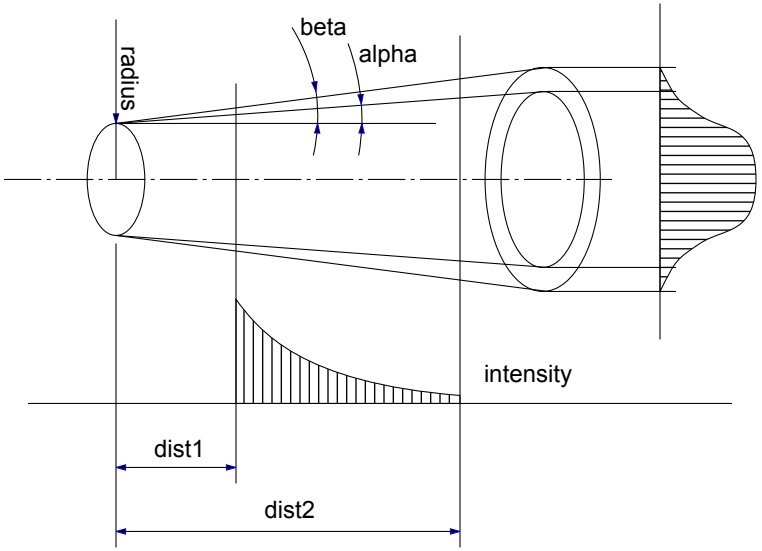
Übergang.) Durch die Werte distance1 und distance 2 kann der Lichteffekt auf einen Bereich begrenzt werden. Auch hier wird die Abschwächung der Intensität in Abhängigkeit von der Entfernung über einen Parameter angegeben (= distance_falloff). Dabei gibt der Wert Null dem Lichtstrahl eine konstante Intensität, höhere Werte bedeuten einen höheren Intensitätsverlust.

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

`shadow:` bestimmt den Schattenwurf.

`0:` das Licht wirft keinen Schatten

`1:` das Licht wirft Schatten



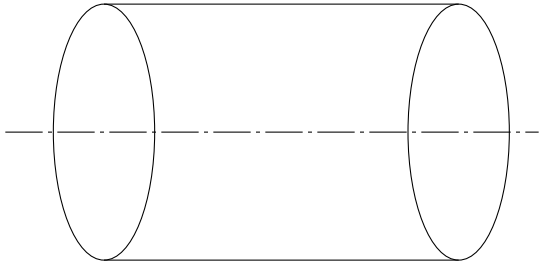
Einschränkung der Parameter:

```
alpha <= beta <= 80°
```

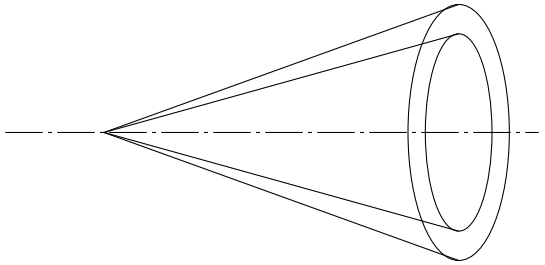
Kombinationen von Parametern mit speziellen Bedeutungen:

`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.

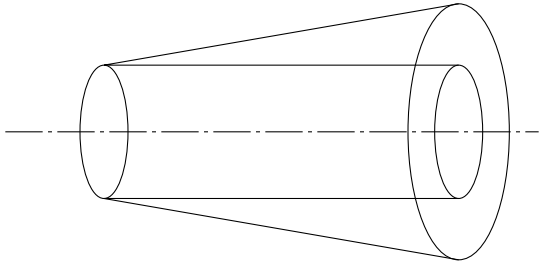
`radius > 0, alpha = 0, beta = 0:` Ein gerichtetes Licht mit parallelen Strahlen.



`r = 0, alpha > 0, beta > 0:` Ein gerichtetes Licht mit konischen Strahlen.



`r > 0, alpha = 0, beta > 0:` Ein gerichtetes Licht mit parallelen Strahlen und konischem Lichtabfall.



Lichtdefinitionen können nach dem Schlüsselwort ADDITIONAL_DATA optionale zusätzliche Datendefinitionen enthalten. Zusätzliche Daten nach dem Schlüsselwort ADDITIONAL_DATA haben einen Namen (namei) und einen Wert (valuei), der ein Ausdruck eines beliebigen Typs sein kann, auch ein Array. Wenn ein String Parametername mit dem Teilstring "_file" endet, wird sein Wert als Dateiname betrachtet und in das Archiv-Projekt eingefügt.

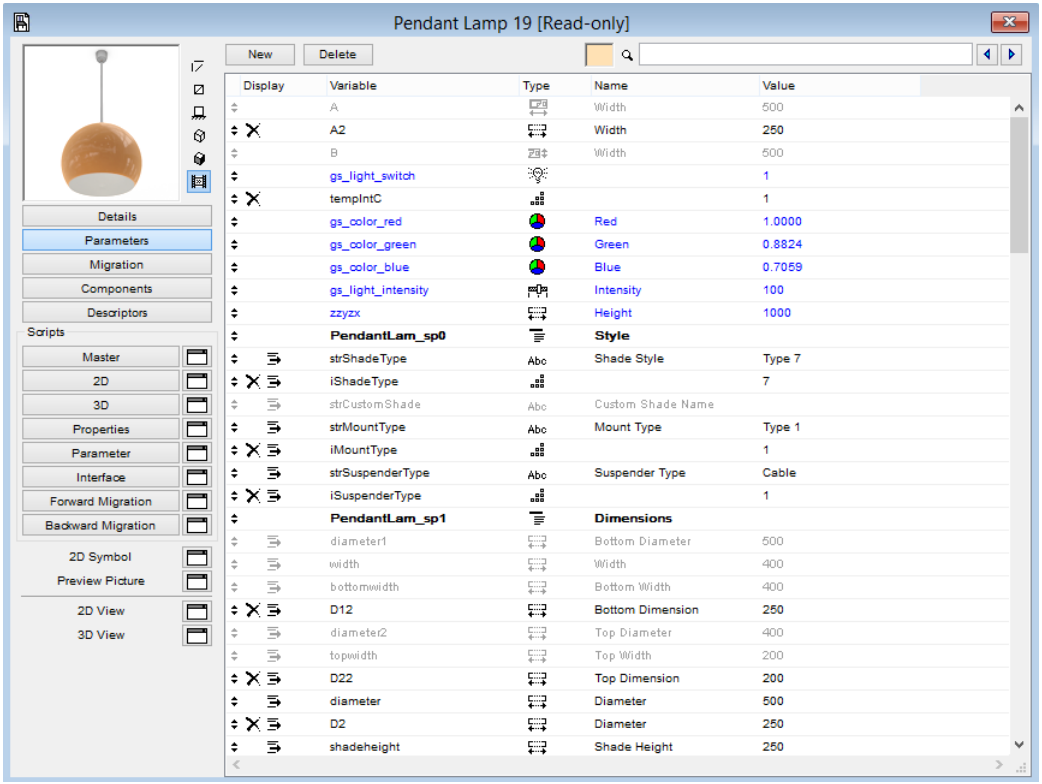
Unterschiedliche Bedeutungen von "additional data" können von der ausführenden Anwendung definiert und verwendet werden.

Beispiel 1:

```
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
```

Beispiel 2:

Bibliothekselement-Dialog für Lichtquellen in ARCHICAD:



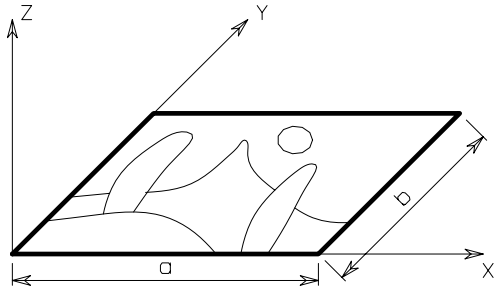
Teil des entsprechenden GDL-Scriptes:

```
if gs_light_switch > 0 then
    LIGHT gs_light_intensity/100*gs_color_red, \
   gs_light_intensity/100*gs_color_green, \
   gs_light_intensity/100*gs_color_blue, ! RGB
            ...
endif
```

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

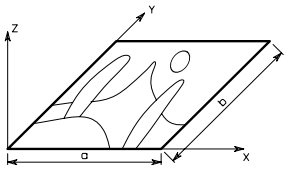
alpha: Steuerung des Alphakanals, wenn in der Bilddatei enthalten

- 0: Kein Verwenden des Alphakanals, das Bild ist ein opakes Rechteck.
- 1: Verwenden des Alphakanals, Teile des Bildes sind transparent.

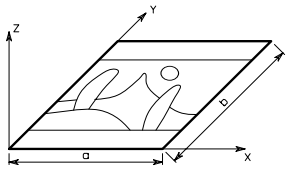
distortion: Kontrolle der Verzerrung

- 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

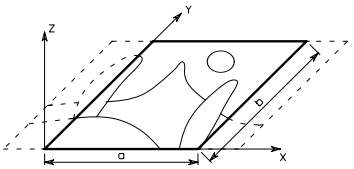
verzerrung=0



verzerrung=2



verzerrung=4



3D-TEXTELEMENTE

TEXT

TEXT d, 0, expression

Eine 3D-Darstellung in dem eingestellten Stil des Wertes eines Text- oder numerischen Ausdruckes (expression).

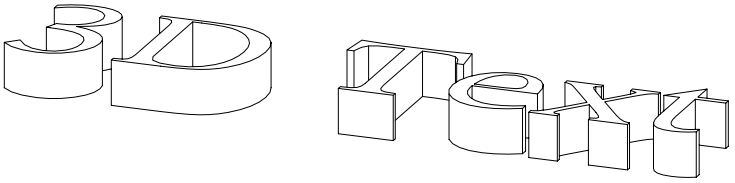
Siehe [SET] STYLE und DEFINE STYLE.

d: Zeichendicke in Meter.

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

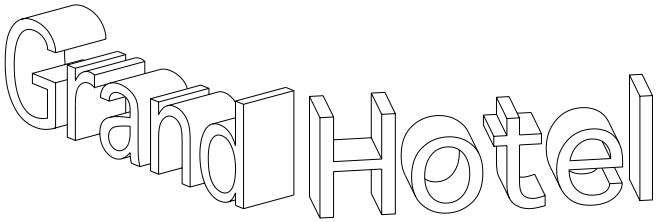
Anmerkung: Aus Kompatibilitätsgründen mit dem 2D GDL Script werden die Zeichenhöhen in den DEFINE STYLE-Anweisungen immer in mm interpretiert.

Beispiel 1:



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

Beispiel 2:




```
name = "Grand"
ROTX 90
ROTY -30
TEXT 0.003, 0, name
ADDX STW (name)/1000
ROTY 60
TEXT 0.003, 0, "Hotel"
```

RICHTEXT

RICHTEXT x, y,
height, 0, textblock_name

Dreidimensionale Darstellung von einem vorher definierten TEXTBLOCK. Für weitere Details, *siehe TEXTBLOCK*.

x, y: X-Y Koordinaten der RICHTEXT-Platzierung

height: Zeichendicke in Meter.

textblock_name: Name eines zuvor 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 (BODY) 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 angegeben 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.

VERT{2}

VERT x, y, z, hard

Erweiterung von VERT ermöglicht es, einen Knoten als harten Scheitelpunkt zu erzeugen. Ein harter Scheitel definiert einen Bruch beim Rendern von weichen Oberflächen.

x, y, z: Koordinaten des Knotens.

hard:

1: Falls der Scheitel einen Bruch beim Rendern von weichen Oberflächen definieren soll

0: andernfalls.

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*).

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 Bedeutungen:

`0:` Seitliche oder alleinstehende Kante,
`< 0:` Mögliche Nachbarpolygone werden automatisch gesucht.

`status:` Status-Werte:
`status = j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 262144*j19`, hierbei kann j jeweils 0 oder 1 sein.
`j1:` Unsichtbare Kante.
`j2:` Kante einer gekrümmten Oberfläche.

Reservierte Status-Bits für zukünftige Anwendungen:

`j3:` Erstes Polygon mit einer gekrümmten Oberfläche (nur im Zusammenhang mit j2=1),
`j4:` Letztes Polygon mit einer gekrümmten Oberfläche (nur im Zusammenhang mit j2=1),
`j5:` Die Kante ist ein Bogen.
`j6:` erstes Segment eines Bogens (nur wirksam, wenn j4=1),
`j7:` letztes Segment eines Bogens (nur wirksam, wenn j4=1),
`j19:` rendert scharfe Kanten zwischen 2 gebogenen Polygonen (nur effektiv wenn j2=1).

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.

`edge1, edge2, ..., edgen:` müssen auf die vorher definierten Kanten mit 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:` Status-Werte:
`status = j1 + 2*j2 + 16*j5 + 32*j6 + 64*j7 + 4*j3 + 8*j4`, hierbei kann j jeweils 0 oder 1 sein.
`j1:` unsichtbares Polygon.
`j2:` Polygon mit einer gekrümmten Oberfläche.
`j5:` Konkaves Polygon.
`j6:` Polygon mit Öffnungen.
`j7:` Die Öffnungen sind konvex (nur im Zusammenhang mit j6=1),

Reservierte Status-Bits für zukünftige Anwendungen:

`j3:` Erstes Polygon mit einer gekrümmten Oberfläche (nur im Zusammenhang mit j2=1),
`j4:` Letztes Polygon mit einer gekrümmten Oberfläche (nur im Zusammenhang mit j2=1).

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

n = 0 ist für spezielle Zwecke gestattet.

PGON{2}

PGON{2} `n, vect, status, wrap, edge_or_wrap1, ..., edge_or_wrapn`

Die ersten drei Parameter sind ähnlich denen beim PGON.

wrap: Hüllmodus + Projektionstyp.
`0:` der globale Hüllmodus wird verwendet,
`> 0:` die Bedeutung ist dieselbe wie bei COOR.

edge_or_wrap1, ..., edge_or_wrapn: Die Anzahl und die Bedeutung dieser Parameter basiert auf der Hülldefinition:
`edge1, ..., edgen:` falls wrap 0 ist; in diesem Fall bedeutet `edgen` das selbe wie PGON, und global definiertes Texturmapping wird angewendet;
`x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4, edge1, ..., edgen:` falls der Hüllmodus nicht 0 ist bei wrap; in diesem Fall definieren die Koordinaten `xi, yi, zi` das Koordinatensystem des Texturmappings für das Polygon;
`edge1, u1, v1, ..., edgen, un, vn:` falls der Hüllmodus 0 ist, aber der Projektionstyp in wrap nicht 0 ist; in diesem Fall sind die Raumkoordinaten der Textur `ui, vi` die gleichen wie beim TEVE; das Mapping beeinflusst nur das zuletzt definierte Polygon.

PGON{3}

PGON{3} `n, vect, status, wrap_method, wrap_flags, edge_or_wrap1, ..., edge_or_wrapn`

Die Parameter sind ähnlich wie bei PGON{2}, ausser`wrap`, welches in 2 Parameter gesplittet wird `wrap_methode` und `wrap_flags`. Die Bedeutung dieser beiden ist die selbe wie bei COOR{2}.

PIPG

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

Definition eines Bild-Polygons Die ersten vier Parameter sind die gleichen wie in PICTURE; die restlichen dieselben wie bei dem PGON.

COOR

COOR `wrap, vert1, vert2, vert3, vert4`

Veraltet. Siehe COOR{3}.

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

wrap: Umhüllungsmodus + Projektionstyp

Wrapping modes:

1: eben (veraltet),
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.
6: eben,

Projection types:

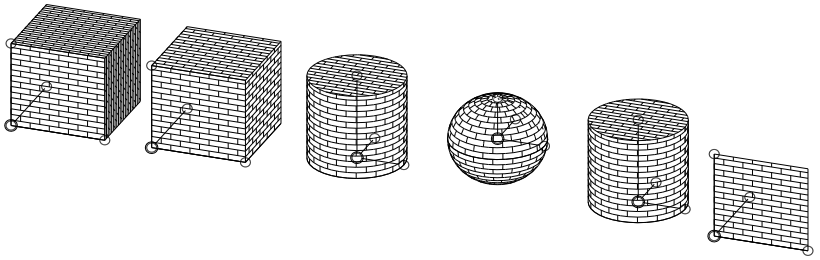
256: die Füllung geht immer vom Ursprung des lokalen Koordinatensystems aus.
1024: quadratische Texturprojektion (empfohlen).
2048: lineare Texturprojektion, die auf dem Durchschnittsabstand basiert.
4096: lineare Texturprojektion, die auf den normalen Dreieckspunkten basiert.

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

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

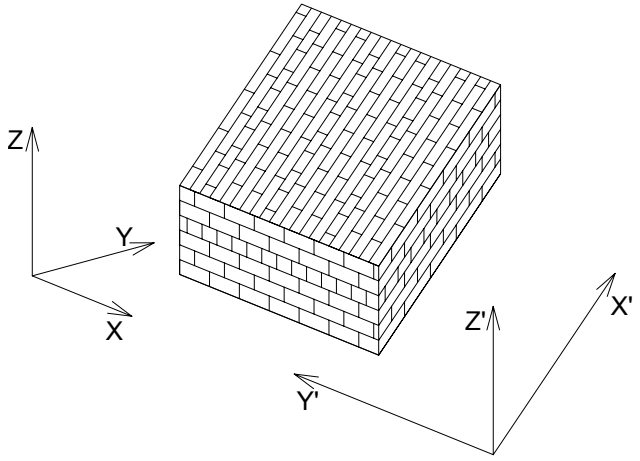
vert2, vert3, vert4: Indizes der die drei Koordinatenachsen definierenden VERTs.

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



Beispiel: für benutzerdefinierte Texturachsen:

```
CSLAB_ "Brick-White", "Brick-White", "Brick-White",
        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
COOR 2, -1, -2, -3, -4
BODY 1
```



COOR{2}

COOR{2} wrap_method, wrap_flags, vert1, vert2, vert3, vert4

Veraltet. Siehe COOR{3}.

Ähnlich wie COOR, wobeiwrap in zwei Parameter gesplittet wird wrap_methode und wrap_flags, und außerdem dessen Möglichkeiten erweitert.

wrap_method: Umhüllungsmoden sind die gleichen wie bei COOR beschrieben. Statt Projektionstyp benutzen Sie `wrap_flags`.

wrap_flags: Umhüllungs-Flags

`wrap_flags = 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8`, hierbei kann j jeweils 0 oder 1 sein.

`j3:` quadratische Texturprojektion (empfohlen).

`j4:` lineare Texturprojektion, die auf dem Durchschnittsabstand basiert.

`j5:` lineare Texturprojektion, die auf den normalen Dreieckspunkten basiert,

`j8:` Überträgt den Ursprung des Textur-Koordinaten-Systems möglichst nahe an den Globalen Ursprung bezüglich der Richtung der X-, Y- und Z-Achse. Zum Beispiel bewirkt, `j6`, dass der Ursprung in die Richtung der X-Achse übertragen wird (entlang`v2 - v1` vector) so dass es die orthogonale Projektion des Globalen Ursprungs der Linie in Bezug auf die X-Achse bildet. Das ist dann der Fall, wenn alle `j6`, `j7` und `j81` sind; dann wird der Ursprung in den Globalen Ursprung übertragen (genau wie wenn der Projektionstyp `256` in demCOOR ist).

Anmerkung: Die Flags `j3`, `j4` und `j5` sind nur dann effektiv, wenn `wrap_method0` ist und nur einer davon kann `1` sein. Die Flags `j6`, `j7` und `j8` sind nur dann effektiv, wenn `wrap_method` nicht `0` ist. Diese können gleichzeitig in jeder beliebigen Kombination `1` sein.

vert1, vert2, vert3, vert4: wie im COOR.

COOR{3}

```
COOR{3} wrapping_method, wrap_flags,
        origin_X, origin_Y, origin_Z,
        endOfX_X, endOfX_Y, endOfX_Z,
        endOfY_X, endOfY_Y, endOfY_Z,
        endOfZ_X, endOfZ_Y, endOfZ_Z
```

Kompatibilität: eingeführt in ARCHICAD 20.

Gleicht dem COOR{2}-Befehl. Kann zusammen mit Array-Parameter-Eingaben verwendet werden (siehe auch WALL_TEXTURE_WRAP global in „Wandparameter, verfügbar für Fenster/Türen, Listen und Etiketten“ für weitere Infos).

Das Koordinatensystem des Projektionskörpers ist enthalten im COOR{3} -Befehl selber, es gibt also keine Notwendigkeit, zusätzliche Scheitelpunkte im bestehenden BODY hinzuzufügen. Kompatibel mit NURBS -Körpern (es werden keine nicht-NURBS Primitive benötigt um das Textur-Koordinatensystem anzulegen).

wrap_method: Umhüllungs-Modi sind die gleichen wie in dem COOR durch den ergänzten NURBS-basierten Umhüllungs-Modus beschrieben. Statt Projektionstyp benutzen Sie `wrap_flags`.

`1:` eben (veraltet),

`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.
`6:` eben,
`7:` NURBS-basiert werden die Texturkoordinaten der Knotenpunkte aus ihren Oberflächenparametern ermittelt, nur im Fall von NURBS-bodies.

`wrap_flags:` Umhüllungs-Flags
`wrap_flags = 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8`, hierbei kann j jeweils 0 oder 1 sein.
`j3:` quadratische Texturprojektion (empfohlen).
`j4:` lineare Texturprojektion, die auf dem Durchschnittsabstand basiert.
`j5:` lineare Texturprojektion, die auf den normalen Dreieckspunkten basiert,
`j8:` Überträgt den Ursprung des Textur-Koordinaten-Systems möglichst nahe an den Globalen Ursprung bezüglich der Richtung der X-, Y- und Z-Achse. Zum Beispiel führt `j6` dazu, dass der Ursprung in Richtung der X-Achse verschoben wird, so dass er die orthogonale Projektion des globalen Ursprungs bezogen auf die Linie der X-Achse ist. Das bedeutet, wenn alle `j6`, `j7` und `j81` sind, wird der Ursprung auf den globalen Ursprung verschoben (gegenteiliger Effekt, wenn der Projektionstyp `256` im COOR ist).

Anmerkung: Die Flags `j3`, `j4` und `j5` sind nur dann effektiv, wenn `wrap_method0` ist und nur einer davon kann `1` sein. Die Flags `j6`, `j7` und `j8` sind nur dann effektiv, wenn `wrap_method` nicht `0` ist. Diese können gleichzeitig in jeder beliebigen Kombination `1` sein.

`origin_X, origin_Y, origin_Z:` Knoten im x-y-z-Raum, definiert durch drei Koordinaten, Texturursprung.

`endOfX_X, endOfX_Y, endOfX_Z:` Knoten im x-y-z-Raum, definiert durch drei Koordinaten, X-Richtung des Texturmappings.

`endOfY_X, endOfY_Y, endOfY_Z:` Knoten im x-y-z-Raum, definiert durch drei Koordinaten, Y-Richtung des Texturmappings.

`endOfZ_X, endOfZ_Y, endOfZ_Z:` Knoten im x-y-z-Raum, definiert durch drei Koordinaten, Z-Richtung des Texturmappings.

Beispiel: COOR{3} und äquivalente COOR{2} Parametrisierung

```
COOR{3} wrapping_method, wrap_flags,
        origin_X, origin_Y, origin_Z,
        endOfX_X, endOfX_Y, endOfX_Z,
        endOfY_X, endOfY_Y, endOfY_Z,
        endOfZ_X, endOfZ_Y, endOfZ_Z

! COOR{2} equivalent
BASE
VERT    origin_X, origin_Y, origin_Z,
VERT    endOfX_X, endOfX_Y, endOfX_Z
VERT    endOfY_X, endOfY_Y, endOfY_Z
VERT    endOfZ_X, endOfZ_Y, endOfZ_Z
COOR{2} wrapping_method, wrap_flags, -1, -2, -3, -4
```

BODY

BODY status

Erzeugt einen Körper unter Verwendung der vorher genannten Grundelemente.

status: Status-Werte:

status = j_1 + 2*j_2 + 4*j_3 + 32*j_6 + 64*j_7, hierbei kann j jeweils 0 oder 1 sein.

j_1: geschlossener Körper (veraltet),

j_2: Der Körper besitzt (eine) gekrümmte Oberfläche(n). (veraltet),

j_3: Flächenmodell: der Körper verhält sich wie ein Hohlkörper. Bei einem Schnitt durch den Körper erscheint keine Schnittfläche.

j_6: Der Körper erzeugt immer einen Schattenwurf, unabhängig von der automatischen Voreinstellung für Schattenwurf.

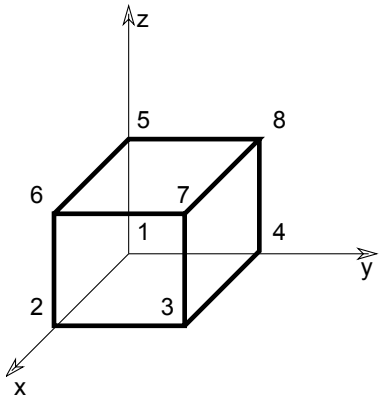
j_7: Körper erzeugt niemals Schattenwurf.

Ist weder j6 noch j7 angegeben, wird die automatische Voreinstellung für Schattenwurf verwendet.

Siehe SHADOW.

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.

NURBS Primitiv-Elemente

Die Primitivelemente der 3D-Datenstruktur der NURBS-Körper sind: NURBSCURVE2D, NURBSCURVE3D, NURBSSURFACE, NURBSVERT, NURBSEDGE, NURBSTRIM, NURBSTRIMSINGULAR, NURBSFACE, NURBSLUMP, und NURBSBODY.

Solide NURBS-Körper werden durch die Grenz-NURBS-Oberflächen der soliden Bereiche dargestellt, laminare Oberflächen-NURBS-Körper werden durch die NURBS-Oberflächen selber dargestellt, Drahtgitter-NURBS-Körper werden durch die NURBS-Kanten dargestellt. Ein NURBS-Körper kann gleichzeitig solide, laminare und Drahtgitter-Bestandteile besitzen, ein NURBS-Körper selber ist nicht nach den Kategorien Solide/Oberfläche/Drahtgitter klassifiziert.

Nurbs-Primitive können nicht bei planaren Oberflächen-Körpern verwendet werden und Nicht-NURBS-Primitive können nicht für NURBS-Körper verwendet werden. Eine Nicht-NURBS-Primitiv Anweisung veranlasst den "im Bau befindlichen" NURBS-Körper sich zu vervollständigen, ein neuer Nicht-NURBS-Körper kann dann begonnen werden (Voraussetzung: BODY und NURBSBODY Ausdrücke).

Auf ähnliche Weise bewirkt eine NURBS-Primitiv-Anweisung bei einem "im Bau befindlichen" Nicht-NURBS-Körper eine Vervollständigung und den Beginn eines neuen NURBS-Körpers. Eine zusammengesetzte Anweisung (BRICK, CYLIND, PRISM, etc.) oder eine MODEL-Anweisung bewirken, dass entweder NURBS oder "im Bau befindliche" Nicht-NURBS Körper fertiggestellt werden. Falls eine NURBSBODY-Anweisung einen Nicht-NURBS-Körper oder eine BODY-Anweisung einen NURBS-Körper schließt, haben die verwendeten Statuswerte keine Wirkung.

Die Indizierung von NURBS-Primitiven beginnt mit 1. Die Indizierung von NURBS-Primitiven und Nicht-NURBS-Primitiven (VERT, TEVE, EDGE, VECT, PGON, PIPG) werden separat behandelt. Die BASE-Anweisung setzt den Zähler auch für NURBS-Körper-Primitive zurück. Alle Primitive, welche von anderen Primitiven referenziert werden, sollten vor dem referenzierendem definiert werden (z.B. Scheitelpunkte (VERT) und 3D-Kurven einer Kante sollten vor der Kante definiert werden).

Die Anweisungen NURBSCURVE2D, NURBSCURVE3D und NURBSSURFACE erzeugen nur geometrische Elemente im NURBS-Körper, welche selber nicht sichtbar sind. Eine NURBS-Kante definiert ihren geometrischen Bezug, indem sie auf eine 3D-NURBS-Kurve referenziert, ähnlich referenziert ein NURBS-Beschneidungselement auf eine 2D-NURBS-Kurve und eine NURBS-Hüllfläche referenziert auf eine NURBS-Oberfläche als deren geometrische Grundstruktur (die Kante, die Beschneidung und die Hüllfläche können nicht über die gesamte geometrische Grundstruktur hinausragen, siehe Details hierzu in jeder Befehls-Beschreibung).

Die NURBS-Kante, ihre 3D-Kurve, ihre Beschneidungen und die 2D-Kurven ihrer Beschneidungen sind immer einheitlich ausgerichtet. Die Außenfläche der NURBS und ihre Oberflächen sind immer einheitlich ausgerichtet.

Die Außenflächen der NURBS kann man in NURBS-Klumpen gliedern. Ein Klumpen definiert einen soliden Bereich, begrenzt von einer oder mehrerer Schalen. Eine Schale ist eine geschlossene und miteinander verbundene Anordnung von Außenflächen, welche den Raum in zwei Bereiche teilt. Ein Klumpen besitzt eine äußere Schale, welche den Klumpen vom Unendlichen trennt, und besitzt unter Umständen leere Schalen, welche den Klumpen von inneren Hohlräumen trennt.

Eine konsistente Ausrichtung der Flächen in einer Schale ist nicht notwendig, zwei benachbarte Flächen können sich auf die gleiche Kante in der gleichen Richtung beziehen. Jedoch müssen Schalen von Klumpen konsistente Ausrichtungen besitzen, die Rückseite einer Schale sollte in Richtung auf das Innere des Klumpen ausgerichtet sein, hierzu kann der Klumpen auf die Flächen mit negativem Vorzeichen für eine umgekehrte Orientierung verweisen.

Außenflächen, welche nicht Teil eines Klumpen sind, werden als laminare Oberflächen behandelt, selbst dann, wenn diese Flächen eine geschlossene Schale bilden. Kanten, welche nicht Teil einer Außenfläche bilden, werden als Drahtgitterkanten behandelt. Ein NURBS-Körper kann gleichzeitig solide Klumpen enthalten, sowie laminare Oberflächen und Drahtkanten.

Die 2-Vielfach-Eigenschaft ist nicht für NURBS Körper erforderlich, es kann aber eine NURBS-Kante mit mehr als zwei Flächen verbunden werden (um mehr als zwei Anschnitte). Selbst eine Schale eines NURBS-Klumpen kann an einer Kante mehr als zwei Flächen besitzen, solange die Schale den Raum immer noch in zwei Bereiche trennt (das bedeutet eine gerade Anzahl von Flächen einer gegebenen Schale an jeder Kante).

Die Anweisungen RADIUS, RESOL und TOLER haben keinen Einfluss auf den Glättungsgrad von NURBS-Oberflächen und -Kanten. Der Glättungsgrad von NURBS-Primitiven wird automatisch berechnet und kann durch die Parameter des NURBSBODY-Befehls für die NURBS-Körper eingeschränkt sein (näheres hierzu unter NURBSBODY).

Details für korrekte Textureinstellungen bei NURBS finden Sie hier COOR{3}.

NURBS-Außenflächen-Anschnitte

Eine NURBS-Außenfläche ist eine zweidimensionale Schicht im dreidimensionalen Raum und wird definiert durch eine geometrische Funktion, welche ein Rechteck in den Raum mappt. Die Geometrie einer NURBS-Außenfläche ist immer Teil einer NURBS-Oberfläche, kann aber komplexer als diese sein. Dies wird durch Anschnitte ermöglicht.

Ein Anschnitt definiert einen Schnitt auf dem dominierenden Rechteck der Oberfläche, und zwar einen Schnitt mit einer 2-dimensionalen NURBS-Kurve. Dies bedeutet einen Schnitt auf der dreidimensionalen Schicht der Oberfläche. Dieser Schnitt befindet sich entlang des NURBS-Begrenzungs-Randes der Außenfläche; die Geometrie des Schnitts entlang der Oberflächenschicht muss mit der Geometrie der NURBS-Kante konsistent sein.

Eine NURBS-Außenfläche besitzt Konturen wie ein traditionelles PGON, aber die Konturen sind keine Listen von NURBS-Kanten, sondern von NURBS-Anschnitten, weil die Anschnitte die Informationen besitzen, um die Außenflächen korrekt zu beschneiden. (Die 2D-Kurven der Anschnitte können aus der 3D-Kurve des Randes berechnet werden, aber es kann im Falle von Oberflächen mit Selbstüberschneidungen oder Singularitäten oder bei fehlerhaften Daten ungenau oder sogar mehrdeutig sein.)

NURBS Geometrie Befehle

Die folgenden Befehle beschreiben die geometrischen Teile eines NURBS-Elements: Kurven und Oberflächen.

NURBSCURVE2D

```
NURBSCURVE2D degree, nControlPoints,
        knot_1, knot_2, ..., knot_m,
        cPoint_1_x, cPoint_1_y, weight_1,
        cPoint_2_x, cPoint_2_y, weight_2,
        ...,
        cPoint_n_x, cPoint_n_y, weight_n
```

NURBSCURVE3D

```
NURBSCURVE3D degree, nControlPoints,
        knot_1, knot_2, ..., knot_m,
        cPoint_1_x, cPoint_1_y, cPoint_1_z, weight_1,
        cPoint_2_x, cPoint_2_y, cPoint_2_z, weight_2,
        ...,
        cPoint_n_x, cPoint_n_y, cPoint_n_z, weight_n
```

2 und 3 dimensionale NURBS-Kurven mit Angabe von Rangstufe, Knotenvektor, Kontrollpunkten und Gewichten.

degree: Rangstufe einer NURBS-Kurve, eins weniger als die Reihenfolge der Kurve (reihenfolge = rangstufe + 1), positiv

nControlPoints: Anzahl der Kontrollpunkte (n), größer als die Rangstufe der Kurve (nicht kleiner als die Reihenfolge)

knot_i: index i Knotenwert

- Anzahl der Knotenwerte (m, Größe des Knotenvektors) wird wie folgt bestimmt: m = grad + 1 + n
- Knoten sind in einer nicht absteigenden Reihenfolge (knoten_i <= knoten_{i+1})
- gleiche Knotenwerte sind erlaubt, mit der Häufigkeit bis zur Rangstufe, oder mit der Häufigkeit bis zu Rangstufe + 1 beim ersten oder letzten Knoten.

cPoint_i_x, cPoint_i_y, cPoint_i_z: Koordinaten des index i Kontrollpunkts

weight_i: Gewicht des index i Kontrollpunkts, positiv

Periodische Kurven werden nicht separat behandelt, sondern als fließende (nicht starre) NURBS-Kurven beschrieben, welche geometrisch geschlossen sind und an ihren Enden geeignete durchgängige Verbindungen besitzen. Dies wird durch das Wiederholen einer ausreichenden Anzahl von Kontrollpunkten und Knotenintervallen am Ende sichergestellt:

- die Vielfach-Kontrollpunkte der letzten Rangstufe sind Duplikate der Vielfach-Kontrollpunkte der ersten Rangstufe (nicht in umgekehrter Reihenfolge),
- die erste doppelte Rangstufe von Knotendifferenzen (knoten_1-knoten_0, knoten_2-knoten_1, ...) sind die gleichen wie bei den letzten der Knotenvektoren (dieses sind die Knoten, welche sich mit denen der ersten (oder letzten) Rangstufen-Vielfach-Kontrollpunkten in Verbindung befinden).

Der nutzbare Bereich einer Kurve ist das geschlossene Intervall zwischen knoten_ {rangstufe + 1} und knoten_ {m - rangstufe}.

NURBSSURFACE

```
NURBSSURFACE degree_u, degree_v, nu, nv,
       knoten_u_1, knoten_u_2, ..., knoten_u_mu,
       knoten_v_1, knoten_v_2, ..., knoten_v_mv,
       cPoint_1_1_x, cPoint_1_1_y, cPoint_1_1_z, weight_1_1,
       cPoint_1_2_x, cPoint_1_2_y, cPoint_1_2_z, weight_1_2,
       ...,
       cPoint_1_nv_x, cPoint_1_nv_y, cPoint_1_nv_z, weight_1_nv,
       cPoint_2_1_x, cPoint_2_1_y, cPoint_2_1_z, weight_2_1,
       ...,
       cPoint_nu_nv_x, cPoint_nu_nv_y, cPoint_nu_nv_z, weight_nu_nv
```

3-dimensionale NURBS-Oberflächen mit u-v Parameterraum, vorgegebener Rangstufe, Knotenvektoren in u und v Richtung und vorgegebenen Kontrollpunkten, Nettogewicht. Rangstufen sind 1 Wert niedriger als die Oberflächenordnung (ordnung_u = rangstufe_u + 1), Rangstufen sind positiv.

degree_u: Rangstufe der Oberfläche in der Richtung des u Parameters

degree_v: Rangstufe der Oberfläche in der Richtung des v Parameters

nu, nv: Anzahl der Kontrollpunkte in u und v Richtung, größer als Rangstufe (nicht kleiner als die Ordnung) der Oberfläche in die angegebene Richtung

knot_u_i, knot_v_i: Index des i Knotenwertes in u und v Richtung

- ihre Anzahl (die Größe des Knotenvektors) wird wie folgt angegeben: mu = degree_u + 1 + nu
- Knoten zählen in nicht absteigender Reihenfolge (knoten_u_i <= knoten_u_{i+1}, knoten_v_i <= knoten_v_{i+1})
- gleiche Knotenwerte sind erlaubt, mit der Häufigkeit bis zur Rangstufe, oder mit der Häufigkeit bis zu Rangstufe + 1 beim ersten oder letzten Knoten.

cPoint_i_j_x, cPoint_i_j_y, cPoint_i_j_z: Kontrollpunkte auf dem Kontrollpunktnetz, Index i in der u Richtung, Index j in der v Richtung

weight_i_j: Gewicht des Kontrollpunktes cPoint_ij, positiv

Oberflächen können periodisch auftreten entweder in eine Richtung (u oder v) oder in beide Richtungen. Periodische Oberflächen werden nicht separat behandelt, sondern werden als fließende (nicht starre) NURBS Oberflächen beschrieben, welche geometrisch geschlossen sind und an ihren Enden geeignete durchgängige Verbindungen besitzen. Dies wird auf die selbe Weise sichergestellt wie im Fall der Kurven.

Der nutzbare Bereich einer Oberfläche ist jeweils das Kreuzprodukt der geschlossene Intervalle zwischen knoten_u_ {rangstufe_u + 1}, knoten_u_ {mu - rangstufe_u} und knoten_v_ {rangstufe_v + 1}, knoten_v {mv - rangstufe_v}.

NURBS Mathematische Befehle

Die folgenden Befehle beschreiben mathematische Teile von NURBS-Elementen.

NURBSVERT

NURBSVERT x, y, z, hard, tolerance

Vertex, ein Knoten eines NURBS-Körpers. Unterscheidet sich von jedem anderen Knoten erzeugt von VERT, und separat von diesen indiziert. Nur verwendbar in NURBS-Körpern, mit Ausnahme von planar-flächigen Körpern.

x, y, z: Koordinaten des Vertex

hard:

1: Falls der Vertex einen Absatz (Kante) beim Rendern von glatten Oberflächen erzeugen soll,

0: andernfalls.

tolerance: maximale geometrische Distanz zwischen NURBS-Vertex und anderen Elementen (NURBS Kante, NURBS Außenfläche) welche mit diesem mathematisch verbunden sind. Falls negativ, ist die Toleranz ein vordefinierter Standardwert.

NURBSEDGE

NURBSEDGE vert1, vert2, curve, curveDomainBeg, curveDomainEnd, status, tolerance

Kante eines NURBS-Körpers. Unterscheidet sich von jeder anderen Kante, die definiert wurde von EDGE, und wird unabhängig von diesen indiziert. Nur verwendbar in NURBS-Körpern, mit Ausnahme von planar-flächigen Körpern.

vert1, vert2: gdl-index des Beginns und Endes von NURBS Scheitelpunkte

- vert1 und vert2 können gleich sein. In diesem Fall ist die Kante eine Schleifen-Kante (und ihre Kurve ist geschlossen oder besitzt einen geschlossenen Teil)
- vert1 und vert2 können Null sein bei einer ringförmigen Kante (welche keine Scheitelpunkte besitzt und ihre Kurve ist geschlossen oder besitzt einen geschlossenen Teil)

curve: gdl-index einer NURBS-Kurve für die Geometrie einer Kante. Positiver Index, Ausrichtung der Kante stimmt immer mit der Ausrichtung der Kurve überein.

curveDomainBeg, curveDomainEnd: Definition des Teils der Kurve, welche geometrisch gesehen die Kante darstellt. Der Wert curveDomainEnd muss größer sein als curveDomainBeg, sie dürfen nicht gleich sein, und beide Werte müssen innerhalb des nutzbaren Bereichs der Kurve liegen.

status: Statuskontrolle der Kante:

status = j_1 + 2*j_2 + 4*j_3, hierbei kann j jeweils 0 oder 1 sein.

j_1: unsichtbare Kante (kann nur eingestellt werden, wenn j2 nicht gesetzt ist).

`j2`: Kante nur sichtbar, wenn Kontur (kann nur eingestellt werden, wenn j1 nicht gesetzt ist).

`j3`: geglättete Kante (die Kante erzeugt keinen Bruch beim Rendern glatter Oberflächen).

Falls sowohl j1 als auch j2 gesetzt sind, erzeugt die Kante einen Fehler, welcher dazu führt, dass der gesamte NURBS-Körper verschwindet.

tolerance: maximaler geometrischer Abstand zwischen NURBS-Kante und anderen Elementen (NURBS Sichtfläche), welche topologisch (mathematisch) damit verbunden sind. Falls negativ, ist die Toleranz ein vordefinierter Standardwert.

Die an jedem Endpunkt ausgewertete Kurve sollte mit der Position des entsprechenden Scheitelpunktes übereinstimmen. Die Kante kann eine ringförmige Kante ohne Scheitelpunkte sein. In diesem Fall muss die auf [curveDomainBeg, curveDomainEnd] eingeschränkte Kante geschlossen sein, d.h. es erfolgt an jedem Endpunkt eine gleichartige Auswertung. Jede Anzahl von Kanten kann einem Scheitelpunkt zugewiesen werden. Die Farbe einer NURBS-Kante wird durch die letzte `PEN` -Anweisung definiert.

NURBSTRIM

NURBSTRIM `edge, curve, curveDomainBeg, curveDomainEnd, tolerance`

NURBSTRIMSINGULAR

NURBSTRIMSINGULAR `vertex, curve, curveDomainBeg, curveDomainEnd, tolerance`

Eine Begrenzungskante einer Sichtfläche. Verwendet zum Trimmen einer Sichtfläche im Parameterraum der sichtbaren Oberfläche. NURBSTRIMSINGULAR wird entlang der singulären Kanten der Oberfläche verwendet (deren Seite zu einem Punkt auf der Oberfläche kontrahiert). Verbindet die Sichtfläche mit einer Kante (oder mit einem Scheitelpunkt im singulären Fall).

edge: gdl-index einer NURBS-Kante, mit der diese Beschneidung verknüpft ist. Positiver Index, Kante und Beschneidung werden immer konsistent ausgerichtet.

vertex: gdl-index eines NURBS-Scheitelpunkts, mit der diese Beschneidung verknüpft ist (singulärer Fall).

curve: gdl-index einer 2D NURBS-Kurve. Positiver Index, Kante und Beschneidung werden immer konsistent ausgerichtet. Dies wird für den Bereich (u-v Parameter-Raum) der sichtbaren Oberfläche definiert.

curveDomainBeg, curveDomainEnd: Definition des Teils der Kurve, welche geometrisch gesehen die Beschneidung darstellt. Der Wert curveDomainEnd muss größer sein als curveDomainBeg, sie dürfen nicht gleich sein, und beide Werte müssen innerhalb des nutzbaren Bereichs der Kurve liegen.

tolerance: maximaler geometrischer Abstand zwischen 2D-Kurve der NURBS-Beschneidung und anderen Elementen (andere NURBS-Beschneidungen) welche topologisch damit verknüpft sind. Falls negativ, ist die Toleranz ein vordefinierter Standardwert.

Die auf das Intervall [curveDomainBeg, curveDomainEnd] beschränkte Kurve sollte komplett innerhalb des nutzbaren Bereichs der sichtbaren Oberfläche liegen (mit vorgegebener Toleranz). Bei NURBSTRIMSINGULAR muss die 2D-Kurve entlang einer singulären Kante des nutzbaren Bereiches (u-v Parameter-Raum) der sichtbaren Oberfläche liegen .

Die Zusammensetzung der beschränkten 2D-Kurve und der Oberfläche ergibt eine 3D-Kurve, die mit der eingeschränkten 3D-Kurve des Randes zusammenfallen sollte. Deshalb sollte die bei curveDomainBeg und curveDomainEnd ausgewertete 2D-Kurve mit der Position des entsprechenden Scheitelpunktes übereinstimmen. Im singulären Fall ergeben die Gestaltung der 2D-Kurve und der Oberfläche einen 3D-Punkt, welcher mit dem entsprechenden Scheitelpunkt übereinstimmen sollte.

Das Indizieren von singulären und nicht-singulären Beschneidungen ist üblich.

Jede Anzahl von Beschneidungen kann sich auf jede Kante beziehen (d.h. indirekt kann jede Anzahl von Sichtflächen mit jeder Kante verknüpft werden). Die Kante kann Nicht-2-mehrfachgefaltet sein.

Zwei Beschneidungen auf einer Kante können zur selben Sichtfläche gehören, in welchem Fall man die Kante als Nahtkante bezeichnet. Zum Beispiel kann ein Zylindermantel eine Sichtfläche mit einer Nahtkante sein.

NURBSFACE

```
NURBSFACE n, surface, tolerance,
        trim1, trim2, ..., trimn
```

Sichtfläche eines NURBS-Körpers. Unterscheidet sich von jedem anderen durch PGON erzeugten Polygon, von diesen separat indiziert. Nur verwendbar in NURBS-Körpern, mit Ausnahme von planar-flächigen Körpern.

n: Anzahl der Begrenzungskanten (einschließlich optionaler Lochschneider Nullen).

surface: gdl-index einer NURBS-Oberfläche, welche die Sichtfläche unterstützt. Positiver Index, Ausrichtung der Sichtfläche ist immer mit der Ausrichtung der Oberfläche identisch.

trimi: gdl-index von NURBS-Beschneidungen, welche die Sichtflächen begrenzen.

- Die Beschneidungen werden bei der äußeren Kontur-Schleife ordnungsmäßig gegen den Uhrzeigersinn gelistet (mathematisch positiv) und im Uhrzeigersinn (negativ) bei Lochkontur-Schleifen.
- Kann 0 sein, was das Ende einer Kontur anzeigt (Loch-Separator).
- Negative Indizes bedeuten, dass Beschneidung und Kontur (der Sichtfläche) entgegengesetzte Orientierungen besitzen.

tolerance: falls negativ, ist die Toleranz ein vorgegebener Standardwert.

Die Beschneidungen müssen mit gewöhnlichen Scheitelpunkten verknüpft sein: der Endscheitelpunkt einer Beschneidung ist der selbe wie der Anfangsscheitelpunkt der nächsten Beschneidung der Sichtfläche. (Die Scheitelpunkte einer Beschneidung sind die Scheitelpunkte der Kante einer Beschneidung bei einer nicht-singulären Beschneidung)

Die aufeinander folgenden Beschneidungen - als 2D-Kurven - sind auch mit dem Bereich der Sichtfläche (Parameterraum) verbunden, und definieren darauf eine oder mehrere geschlossene Kontur-Schleifen. Die erste Schleife ist immer die äußere Schleife, welche eine unendliche äußere Region von einer endlichen inneren Region abtrennt. Die möglichen nachfolgenden Schleifen sind Lochkonturen.

Die 2D-Kurve jeder Beschneidung sollte vollständig innerhalb des nutzbaren Bereichs der Sicht-Oberfläche liegen und sollte nicht mit sich selbst oder Kurven anderer Beschneidungen der Fläche überschneiden. Jede Beschneidung darf nur auf einer Sichtfläche angewendet werden.

Die Attribute von Material und Schnitt einer Sichtfläche werden von den letzten MATERIAL und `SECT_ATTRS` (oder SECT_FILL) Anweisungen bestimmt. Die Farbe der Kanten innerhalb der Sichtfläche, welche für polygonale Segmentierung erzeugt werden, wird durch die letzte `PEN` Anweisung definiert.. Dies ist in der Praxis bei Silhouetten sichtbar, welche aus dem inneren dieser Sichtfläche kommen.

NURBSFACE{2}

```
NURBSFACE{2} n, surface, tolerance,
        wrap_method, wrap_flags,
        x1, y1, z1,
        x2, y2, z2,
        x3, y3, z3,
        x4, y4, z4,
         trim1, trim2, ..., trimn
```

Ähnlich wieNURBSFACE, erweitert um die Eigenschaft, Textur-Mapping auf der NURBS-Außenseite zu beschreiben, wie in PGON{3}.

`n, surface, tolerance:` gleich wie NURBSFACE.

`wrap_method:` gleich wie PGON{3}.

- 0: Der globale Hüllmodus wird angewendet (Parameter x1 ... z4 sind erforderlich, werden aber ignoriert)
- > 0: gleich wie PGON{3}

`wrap_flags:` ähnlich wie PGON{3}, mit Ausnahme, dass Projektionstyp-Flags (j3, j4 und j5) ignoriert werden (Texturkoordinaten können nicht auf NURBS-Außenflächen angewendet werden).

`x1, y1, z1 ... x4, y4, z4:` Koordinaten, die das Koordinatensystem der Texturabbildung für die NURBS-Außenfläche definieren (diese Parameter sind nur wirksam, wenn wrap_method> 0).

`trim1 ... trimn:` gleich wieNURBSFACE.

NURBSLUMP

```
NURBSLUMP n, face1, face2, ..., facen
```

Definiert einen soliden Teil ("Klumpen") - eine geometrisch verbundene Teilmenge - eines soliden NURBS-Körpers.

`n:` Anzahl der Verbindungsflächen (einschließlich optionaler Hohlraum-Unterteiler-Nullen).

`facei:` gdl-index der NURBS-Sichtffläche, welche den Klumpen einbindet

- Darf Null sein, was das Ende einer Schale und den Anfang einer anderen Schale anzeigt (Hohlraum-Unterteiler).

- Ein negativer Index bedeutet, dass die Sichtfläche in entgegengesetzter Richtung verwendet wird. Bei positivem Index entspricht die Rückseite der Sichtfläche dem Inneren des Klumpen, bei negativem Index ist die Frontseite nach innen gerichtet.

Die Begrenzung eines Klumpen kann auf mehrere geschlossene Schalen fallen: eine äußere Schale, welche den Klumpen von dem unendlichen äußeren Bereich des Raumes abteilt; und Null oder mehr innere - hohle- Schalen, welche den Klumpen von Hohlraumbereichen trennen Die Sichtflächen einer Schale müssen einen fortlaufenden Teil einer Sichtflächen-Liste bilden. Diese unterschiedlichen Teile für unterschiedliche Schalen müssen durch einen Wert 0 getrennt sein. Die erste Schale muss die Außenschale sein. Die Sichtflächen einer Schale müssen an den gemeinsamen Kanten verbunden sein, aber in der Liste spielt die Reihenfolge keine Rolle.

Beachten Sie, dass die Sichtflächen einer Schale mit anderen Sichtflächen verbunden sein können, welche sich nicht in der Schale befinden oder sich in einer anderen Schale befinden (weil Kanten mehr als 2 Sichtflächen haben können.) Jede Sichtfläche darf nur in einem einzigen Klumpen verwendet werden. Die Schale eines Klumpen kann nicht offen sein - offene Körper besitzen weder Klumpen noch Schalen.

NURBSBODY

NURBSBODY `shadowStatus, smoothnessMin, smoothnessMax`

Komponiert einen NURBS-Körper, definiert mit den oben genannten NURBS-Primitiven.

shadowStatus: Status für Schattenkontrolle:

`shadowStatus = 32*j6 + 64*j7`, hierbei kann j jeweils 0 oder 1 sein.

`j6`: NURBS-Körper werfen immer Schatten, unabhängig vom automatischen Vorauswahl-Algorithmus,

`j7`: NURBS-Körper werfen niemals Schatten.

Ist weder j6 noch j7 angegeben, wird die automatische Voreinstellung für Schattenwurf verwendet. Sieh auch SHADOW-Befehl.

smoothnessMin, smoothnessMax: Grenzen von automatisch berechneten Glättungsparametern für den Flächenabschluss der Oberflächen und Kurven des Körpers. Der automatisch berechnete Parameter liegt immer innerhalb des Bereiches von 0 bis 1, so dass smoothnessMin <= 0 keine untere Begrenzung bedeutet und smoothnessMax >= 1 keine obere Begrenzung. Falls smoothnessMin > smoothnessMax ist, werden die Werte die automatisch berechnete Glättung nicht beeinflussen.

Jeder nicht-NURBS Primitiven-Ausdruck (VERT, TEVE, EDGE, VECT, PGON, PIPG, BODY) oder jede zusammengesetzte Anweisung (BRICK, CYLIND, PRISM, REVOLVE, etc.) bewirken, dass ein im Aufbau befindlicher NURBS-Körper fertiggestellt wird (implizite NURBSBODY-Anweisung). In diesem Fall wird keine Begrenzung des Glättungsgrades vorgegeben und shadowStatus ist dann Null (Status-Parameter des BODY-Ausdrucks wird nicht weitergegeben).

PUNKTWOLKEN

POINTCLOUD

POINTCLOUD `"data_file_name"`

Erzeugt eine Punktwolke im 3D-Modell. Eine Punktwolke ist eine Reihe von Punkten, deren Farbe und andere möglichen Metadaten in jedem Punkt gespeichert werden.

data_file_name: der Name des geladenen Bibliothekselementes, welches die Daten der Punktwolke enthält. Muss ein String-Ausdruck sein.

Punktwolken werden nicht mit der internen 3D-Engine angezeigt. Die 2D-Darstellung ist eine Projektion unter Verwendung von Schnittbefehlen, welche unnötige Punkte herausfiltern.

VERSCHNEIDEN IM 3D-RAUM

CUTPLANE

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

CUTPLANE{2}

```
CUTPLANE{2} angle [, status]
[statement1 ... statementn]
CUTEND
```

CUTPLANE{3}

```
CUTPLANE{3} [x [, y [, z [, side [, status]]]]]
[statement1 ... statementn]
CUTEND
```

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

Falls CUTPLANE die folgenden Anzahl von Parametern hat:

0: Schnittfläche ist die 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: Definition der Seite, welche zu beschneiden ist

0: entfernt Teile oberhalb der Schneidefläche (standard)

`1:` entfernt Teile unterhalb der Schneidefläche; im Fall von x-y, x-z, y-z, werden die Teile in negativer Richtung der Achse entfernt.

`status:` Statuskontrolle der beschnittenen Oberflächen Falls der Statuswert nicht angegeben wird, wird er automatisch auf 1+2 gesetzt.
`status = j1 + 2*j2 + 4*j3 + 256*j9`, hierbei kann j jeweils 0 oder 1 sein.
`j1:` verwendet die eigenen Attribute des Körpers der erzeugten Polygone und Kanten.
`j2:` erzeugte Schnittflächenpolygone werden wie normale Polygone behandelt.
`j3:` erzeugte Schnittkanten sind unsichtbar.
`j9:` Eckpunkte der Schnittfläche werden behandelt als wären sie entfernt worden.

Der Schnitt (ohne den Parameter side) 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 enthalten sein. 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. Deshalb wird empfohlen, alle notwendigen Transformationen zu definieren, um das CUTPLANE aufzubauen, danach löschen Sie all diese Transformationen bevor Sie die zu schneidenden Körper definieren.

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 platziert wurde, während wahrscheinlich die Körper komplett verschwinden.

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.

Anmerkung 1: Falls CUTPLANE nicht mit CUTEND geschlossen wird, werden möglicherweise alle Körper komplett entfernt. Deswegen bekommen Sie stets eine Warnung über fehlende CUTENDs.

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

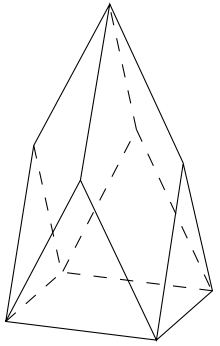
Wird ein Makro zwischen CUTPLANE und CUTEND aufgerufen, dann werden auch die Körper aus dem Makro geschnitten.

Anmerkung 2: Falls Sie CUTPLANE{2} mit mehr als 2 Parametern verwenden, hat dies denselben Effekt wie ein CUTPLANE.

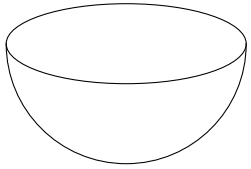
Anmerkung 3: Bevorzugen Sie die Verwendung von CUTPLANE{3} anstatt von CUTPLANE. Falls Sie CUTPLANE mit 5 Parametern verwenden, wird der 4. Parameter übergangen. Bei CUTPLANE{3}, hat dieser Parameter einen Effekt, unabhängig vom 5. Parameter.

Beispiel 1:

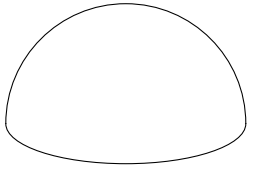
```
CUTPLANE 2, 2, 4
CUTPLANE -2, 2, 4
CUTPLANE -2, -2, 4
CUTPLANE 2, -2, 4
ADD -1, -1, 0
BRICK 2, 2, 4
DEL 1
CUTEND
CUTEND
CUTEND
CUTEND
```



Beispiel 2:

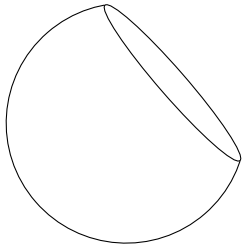


```
CUTPLANE
SPHERE 2
CUTEND
```

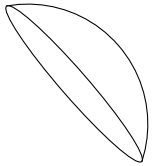


```
CUTPLANE 1, 1, 0, 1
SPHERE 2
CUTEND
```

Beispiel 3:

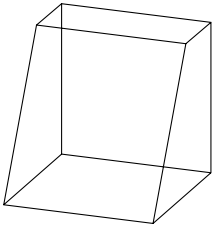


```
CUTPLANE 1.8, 1.8, 1.8
SPHERE 2
CUTEND
```

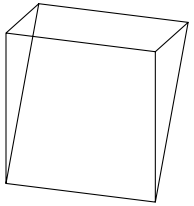


```
CUTPLANE 1.8, 1.8, 1.8, 1
SPHERE 2
CUTEND
```

Beispiel 4:



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



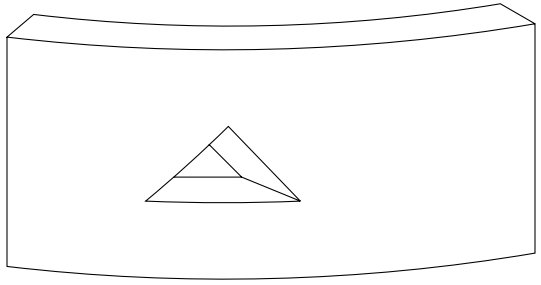
```
CUTPLANE -120
BRICK 2, 2, 2
CUTEND
```

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 kann konkav oder konvex sein, darf sich aber nicht überschneiden. Der Schnitt erfolgt in der Richtung der z-Achse oder eines optional bestimmten (x, y, z) Richtungsvektors. Spiegel-Transformationen beeinflussen die Schnittrichtung auf unvorhersehbare Weise - um unkomplizierte Ergebnisse zu erhalten, sollten Sie CUTFORM.

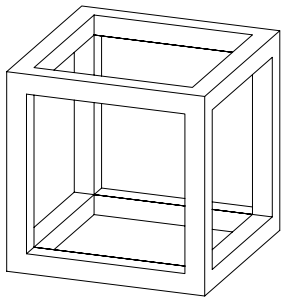
Beispiel 1:



```
ROTX 90
MULZ -1
CUTPOLY 3,
        0.5, 1,
        2, 2,
        3.5, 1,
        -1.8, 0, 1
DEL 1
BPRISM_ "Brick-Red", "Brick-Red", "Brick-White",
        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
```

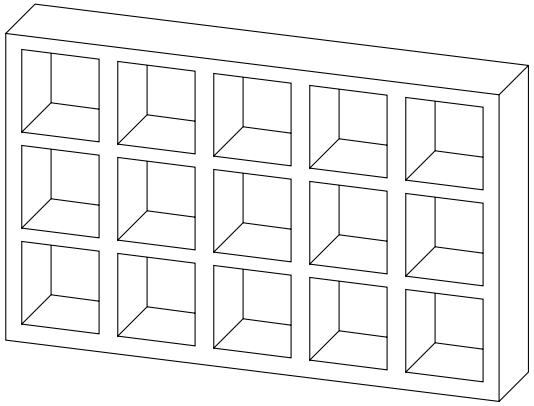
Beispiel 2:

```
a=1.0
d=0.1
GOSUB "rect_cut"
ROTX 90
GOSUB "rect_cut"
DEL 1
ROTY -90
GOSUB "rect_cut"
DEL 1
BLOCK a, a, a
CUTEND
CUTEND
CUTEND
END
"rect_cut":
    CUTPOLY 4,
            d, d,
            a-d, d,
            a-d, a-d,
            d, a-d
    RETURN
```



Beispiel 3:

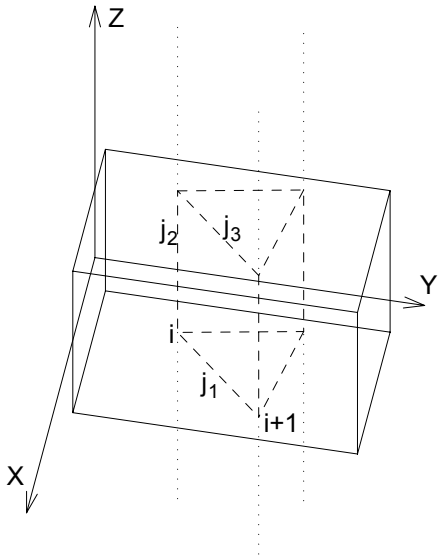
```
ROTX 90
FOR i=1 TO 3
    FOR j=1 TO 5
        CUTPOLY 4,
                0, 0, 1, 0,
                1, 1, 0, 1
        ADDX 1.2
    NEXT j
    DEL 5
    ADDY 1.2
NEXT i
DEL NTR()-1
ADD -0.2, -0.2, 0
BRICK 6.2, 3.8, 1
FOR k=1 TO 15
    CUTEND
NEXT k
DEL TOP
```



CUTPOLYA

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

Ähnlich wie CUTPOLY, aber mit der Möglichkeit die Kantensichtbarkeit der erzeugten Polygone. Der Schnittkörper ist eine in mindestens eine Richtung unendliche Röhre mit der definierten Polygonschnittfläche. Ragt das Ende des Schnittskörpers in die nachfolgenden Körper, wird dieser Teil der Körper ausgeschnitten.



status: steuert die Behandlung der erzeugten Schnittpolygone.

`1:` verwendet die eigenen Attribute des Körpers der erzeugten Polygone und Kanten.

`2:` erzeugte Schnittflächenpolygone werden wie normale Polygone behandelt.

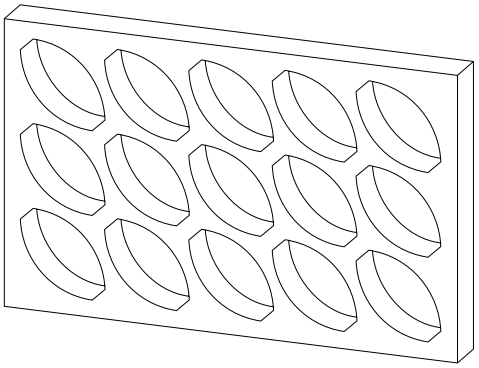
d: der Abstand zwischen dem lokalen Ursprung und dem Ende der unendlichen Röhre.

`0:` bedeutet einen Schnitt mit einer unendlichen Röhre.

maski: gleicht dem PRISM_-Befehl.

`maski = j1 + 2*j2 + 4*j3 + 64*j7`, hierbei kann j jeweils 0 oder 1 sein.

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 [, status]
[statement1 statement2 ... statementn]
CUTEND
```

Schneidet einen Block mit Dicke "d", unendlicher Länge (beide Seiten der y-Achse) und semi-unendlicher Höhe (über der xy Ebene).

status: steuert die Behandlung der erzeugten Schnittpolygone. Ist kein Wert angegeben (aus Kompatibilitätsgründen) wird als Standardwert 3 angenommen.

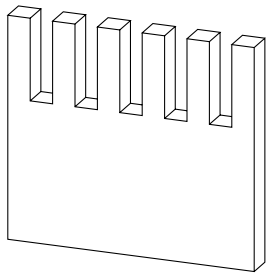
status = j_1 + 2*j_2, hierbei kann j jeweils 0 oder 1 sein.

j_1: verwendet die eigenen Attribute des Körpers der erzeugten Polygone und Kanten.

j_2: erzeugte Schnittflächenpolygone werden wie normale Polygone behandelt.

Beispiel:

```
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
```



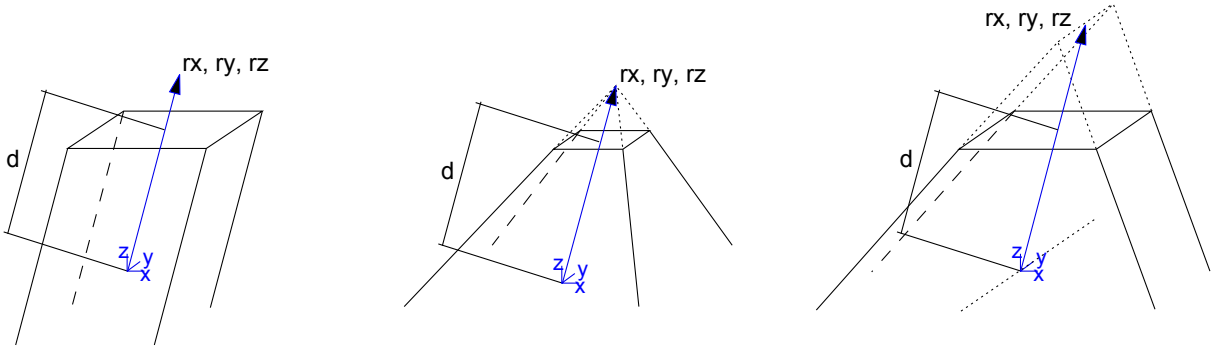
CUTFORM

```
CUTFORM n, method, status,
        rx, ry, rz, d,
        x1, y1, mask1 [, mat1],
        ...
        xn, yn, maskn [, matn]
```

Ähnlich zu CUTPOLYA, aber mit der Möglichkeit die Form und die Ausdehnung des Schnittkörpers zu beeinflussen.

method: Steuert die Form des Schnittkörpers.

1: prismenförmig
2: pyramidenförmig
3: keilförmiger Schnittkörper. Die Richtung der Firstkante 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 = j_1 + 2*j_2 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9, hierbei kann j jeweils 0 oder 1 sein.
j_1: verwendet die eigenen Attribute des Körpers der erzeugten Polygone und Kanten.
j_2: erzeugte Schnittflächenpolygone werden wie normale Polygone behandelt.
j_4: definiert die Begrenzung des Schnittes (mit j5),
j_5: definiert die Begrenzung des Schnittes (mit j4):
j_6: generiert eine Boole'sche Schnittmenge mit dem Schnittkörper statt eines Boole'schen Unterschieds. (kann nur mit dem Befehl CUTFORMverwendet werden)
j_7: Kanten, generiert durch den Schnittkörperboden werden unsichtbar
j_8: Kanten, generiert durch die Schnittkörperspitze werden unsichtbar
j_9: Schnittflächen haben benutzerdefinierte Seitenmaterialien (mati)
j4 = 0 und j5 = 0: endlicher Schnitt
j4 = 0 und j5 = 1: halb-unendlicher Schnitt
j4 = 1 und j5 = 1: unendlicher Schnitt

rx, ry, rz: definiert die Ausrichtung des Schnitts, wenn der Schnitt prismenförmig ist, oder die Spitze der Pyramide, wenn die Schnittmethode pyramidenförmig ist; Ist die Schnittmethode keilförmig geben rx und rz die Position der zur Y-Achse parallelen Keillinie an

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 Ursprung des lokalen Koordinatensystems 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.

mask = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 64*j_7, hierbei kann j jeweils 0 oder 1 sein.

j_1: das Polygon wird eine sichtbare Kante erzeugen beim Eintritt in den geschnittenen Körper (außer es wird ein Solid-Körper mit einer keilförmigen Schnittform geschnitten, siehe unten),

j_2: Die Längskante der Schnittform ist sichtbar

j_3: das Polygon wird eine sichtbare Kante erzeugen beim Verlassen des geschnittenen Körpers (außer es wird ein Solid-Körper mit einer keilförmigen Schnittform geschnitten, siehe unten),

j_4: Die untere Kante der Schnittform ist sichtbar

j_5: Die obere Kante der Schnittform ist sichtbar

j_7: Steuert die vom Blickpunkt abhängige Sichtbarkeit der Längskante

Im Fall, dass ein Solid-Körper mit einer keilförmigen Schnittform geschnitten wird, werden die Werte für die Sichtbarkeit von Eintritts- und Austritts-Kanten (j1 und j3) getauscht. Dieses Verhalten wurde aus Kompatibilitätsgründen beibehalten.

mati: Seitenmaterialien der Schnittflächen (wenn Status j9 = 1)

CUTFORM{2}

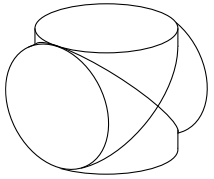
```
CUTFORM{2} n, method, status,
        rx, ry, rz, d,
        x1, y1, mask1 [, mat1],
        ...
        xn, yn, maskn [, matn]
```

CUTFORM{2} ist eine Erweiterung von CUTFORM mit der Möglichkeit, die Inline-Materialdefinition zu verwenden, was bedeutet, dass Materialien lokal innerhalb des GDL-Scriptes definiert werden können und neben den globalen Materialdefintionen verwendet werden können.

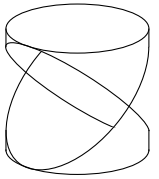
SOLID-ELEMENT-BEFEHLE

GDL kann spezielle 3D-Operationen zwischen Massivkörpern, die durch Gruppierung festgelegt werden, durchführen. Diese Operationen können sein:

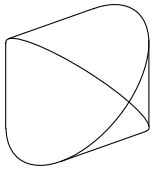
`ADDGROUP` Bildet eine Boole'sche Verbindung aus zwei Massivkörpern;



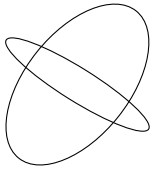
`SUBGROUP` Bildet einen Boole'schen Unterschied aus zwei Massivkörpern;



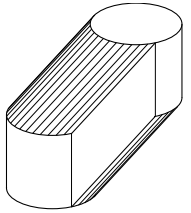
`ISECTGROUP` Bildet eine Boole'sche Schnittmenge aus zwei Massivkörpern;



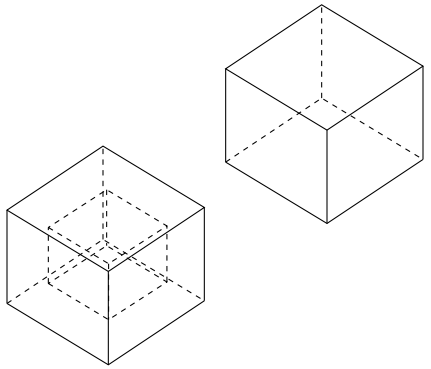
`ISECTLINES` Berechnet die Schnittlinie aus zwei Massivkörpern;



`SWEEPGROUP` Verzieht 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 der Art, 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 sich als Ergebnis eines Boole'schen Unterschieds (z. B. Subtraktion eines kleinen Würfels aus der Mitte eines großen) ergeben.

Siehe auch „Grundelemente".

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:

Tabelle 1. Vereinigung (Basis-Werkzeug)

	Massivbasis	Oberflächenbasis	Drahtmodellbasis	Hybridbasis
Massivwerkzeug	Massivergebnis	Oberflächenergebnis (dazuladen)	Drahtmodellergebnis (dazuladen)	Hybridergebnis (dazuladen)
Oberflächen- werkzeug	Oberflächenergebnis (dazuladen)	Oberflächenergebnis (dazuladen)	Hybridergebnis (dazuladen)	Hybridergebnis (dazuladen)
Drahtmodell- werkzeug	Drahtmodellergebnis (dazuladen)	Hybridergebnis (dazuladen)	Drahtmodellergebnis (dazuladen)	Hybridergebnis (dazuladen)
Hybridwerkzeug	Hybridergebnis (dazuladen)	Hybridergebnis (dazuladen)	Hybridergebnis (dazuladen)	Hybridergebnis (dazuladen)

Tabelle 2. Unterschied (Basis-Werkzeug)

	Massivbasis	Oberflächenbasis	Drahtmodellbasis	Hybridbasis
Massivwerkzeug	Massivergebnis	Oberflächen- ergebnis	Drahtmodell- ergebnis	Hybridergebnis
Oberflächen- werkzeug	Oberflächenbasis (keine Auswirkung)	Oberflächenbasis (keine Auswirkung)	Hybridbasis (keine Auswirkung)	Hybridbasis (keine Auswirkung)
Drahtmodell- werkzeug	Drahtmodellbasis (keine Auswirkung)	Hybridbasis (keine Auswirkung)	Drahtmodellbasis (keine Auswirkung)	Hybridbasis (keine Auswirkung)
Hybridwerkzeug	Hybridbasis (keine Auswirkung)	Hybridbasis (keine Auswirkung)	Hybridbasis (keine Auswirkung)	Hybridbasis (keine Auswirkung)

Tabelle 3. Schnittmenge (Basis-Werkzeug)

	Massivbasis	Oberflächenbasis	Drahtmodellbasis	Hybridbasis
Massivwerkzeug	Massivergebnis	Oberflächen- ergebnis	Drahtmodell- ergebnis	Hybridergebnis
Oberflächen- werkzeug	Oberflächen- ergebnis	keine Auswirkung	keine Auswirkung	keine Auswirkung
Drahtmodell- werkzeug	Drahtmodell- ergebnis	keine Auswirkung	keine Auswirkung	keine Auswirkung
Hybridwerkzeug	Hybridergebnis	keine Auswirkung	keine Auswirkung	keine Auswirkung

Tabelle 4. Schnittlinien (Basis-Werkzeug)

	Massivbasis	Oberflächenbasis	Drahtmodellbasis	Hybridbasis
Massivwerkzeug	Drahtmodell- ergebnis	Drahtmodell- ergebnis	keine Auswirkung	Drahtmodell- ergebnis
Oberflächen- werkzeug	Drahtmodell- ergebnis	keine Auswirkung	keine Auswirkung	keine Auswirkung
Drahtmodell- werkzeug	keine Auswirkung	keine Auswirkung	keine Auswirkung	keine Auswirkung
Hybridwerkzeug	Drahtmodell- ergebnis	keine Auswirkung	keine Auswirkung	keine Auswirkung

Tabelle 5. Verziehen

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

Oberflächen 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 benachbarter Polygonflächen 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 stark gefordert 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 diese Regeln für Schalen (durch Körper definiert) eingehalten werden, nicht jedoch für Massivkörper (durch Gruppen definiert). (Der im obigen Script in der Gruppenkonstruktion erzeugte Massivkörper ist korrekt modelliert, da sich die Schalenflächen, aus denen er besteht, zwar berühren, aber geometrisch korrekt sind.)

GROUP - ENDGROUP

```
GROUP "name"
    [statement1 ... statementn]
ENDGROUP
```

Beginn eine neue 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 PLACEGROUPplatziert werden. Gruppendefinitionen können nicht verschachtelt werden, es können jedoch Makroaufrufe eingeschlossen werden, die Gruppendefinitionen und PLACEGROUP-Befehle enthalten, die wiederum weitere Gruppen verwenden.

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

ADDGROUP

```
ADDGROUP (g_expr1, g_expr2)
ADDGROUP{2} (g_expr1, g_expr2, edgeColor, materialId, materialColor [, operationStatus])
ADDGROUP{3} (g_expr1, g_expr2, edgeColor, materialId, materialColor [, operationStatus])
```

SUBGROUP

```
SUBGROUP (g_expr1, g_expr2)
SUBGROUP{2} (g_expr1, g_expr2, edgeColor, materialId, materialColor [, operationStatus])
SUBGROUP{3} (g_expr1, g_expr2, edgeColor, materialId, materialColor [, operationStatus])
```

ISECTGROUP

```
ISECTGROUP (g_expr1, g_expr2)
ISECTGROUP{2} (g_expr1, g_expr2, edgeColor, materialId, materialColor [, operationStatus])
ISECTGROUP{3} (g_expr1, g_expr2, edgeColor, materialId, materialColor [, operationStatus])
```

g_expr1: Bezeichner der Zielgruppe

g_expr2: Bezeichner der Werkzeug-Gruppe

edgeColor: Farbe der neuen Kante, wenn diese von 0 abweicht

materialId: Material der neuen Außenfläche, wenn es von 0 abweicht

materialColor: Farbe der neuen Außenfläche wenn es von 0 abweicht und materialId = 0

operationStatus: Status-Kontrolle der Operation.

operationStatus = j_1 + 2*j_2, hierbei kann j jeweils 0 oder 1 sein.
j_1: neu erzeugte Schnittkanten sind unsichtbar.
j_2: geschnittene Polygone des Ergebnisses erben Material und Texturprojektion aus den entsprechenden Polygonen der Werkzeuggruppe.

ISECTLINES

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. Beachten Sie, dass die Operationen ADDGROUP, ISECTGROUP und ISECTLINES in ihrer Parametrisierung symmetrisch sind während die Reihenfolge der Parameter bei dem Befehl SUBGROUP eine Rolle spielt.

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.

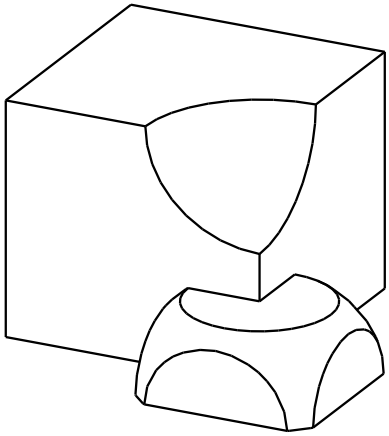
KILLGROUP

KILLGROUP g_expr

Löscht die Körper der angegebenen Gruppe aus dem Speicher. Nach einer Operation KILLGROUP ist die Gruppe leer. Die Namen von Killgroups können innerhalb eines Scriptes nicht mehrfach verwendet werden. Das Löschen erfolgt automatisch am Ende der Interpretation des Objektes oder nach der Rückkehr aus dem Makroaufruf. Zur Verbesserung der Leistung der Objekte 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 Verziehen des Gruppenparameters entlang der angegebenen Ausrichtung erstellt wird. Der Befehl wirkt nur für Massivmodelle.

SWEEPGROUP{2} (g_expr, x, y, z)

Der Unterschied zwischen SWEEPGROUP und SWEEPGROUP {2} ist, dass im ersten Fall die aktuelle Transformationsmatrix für den Richtungsvektor des Zeichnens mit Bezug auf das aktuelle Koordinatensystem erneut verwendet wird. (im Fall von SWEEPGROUP wird die aktuelle Transformation zweimal für den Richtungsvektor mit Bezug auf das globale Koordinatensystem verwendet.)

SWEEPGROUP{3} (g_expr, x, y, z, edgeColor, materialId, materialColor, method)

Diese Version ergänzt eine neue Methodenauswahl zu SWEEPGROUP{2} und funktioniert auch mit Oberflächen-Modellen.

edgeColor: Farbe der neuen Kante, wenn diese von 0 abweicht

materialId: Material der neuen Außenfläche, wenn es von 0 abweicht

materialColor: Farbe der neuen Außenfläche wenn es von 0 abweicht und materialId = 0

method: kontrolliert die Endkontur des resultierenden Körpers.

0: genau wie bei SWEEPGROUP{2}, beide Ende stammen von dem Ursprungskörper

1: der Anfang stammt von dem Ursprungskörper, das verzogene Ende ist flach

SWEEPGROUP{4} (g_expr, x, y, z, edgeColor, materialId, materialColor, method, status)

Diese Version fügt einen neuen Staus-Parameter zu SWEEPGROUP{3} hinzu.

status: Kontrolliert die Attribute des Ergebnisses.

status = $2*j_2$, hierbei kann j jeweils 0 oder 1 sein.

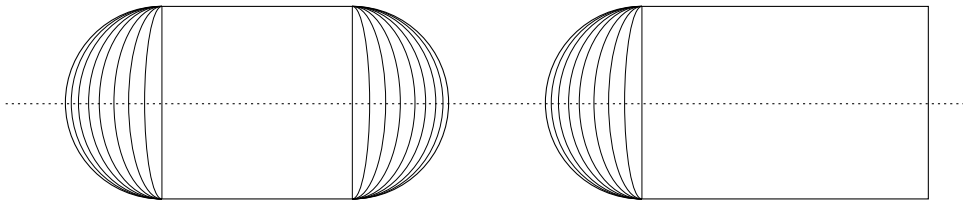
j_2: Behalten pro Polygon die Texturmapping-Parameter auf dem verzogenen Ergebnis (siehe PGON für Details).

SWEEPGROUP{5} (g_expr, x, y, z, edgeColor, materialId, materialColor, method, status)

SWEEPGROUP{5} ist eine Erweiterung des SWEEPGROUP{4} - Befehls mit der Möglichkeit der Verwendung von Inline-Material-Definitionen, d.h. lokal in GDL-Scripten definierte Materialien können neben denen verwendet werden, welche global definiert wurden.

Kompatibilität: eingeführt in ARCHICAD 22.

Beispiel:



```
GROUP "the_sphere"
    SPHERE 1
ENDGROUP
PLACEGROUP SWEEPGROUP{2} ("the_sphere", 2, 0, 0)
ADDX 5
PLACEGROUP SWEEPGROUP{3} ("the_sphere", 2, 0, 0, 4, 0, 4, 1)
del 1
```

CREATEGROUPWITHMATERIAL

CREATEGROUPWITHMATERIAL (g_expr, repl_directive, pen, material)

Gibt eine Gruppe zurück, welche erzeugt wird, indem alle Stiftfarben und/oder Materialien der Gruppe g_expr ersetzt werden.

g_expr: Gruppenbezeichnung, welche die Basis-Gruppe identifiziert.

repl_directive:

repl_directive = $j_1 + 2*j_2 + 4*j_3 + 8*j_4$, hierbei kann j jeweils 0 oder 1 sein.

j_1: ersetzt die Stiftfarbe ,

j_2: ersetzte das Material.

j_4: macht Kanten unsichtbar.

pen: ersetzter Stiftfarbenindex.

material: ersetzter Materialindex.

BINäRES 3D

BINARY

BINARY mode [, section, elementID]

Spezieller Befehl, um binäre Elemente eines Objekte in ein GDL Makro einzuschließen. Ein Satz von Punkten, 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 Binärteiles, von 1 bis 16.

`0:` Mit section=0 beziehen Sie sich gleichzeitig auf sämtliche vorhandenen Binärteile.

`1:` Nur Schnitte (section) mit dem Index-Wert 1 können innerhalb von GDL abgespeichert werden, auch BINARY- Befehle ohne Argument section beziehen sich darauf,

`2-16:` Die anderen Schnitt-Indizes werden von Drittanbieter-Werkzeugen verwendet.

`elementID:` ID eines Elementes dieses Binär-Teiles. Dieser Parameter wird während des Import-Prozesses erzeugt.

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 der Bibliothekselemente über den Befehl Sichern als.Bibliothekselemente im Binärformat.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.

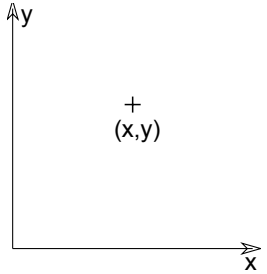
2D ELEMENTE

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

HOTSPOT2 x, y [, unID [, paramReference [, flags [, displayParam [, "customDescription"]]]]]



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

paramReference: Parameter, der durch diesen Fixpunkt mit der Parameter-Bearbeitungsmethode auf Basis grafische veränderbarer.

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

customDescription: benutzerdefinierter Beschreibungs-String des in der Infopalette angezeigten Parameters. Wenn Sie diese Option verwenden, muss displayParam ebenfalls gesetzt sein (verwenden Sie paramReference als Standardwert).

Siehe Bearbeitungsbefehle auf Hotspot-Basis für weitere Informationen über HOTSPOT2.

HOTLINE2

HOTLINE2 x1, y1, x2, y2, unID

Definition einer Linie zwischen zwei Punkten, die zwar nicht dargestellt wird, aber an der der Cursor einrasten kann. Kann eine unique ID für assoziative Bemaßungszwecke besitzen.

HOTARC2

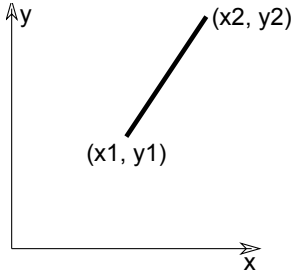
HOTARC2 x, y, r, startangle, endangle, unID

Definiert einen Bogen mit dem Mittelpunkt bei (x, y) zwischen Start- und Endwinkel (startangle, endangle) und dem Radius r. Der Bogen wird zwar nicht dargestellt, aber der Cursor kann an ihm einrasten. Kann eine unique ID für assoziative Bemaßungszwecke besitzen.

LINE2

LINE2 x1, y1, x2, y2

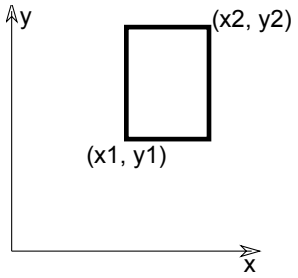
Definition von Linien zwischen zwei Punkten.



RECT2

RECT2 x1, y1, x2, y2

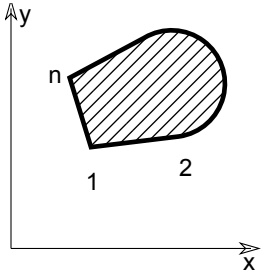
Definition von Rechtecken durch zwei Eckpunkten. Die beiden Punkte befinden sich auf der Diagonalen des Rechtecks, die Seiten sind parallel zur aktuellen X- und Y-Achse.



POLY2

POLY2 n, frame_fill, x1, y1, ..., xn, yn

Ein offenes oder geschlossenes Polygon mit n Eckpunkte.



Einschränkung der Parameter:

n >= 2

n: die Anzahl der Eckpunkte

x1, y1, ..., xn, yn: Koordinaten jedes Eckpunktes.

frame_fill:

frame_fill = j_1 + 2*j_2 + 4*j_3, hierbei kann j jeweils 0 oder 1 sein.

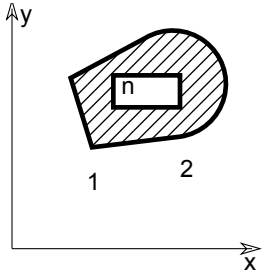
j_1: nur Kontur

j_2: nur Schraffur

j_3: Schließen eines offenen Polygons

POLY2_

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



Gleicht dem POLY2-Befehl, jedoch kann die Sichtbarkeit jeder beliebigen Kante unterdrückt werden. Ist `si = 0`, ist die Kante ausgehend vom Punkt (xi,yi) unsichtbar. Ist `si = 1`, sollte der Kreuzungspunkt sichtbar sein. `si = -1` wird verwendet um Löcher direkt zu definieren. Sie können mit zusätzlichen Statuscodewerten auch Bögen und Segmente in der Polylinie definieren, sowie Durchbrüche in der Fläche definieren.

Einschränkung der Parameter:

`n >= 2`

n: die Anzahl der Eckpunkte

x1, y1, ..., xn, yn: Koordinaten jedes Eckpunktes.

frame_fill:

`frame_fill = j1 + 2*j2 + 4*j3 + 8*j4 + 32*j6 + 64*j7`, hierbei kann j jeweils 0 oder 1 sein.

`j1`: nur Kontur
`j2`: nur Schraffur
`j3`: Schließen eines offenen Polygons
`j4`: lokale Schraffurausrichtung
`j6`: Schraffur ist Bauteilschraffur (Grundeinstellung ist Zeichenschraffur)
`j7`: Schraffur ist Deckschraffur (nur falls j6 = 0, Grundeinstellung ist Zeichenschraffur)

si: Statuswerte:

`si = j1 + 16*j5 + 32*j6`, hierbei kann j jeweils 0 oder 1 sein.

`j1`: das nächste Segment ist sichtbar
`j5`: das nächste Segment ist eine Innenlinie (wenn 0, spezifische Linie)
`j6`: das nächste Segment ist eine Kontur (nur wirksam, wenn j5 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.

Siehe „Zusätzliche Statuscodes" für 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_, mit weiteren Parametern: der Stift-Schraffur und des Hintergrundstiftes. Alle anderen Parameter entsprechen denen, die unter dem Befehl POLY2_beschrieben wurden.

`fill_pen:` Nummer des Schraffurstifts

`fill_background_pen:` Nummer des Schraffurhintergrundstifts

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

Siehe „Zusätzliche Statuscodes" für 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.

`frame_fill:`

`frame_fill` = $j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7$, hierbei kann j jeweils 0 oder 1 sein.

j_1: nur Kontur

j_2: nur Schraffur

j_3: Schließen eines offenen Polygons

j_4: lokale Schraffurausrichtung

j_5: globale Schraffurursprung (nur wirksam, wenn j4 eingestellt ist)

j_6: Schraffur in Bauteilkategorie (abgrenzend mit j7, Zeichnungskategorie wenn nichts anders eingestellt ist)

j_7: Schraffur in Deckschraffurkategorie (abgrenzend mit j6, Zeichnungskategorie wenn nichts anders eingestellt ist)

fillOrigoX: X-Koordinate des Schraffurursprungs

fillOrigoY: Y-Koordinate des Schraffurursprungs

fillAngle: Ausrichtung des Schraffurwinkels

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

Siehe „Zusätzliche Statuscodes" für Details.

POLY2_B{3}

```
POLY2_B{3} n, frame_fill,
        fill_pen, fill_background_pen,
        fillOrigoX, fillOrigoY,
        mxx, mxy, myx, myy, x1, y1, s1, ..., xn, yn, sn
```

Erweiterte Version des Befehls POLY2_B, wo die Ausrichtung der Schraffur mithilfe einer Matrix definiert werden kann.

frame_fill:

frame_fill = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8, hierbei kann j jeweils 0 oder 1 sein.

j_1-j_7: ähnlich wie für die vorherigen POLY2_ Befehle

j_8: verwenden Sie geneigte Schraffur

mxx, mxy, myx, myy: wenn j8 eingestellt wurde, diese Matrix definiert die Richtung der Schraffur

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

Siehe „Zusätzliche Statuscodes" für Details.

POLY2_B{4}

```
POLY2_B{4} n, frame_fill,
        fill_pen, fill_background_pen,
        fillOrigoX, fillOrigoY,
        mxx, mxy, myx, myy,
        gradientInnerRadius,
        x1, y1, s1, ..., xn, yn, sn
```

Erweiterte Version des Befehls POLY2_ B{3}, wo der Innenradius der Verlaufsschraffur definiert werden kann.

gradientInnerRadius: Innenradius des Farbverlaufs, falls eine Verlaufsschraffur für das Polygon ausgewählt ist.

POLY2_B{5}

```
POLY2_B{5} n, frame_fill, fillcategory, distortion_flags,
        fill_pen, fill_background_pen,
        fillOrigoX, fillOrigoY,
        mxx, mxy, myx, myy,
        gradientInnerRadius,
        x1, y1, s1, ..., xn, yn, sn
```

Erweiterte Version des Befehls POLY2_ B{4}, wo die Schraffurverformung auf verbesserte Weise eingestellt werden kann.

frame_fill:

frame_fill = j_1 + 2*j_2 + 4*j_3, hierbei kann j jeweils 0 oder 1 sein.

j_1: nur Kontur

j_2: nur Schraffur

j_3: Schließen eines offenen Polygons

fillcategory:

0: Zeichnungsschraffur

1: Bauteilschraffur

2: Deckschraffur

distortion_flags:

distortion_flags = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7, hierbei kann j jeweils 0 oder 1 sein.

Der gültige Wert der Verformungs_flags liegt zwischen 0 und 127. Verwenden Sie keine Werte außerhalb dieses Bereiches.

j_1: die X-Koordinate des Schraffurursprungs ist die X-Koordinate des globalen Ursprungs, bedeutsam nur wenn j4 gesetzt ist. Das fillOrigo ist der projizierte Ursprung (0,0) auf der Linie des (mxx, mxy) Vektors.

j_2: die Y-Koordinate des Schraffurursprungs ist die Y-Koordinate des globalen Ursprungs, bedeutsam nur wenn j4 gesetzt ist.

j_3: kreisförmige Verformung erstellen unter Verwendung des Parameters InnererRadius

j_4: lokale Ausrichtung verwenden, Verformungsmatrix verwenden (mij Parameter)

j_5: (nur wirksam für Symbolschraffuren) X-Größe des Musters auf die Länge des definierten X-Vektors zurücksetzen (mxx, mxy)

j_6: (nur wirksam für Symbolschraffuren) Y-Größe des Musters auf die Länge des definierten Y-Vektors zurücksetzen (mxx, mxy)

j_7: (nur wirksam für Symbolschraffuren) Proportionen des Symbol-Schraffurmusters beibehalten; nur wirksam wenn eines von j5 und j6 eingestellt ist

innerRadius: Radius für kreisförmige Schraffurverformung; der Ursprung des Basiskreises wird auf der Y-Schraffur-Achse in der Position (0, -InnererRadius) platziert

POLY2_B{6}

```
POLY2_B{6} n, frame_fill, fillcategory, distortion_flags,
                   fill_pen, fill_background_pen,
                   fillOrigoX, fillOrigoY,
                   mxx, mxy, myx, myy,
                   gradientInnerRadius,
                   x1, y1, s1, pen1, linetype1, ..., xn, yn, sn, penn, linetypen
```

Erweiterte Version von POLY2_B {5}, wobei Konturattribute (Stift und Linientyp) individuell für jedes Kontursegment gesteuert werden können.

peni: Stift-Index der Konturlinie ab Kontrollpunkt i.

linetypei: Linientyp-Index der Konturlinie ab Kontrollpunkt i.

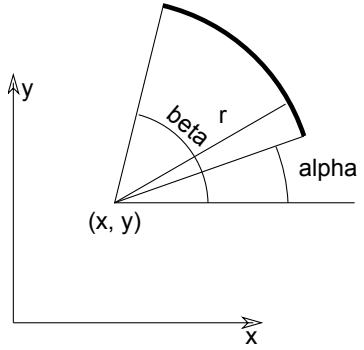
Kompatibilität: eingeführt in ARCHICAD 21.

ARC2

```
ARC2 x, y, r, alpha, beta
```

Ein Bogen zwischen den Winkeln alpha und beta, einem Mittelpunkt bei (x, y) und dem Radius r.

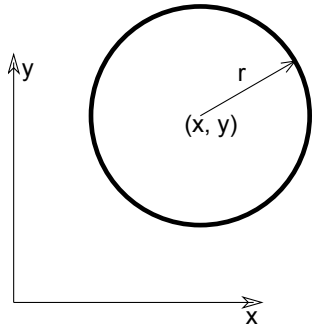
alpha und beta werden in Grad angegeben.



CIRCLE2

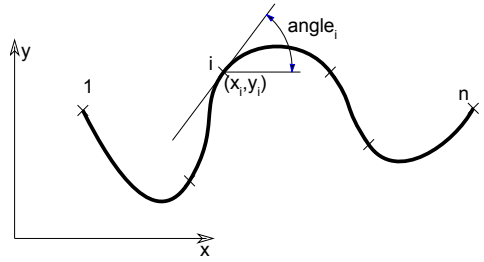
```
CIRCLE2 x, y, r
```

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



SPLINE2

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



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

Einschränkung der Parameter:

n >= 2

si: Statuswerte:

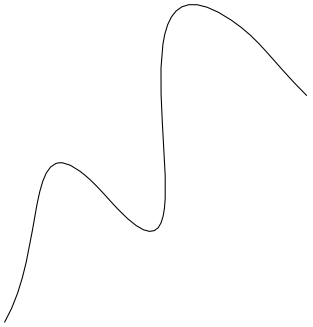
0: standard

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

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

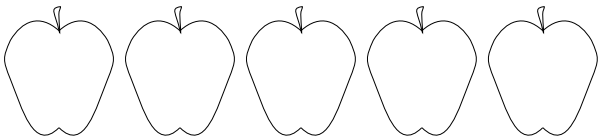
Beispiel 1:

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



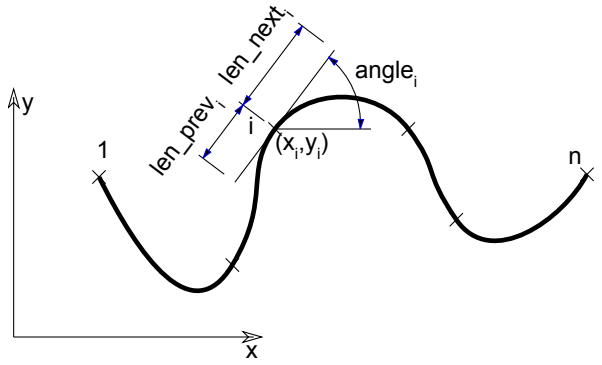
Beispiel 2:

```
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 "Linien/ Zeichnen von Splines" im Kapitel Dokumentation, in der ARCHICAD Hilfe.*

si: Statuswerte:

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 $_i$ und length_next$_i$ 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.

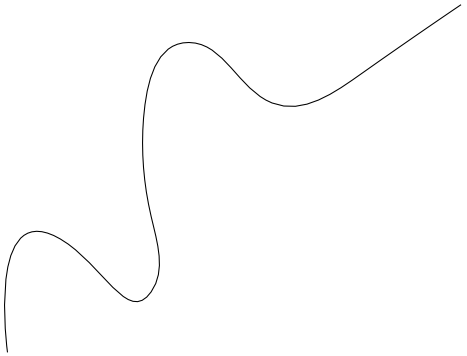
xi, yi: Kontrollpunkt-Koordinaten.

length_previousi, length_nexti: Längen der Tangenten des vorherigen und des folgenden Kontrollpunktes.

anglei: 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

PICTURE2 expression, a, b, mask

PICTURE2{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.

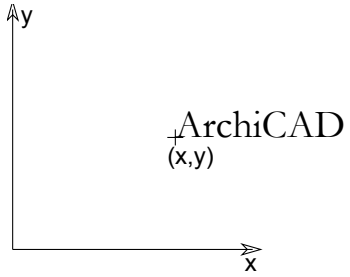
Andere Bilder können in Bibliothekselementen nur dann gespeichert werden, wenn das Projekt oder die ausgewählten Elemente, die die Bilder beinhalten, als GDL-Objekte gesichert werden.

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 und DEFINE STYLE.



RICHTEXT2

RICHTEXT2 `x, y, textblock_name`

Platzieren Sie einen zuvor definierten TEXTBLOCK.

Für weitere Details, *siehe TEXTBLOCK.*

`x, y:` X-Y Koordinaten der RICHTEXT-Platzierung

`textblock_name:` Name eines zuvor definierten TEXTBLOCK

Binäres 2D

FRAGMENT2

FRAGMENT2 `fragment_index, use_current_attributes_flag`
FRAGMENT2 `ALL, use_current_attributes_flag`

Das 2D-Element mit dem angegebenen Index wird mit den aktuellen Transformationen in das 2D-Vollbild eingefügt. Falls `ALL` angegeben ist, werden alle Fragmente 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

PROJECT2{2}

PROJECT2{2} projection_code, angle, method [, backgroundColor,
fillOrigoX, fillOrigoY, filldirection]

Erstellt eine Projektion des 3D-Scripts in demselben Bibliothekselement und addiert die erzeugten Linien zum parametrischen 2D Symbol. Die zweite Version, PROJECT2{2} ermöglicht dem Benutzer zusammen mit einem vorherigen Befehl [SET] FILLdie Steuerung der Hintergrundschraffur, des Ursprungs und die Steuerung der aus dem 2D-Script resultierenden Zeichnung. Der SET FILL 0 Shortcut zum Erzeugen einer leeren Schraffur, funktioniert in diesem Fall nicht, Sie müssen stattdessen auf eine wirklich leere Schraffur referenzieren.

projection_code: Projektionsart

- 3: Draufsicht
- 4: Seitenansicht
- 5: Seitenansicht 2
- 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.Falls ungültig oder kein Wert gesetzt ist, ist der Standard Verdeckte Linien (2).

- 1: Drahtmodell
- 2: Verdeckte Kanten (analytisch)
- 3: Schattierung
- 16: Zusätzlicher Modifizierer: zeichnet Vektorschraffuren (nur wirksam im Verdeckte Kanten und Schattierten Modus),
- 32: Zusätzlicher Modifizierer: verwendet aktuelle Attribute anstatt Attributen aus 3D (nur wirksam im Schattierten Modus),

64: Zusätzlicher Modifizierer: lokale Schraffur-Ausrichtung (nur wirksam im Schattierten Modus),
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 512 nicht aktiviert wurde). Grundeinstellung ist Zeichnungsschraffuren.

BackgroundColor: Hintergrundfarbe der Schraffur.

fillOrigoX: X-Koordinate des Schraffurursprungs

fillOrigoY: Y-Koordinate des Schraffurursprungs

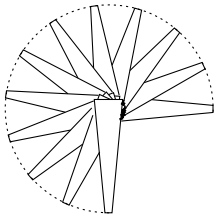
filldirection: Ausrichtung des Schraffurwinkels

Anmerkung: [SET] FILL SET FILL ist wirksam für PROJECT2{2}

Beispiel:

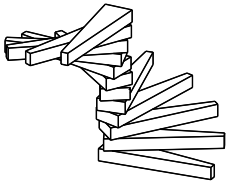
2D

```
PROJECT2 3, 270, 2

LINE_TYPE "DASHED"
ARC2 0, 0, A-B/3, 0, E

E = 270
A = 1
B = 0.2

ROT2 E
ADD2 A-B/3, 0
LINE2 0, 0, -0.05, -0.1
LINE2 0, 0,  0.05, -0.1

DEL 2
```



3D

```
n = 12
E = 270
D = 0.2
A = 1
B = 0.2

FOR i=1 TO n
    prism  4, D,
         -B/3, -B/2,
         -B/3,  B/2,
        A-B/3,  B/8,
        A-B/3, -B/8
    ADDZ D
    ROTz E/(n-1)
NEXT i

DEL n*2
```



PROJECT2{3}

PROJECT2{3} projection_code, angle, method, parts [, backgroundColor, fillOrigoX, fillOrigoY, filldirection][[,] PARAMETERS name1=value1, ..., namen=valuen]

Erstellt eine Projektion des 3D-Scripts in demselben Bibliothekselement und addiert die erzeugten Linien zum parametrischen 2D Symbol. Mit der dritten Version, PROJECT2{3} können Sie bestimmen, welche Teile des projizierten Modells nötig sind und Sie haben die Möglichkeit, die Attribute des Schnitt und Ansichtsteiles, einschließlich des Linientyps getrennt zu steuern. Sie können die Projektion mit den im Befehl bestimmten Parametern erstellen.

method: die ausgewählte Visualisierungsmethode. Falls ungültig oder kein Wert gesetzt ist, ist der Standard Verdeckte Linien (2).

1: Drahtmodell

2: Verdeckte Kanten (analytisch)

3: Schattierung

16: Zusätzlicher Modifizierer: zeichnet Vektorschraffuren (nur wirksam im Verdeckte Kanten und Schattierten Modus),

32: Zusätzlicher Modifizierer: verwendet aktuelle Attribute anstatt Attributen aus 3D (nur wirksam im Schattierten Modus),

64: Zusätzlicher Modifizierer: lokale Schraffur-Ausrichtung (nur wirksam im Schattierten Modus),

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 512 nicht aktiviert wurde). Grundeinstellung ist Zeichnungsschraffuren.

2048: Zusätzlicher Modifizierer: Modifizierer 16, 32, 64, 128, 256, 512, 1024 und Schraffurattributparameter sind nur für den Ansichtsteil der Projektion wirksam. Grundeingestellt gelten diese für alle Teile.

4096: Zusätzlicher Modifizierer: Modifizierer 16, 32, 64, 128, 256, 512, 1024 und Schraffurattributparameter sind nur für den Ausschnittsteil der Projektion wirksam. Grundeingestellt gelten diese für alle Teile.

8192: Zusätzlicher Modifizierer: Bauteilschraffuren sind geneigt

16384: Zusätzlicher Modifizierer: ermöglicht Transparenz bei transparenten Oberflächen. Beachten Sie, dass Transparenz in diesem Fall volle Transparenz für Oberflächen mit einer Durchlässigkeit von über 50 bedeutet, alles andere ist nicht-transparent.

Bekannte Einschränkung: Linien des Schnitteils können nicht getrennt behandelt werden, nur alle Linien zusammen können entweder auf Innen- oder auf Konturlinien gestellt werden.

Kompatibilitätshinweis: Bis ARCHICAD 19 wurden Schnittpolygone mit Attributen generiert, die im 3D-Script definiert wurden von SECT_FILL oder SECT_ATTRS. Ab ARCHICAD 20 werden die Attribute der geschnittenen Polygone durch die Deckschraffur der Außenflächen definiert (falls der Zusatzmodifikator 32 nicht gesetzt ist)).

parts: definiert die zu erstellenden Teile. Die Werte 1+2+4+8+16+32 bedeuten alle Teile.

`parts = j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6`, hierbei kann j jeweils 0 oder 1 sein.

Die Werte j1j2j3, j4, j5, j6 geben an, ob die entsprechenden Teile des projizierten Modells vorhanden sind (1) oder nicht (0).

`j1`: geschnittene Polygone (nur im Schattierungsmodus aktiv),
`j2`: geschnittene Polygonkonturen
`j3`: Ansichtspolygone
`j4`: Ansichtspolygonkonturen,
`j5`: projizierte 3D-Hotspots als statische 2D-Hotspots,
`j6`: projizierte 3D-Hotlines und Hotarcs (einschließlich zugehöriger 3D-Hotspots umgewandelt in statische 2D-Hotspots).

PROJECT2{4}

```
PROJECT2{4} projection_code, angle,
       useTransparency, statusParts,
       numCutplanes,
       cutplaneHeight1, ..., cutplaneHeightn,

       method1, parts1,
       cutFillIndex1,
       cutFillFgPen1, cutFillBgPen1,
       cutFillOrigoX1, cutFillOrigoY1, cutFillDirection1,
       cutLinePen1, cutLineType1,
       projectedFillIndex1,
       projectedFillFgPen1, projectedFillBgPen1,
       projectedFillOrigoX1, projectedFillOrigoY1,
       projectedFillDirection1,
       projectedLinePen1, projectedLineType1,
       ...
       method(numCutplanes+1)), parts(numCutplanes+1),
       cutFillIndex(numCutplanes+1),
       cutFillFgPen(numCutplanes+1), cutFillBgPen(numCutplanes+1),
       cutFillOrigoX(numCutplanes+1), cutFillOrigoY(numCutplanes+1),
       cutFillDirection(numCutplanes+1),
       cutLinePen(numCutplanes+1), cutLineType(numCutplanes+1),
       projectedFillIndex(numCutplanes+1),
       projectedFillFgPen(numCutplanes+1), projectedFillBgPen(numCutplanes+1),
       projectedFillOrigoX(numCutplanes+1), projectedFillOrigoY(numCutplanes+1),
       projectedFillDirection(numCutplanes+1),
       projectedLinePen(numCutplanes+1), projectedLineType(numCutplanes+1)
```

Kompatibilität: eingeführt in ARCHICAD 20.

Erstellt eine Projektion des 3D-Scripts in demselben Bibliothekselement und addiert die erzeugten Linien zum parametrischen 2D Symbol. Die vierte Version, PROJECT2{4}, bietet als Zusatzfunktion die Möglichkeit mehrere Schneideebenen parallel zur X-Y-Ebene zu definieren, und die Attribute der Schnitte und projizierten Teile der Beschneidungskörper zu kontrollieren, einschließlich Linientyp,Stifte und Schraffuren. Die Anzahl der Schneideebenen kann Null sein, wodurch exakt ein unbeschnittener Beschneidungskörper entsteht (numCutplanes+1).

`useTransparency:` kann 0 sein (keine Transparenz) oder eine positive Ganzzahl (1: Transparenz aktiviert).

`statusParts:` definiert die zu erzeugenden Status-Teile (hotlines, hotspots, hotarcs). Die Werte 1+2 value bedeuten alle Teile. Einstellung gilt für alle Beschneidungskörper.

`statusParts = j1 + 2*j2`, hierbei kann j jeweils 0 oder 1 sein.

Die Zahlen j1, j2 bedeuten, dass die zugehörigen Staus-Teile des projizierten Modells sichtbar sind (1) oder verborgen (0):

`j1:` projizierte 3D-Hotspots als statische 2D-Hotspots,

`j2:` projizierte 3D-Hotlines und Hotarcs (einschließlich zugehöriger 3D-Hotspots umgewandelt in statische 2D-Hotspots).

`numCutplanes:` die Anzahl der definierten Schnittebenen. Kann Null sein, vorzugsweise jedoch mehr.

`cutplaneHeighti:` Pie Position jeder individuell definierten Schneideebene. Gemessen als Länge rechtwinklig von der X-Y-Ebene des Objekts.

`method:` die ausgewählte Visualisierungsmethode. Falls ungültig oder kein Wert gesetzt ist, ist der Standard Verdeckte Linien (2).

`0:` der aktuelle Beschneidungskörper ist nicht Teil der Projektion,

`1:` Drahtmodell

`2:` Verdeckte Kanten (analytisch)

`3:` Schattierung

`4:` Verdeckte Linien mit Polygon: Das Polygon beseitigt nicht irgendein anderes Polygon oder eine Linie, welche zu den Teilen gehören, die mit der Schattierungsmethode erstellt wurden, aber es überdeckt/beseitigt Polygone und Linien, welche zu anderen Drahtgitter/Verdeckten-Linien-Teilen gehören. Für optimale Ergebnisse sollten Sie Luftschicht einstellen. Solche zerlegten Polygone verhalten sich in 2D entsprechend der Reihenfolge der Beschneidungskörper (es werden schattierte Teile abgedeckt, aber nicht ausgeblendet).

`16:` Zusätzlicher Modifizierer: zeichnet Vektorschraffuren (nur wirksam im Verdeckte Kanten und Schattierten Modus),

`32:` Zusätzlicher Modifizierer: verwendet aktuelle Attribute anstatt Attributen aus 3D (nur wirksam im Schattierten Modus und Verdeckte-Linien-mit-Polygon-Modus),

`64:` Zusätzlicher Modifizierer: lokale Schraffur-Ausrichtung (nur wirksam im Schattierten Modus und Verdeckte-Linien-mit-Polygon-Modus),

`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 512 nicht aktiviert wurde). Grundeinstellung ist Zeichnungsschraffuren.

`2048:` Zusätzlicher Modifizierer: Modifizierer 16, 32, 64, 128, 256, 512, 1024 und Schraffurattributparameter sind nur für den Ansichtsteil der Projektion wirksam. Grundeingestellt gelten diese für alle Teile.

`4096:` Zusätzlicher Modifizierer: Modifizierer 16, 32, 64, 128, 256, 512, 1024 und Schraffurattributparameter sind nur für den Ausschnittsteil der Projektion wirksam. Grundeingestellt gelten diese für alle Teile.

8192: Zusätzlicher Modifizierer: Bauteilschraffuren sind geneigt

partsi: definiert die zu erstellenden Teile. Die Werte 1+2+4+8+64 bedeuten alle Teile.
partsi = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 64*j_7, hierbei kann j jeweils 0 oder 1 sein.
Die Werte j1, j2, j3, j4, j7 geben an, ob die entsprechenden Teile des projizierten Modells vorhanden sind (1) oder nicht (0).
j_1: cut polygons (effective only in shading mode),
j_2: geschnittene Polygonkonturen
j_3: Ansichtspolygone
j_4: Ansichtspolygonkonturen,
j_7: Projekt Punktwolken.

cutFillIndexi: Schraffurtyp-Index des geschnittenen Bereiches des aktuellen Beschneidungskörpers.

cutFillFgPeni: Schraffurstift des geschnittenen Bereiches des aktuellen Beschneidungskörpers.

cutFillBgPeni: Schraffurhintergrundstift des geschnittenen Bereiches des aktuellen Beschneidungskörpers.

cutFillOrigoXi: X Koordinate des Schnittschraffurursprungs des aktuellen Beschneidungskörpers.

cutFillOrigoYi: Y Koordinate des Schnittschraffurursprungs des aktuellen Beschneidungskörpers.

cutFillDirectioni: Richtungswinkel der Schnittschraffur des aktuellen Beschneidungskörpers.

cutLinePeni: Stift-Index der Schnittlinien des aktuellen Beschneidungskörpers.

cutLineTypei: Linientyp der Schnittlinien des aktuellen Beschneidungskörpers.

projectedFillIndexi: Schraffurtyp-Index des projizierten Bereiches des aktuellen Beschneidungskörpers.

projectedFillFgPeni: Schraffurstift des projizierten Bereiches des aktuellen Beschneidungskörpers.

projectedFillBgPeni: Schraffurhintergrundstift des projizierten Bereiches des aktuellen Beschneidungskörpers.

projectedFillOrigoXi: X Koordinate des projizierten Schraffurursprungs des aktuellen Beschneidungskörpers.

projectedFillOrigoYi: Y Koordinate des projizierten Schraffurursprungs des aktuellen Beschneidungskörpers.

projectedFillDirectioni: Richtungswinkel der projizierten Schraffur des aktuellen Beschneidungskörpers.

projectedLinePeni: Stift-Index der projizierten Linien des aktuellen Beschneidungskörpers.

projectedLineTypei: Linientyp der projizierten Linien des aktuellen Beschneidungskörpers.

ZEICHNUNGEN IN DER LISTE

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

Ist das Element ein spezielles Eigenschaftenbibliothekselement und ist irgendwie mit einem im Grundriss platzierten 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,
fillOrigoX, fillOrigoY, filldirection]

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

method: Neue methode-Flags in DRAWING3{2}:

3: Schattierung

32: aktuelle Attribute verwenden statt der Attribute aus 3D

64: lokale Schraffurausrichtung

DRAWING3{3}

DRAWING3{3} projection_code, angle, method, parts [, backgroundColor,
fillOrigoX, fillOrigoY, filldirection][[,]
PARAMETERS name1=value1, ..., namen=valuen]

Erstellt ähnlich wie PROJECT2, eine Projektion des 3D-Scriptes des Bibliothekselementes, dass dem Eigenschaftenbibliothekselement zugeordnet ist, dass diesen Befehl enthält. Alle Parameter sind denen von PROJECT2, PROJECT2{2} und PROJECT2{3} ähnlich.

method: Neue methode-Flags in DRAWING3{3}:

2048: Zusätzlicher Modifizierer: Modifizierer 16, 32, 64, 128, 256, 512, 1024 und Schraffurattributparameter sind nur für den Ansichtsteil der Projektion wirksam. Grundeingestellt gelten diese für alle Teile.

4096: Zusätzlicher Modifizierer: Modifizierer 16, 32, 64, 128, 256, 512, 1024 und Schraffurattributparameter sind nur für den Ausschnittsteil der Projektion wirksam. Grundeingestellt gelten diese für alle Teile.

`8192:` Zusätzlicher Modifizierer: Bauteilschraffuren sind geneigt
`16384:` Zusätzlicher Modifizierer: ermöglicht Transparenz bei transparenten Oberflächen. Beachten Sie, dass Transparenz in diesem Fall volle Transparenz für Oberflächen mit einer Durchlässigkeit von über 50 bedeutet, alles andere ist nicht-transparent.

BEARBEITUNGSBEFEHLE AUF HOTSPOT-BASIS

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

HOTSPOT x, y, z [, unID [, paramReference [, flags [, displayParam [, "customDescription"]]]]]
HOTSPOT2 x, y [, unID [, paramReference [, flags [, displayParam [, "customDescription"]]]]]

unID: eindeutige ID-Nr. >0; muss innerhalb der im Bibliothekselement definierten Hotspots eindeutig sein.

paramReference: Parameter, der durch diesen Fixpunkt mit der Parameter-Bearbeitungsmethode auf Basis grafische veränderbarer.

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

customDescription: benutzerdefinierter Beschreibungs-String für in der Infopalette angezeigte Parameter. Wenn Sie diese Option verwenden, muss displayParam ebenfalls gesetzt sein (verwenden Sie paramReference als Standardwert). Es wird nur der Wert für die Hotspots vom Bewegungs-Typ angezeigt. Es wird empfohlen, die gleiche Beschreibung für sämtliche Bewegungs-Hotspots zu verwenden, die den selben Basis-Hotspot benutzen.

Beispiele gültiger Argumente:

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

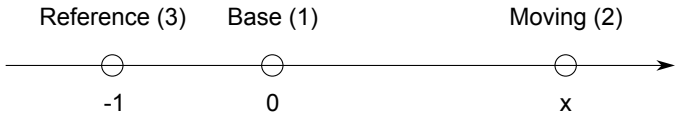
flags: Hotspottyp + Attribute:

type:

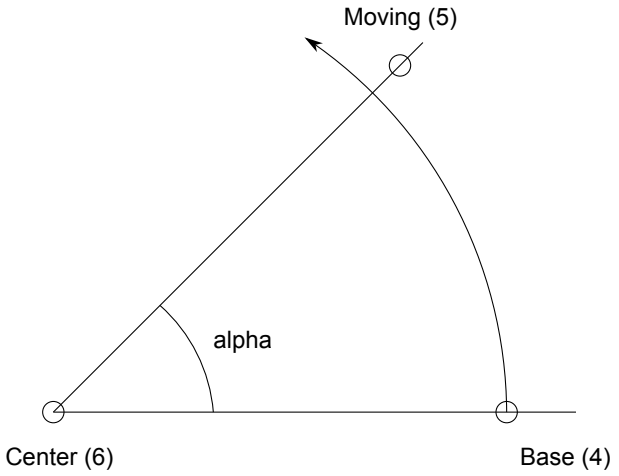
1: Typ Längenbearbeitung, Basis-Hotspot
2: Typ Längenbearbeitung, beweglicher Hotspot
3: Typ Längenbearbeitung, Referenz-Hotspot (immer verborgen)
4: Typ Winkelbearbeitung, Basis-Hotspot
5: Typ Winkelbearbeitung, beweglicher Hotspot
6: Typ Winkelbearbeitung, Mittelpunkt des Winkels (immer verborgen)
7: Typ Winkelbearbeitung, Referenz-Hotspot (immer verborgen)

attribute: Attribut kann eine Kombination der folgenden Werte oder Null sein:
attribute = $128*j_8 + 256*j_9 + 512*j_{10} + 1024*j_{11}$, hierbei kann j jeweils 0 oder 1 sein.
j_8: Hotspot verbergen (wichtig für Typen: 1,2,4,5),
j_9: Bearbeitbarer Basis-Hotspot (für Typen: 1,4),
j_{10}: Umkehrung des Winkels in 2D (für Typ 6),
j_{11}: verwenden Sie den Wert paramReference in Metern auf dem Papierbereich.

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 Winkelebene 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 Gruppen von Hotspots zur Bearbeitung des gleichen Parameters definiert werden, werden die Hotspots in der Reihenfolge der Ausführung der Hotspot-Befehle gruppiert. Wenn das Attribut "bearbeitbar" 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 verschoben bzw. gedreht. (Bei einer Änderung des Parameterwerts ändert der bewegliche Hotspot seine Position nicht.)

Zwei Hotspot-Sets des Typs Länge können kombiniert werden, um das Bearbeiten von zwei 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. Ein Bearbeiten kombinierter Parameter wird gestartet, wenn an der Position des ausgewählten Punkts zwei bearbeitbare Hotspots (bewegliche oder bearbeitbare Basis) mit unterschiedlichen zugeordneten Parametern liegen. Bei Parametern, die für eine kombinierte

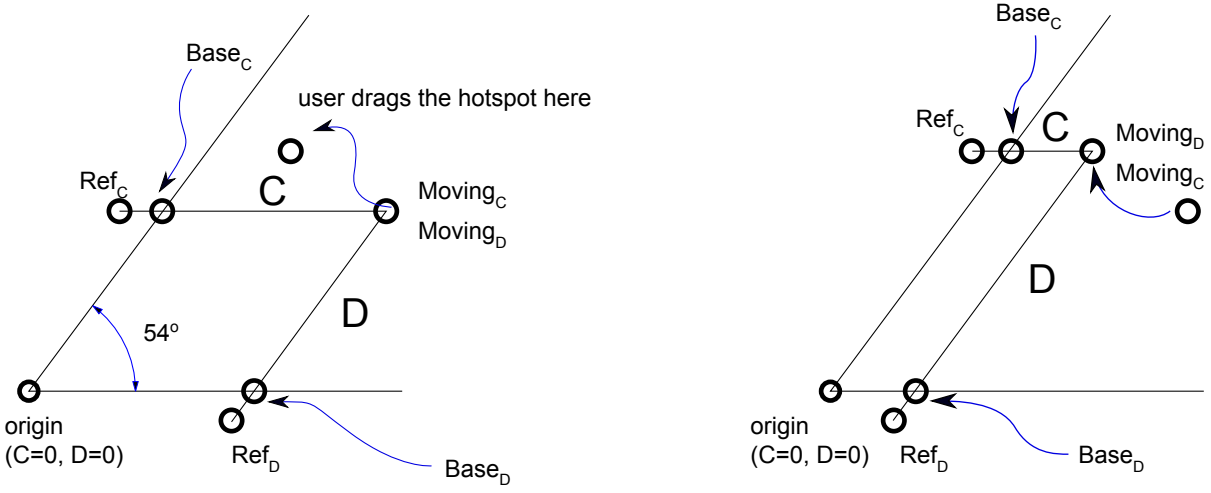
Bearbeitung konzipiert wurden, sind Basis- und Referenz-Hotspots im Koordinatenrahmen nicht fixiert, sondern werden bei der Änderung anderer Parameterwerte verschoben.

Siehe hierzu die Illustration und Beispiel 2.

Beispiel 1: Winkelbearbeitung in 2D:

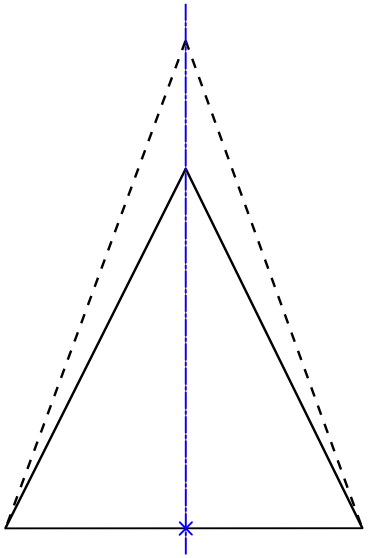
```
LINE2 0, 0, A, 0
LINE2 0, 0, A*COS(angle), A*SIN(angle)
ARC2 0, 0, 0.75*A, 0, angle
HOTSPOT2 0, 0, 1, angle, 6
HOTSPOT2 0.9*A, 0, 2, angle, 4
HOTSPOT2 0.9*A*COS(angle), 0.9*A*SIN(angle), 3,
angle, 5
```

Beispiel 2: kombinierte Längenbearbeitung mit 2 Parametern in 2D:



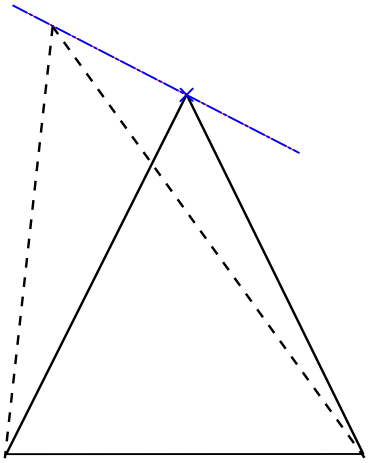
```
! sideX, sideY parameters
RECT2 0, 0, A, B
RECT2 0, 0, sideX, sideY
HOTSPOT2 sideX, 0, 1, sideY, 1
HOTSPOT2 sideX, -0.1, 2, sideY, 3
HOTSPOT2 sideX, sideY, 3, sideY, 2
HOTSPOT2 0, sideY, 4, sideX, 1
HOTSPOT2 -0.1, sideY, 5, sideX, 3
HOTSPOT2 sideX, sideY, 6, sideX, 2
```

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, 6, corner_x, 1+128
HOTSPOT2 -1, corner_y, 7, corner_x, 3
HOTSPOT2 corner_x, corner_y, 8, 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
```

STATUSCODES

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

Planare Polylinien bilden mit den Statuswerten der Eckpunkte die Ausgangsbasis für viele GDL-Elemente: `POLY2_`, `POLY2_A`, `POLY2_B`, `POLY2_B{2}`, `POLY2_B{3}`, `POLY2_B{4}`, `POLY2_B{5}`, `POLY_`, `PLANE_`, `PRISM_`, `CPRISM_`, `BPRISM_`, `FPRISM_`, `HPRISM_`, `SPRISM_`, `SLAB_`, `CSLAB_`, `CROOF_`, `EXTRUDE`, `PYRAMID`, `REVOLVE`, `SWEEP`, `TUBE`, `TUBEA`

Statuscodes ermöglichen:

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

STATUSWERT-SYNTAX

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

`si = j1 + 2*j2 + 4*j3 + 8*j4 + 64*j7 [+ a_code]` , hierbei kann j jeweils 0 oder 1 sein.

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

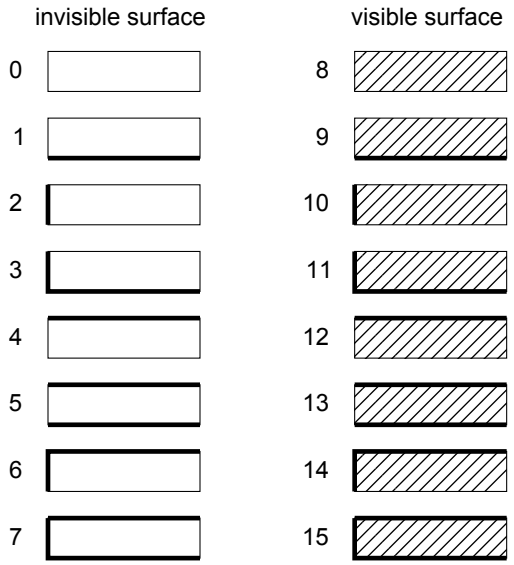
`j1:` untere horizontale Kante
`j2:` vertikale Kante
`j3:` obere horizontale Kante
`j4:` Seitenfläche
`j7:` spezieller zusätzlicher Statuswert, der nur gültig ist, wenn j2 1 ist und die vom Blickpunkt abhängige Sichtbarkeit der aktuellen vertikalen Kante kontrolliert
`a_code:` Zusätzlicher Statuscode (optional), der das Erstellen von Segmenten und Bögen in der Polylinie ermöglicht.

`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):



`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.

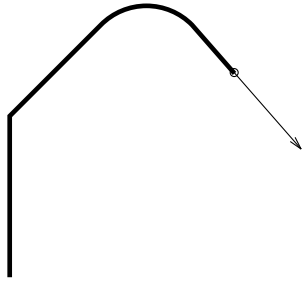
Zusätzliche Statuscodes

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

Anmerkung

Die Auflösung von Kreisbögen wird durch die im Kapitel „Anweisungen für 3D- und 2D-Scripts“ 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.

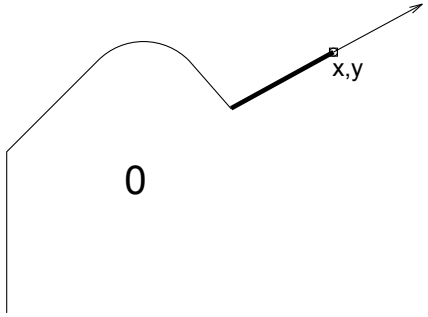
Vorgegebener erster Teil eines Polygonzuges: aktuelle Position und Tangente ist definiert



Segment, definiert durch den absoluten Endpunkt

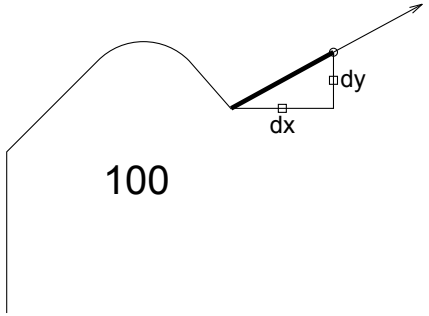
`x, y, s`

wobei 0 < s < 100



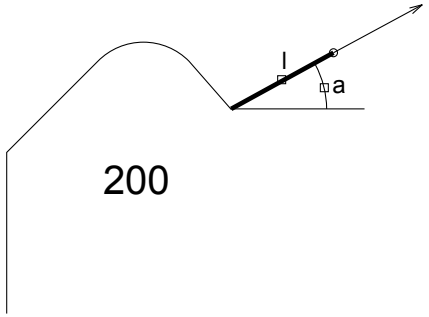
Segment, definiert durch den relativen Endpunkt

`dx, dy, 100+s,`
wobei 0 < s < 100



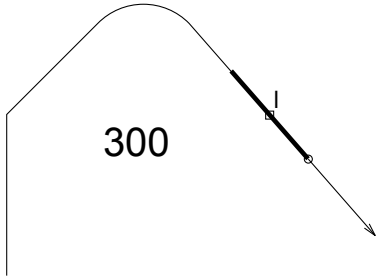
Segment, definiert durch Längen- und Richtungsangabe

`l, a, 200+s,`
wobei 0 < s < 100



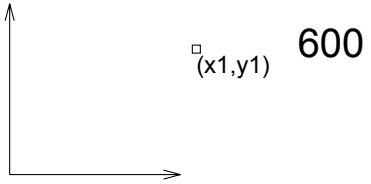
Tangentiales Segment, definiert durch Längenangabe

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



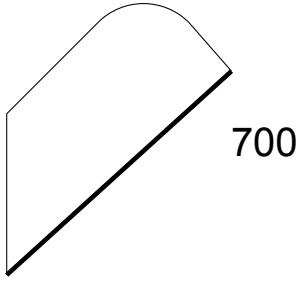
Angabe des Startpunktes

```
x1, y1, 600,
```



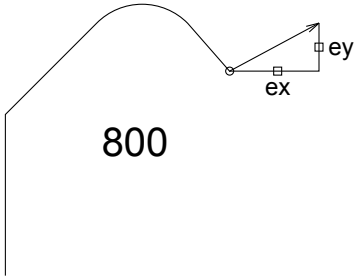
Schließen des Polygonzuges

```
0, 0, 700,
```



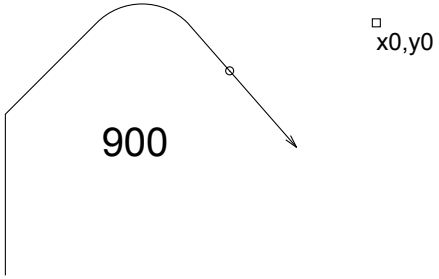
Angabe der Tangente

```
ex, ey, 800,
```



Angabe des Mittelpunktes

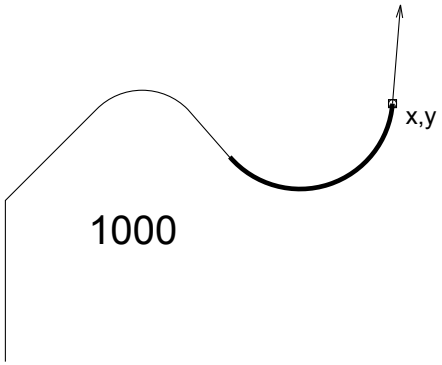
`x0, y0, 900,`



Tangentialer Bogen zum Endpunkt

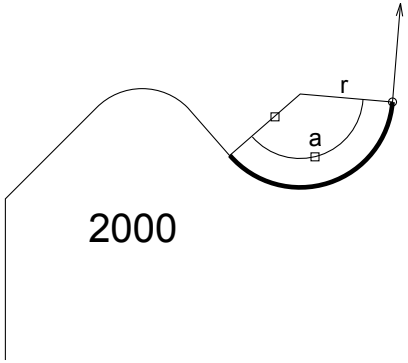
`x, y, 1000+s,`

wobei $0 < s < 100$



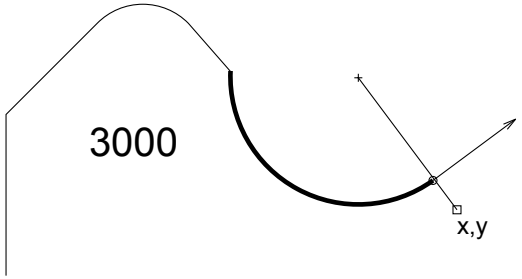
Tangentialer Bogen, definiert durch Radius und Winkel

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



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

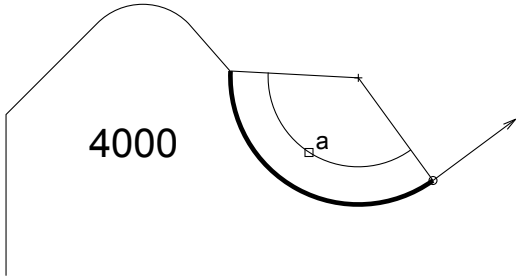
`x, y, 3000+s,`
wobei 0 < s < 100



Kreisbogen, definiert durch Mittelpunkt und Winkel

`0, a, 4000+s,`

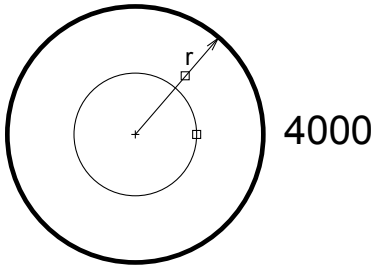
wobei 0 < s < 100



Geschlossener Kreis, definiert durch Mittelpunkt und Radius

`r, 360, 4000+s,`

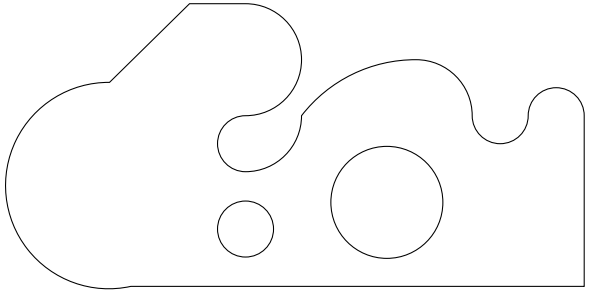
wobei 0 < s < 100

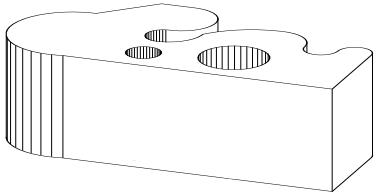


In diesem Fall bezieht sich der Status auf den ganzen Kreis.

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

Beispiel 1:

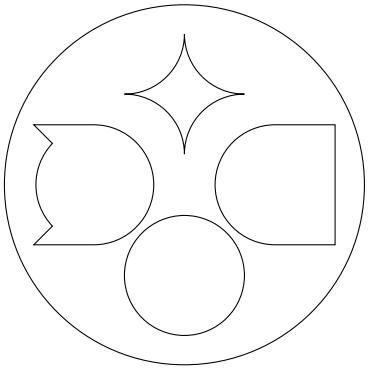




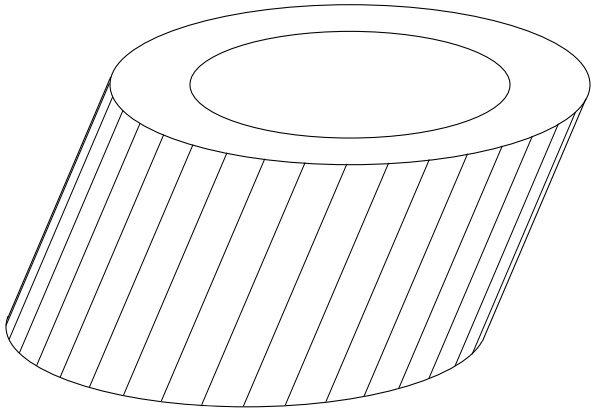
```
EXTRUDE 21, 0, 0, 3, 1+2+4+16+32,
        0, 0, 0,
        7,   0,    0,
        7,   3,    1,
        6,   3,    1000, ! tangentialer Bogen zum Endpunkt
        5,   3,    1001, ! tangentialer Bogen zum Endpunkt
        1,   90,   2000, ! tangentialer Bogen mit Radius und Winkel
        2,   3,    1001, ! tangentialer Bogen zum Endpunkt
        1,   3,    900,  ! setzt Mittelpunkt
     1,  2,   3000, ! Bogen verwendet Startpunkt, Mittelpunkt und Punkt auf endgültigem Radius
        1,   2.5,  900,  ! setzt Mittelpunkt
        0, -180, 4001,   ! Bogen verwendet Startpunkt, Mittelpunkt und Winkel
        1,   5,    1000, ! tangentialer Bogen zum Endpunkt
        -1,  0,    100,  ! Segment mit (dx, dy)
        2,   225,  200,  ! Segment mit (len, angle)
        -1,  0,    800,  ! Setzt Tangente
        -1,  0,    1000, ! tangentialer Bogen zum Endpunkt
        0,   0,    -1,   ! Ende der Kontur
        1,   1,    900,  ! setzt Mittelpunkt
        0.5, 360,  4000, ! Vollkreis mit Mittelpunkt und Radius
        3.5, 1.5,  900,  ! setzt Mittelpunkt
        1, 360, 4001     ! Vollkreis mit Mittelpunkt und Radius
```

Beispiel 2:

```
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
```

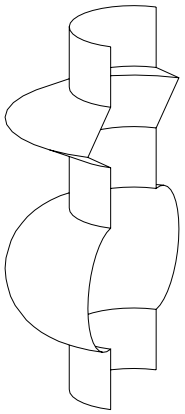


Beispiel 3:



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

Beispiel 4:



```
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, ! setzt Mittelpunkt
        0, 180, 4001, ! Bogen verwendet Startpunkt, Mittelpunkt und Winkel
        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 bei zylindrischen Elementen bestimmen, für ihren Darstellungsmodus in der 3D-Ansicht oder für die Zuweisung von Attributen (Farbe, Material, Textstil, usw.) für nachfolgende Elemente 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 Skriptes. Aufgerufene Makros übernehmen die aktuellen Einstellungen: die Änderungen wirken sich lokal aus. Nach Beendigung des Scripts werden die Werte vor dem Makro-Aufruf wiederhergestellt.

Anweisungen 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 < radius_min als Sechseck,
- wenn r >= radius_max als 36-eckiges Polygon,
- wenn radius_min < r < radius_max als ein (6+30*(r-radius_min)/(radius_max-radius_min))-eckiges Polygon.

Die Bogendarstellung erfolgt dazu analog.

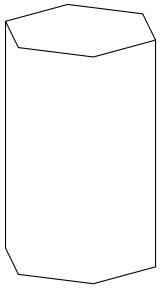
Nach einer RADIUS-Anweisung verlieren die vorhergehenden RESOL- und TOLER-Anweisungen in Wirkung.

Einschränkung der Parameter:

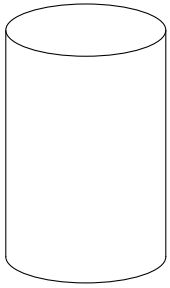
```
r_min <= r_max
```

Beispiel:

```
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 RADIUS- und TOLER-Anweisungen ihre Wirkung.

Einschränkung der Parameter:

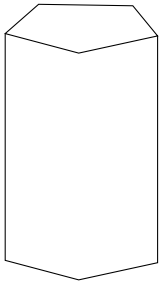
```
n >= 3
```

Grundeinstellung:

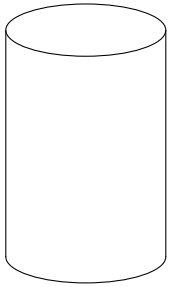
```
RESOL 36
```

Beispiel:

```
RESOL 5
CYLIND 3.0, 1.0
```



```
RESOL 36
CYLIND 3.0, 1.0
```



TOLER

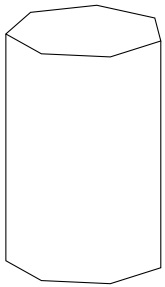
TOLER d

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.

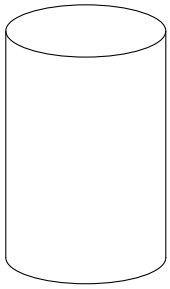
Nach einer TOLER-Anweisung verlieren die vorhergehenden RESOL- und RADIUS-Anweisungen ihre Wirkung.

Beispiel:

```
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, sowie in Polygonzügen, in denen gekrümmte Kanten verwendet werden .

Siehe „Zusätzliche Statuscodes“.

PEN

PEN n

Legt die Stiftfarbe fest.

Einschränkung der Parameter:

```
0 < n <= 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 nicht existierenden Index bezieht, so wird die PEN 1-Anweisung die Grundeinstellung.)

LINE_PROPERTY

LINE_PROPERTY expr

Definiert die Eigenschaft für alle anschließenden Linien im 2D-Script (RECT2, LINE2, ARC2, CIRCLE2, SPLINE2, SPLINE2A, POLY2, FRAGMENT2 -Befehle) bis zum nächsten LINE_PROPERTY-Ausdruck. Grundwert ist allgemein.

expr: Mögliche Werte:

- 0: alle Linien sind allgemeine Linien
- 1: alle Linien sind innen liegend
- 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 ständige Referenz auf eine Stilmenge in der internen Datenstruktur (negative Indizes bedeuten Indizes in der Datenstruktur der Inline-Materialien, die früher im GDL Script definiert wurden). 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.

Anweisungen nur für 3D-Scripts

MODEL

MODEL WIRE
MODEL SURFACE
MODEL SOLID

Bestimmt die Darstellungsform im aktuellen Script.

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

MODEL SURFACE, MODEL SOLID: 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.

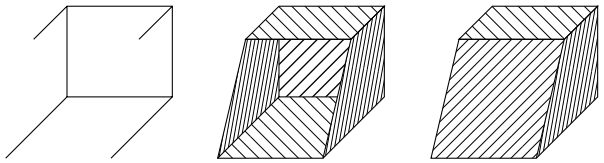
Grundeinstellung:

MODEL SOLID

Beispiel: 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_or_index

Alle nach dieser Anweisung erzeugten Oberflächen erhalten diese Materialzuweisung, bis eine neue MATERIAL-Anweisung die vorhergehende ersetzt. Die Oberflächen der Körper von BPRISM_, CPRISM_, FPRISM_, HPRISM_, SPRISM_, CSLAB_, CWALL_, BWALL_, XWALL_, CROOF_, MASS, sind von dieser Regel ausgenommen.

Der Index ist eine konstante Referenz auf eine Stilmenge in der internen Datenstruktur (negative Indizes bedeuten Indizes in der Datenstruktur der Inline-Materialien, die früher im GDL Script definiert wurden)). Diese Struktur wird während der GDL-Analyse modifiziert und kann auch innerhalb des Programms verändert werden. Die Benutzung des Index anstelle des Materialnamens 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 verwendet.)

[SET] BUILDING_MATERIAL

```
[SET] BUILDING_MATERIAL name_or_index
     [, cut_fill_pen [, cut_fill_bkgd_pen, [iOverrideFlag]]]
```

Kompatibilität: eingeführt in ARCHICAD 21.

Alle nachfolgend erzeugten Körper werden die Oberfläche, die Schnittschraffur (in Schnitt/Ansicht), die Vorder- und Hintergrundstifte des eingestellten Baustoffs darstellen.

cut_fill_pen: Benutzerdefinierter Schnittschraffur-Vordergrund-Stift-Index, um den Index des aktiven Baustoffattributs zu überschreiben

cut_fill_bkgd_pen: Benutzerdefinierter Schnittschraffur-Hintergrund-Stift-Index, um den Index des aktiven Baustoffattributs zu überschreiben

iOverrideFlag: Aktivieren Sie "cut_fill_pen" und / oder "cut_fill_bkgd_pen", um zu das zu bewirken
`iOverrideFlag = j1 + 2*j2:` , wobei jedes j 0 oder 1 sein kann.
`j1:` Überschreibt den Schnittschraffur-Vordergrundstift mit cut_fill_pen
`j2:` Überschreibt den Schnittschraffur-Hintergrundstift mit cut_fill_bkgd_pen
Überschreibungs-Parameter sind optional: Wenn das "iOverrideFlag" nicht gesetzt ist oder das Schlüsselwort DEFAULT in einem der Override-Stift-Indexparameter verwendet wird, werden die Baustoffattribute wirksam.

Beispiel: Überschreiben des Schnittschraffur-Hintergrund-Stiftes

```
BUILDING_MATERIAL buildingMatIndex, DEFAULT, cut_fill_bkgd_pen
```

Alle nachfolgend erzeugten Oberfläche repräsentieren die Oberfläche des Baustoffs bis zur nächsten BUILDING_MATERIAL-, MATERIAL-, SECT_FILL- oder SECT_ATTRS-Anweisung. Die Oberflächen der Körper von `BPRISM_`, `CPRISM_`, `FPRISM_`, `HPRISM_`, `SPRISM_`, `CSLAB_`, `CWALL_`, `BWALL_`, `XWALL_`, `CROOF_`, `MASS`, sind von dieser Regel ausgenommen.

In Schnitten/Ansichten werden die angezeigten Schnittschraffur-Vorder- und Hintergrundstifte mit den gleichen Attributen des Baustoffs übereinstimmen (oder die im Befehl selbst eingestellten Override-Parameter) bis zur nächsten Anweisung BUILDING_MATERIAL, MATERIAL, SECT_FILL oder SECT_ATTRS.

Eine vorher gesetzte BUILDING_MATERIAL-Anweisung hat keinen weiteren Einfluss auf Körper, die nach der Verwendung von SECT_FILL- oder SECT_ATTRS-Anweisungen erzeugt wurden. Körper, die nach der Verwendung der folgenden Anweisungen erzeugt

werden, behalten ihre BUILDING_MATERIAL-Einstellungen bei: die MATERIAL-Anweisung überschreibt nur die Oberflächen der erzeugten Körper, die Anweisung SECT_ATTRS {2} steuert nur die Darstellung des Konturstiftes und des Linientyps in der Schnittansicht, während der Rest der Attribute noch durch das Baustoff selbst kontrolliert wird.

Der Index ist eine Konstante, die sich auf einen Baustoffstapel in der internen Datenstruktur bezieht. Die Verwendung des Index anstelle des Baustoffnamens wird nur bei vorheriger Verwendung der Funktion `IND` empfohlen.

index 0 hat eine besondere Bedeutung: Der generierte Schnitt eliminiert die Linien auf der Grundlage der Schraffuren.

Grundeinstellung:

`BUILDING_MATERIAL 0`

falls es im Script keine BUILDING_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 Baustoff-Einstellung 0 der voreingestellte Wert verwendet.)

SECT_FILL

```
SECT_FILL fill, fill_background_pen,
        fill_pen, contour_pen
```

oder

SECT_ATTRS

```
SECT_ATTRS fill, fill_background_pen,
        fill_pen, contour_pen [, line_type]
```

Definiert die Attribute, die für den geschnittenen Teil der 3D-Elemente im Schnitt- / Ansichts-Fenster verwendet werden. *Kompatibilität: Bis ARCHICAD 19 PROJECT2{3} ist dies ebenfalls betroffen.* Inline-Schraffur- und Linientyp-Attribute (definiert im Master-Script oder 3D-Script) werden nicht akzeptiert.

`fill:` Name oder Indexnummer der Schraffur

`fill_background_pen:` Nummer des Schraffurhintergrundstifts

`fill_pen:` Nummer des Schraffurstifts

`contour_pen:` Nummer des Schraffurkonturenstifts

`line_type:` Linientyp der Polygonkanten

SECT_ATTRS{2}

```
SECT_ATTRS{2} contour_pen [, line_type]
```

Kompatibilität: eingeführt in ARCHICAD 21.

Definiert den Konturstift und den Linientyp, der für den geschnittenen Teil der 3D-Elemente in Schnitt / Ansicht verwendet wird. Kann mit der Anweisung BUILDING_MATERIAL für die Bearbeitung aller Schnitt/Ansichts-Attribute kombiniert werden. Inline-Linientyp-Attribute (definiert im Master-Script oder 3D-Script) werden nicht akzeptiert.

`contour_pen:` Nummer des Schraffurkonturenstifts

`line_type:` Linientyp der Polygonkanten

SHADOW

`SHADOW` `casting[, catching]`

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

`casting:` ON, AUTO oder 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.

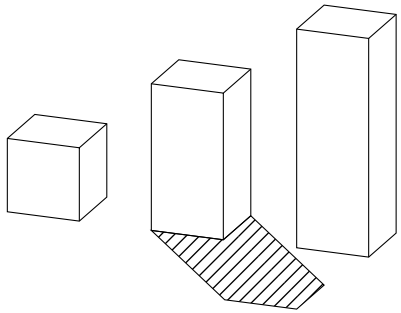
`catching:` ON oder OFF

Dieser optionale Parameter kontrolliert die Sichtbarkeit von Schatten (von anderen Körpern) auf Oberflächen.

Falls Schattenwurf nicht spezifiziert ist, ist der Defaultwert AUTO.

Beispiel:

```
SHADOW OFF
! horizontal surface
PRISM 4, 0.2,
      0, 0,
      6, 0,
      6, 6,
      0, 6

ADDX 0.5
ADDY 2.5

BRICK 1, 1, 1
ADDX 2
SHADOW ON
BRICK 1, 1, 2
ADDX 2
SHADOW OFF
BRICK 1, 1, 3

DEL 4
```



Anweisungen nur für 2D-Scripts

DRAWINDEX

DRAWINDEX number

Definiert die Darstellungsreihenfolge von 2D-Scriptelementen. 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.

[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 Linientypnamens 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.

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 dazugeladen; Attribute mit demselben 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. Im Master-Script oder 2D-Script definierte und verwendete Schraffuren und Linientypen besitzen das

selbe Verhalten als wenn sie im Script einer MASTER_GDL-Datei definiert wären, aber nur wenn genutzt mit Namen oder Index (nicht durch einen Parameter). Auf im Master-Script oder 3D-Script definierte Linientypen kann im 3D-Script nicht zugegriffen werden.

Über den Befehl GDL Script prüfen 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 GDL Script prüfen finden Sie diese Parameter.

Materialien

DEFINE MATERIAL

```
DEFINE MATERIAL name type,
        surface_red, surface_green, surface_blue
        [, ambient_ce, diffuse_ce, specular_ce, transparent_ce,
         shining, transparency_attenuation
         [, specular_red, specular_green, specular_blue,
          emission_red, emission_green, emission_blue, emission_att]]
        [, fill_index [, fillcolor_index, texture_index]]
```

Anmerkung: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe „Zusätzliche Daten“ für 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: allgemeine 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
20-27: Spezielle Bedeutungen für Typen 20-27: Ist die Stiftnummer Null, so werden die vektoriellen Schraffuren mit dem aktiven Stift erstellt. Der Null Wert für den Textur-Index ermöglicht die Definition von Materialien ohne vektorielle Schraffuren oder Texturen.

Beispiel 1: Material mit voller Farbe

```
DEFINE MATERIAL "water" 0,
        0.5284, 0.5989, 0.6167,! surface RGB [0.0..1.0]
        1.0,                   ! ambient coefficient [0.0..1.0]
        0.5,                   ! Diffuser Koeffizient[0.0..1.0]
        0.5,                   ! specular coeff. [0.0..1.0]
        0.9,                   ! transparent coeff. [0.0..1.0]
        2.0,                   ! shining [0.0..100.0]
        1,                     ! transparency atten. [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 atten. [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]
```

Beispiel 2: Material mit Schraffur

```
DEFINE MATERIAL "Brick-Red" 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")    ! Schraffur-Index
```

Beispiel 3: Material mit Schraffur und Textur

```
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")
        !      Schraffur-Index, color index, texture index
```

DEFINE MATERIAL BASED_ON

DEFINE MATERIAL name [,] **BASED_ON** orig_name [,] PARAMETERS name1 = expr1 [, ...]
[[,] **ADDITIONAL_DATA** name1 = expr1 [, ...]]

Materialdefintion 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", ...) zugegriffen werden.

orig_name: Name des Originalmaterials (Name eines vorhandenen Materials das bereits in GDL oder einem Grundriss definiert wurde)

namei: 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:` (Oberflächen-RGB [0.0..1.0])
`gs_mat_ambient:` (Streulicht Koeffizient[0.0..1.0])
`gs_mat_diffuse:` (Diffuswert Koeffizient[0.0..1.0])
`gs_mat_specular:` (Glanz Koeffizient[0.0..1.0])
`gs_mat_transparent:` (Transparenz Koeffizient[0.0..1.0])
`gs_mat_shining:` (Glanzlicht [0.0..100.0])
`gs_mat_transp_att:` (Transparenzabschwächung[0.0..4.0])
`gs_mat_specular_r, gs_mat_specular_g, gs_mat_specular_b:` (Gespiegelte Farbe RGB [0.0..1.0])
`gs_mat_emission_r, gs_mat_emission_g, gs_mat_emission_b:` (Abstrahl-Farbe RGB [0.0..1.0])
`gs_mat_emission_att:` (Abstrahlungsabschwächung[0.0..65.5])
`gs_mat_fill_ind:` (Schraffurindex)
`gs_mat_fillcolor_ind:` (Schraffurfarbenindex)
`gs_mat_texture_ind:` (Texturindex)

`expri:` neuer Wert, der den ausgewählten Parameter des Materials überschreibt. Der Wertebereich 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 werden.

name: Name der Textur

expression: Der Schraffur zugeordnetes Bild. Ein Zeichenfolgenausdruck bedeutet ein Dateiname, ein numerischer Ausdruck 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:

mask = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9, hierbei kann j jeweils 0 oder 1 sein.

Alphakanal-Kontrollen (j1... j6):

j_1: alpha Kanal ändert die Transparenz der Textur

j_2: Bump mapping (3D-Texturen) oder ermittelt die Oberflächenhöhen durch Normalen Oberflächen-Störung. Bump mapping (3D-Texturen) verwendet Alpha Kanal zur Bestimmung der Amplitude der normalen Oberfläche.

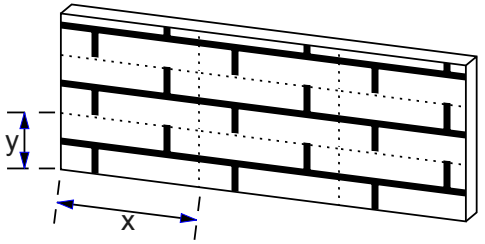
j_3: Alpha Kanal ändert die diffuse Farbe der Textur

j_4: Alpha Kanal ändert die spiegelnde Farbe der Textur

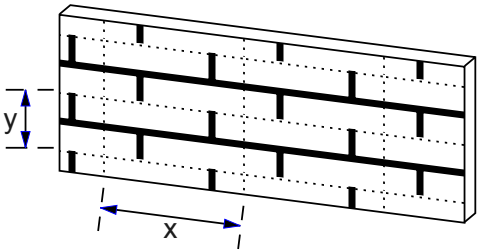
j_5: Alpha Kanal ändert die Streulichtfarbe der Textur

j_6: Alpha Kanal ändert die Oberflächenfarbe der Textur

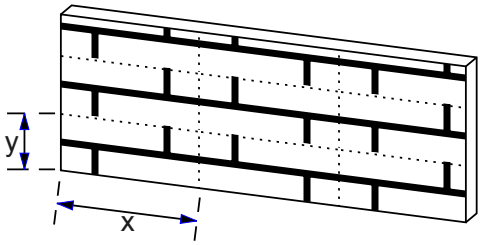
Verbindungskontrollen (j7 j9): (Ist der Wert Null, so wird der normale Modus gewählt)



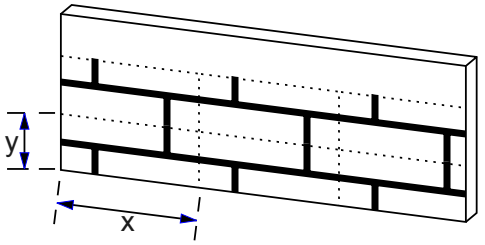
j_7: die Textur wird zufälligerweise verschoben.



`j8:` Spiegelung in die 'x'-Richtung



`j9:` Spiegelung in die 'y'-Richtung



`angle:` Winkel der Drehung

Beispiel:

```
DEFINE TEXTURE "Brick" "Brick.PICT", 1.35, 0.3, 256+128, 35.0
```

Schraffuren

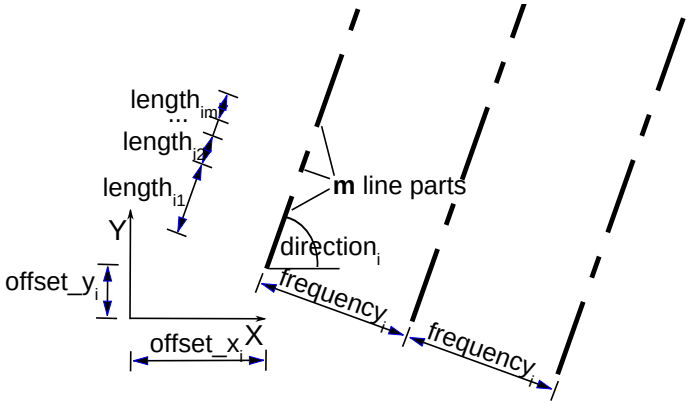
DEFINE FILL

```
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
```

Anmerkung 1: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe „Zusätzliche Daten" für Details.

In jedem GDL-Script können Sie Schraffuren definieren, die später über ihren Namen aufgerufen werden können. Die auf diese Art definierte Schraffur kann nur für 2D-Elemente in dem Script verwendet werden, in welchem sie definiert wurde; gleiches gilt für seine nachfolgenden Scripte der zweiten Generation..



name: Name der Schraffur

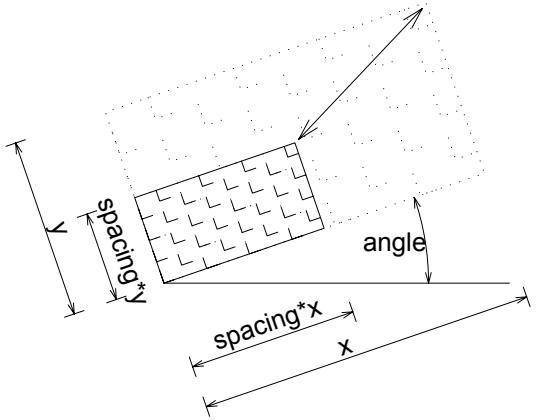
fill_types:

fill_types = j_1 + 2*j_2 + 4*j_3, hierbei kann j jeweils 0 oder 1 sein.

j_1: Bauteilschraffuren
j_2: Deckschraffuren
j_3: Zeichenschraffuren

Ist j gesetzt, kann die definierte Schraffur ihrem Schraffurtyp entsprechend verwandt werden. Grundeinstellung ist Bauteil- und Zeichnungsschraffur (5).

pattern definition: 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)

diri: 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 Ansicht / Bildschirmdarstellungsoptionen / Vektorschraffur auf aus 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 Schraffur 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 (lenghij) definiert, beschrieben.

Anmerkung 2: Mit dem Befehl DEFINE FILL können nur einfache Schraffuren definiert werden. Es gibt keine Möglichkeit zur Definition von Symbolschraffuren.

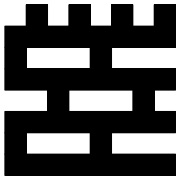
Beispiel:

```
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
```

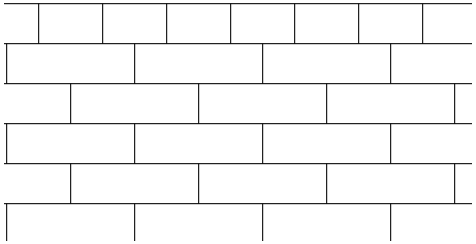
Bitmapmuster:

```
Muster:          Binärwert:
pattern1 = 85     01010101  • • • •
pattern2 = 255    11111111 ••••••••
pattern3 = 136    10001000 •   •
pattern4 = 255    11111111 ••••••••
pattern5 = 34     00100010   •   •
pattern6 = 255    11111111 ••••••••
pattern7 = 136    10001000 •   •
pattern8 = 255    11111111 ••••••••
```

Ansicht:



Vektorschraffur:

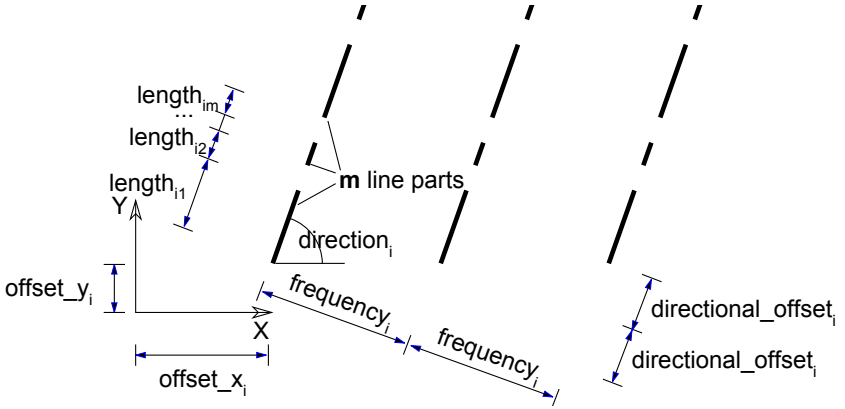


DEFINE FILLA

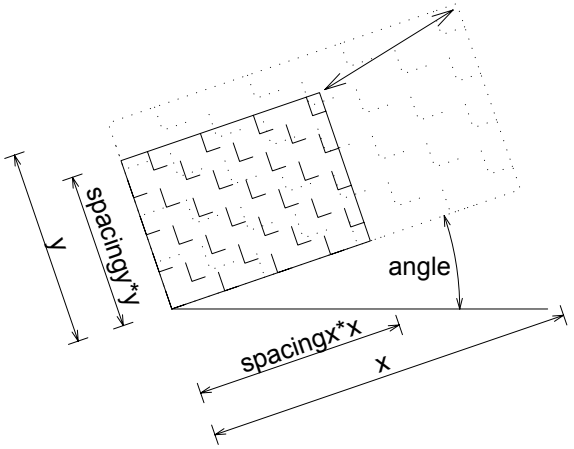
```
DEFINE FILLA name [,] [FILLTYPES_MASK fill_types,]
        pattern1, pattern2, pattern3, pattern4,
        pattern5, pattern6, pattern7, pattern8,
        spacing_x, spacing_y, angle, n,
        frequency1, directional_offset1, direction1,
        offset_x1, offset_y1, m1,
        length11, ..., length1m,
        ...
        frequencyn, directional_offsetn, directionn,
        offset_xn, offset_yn, mn,
        lengthn1, ..., lengthnm
```

Anmerkung: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe „Zusätzliche Daten“ für Details.



Ein erweiterter DEFINE FILL-Befehl.



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_offseti: Versatz des Anfangs der nächsten ähnlichen Schraffurlinie, die entlang der Linienrichtung gemessen wird. Jede Linie der Serie wird in einer durch frequencyi definierten Abstand mit einem durch offset gegebenen Versatz gezeichnet. Die wirkliche Länge des Versatzes wird durch spacing * directional_offseti definiert.

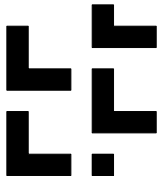
Beispiel:

```
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
```

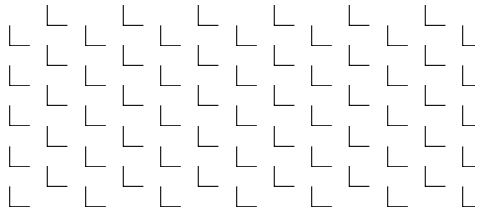
Bitmapmuster:

```
Muster:      Binärwert:
pat1 = 8      00001000      •
pat2 = 142    10001110  •   •••
pat3 = 128    10000000  •
pat4 = 232    11101000  ••• •
pat5 = 8      00001000      •
pat6 = 142    10001110  •   •••
pat7 = 128    10000000  •
pat8 = 232    11101000  ••• •
```

Ansicht:



Vektorschraffur:

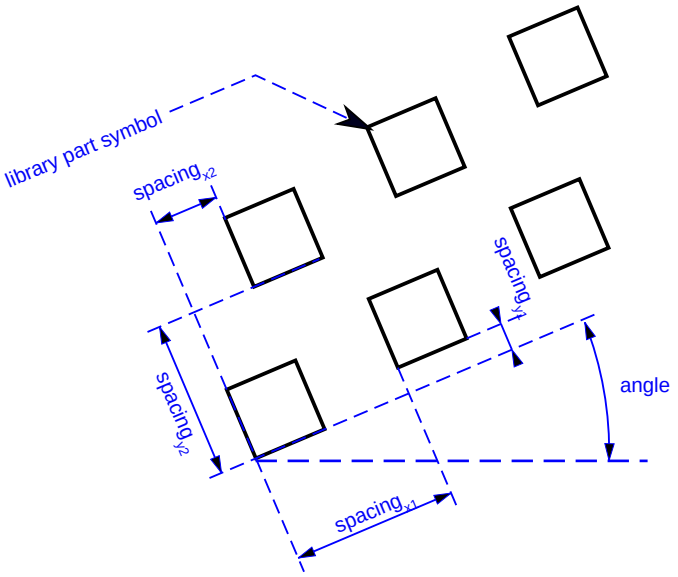


DEFINE SYMBOL_FILL

```
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]
```

Anmerkung: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe „Zusätzliche Daten“ für Details.



Eine Erweiterung zu der Anweisung DEFINE FILL, die mit einem Bibliothekselement (als Makro) erzeugte Zeichnungselemente in einer Schraffurdefinition ermöglicht. Die Verwendung von macro_name und den Parametern entspricht der beim Befehl CALL.

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 Vollschraffur.

Anmerkung: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe „Zusätzliche Daten" für Details.

DEFINE EMPTY_FILL

```
DEFINE EMPTY_FILL name [[,] FILLTYPES_MASK fill_types]
```

Definiert eine leere Schraffur.

Anmerkung: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe „Zusätzliche Daten" für Details.

DEFINE LINEAR_GRADIENT_FILL

```
DEFINE LINEAR_GRADIENT_FILL name [[,] FILLTYPES_MASK fill_types]]
```

Linearen Farbverlauf definieren.

DEFINE RADIAL_GRADIENT_FILL

```
DEFINE RADIAL_GRADIENT_FILL name [[,] FILLTYPES_MASK fill_types]]
```

Radialen Farbverlauf definieren

DEFINE TRANSLUCENT_FILL

```
DEFINE TRANSLUCENT_FILL name [[,] FILLTYPES_MASK fill_types]
        pat1, pat2, pat3, pat4, pat5, pat6, pat7, pat8,
        percentage
```

Definieren Sie eine Schraffur, die eine Mischung der Hinter-und Vordergrundfarben darstellt, welche durch den angegebenen Prozentwert definiert wird.

percentage: Volumen der Vordergrundfarbe; 0 stellt nur die Hintergrundfarbe dar (wie leere Schraffur), 100 stellt nur die Vordergrundfarbe dar (wie feste Schraffur).

DEFINE IMAGE_FILL

```
DEFINE IMAGE_FILL name image_name [[,] FILLTYPES_MASK fill_types]
        part1, part2, part3, part4, part5, part6, part7, part8,
        image_vert_size, image_hor_size, image_mask, image_rotangle
```

Definieren Sie eine Schraffur aufgrund eines Bildmusters (Bitmap).

image_name: Name der in der aktuellen Bibliothek geöffneten Musterabbildung.

image_vert_size, image_hor_size: Modellgröße des Musters.

image_mask: Art der Kachelung

image_mask = 1024*j_{11} + 2048*j_{12}, hierbei kann j jeweils 0 oder 1 sein.

Für weitere Informationen über das Layouten von Grafiken auf Oberflächen siehe DEFINE TEXTURE.

j_{11}: Spiegelung in die 'x'-Richtung
j_{12}: Spiegelung in die 'y'-Richtung

image_rotangle: Drehwinkel des Musters in Bezug zu dem normalen Koordinatensystem.

Linientypen

DEFINE LINE_TYPE

```
DEFINE LINE_TYPE name spacing, n,
        length1, ..., lengthn
```

Anmerkung 1: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe „Zusätzliche Daten" für 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, wird als Linientyp in die Grundrissattribute des Projektes aufgenommen und steht allen Dialogen und Objekten nach der Abarbeitung des Skriptes zur Verfügung.

name: Name des Linientyps

spacing: Faktor Zwischenraum

n: Anzahl der Schraffurelemente

lengthi: Länge der einzelnen Schraffurelemente (die wirkliche Länge ist spacing * lengthi). Linienelemente bestehen aus aufeinanderfolgenden Segmenten und Leerräumen. Das erste Schraffurelement ist ein Strich, die Länge Null kennzeichnet einen Punkt.

Anmerkung 2: Nur einfache Linientypen - d.h. bestehend ausschließlich aus Segmenten und Zwischenräumen - können mit diesem Befehl definiert werden; die Definition von Symbollinien erfolgt mit DEFINE SYMBOL_LINE.

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]
```

Anmerkung: Dieser Befehl kann zusätzliche Datendefinitionen enthalten.

Siehe „Zusätzliche Daten" für Details.

Eine Erweiterung zu der Anweisung DEFINE LINE_TYPE, die mit einem Bibliothekselement (als Makro) erzeugte Zeichnungselemente in einer Liniendefinition ermöglicht. Die Verwendung von macro_name und den Parametern entspricht der beim Befehl CALL.

dash: Skalierung beider Linienkomponenten

gap: Abstand zwischen den einzelnen Komponenten

Text und Stile

DEFINE STYLE

```
DEFINE STYLE name font_family, size, anchor, face_code
```

Empfohlen bei der Verwendung der Befehle TEXT2 undTEXT.

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

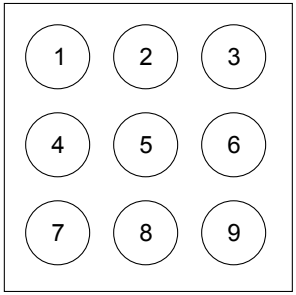
font_family: Name der verwendeten Schriftfamilie (z.B. Geramont).

size: Größe des Buchstabens "I" in mm im Papierraum 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 dem Absatzbefehl PARAGRAPH in den Befehlen RICHTEXT2 und RICHTEXT hängt die Höhenangabe in mm oder m vom festgelegten Höhenparameter (fixed_height) der TEXTBLOCK-Definition ab. Gleichzeitig sind Textstile für Kontur (outline) und Schatten nicht wirksam.

anchor: Code des Positionspunktes im Text



face_code: eine Kombination der folgenden Werte:

face_code = j_1 + 2*j_2 + 4*j_3, hierbei kann j jeweils 0 oder 1 sein.

j_1: fett

j_2: kursiv

j_3: unterstrichen

Falls der face_code = 0, dann ist der Textstil normal.

DEFINE STYLE{2}

DEFINE STYLE{2} name font_family, size, face_code

Neue Version der Stildefinition, die mit PARAGRAPH benutzt werden sollte.

name: Name des Stils

font_family: Name der verwendeten Schriftfamilie (z.B. Geramont).

size: Größe der Buchstaben in mm oder m im Modellraum.

face_code: eine Kombination der folgenden Werte:

face_code = j_1 + 2*j_2 + 4*j_3 + 32*j_6 + 64*j_7 + 128*j_8, hierbei kann j jeweils 0 oder 1 sein.

j_1: fett

j_2: kursiv

j_3: unterstrichen

j_6: hochgestellt
j_7: tiefergestellt
j_8: durchgestrichen

Falls der face_code = 0, dann ist der Textstil normal.

Falls der definierte Stil mit TEXT2 verwendet wird, bedeutet die Größe die Buchstabenhöhe in Millimetern, wobei die Facecode-Werte für die hochgestellte, tiefgestellte und durchgestrichene Schrift nicht wirksam sind. Bei Verwendung mit dem Absatzbefehl PARAGRAPH in den Befehlen RICHTEXT2 und RICHTEXT hängt die Höhenangabe in mm oder m vom festgelegten Höhenparameter (fixed_height) der TEXTBLOCK-Definition ab. Gleichzeitig sind Textstile für Kontur (outline) und Schatten nicht wirksam.

PARAGRAPH

```
PARAGRAPH name alignment, firstline_indent,
        left_indent, right_indent, line_spacing [,
        tab_position1, ...]
    [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. Zeilenumbrüche, die in ein Absatz-String aufgenommen werden (durch das Sonderzeichen '\n'), unterteilen den String automatisch in identisch formatierte Absätze, die jeweils eine Zeile enthalten. Im Befehl TEXTBLOCKkann

namentlich auf Absatzdefinitionen verweisen. 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. Kann entweder String oder Ganzzahl sein. Ganzzahl-Identifizierer funktionieren nur mit TEXTBLOCK_

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 Absatzs-Strings springen an diese Position. Sind keine Tabulatorpositionen angegeben, werden die Grundwerte benutzt (12.7 mm). Funktioniert nur mit der speziellen Zeichenfolge '\t'.

stringi: Teil des Textes. Kann entweder ein konstanter String oder ein Parameter vom String-Typ sein..

TEXTBLOCK

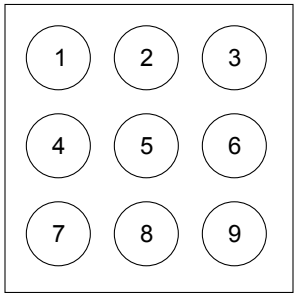
```
TEXTBLOCK name width, anchor, angle, width_factor, charspace_factor, fixed_height,
        'string_expr1' [, 'string_expr2', ...]
```

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 und RICHTEXT ausgegeben werden können. Der Befehl ("TEXTBLOCK_INFO", ...) zeigt Informationen zur berechneten Höhe und Breite des Textblocks an.

name: Name des Textblocks, vom Typ String.

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: Die Buchstabenbreite des aktuellen Stils wird um diese Zahl multipliziert.

charspace_factor: Die horizontale Entfernung zwischen den Buchstaben wird mit dieser Zahl multipliziert.

fixed_height: 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 sind als m im Modellraum.

string_expri: bedeutet Absatzname, wenn dieser vorher definiert wurde, sonst einen einfachen String (mit Standardabsatzparametern).

TEXTBLOCK_

```
TEXTBLOCK_ name width, anchor, angle, width_factor, charspace_factor, fixed_height, n,
        'expr_1' [, 'expr_2', ..., 'expr_n']
```

Gleicht dem TEXTBLOCK-Befehl. Die Bedeutung von allen Parametern ist die gleiche, mit folgenden Zusätzen:

expr_i: Paragraph-Namen können entweder vom Typ String oder Ganzzahl sein, innerhalb eines Textblocks.

n: Anzahl der gelisteten expr_i Namen

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 (Namei) und einen Wert (Werti), der ein Ausdruck eines beliebigen Typs sein kann, auch ein Array. Wenn ein String Parametername mit dem Teilstring "_file" endet, wird sein Wert als Dateiname betrachtet und in das Archiv-Projekt eingefügt. Unterschiedliche Bedeutungen von "additional data" können von der ausführenden Anwendung definiert und verwendet werden.

Zusätzliche Datendefinitionen sind bei den folgenden Befehlen verfügbar:

```
DEFINE MATERIAL parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE MATERIAL name [,] BASED_ON orig_name [,] PARAMETERS name1 = expr1 [, ...]
        [[,] ADDITIONAL_DATA name1 = expr1 [, ...]]
DEFINE FILL parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE FILLA parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE SYMBOL_FILL parameters
        [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE SOLID_FILL name [[,] FILLTYPES_MASK fill_types]
        [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE EMPTY_FILL name [[,] FILLTYPES_MASK fill_types]
        [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE LINEAR_GRADIENT_FILL name [[,] FILLTYPES_MASK fill_types]
        [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE RADIAL_GRADIENT_FILL name [[,] FILLTYPES_MASK fill_types]
        [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE TRANSLUCENT_FILL name [[,] FILLTYPES_MASK fill_types]
        pat1, pat2, pat3, pat4, pat5, pat6, pat7, pat8,
        percentage [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE IMAGE_FILL name image_name [[,] FILLTYPES_MASK fill_types]
        part1, part2, part3, part4, part5, part6, part7, part8,
        image_vert_size, image_hor_size, image_mask, image_rotangle
        [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE LINE_TYPE parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
DEFINE SYMBOL_LINE parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]
```

EXTERNE DATEIABHÄNGIGKEIT

FILE_DEPENDENCE

```
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). Dieser Befehl funktioniert nur auf dieser Ebene: Sind die angegebenen Dateien selbst Bibliothekselemente, werden deren aufgerufene Makros nicht mit eingeschlossen.

Dieser Befehl ist nützlich, wenn im GDL-Script mit benutzerdefinierten Variablen oder Parametern auf externe Dateien verwiesen wird, z.B: Dateiparameter mit ADDITIONAL_DATA oder Datendateien in Dateioperationen.

NICHT GEOMETRISCHE SCRIPTS

Neben den 3D- und 2D-Scripts, die das Erscheinen des GDL-Objektes definieren, sind weitere Scripts zum Hinzufügen von ergänzenden Informationen verfügbar. Dieses sind das Eigenschaften-Script (verwendet für Mengenermittlungen), das Parameter-Script (enthält die Listen der möglichen Werte der verschiedenen Parameter), das Interface-Script (dient der Erzeugung einer individuellen Schnittstelle für die Parametereingabe), das Vorwärts-Migration Script und Rückwärts-Migration Script (um alte Instanzen auf den aktuellen Stand zu migrieren oder um eine Element auf den Stand einer älteren Instanz zurück zu migrieren). Die Befehle für alle diese 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. Durch Verwendung weniger Befehle ist es möglich, im Script lokale Beschreibungen und Komponenten zu definieren. Beschreibungen und Bestandteile können auch von externen Datenbanken bezogen werden. Die Länge des Codes darf 32 Zeichen nicht überschreiten.

Im Eigenschaften-Script können Sie alle GDL-Befehle verwenden, die graphischen Elemente 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 Parameter set_name als Anweisung 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. Alle diese Namen werden in den lokalisierten ARCHICAD-Versionen übersetzt.

Scripts 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. Scripts 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 abgeschnitten. Wenn Sie einen längeren Text benötigen, verwenden Sie mehrere DESCRIPTORs.

code: Text, definiert einen Code für die Beschreibungen

keycode: Text, Bezug auf einen Key in externer Datenbank.

Der Key wird der Beschreibung zugewiesen.

REF DESCRIPTOR

REF DESCRIPTOR code [, keycode]

Bezug durch einen code und keycode-Text auf eine Beschreibung in einer externer Datenbank. COMPONENT .

COMPONENT

COMPONENT name, quantity, unit [, proportional_with, code, keycode, unitcode]

Definition des lokalen Bestandteiles. Scripte können eine 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: Text, 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 der Menge des Bestandteils kontrolliert. Dadurch wird der lokale definierte Einheitentext ersetzt.

REF COMPONENT

REF COMPONENT code [, keycode [, numeric_expression]]

Bezug durch einen code oder keycode Text 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 mehre 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 mehre 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 die Beschreibungen und Komponenten bei ihren Standardelementen aufgelistet.

position_keyword: Folgende sind die Codewörter:

```
WALLS
COLUMNS
BEAMS
DOORS
WINDOWS
OBJECTS
CEILS
PITCHED_ROOFS
LIGHTS
HATCHES
ROOMS
MESHES
```

Eine Zuweisung bleibt für alle erfolgreichen DESCRIPTORs und COMPONENTs nur solange gültig, bis die nächste Anweisung zugewiesen wird. Ein Script kann eine beliebige Anzahl von Zuweisungen beinhalten.

Beispiel:

```
DESCRIPTOR "\tPainted box.\n\t Properties:\n\
\t\t - swinging doors\n\
\t\t - adjustable height\n\
\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 ihrer 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 Parameter-Script des Bibliothekselements, in dem ARCHICAD_LibraryMaster-Objekt oder im MASTER_GDL-Script angewandt werden. Die Typenkompatibilität wird vom GDL-Compiler überprüft.

Das Parameterscript wird jedesmal interpretiert, wenn ein Parameter mit Werteliste geändert werden muss. Die möglichen im Script definierten Werte erscheinen in einem Popup-Menü. Für numerische Parameter können Popup-Menü-Elementwerte als String definiert werden mit Hilfe des Befehls VALUES{2} .

VALUES

```
VALUES "parameter_name" [,]value_definition1 [, value_definition2, ...]
VALUES "fill_parameter_name" [[,] FILLTYPES_MASK fill_types], value_definition1
   [, value_definition2, ...]
VALUES "profile_parameter_name" [[,] PROFILETYPES_MASK profile_types], value_definition1
            [, value_definition2, ...]
```

Definiert eine Wertebeschränkung für einen Parameter (außer bei Dictionary-Typen). Der Befehl besitzt eine spezielle Syntax für Schraffurtyp und Profiltypparameter. Falls in Array-Parametern verwendet, bezieht sich die Einschränkung auf alle Array-Elemente einzeln.

parameter_name: Name eines existierenden Parameters

fill_parameter_name: Name eines existierenden Parameters vom Typ Schraffurmuster

fill_types:

fill_types = j_1 + 2*j_2 + 4*j_3, hierbei kann j jeweils 0 oder 1 sein.

j_1: Bauteilschraffuren

j_2: Deckschraffuren

j_3: Zeichenschraffuren

Nur verwendbar für Parameter vom Typ Schraffurmuster. Das Schraffur-Popup für diesen Parameter enthält nur diejenigen Typen von Schraffuren, welche von den auf 1 gesetzten Bits angegeben werden. Grundeinstellung ist Bauteil- und Zeichnungsschraffur (5).

profile_parameter_name: Name eines existierenden Profiltyp-Parameters

profile_types:
profile_types = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5, hierbei kann j jeweils 0 oder 1 sein.
j_1: Wand,
j_2: Träger,
j_3: Stütze,
j_4: Geländer,
j_5: andere.

Kann nur für Profiltypparameter eingestellt werden. Die Werteliste für jeden Profiltyp-Parameter enthält automatisch alle vorhandenen Profile der Plandatei, es wird keine individuelle VALUES-Definition benötigt. Die Verwendung von VALUES ohne Maskierung (0) hat genau das gleiche Ergebnis. Bei Verwendung von VALUES mit Maskierung kann die Werteliste filtern, wobei nur die entsprechenden Profile der Bits auf 1 gesetzt werden. Einzelne Profilindizes können auch als reguläre VALUES-Definitionen aufgelistet werden.

value_definitioni: Werte-Definition, kann sein:
expression: numerisch oder Stringausdruck, oder
CUSTOM: Schlüsselwort, das bedeutet, dass ein beliebiger benutzerdefinierter Wert eingegeben werden kann, oder
RANGE: Bereichsdefinition mit optionaler Schrittweite

RANGE left_delimiter [lower_limit], [upper_limit]right_delimiter [STEP step_start_value, step_value]

left_delimiter: [, größer-gleich >=, oder (, größer >; lower_limit: untere Grenze; upper_limit: obere Grenze; right_delimiter:], kleiner-gleich<=, oder), kleiner <; step_start_value: Startwert; step_value: Schrittwert.

VALUES{2}

VALUES{2} "parameter_name" [,]num_expression1, description1,
[, num_expression2, description2, ...]
VALUES{2} "parameter_name" [,]num_values_array1, descriptions_array1
[, num_values_array2, descriptions_array2, ...]

parameter_name: Name eines existierenden Parameters vom Typ Winkel, Länge, Fliesskommazahl, Ganzzahl

num_expressioni, num_values_arrayi: einfache Wertedefinition für einen numerischen Parameter oder einen Array-Parameter mit multiplen numerischen Werten. Nur verfügbar für VALUES{2}.

descriptioni, descriptions_arrayi: Beschreibungstext (String) des numerischen Wertes i oder Array-Ausdruck mit multiplen Beschreibungs-Strings definiert durch die num_values_arrayi (Die Array-Dimension muss passend sein). Nur verfügbar für VALUES{2}.

Beispiel 1: Einfache Werteliste

```
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 2: Das Lesen aller String-Werte einer Datei und deren Verwendung in einer Werteliste

```
DIM sarray[]
! Datei in der geladenen Bibliothek, enthält
die Parameterdaten= "ProjectNotes.txt
ch1 = OPEN ("text", filename, "MODE=RO, LIBRARY")
i = 1
j = 1
sarray[1] = ""
! sammelt alle 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 entnommen wurden
VALUES "RefNote" sarray
```

PARAMETERS

```
PARAMETERS name1 = expression1 [,
        name2 = expression2, ...,
        namen = expressionn]
```

namei: 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 Variable GLOB_MODPAR_NAME (Typ Text) den Namen des letzten vom Anwender geänderten Parameters.

LOCK

LOCK "name1" [, "name2", ..., "namen"]

Sperrt die benannten Parameter im Einstellungs-Dialog. Ein geschützter Parameter erscheint abgeblendet im Dialogfeld und sein Wert kann nicht vom Anwender geändert werden.

namen: String-Ausdruck, Name des zu sperrenden Parameters.

Kompatibilität: beginnend mit ARCHICAD 22, wurde das Sperren / Ausblenden ausgewählter ARCHICAD-Schnittstellensteuerelemente erweitert.

Das erweiterte Feature kann mit der Einstellung "Ausblenden/ Sperren spezifischer optionaler Parametern mit festem Namen aktivieren" aktiviert werden (siehe Dialog "Details / Kompatibilitätsoptionen" des Objekts im Editor des Bibliothekselements). Diese erweitere Auswahl enthält optionale Parametern mit festem Namen im Zusammenhang mit:

- Standardsteuerelemente für die Textverarbeitung im Dialogfeld "Textstil" - siehe „Parameter für die Textverarbeitung“,
- erweiterte Etikett-Stilsteuerelemente im Dialogfeld "Textstil" im Etiketten-Werkzeug - siehe „Parameter für Etiketten“,
- und ausgewählte Steuerelemente von Etiketten-Zeigern des Dialogfelds "Zeiger" - siehe „Parameter für Etiketten“.

LOCK ALL ["name1" [, "name2", ..., "namen"]]

Sperrt alle Parameter im Einstellungsdialog mit Ausnahme von denen, die nach dem Schlüsselwort ALL folgen.

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.

namen: String-Ausdruck, Name des zu verbergenden Parameters.

Kompatibilität: beginnend mit ARCHICAD 22, wurde das Sperren / Ausblenden ausgewählter ARCHICAD-Schnittstellensteuerelemente erweitert. Weitere Details hierzu, siehe LOCK.

HIDEPARAMETER ALL ["name1" [, "name2", ..., "namen"]]

Verbirgt alle Parameter und deren Kind-Parameter im Einstellungsdialog mit Ausnahme von denen (und deren Kinder), welche nach dem Schlüsselwort ALL folgen.

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. Der grundeingestellte Titel heißt: 'Individuelle Einstellungen'. Die Größe des verfügbaren Bereichs ist jetzt auf 444 x 296 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 [, parent_id, page_title [, image]]`

Seitenanweisung, definiert die Seite, auf der die Schnittstellenelemente platziert werden. Die Standardseitennummerierung beginnt mit 1, es ist aber jede Startnummer verwendbar. Wenn das Schnittstellenscript keinen UI_PAGE-Befehl enthält, wird jedes Element standardmäßig auf der ersten Seite platziert. Die Bewegung zwischen Seiten kann auf unterschiedliche Arten ausgeführt werden:

- Der einfachste Weg ist, das von ARCHICAD erledigen zu lassen: drücken Sie im Objekt-Editor im User Interface Script auf den Button "Hierarchische Seiten" und tragen die optionalen Parameter des UI_PAGE-Befehls ein. In diesem Fall wird die (Seitennummer) page_number, welche im Seitenbaum ausgewählt wird mittels des Parameters gs_ui_current_page an das Bibliothekselement übergeben. . Es besteht keine Notwendigkeit, eine Value-Liste für die Seitennummer-Parameter anzulegen: ARCHICAD sammelt und sortiert alle gültigen Seiten-IDs aus den Parametern des UI_PAGE-Befehls, indem sie das UI-Skript des Objekts vorher durchlaufen.
- Eine andere Methode ist es, zwei Buttons zu verwenden, welche mit den Befehlen UI_NEXT und UI_PREV erzeugt werden, und diese auf jeder Seite zu platzieren um die Werte des Parameters "gs_ui_current_page" zu beeinflussen. Für weitere Details, siehe UI_BUTTON.
- Falls die neue hierarchische Seitennavigation nicht benötigt wird, um dynamischen Seitenaufbau zu erzeugen, verwenden Sie UI_INFIELD{3}. Erzeugen Sie eine Value-Liste für den Parameter "gs_ui_current_page" und platzieren Sie ein Popup unter Verwendung dieser Values auf jeder Seite.

page_number: die Seitennummer, eine positive Ganzzahl. Die folgenden Interface-Elemente werden auf dieser Seite platziert.

parent_id: positive Ganzzahl, die Eltern-ID der Seite Der spezielle Wert -1 bedeutet Wurzel-Elternelement. Nur dann wirksam, wenn "Hierarchische Seiten" aktiviert ist.

page_title: Titelstring der Seite, erscheint oben auf der Seite und in dem der Baumstruktur des Popups der Seiten. Nur dann wirksam, wenn "Hierarchische Seiten" aktiviert ist.

image: Dateiname, oder Indexnummer eines im Objekt gespeicherten Bitmaps. Falls festgelegt und nicht leer oder 0, wird das entsprechende Icon der Seite zugewiesen, welche oben auf der Seite und in der Baumstruktur des Seitenpopups angezeigt wird, neben dem Titel. Nur dann wirksam, wenn "Hierarchische Seiten" aktiviert ist.

Warnung: 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. Diese Einschränkung kann umgangen werden unter der Verwendung von UI_CURRENT_PAGE.

UI_CURRENT_PAGE

UI_CURRENT_PAGE `index`

Angabe, welche Benutzerschnittstellenseite angezeigt werden soll.

Warnung: Springt es auf eine nicht-existierende Seite, wird eine neue Seite ohne Schaltflächen eingefügt, so dass es nicht möglich ist, von dieser Seite auf eine andere Seite zu blättern.

index: gültiger Index einer mit UI_PAGE definierten Seite

UI_BUTTON

UI_BUTTON `type, text, x, y [, width, height, id [, url]]`

Schaltflächendefinition auf der aktuellen Seite. Schaltflächen können verschiedene Funktionen übernehmen: Seitenwechsel, Öffnen einer Internetseite mit einem externen Browser oder eine im Parameterscript festgelegte Aktion. Die Schaltflächen können Text enthalten.

type: Typ der Schaltfläche:

`UI_PREV:` Wechselt nach dem Drücken zur vorherigen Benutzerschnittstellenseite.

`UI_NEXT:` Wechselt nach dem Drücken zur nächsten Benutzerschnittstellenseite.

`UI_FUNCTION:` Wenn die Schaltfläche gedrückt wird, wird die globale Variable GLOB_UI_BUTTON_ID auf den in id spezifizierten Wert gesetzt.

`UI_LINK:` öffnet nach dem Drücken der Schaltfläche den externen Standardbrowser an der in Url angegebenen Stelle.

text: der Text, welcher auf einer Schaltfläche vom Typ Text erscheinen soll.

x, y: die Position der Schaltfläche

width, height: Breite und Höhe in Pixel. Falls (aus Kombatibilitätsgründen) nicht spezifiziert, ist der Default-Wert für Breite und Höhe 60 x 20 Pixel.

id: Eindeutige, ID-Nummer vom Typ integer

url: Text, der eine gültige URL enthält.

UI_PREV und UI_NEXT sind deaktiviert, wenn es keine folgende oder vorherige Seite gibt. Wenn diese Schaltflächen gedrückt werden, wird, sofern in der Parameterliste des Objektes vorhanden, der Wert der Variablen gs_ui_current_page auf die angezeigte Seite gesetzt.

Beispiel:

```
! UI Script
UI_CURRENT_PAGE gs_ui_current_page
UI_BUTTON UI_FUNCTION, "Go to page 9", 200,150, 70,20, 3
UI_BUTTON UI_LINK, "Visit Website", 200,180, 100,20, 0,
        "http://www.graphisoft.com"
! parameter script
if GLOB_UI_BUTTON_ID = 3 then
        parameters gs_ui_current_page = 9, ...
endif
```

UI_PICT_BUTTON

UI_PICT_BUTTON type, text, picture_reference,
x, y, width, height [, id [, url]]

Gleicht dem UI_BUTTON-Befehl. Diese Typen von Schaltflächen können jedoch Bitmapbilder enthalten.

picture_reference: Dateiname oder Indexnummer des im Bibliothekselement gespeicherten Bildes. Der Index 0 bezieht sich auf das Vorschaubild des Bibliothekselements. Pixeltransparenz ist bei den Bildern erlaubt.

text: hat keinen Effekt auf die Bild-Schaltfläche.

UI_SEPARATOR

UI_SEPARATOR x1, y1, x2, y2

Erzeugt ein Trennungsrechteck. Das Rechteck kann zu einer vertikalen (x1 = x2) oder horizontalen (y1 = y2) Trennlinie werden.

x1, y1: Koordinaten des Ausgangspunkts oben links (Koordinaten des Startpunktes der Linie)

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 Trennungslinie. Sie kann dazu verwendet werden, logisch miteinander verbundene Parameter optisch als Gruppe darzustellen.

text: der Titel des Gruppenfeldes

x, y: die Position der linken oberen Ecke

width, height: Breite und Höhe in Pixeln.

UI_PICT

UI_PICT `picture_reference, x, y [, width, height [, mask]]`

Bildelement im Dialogfenster. Die Bilddatei muss sich in einer der geladenen Bibliotheken befinden.

picture_reference: Dateiname oder Indexnummer des im Bibliothekselement gespeicherten Bildes. Der Index 0 bezieht sich auf das Vorschaubild des Bibliothekselements.

x, y: Position der oberen linken Ecke des Bildes.

width, height: Breite und Höhe in Pixeln standardmäßig werden die Originalwerte für Breite und Höhe des Bildes verwendet.

mask: alpha + verzerrung

Siehe PICTURE-Befehl für weitere Erläuterungen.

UI_STYLE

UI_STYLE `fontsize, face_code`

Alle UI_OUTFIELDs und UI_INFIELDs, 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 DEFINE STYLE, aber die Werte können nicht miteinander kombiniert werden.

- `0:` normal
- `1:` fett
- `2:` kursiv

4: unterstrichen

UI_OUTFIELD

UI_OUTFIELD expression, x, y [, width, height [, flags]]

Erzeugt ein statisches Textfeld.

expression: numerischer oder Zeichenfolge-Ausdruck

x, y: Position der linken oberen Ecke des Textblocks

width, height: Breite und Höhe des Textfeldes in Pixel. Falls leer gelassen, legt sich die Textbox so dicht wie möglich um den Text des vorgegebenen Schrifttyps herum.

flags:

flags = j_1 + 2*j_2 + 4*j_3, hierbei kann j jeweils 0 oder 1 sein.

j_1: Horizontale Ausrichtung des Textes (mit j2),

j_2: Horizontale Ausrichtung des Textes (mit j1):

j1 = 0, j2 = 0: Ausrichtung linksbündig (Standard)

j1 = 1, j2 = 0: Rechtsbündig an der rechten Blockkante

j1 = 0, j2 = 1: Mittige Ausrichtung innerhalb des Blocks

j1 = 1, j2 = 1: unbenutzt

j_3: Ausgegrauter Text

UI_INFIELD

```
UI_INFIELD "name", x, y, width, height [,
        method, 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 [,
        method, picture_name,
        images_number,
        rows_number, cell_x, cell_y,
        image_x, image_y,
        expression_image1, text1,
        ...
        expression_imagen, textn]
```

UI_INFIELD{3}

```
UI_INFIELD{3} name, x, y, width, height [,
        method, picture_name,
        images_number,
        rows_number, cell_x, cell_y,
        image_x, image_y,
        expression_image1, text1, value_definition1,
        ...
        [picIdxArray, textArray, valuesArray,
        ...]
        expression_imagen, textn, value_definitionn]
```

UI_INFIELD{4}

```
UI_INFIELD{4} "name", x, y, width, height [,
        method, picture_name,
        images_number,
        rows_number, cell_x, cell_y,
        image_x, image_y,
        expression_image1, text1, value_definition1,
        ...
        [picIdxArray, textArray, valuesArray,
        ...]
        expression_imagen, textn, value_definitionn]
```

Erzeugt ein Textbearbeitungs- oder Popup-Menü für die Parametereingabe. Ein Popup wird erzeugt, wenn der Parameter eine Werteliste (Parameterscript) besitzt oder vom Typ Material, Schraffur, Linientyp oder Stiftfarbe ist.

Wenn die optionalen Parameter des Befehls verwendet werden, können die Wertelisten alternativ als Miniaturansichten angezeigt werden. Verschiedene Steuerungsmöglichkeiten für die Miniaturansichten sind verfügbar. Sie zeigen die spezifizierten Bilder und assoziierten Text an und erlauben die Auswahl genau eines einzelnen Elements, genau wie in einem Pop-up-Menü.

In den Versionen 1 und 2 des UI_INFIELD-Befehls sind die Miniaturansichten und die Wertelistenelemente durch Indizes verknüpft.

In Version 3 und 4 definiert das Eingabefeld Verbindungen mit Werten, welche die Miniaturansicht mit den Wertelistenelementen des assoziierten Parameters verknüpft. Ist ein Wert, der zu einer Miniaturansicht definiert wird in der Werteliste nicht vorhanden, wird er in dem Eingabefeld nicht dargestellt. Arrays mit identischer Größe können ebenfalls für Definitionen von Zeilen verwendet werden.

Das Schnittstellenscript wird mit dem neuen Wert neu aufgebaut, nachdem ein Parameter geändert wurde.

`name:` Parametername als Zeichenfolge-Ausdruck für alle 4 Befehlsvarianten, mit Parameternamesoption für UI_INFIELD{2} und UI_INFIELD{3} und Parametername als Text-Array-Wertelisten-Option für UI_INFIELD{4}.

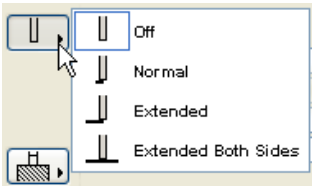
`x, y:` Die Postion des editierbaren Textes, Popups oder Steuerelementes

`width, height:` Breite und Höhe in Pixeln.

`method:` Die Methode des Eingabefeldes mit Miniaturdarstellungen

`1:` Listenansicht

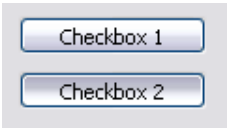
`2:` Steuerungen des Popup-Menüs



3: Popup-Icon Radio-Button (Pfeil auf Bild).

4: Knopf-icon mit Radio-Button

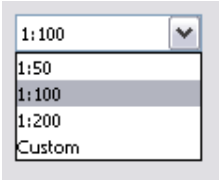
5: Pushbutton mit Text



6: Pushbutton mit Bild

7: Checkbox mit Text

8: Popup-Liste mit Text



`9:` Popup-Icon Radiobutton (Pfeil neben Bild).

`picture_name:` Name der gemeinsamen Bilddatei, die eine Matrix verketteter Bilder enthält, oder leere Zeichenfolge

`images_number:` Anzahl der Bilder in der Matrix, für Boole'sche Parameter kann sie den Wert von 0 oder 2 besitzen.

`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

`value_definitioni:` Wertedefinition, welche diesen Teil der Miniaturansicht mit dem Wertelistenskript verbindet:
`expression:` numerisch oder Stringausdruck, oder
`CUSTOM:` Schlüsselwort, das bedeutet, dass ein beliebiger benutzerdefinierter Wert eingegeben werden kann

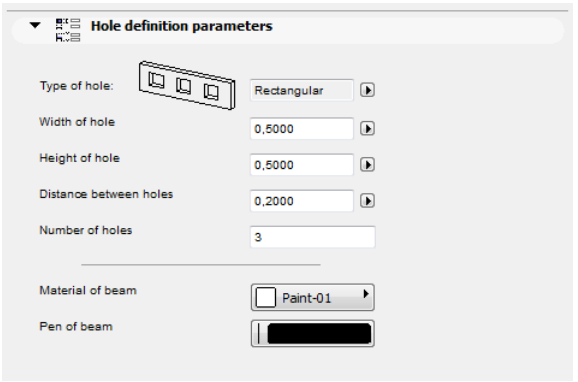
`picIdxArray:` Dynamisches Array der Bildnamen (Strings) oder Indizes (Ganzzahlen) in Zellen. Verwenden Sie keine Mischtypen-Arrays

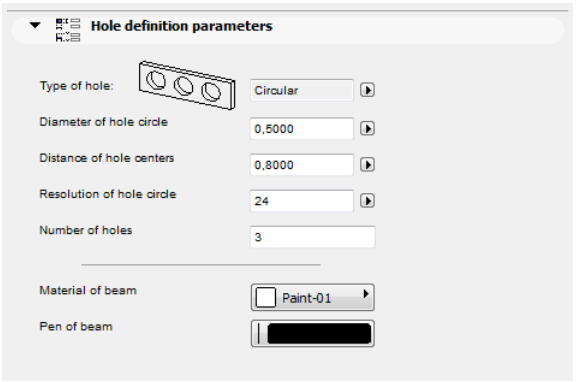
`textArray:` Dynmamisches Array mit Text in Zellen

`valueArray:` Dynmamisches Array mit Parameterwerten in Zellen

Beispiel 1:

```
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
```





Beispiel 2:

```
! Parameter Script:
VALUES "myParameter" "Two", "Three", "Five", CUSTOM

! Interface Script:
px = 80
py = 60
cx = px + 3
cy = py + 25

UI_INFIELD{3} "myParameter", 10, 10, 4 * cx + 21, cy + 5,
        1, "myPicture", 6,
        1, cx, cy, px, py,
        1, "1 - one", "One",
        2, "2 - two", "Two",
        3, "3 - three", "Three",
        4, "4 - four", "Four",
        5, "5 - five", "Five",
        6, "custom value", CUSTOM
```

Beispiel 3:

```
! Parameter Script:
VALUES "myParameter" "Two", "Three", "Five", CUSTOM

! Interface Script:
px = 80
py = 60
cx = px + 3
cy = py + 25

paramNameVar = "myParameter"
UI_INFIELD{4} paramNameVar, 10, 10, 4 * cx + 21, cy + 5,
        1, "myPicture", 6,
        1, cx, cy, px, py,
        1, "1 - one", "One",
        2, "2 - two", "Two",
        3, "3 - three", "Three",
        4, "4 - four", "Four",
        5, "5 - five", "Five",
        6, "custom value", CUSTOM
```

Beispiel 4:

```
! Master Script
dim picIdxValuesUI[]
dim textValuesUI[]
dim parameterValues[]

if myTypeParameter = 1 then
    picIdxValuesUI[1] = 6
    picIdxValuesUI[2] = 7
    picIdxValuesUI[3] = 8

    textValuesUI[1] = "6 - six"
    textValuesUI[2] = "7 - seven"
    textValuesUI[3] = "8 - eight"

    parameterValues[1] = "Six"
    parameterValues[2] = "Seven"
    parameterValues[3] = "Eight"
else
    picIdxValuesUI[1] = 6
    picIdxValuesUI[2] = 7

    textValuesUI[1] = "6 - six"
    textValuesUI[2] = "7 - seven"

    parameterValues[1] = "Six"
    parameterValues[2] = "Seven"
endif
```

```
     ! Parameter Script:
VALUES "myTypeParameter" 1, 2
VALUES "myStringParameter" "Two", "Three", "Five", parameterValues, CUSTOM

! Interface Script:
px = 80
py = 60
cx = px + 3
cy = py + 25

paramNameVar = "myStringParameter"
UI_INFIELD{4} paramNameVar, 10, 10, 4 * cx + 21, cy + 5,
        1, "myPicture", 6,
        1, cx, cy, px, py,
        1, "1 - one", "One",
        2, "2 - two", "Two",
        3, "3 - three", "Three",
        4, "4 - four", "Four",
        5, "5 - five", "Five",
        picIdxValuesUI, textValuesUI, parameterValues,
        9, "custom value", CUSTOM
```

UI_CUSTOM_POPUP_INFIELD

```
UI_CUSTOM_POPUP_INFIELD "name", x, y, width, height,
        storeHiddenId, treeDepth,
        groupingMethod, selectedValDescription,
        value1, value2, valuesArray1, .... valuen, valuesArrayn
```

UI_CUSTOM_POPUP_INFIELD{2}

```
UI_CUSTOM_POPUP_INFIELD{2} name, x, y, width, height,
        storeHiddenId, treeDepth,
        groupingMethod, selectedValDescription,
        value1, value2, valuesArray1, .... valuen, valuesArrayn
```

Kompatibilität: eingeführt in ARCHICAD 20.

Erzeugt ein Popup für eine Parameter-Auswahlliste, welche im User Interface Script erzeugt wird, damit die Verwendung des Parameter-Scripts vermieden wird.

Geeignet für Listen, welche nicht im Parameter-Script abgefragt werden können. *Zu den Einschränkungen im Parameter-Script siehe „REQUEST Optionen“.*

`name:` Parametername als Stringausdruck für UI_CUSTOM_POPUP_INFIELD oder Parametername mit optionalen aktuellen Indexwerten, falls Array, für UI_CUSTOM_POPUP_INFIELD{2}.

`x, y:` Die Position des editierbaren Textes, Popup.

`width, height:` Breite und Höhe in Pixeln.

`storeHiddenId, treeDepth:` zum Einrichten von automatischen oder manuellen Bäumen.

`storeHiddenId = 0, treeDepth = 0:` funktioniert nur bei Array-Parametern.

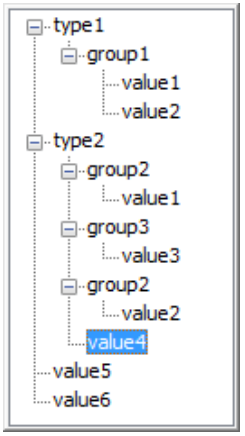
Der Parameter "treeDepth" wird automatisch von der zweiten Dimension des Arrays (Anzahl der Spalten) gesetzt..

`storeHiddenId = 1, treeDepth > 0:` funktioniert nur bei Einzelparametern.

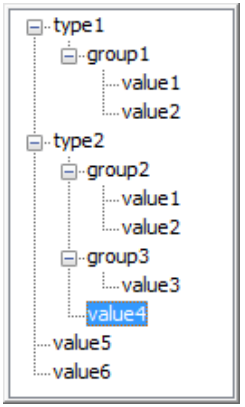
Es müssen **n * (1 + treeDepth)** Werte definiert werden (der erste für die gespeicherte ID und der Rest zur Definition des benutzerdefinierten Baumes).

`groupingMethod:` Gruppierungsmethode für die Sortierung des Baumes.

`1:` gruppiert nicht die Gruppen und Werte unterhalb des selben Elternelements.



`2:` gruppiert die Gruppen und Werte unterhalb des selben Elternelements..

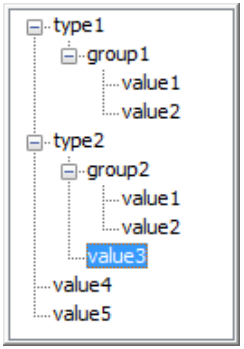


selectedValDescription: der im Feld eingetragene Text; falls ein Leerstring, wird der Text zu einer gespeicherten ID des ausgewählten Elementes.

valuei, valuesArrayi: definiert Baumwerte einen nach dem anderen und/oder mit einen 1-dimensionalen Array.

Beispiel:

```
UI_CUSTOM_POPUP_INFIELD "stParameterName", x, y, width, height,
    1, 3, 2, "",    ! storeHiddenId, treeDepth, groupingMethod, selectedValDescription
    "hiddenID1",    "type1",    "group1",   "value1",
    "hiddenID2",    "type1",    "group1",   "value2",
    "hiddenID3",    "type2",    "group2",   "value1",
    "hiddenID4",    "type2",    "group2",   "value2",
    "hiddenID5",    "type2",    "",         "value3",
    "hiddenID6",    "",         "",         "value4",
    "hiddenID7",    "",         "",         "value5"
```



UI_RADIOBUTTON

UI_RADIOBUTTON "name", value, text, x, y, width, height

UI_RADIOBUTTON{2}

UI_RADIOBUTTON{2} name, value, text, x, y, width, height

Version {2} Kompatibilität: eingeführt in ARCHICAD 20.

Erzeugt eine Optionsfeld einer Optionsfeldgruppe. Optionsfeldgruppen sind durch den Parameternamen definiert. Elemente in derselben Gruppe deaktivieren sich gegenseitig, wenn jeweils 1 Button geklickt ist.

name: Parametername oder Name als String-Ausdruck für UI_RADIOBUTTON und Parametername als String-Ausdruck (oder Text-Array-Indizierter Wert) für UI_RADIOBUTTON {2}.

value: Der Parameter ist auf diesen Wert eingestellt, wenn diese Optionsfeld eingestellt ist

text: Dieser Text wird neben der Optionsfeld angezeigt

x, y: die Position des Radio-Buttons

width, height: Breite und Höhe in Pixeln.

Beispiel:

```
UI_RADIOBUTTON "ceilingPlan", 0, `Floor Plan`, 10, 140, 100, 20
UI_RADIOBUTTON "ceilingPlan", 1, `Ceiling Plan`, 10, 160, 100, 20
```

UI_PICT_RADIOBUTTON

```
UI_PICT_RADIOBUTTON name, value, text,
        picture_reference, x, y, width, height [UI_TOOLTIP tooltip]
```

UI_PICT_RADIOBUTTON{2}

```
UI_PICT_RADIOBUTTON{2} "name", value, text,
        picture_reference, x, y, width, height [UI_TOOLTIP tooltip]
```

Kompatibilität: eingeführt in ARCHICAD 22.

Erzeugt ein Optionsfeld mit dem Symbol einer Optionsfeldgruppe. Optionsfeldgruppen sind durch den Parameternamen definiert. Elemente in derselben Gruppe deaktivieren sich gegenseitig, wenn jeweils 1 Button geklickt ist.

name: Parametername oder Name als String-Ausdruck für UI_PICT_RADIOBUTTON und Parametername als String-Ausdruck (oder indizierter Text-Arraywert) für UI_PICT_RADIOBUTTON {2}.

value: Der Parameter ist auf diesen Wert eingestellt, wenn diese Optionsfeld eingestellt ist

text: dieser Text wird auf der Schaltfläche angezeigt, falls keine Grafik deklariert wurde.

picture_reference: Dateiname oder Indexnummer des im Bibliothekselement gespeicherten Bildes. Der Index 0 bezieht sich auf das Vorschaubild des Bibliothekselements. Pixeltransparenz ist bei den Bildern erlaubt.

x, y: die Position des Optionsfeldes (oberer linker Anker).

width, height: Breite und Höhe in Pixel. Die Bildgröße wird nicht individuell deklariert: diese sollte auf die Schaltfläche passen, weil die Bilder nicht automatisch skaliert werden, damit sie passen; außerdem wird das Bild zentrisch angeordnet.

UI_PICT_PUSHCHECKBUTTON

```
UI_PICT_PUSHCHECKBUTTON name, text, picture_reference,
        frameFlag, x, y, width, height [UI_TOOLTIP tooltip]
```

UI_PICT_PUSHCHECKBUTTON{2}

```
UI_PICT_PUSHCHECKBUTTON{2} "name", text, picture_reference,
        frameFlag, x, y, width, height [UI_TOOLTIP tooltip]
```

Kompatibilität: eingeführt in ARCHICAD 22.

Erzeugt eine Druck-Check-Schaltfläche mit einem Symbol für einen booleschen Parameter. Ähnlich zu UI_INFIELD{3} mit der Methode 6, mit der zusätzlichen Option die Sichtbarkeit des Rahmens der Schaltfläche zu steuern.

name: Parametername oder Name als String-Ausdruck für UI_PICT_PUSHCHECKBUTTON und Parametername als String-Ausdruck (oder indiziertem Text-Array-Wert) für UI_PICT_PUSHCHECKBUTTON{2}.

text: dieser Text wird auf der Schaltfläche angezeigt, falls keine Grafik deklariert wurde.

picture_reference: Dateiname oder Indexnummer des im Bibliothekselement gespeicherten Bildes. Der Index 0 bezieht sich auf das Vorschaubild des Bibliothekselements. Pixeltransparenz ist bei den Bildern erlaubt.

frameFlag: 1 - Rahmen ist sichtbar, 0 - Rahmen ist unsichtbar. Verwenden Sie diese Option, um das Steuerelement an andere Elemente des User Interface im Stil anzupassen.

x, y: die Position der Schaltfläche (ober linker Anker).

width, height: Breite und Höhe in Pixel. Die Bildgröße wird nicht individuell deklariert: diese sollte auf die Schaltfläche passen, weil die Bilder nicht automatisch skaliert werden, damit sie passen; außerdem wird das Bild zentrisch angeordnet.

UI_TEXTSTYLE_INFIELD

```
UI_TEXTSTYLE_INFIELD name, faceCodeMask, x, y,
        buttonWidth, buttonHeight[, buttonOffsetX]
```

UI_TEXTSTYLE_INFIELD{2}

```
UI_TEXTSTYLE_INFIELD{2} "name", faceCodeMask, x, y,
        buttonWidth, buttonHeight [, buttonOffsetX]
```

Kompatibilität: eingeführt in ARCHICAD 22.

Erzeugt eine Zeile von Druck-Check-Schaltflächen, die speziell dazu verwendet werden, den Schriftstil über einen Ganzzahl-Parameter einzustellen, der ähnlich aussieht wie in der allgemeinen Programmoberfläche. Das Format des eingestellten Wertes entspricht dem Eingabeparameter von DEFINE STYLE{2}. Sowohl Icons als auch Tooltipps werden von ARCHICAD selber, entsprechend der lokalisierten Version referenziert. Die aktivierten Schaltflächen werden in einer einreihigen Anordnung angezeigt.

name: Parametername oder Name als String-Ausdruck für UI_TEXTSTYLE_INFIELD und Parametername als String-Ausdruck (oder indizierter Textarraywert) für UI_TEXTSTYLE_INFIELD{2}.

faceCodeMask: folgende verwendete Bits fügen den aufgereihten Steuerelementen die passende Schriftschnittoption hinzu:

faceCodeMask = $j_1 + 2*j_2 + 4*j_3 + 32*j_6 + 64*j_7 + 128*j_8$, hierbei kann j jeweils 0 oder 1 sein.

j_1: fett

j_2: kursiv

`j3`: unterstrichen
`j6`: hochgestellt
`j7`: tiefergestellt
`j8`: durchgestrichen

Falls faceCodeMask = 0, werden sämtliche möglichen Schriftschnitt-Schaltflächen angezeigt. Im Falle eines ungültigen `faceCodeMask`, gibt "Script prüfen" eine Warnmeldung zurück ("Ungültiger Maskenwert verwendet").

x, y: Die Position der ersten Schaltfläche (linke obere Ecke).

buttonWidth, buttonHeight: Breite und Höhe der Schaltflächen in Pixeln. Die Gesamtbreite kann ausgerechnet werden, indem man `faceCodeMask` verwendet, die Werte für `buttonWidth` und `buttonOffsetX` values, falls nötig.

buttonOffsetX: Abstand zwischen benachbarten Schaltflächen in der Zeile in Pixeln. Automatisch, falls nicht gesetzt.

UI_LISTFIELD

UI_LISTFIELD `fieldID, x, y, width, height [, iconFlag [, description_header [, value_header]]]`

Erzeugt eine Eingabemöglichkeit für Parameter als scrollbare Liste, welche eine beliebige Anzahl an Zeilen enthält, und den folgend aufgeführten Spalten: Icon, Beschreibung und Input-Feld des Parameter-Wertes. Zeilen der Liste können definiert werden mit dem Befehl UI_LISTITEM. Definitionen von UI_LISTFIELD und UI_LISTITEM können in beliebiger Reihenfolge im Script erzeugt werden. Leere Listenfelder (ohne Listenpunkte) werden nicht angezeigt.

fieldID: der unique Identifizierer (die ID) des Listenfeldes. Diese ID wird auch in dem Befehl UI_LISTITEM verwendet und gibt das Listenfeld an, zu welchem die Listenpunkte gehören. Duplikate innerhalb eines User Interface Scriptes sind nicht erlaubt.

x, y: Position der linken oberen Ecke des Listenfeldes.

width, height: Breite und Höhe in Pixeln.

iconFlag:

`iconFlag = 0`: eine Icon-Spalte wird für dieses Listenfeld nicht erzeugt.

`iconFlag = 1`: eine Icon-Spalte wird für dieses Listenfeld erzeugt (grundeingestellter Wert ist nicht vorgegeben).

Falls das Paneel der benutzerdefinierten Einstellungen nur einen einzigen Eintrag besitzt und dieser Eintrag ein Listenfeld ist, haben die Parameter für x, y, Breite, Höhe keinen Effekt. In diesem Fall ist die Breite des Listenfeldes genauso breit wie das Paneel der benutzerdefinierten Einstellungen.

description_header: der Titel der Beschreibungs-Spalte

value_header: der Titel der Werte-Spalte.

Wenn sowohl description_header als auch value_header leere Strings sind oder nicht spezifiziert sind, wird das Listenfeld ohne Überschrift erzeugt. Falls der String wenigstens ein Leerzeichen enthält, wird das Listenfeld mit einer leeren Überschrift erzeugt.

UI_LISTITEM

UI_LISTITEM itemID, fieldID, "name" [, childFlag [, image [, paramDesc]]]

UI_LISTITEM{2}

UI_LISTITEM{2} itemID, fieldID, name [, childFlag [, image [, paramDesc]]]

Hängt einen Listenpunkt an das Listenfeld an, definiert durch den Parameter fieldID.

itemID: der unique Identifizierer (ID) des Listenpunktes. Listenpunkte können in beliebiger Reihenfolge im Script abgelegt werden; sie werden sortiert durch den Wert itemID. Doppelte Listenwert-IDs innerhalb eines Listenfeldes sind nicht erlaubt.

fieldID: der unique Identifizierer des Listenfeldes, welches diesen Listenpunkt enthält.

name: Parameternamen als String-Ausdruck für UI_LISTITEM oder Parametername mit optionalem aktuellem Indexwert, falls ein Array für UI_LISTITEM{2}.

childFlag:

childFlag = 0: der Listenpunkt ist ein Gruppenelement (grundeingestellter Wert ist nicht spezifiziert).

childFlag = 1: der Listenpunkt ist ein Tochterelement. Das Elternelement ist das erste oberhalb liegende Gruppenelement.

image: Dateiname oder Indexnummer des im Bibliothekselement gespeicherten Bildes. Falls gültig, wird es als Icon in der ersten Spalte des Listenfeldes in der dazugehörigen Zeile des Listenpunktes angezeigt.

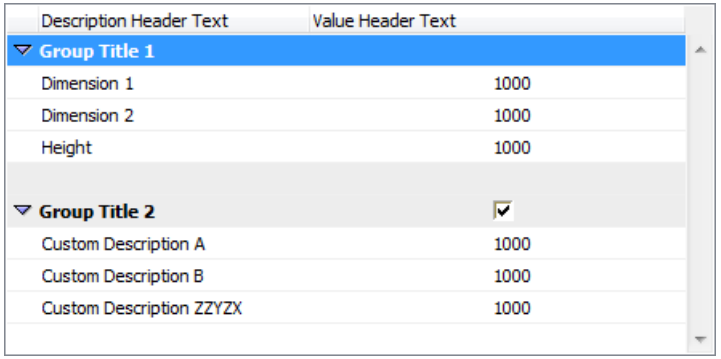
paramDesc: der sichtbare Name des Listenpunktes in der Beschreibungs-Spalte. Falls leergelassen, wird die Beschreibung automatisch mit der Listenbeschreibung des Parameters des Bibliothekselementes ausgefüllt. Falls es dort keine Beschreibung gibt, wird stattdessen der Name des Parameters angezeigt.

Falls der String "name" leer ist, ist der Listenpunkt eine Gruppe mit fettem Schrifttyp. Falls sowohl der String "name" als auch paramDesc leer sind, ist der Listenpunkt eine Trennlinie. Der Befehl HIDEPARAMETER funktioniert bei Listenpunkten nicht; das Script sollte in diesem Falls den Punkt einfach weglassen anstatt ihn hinzuzufügen. Der Befehl LOCK kann bei Listenpunkten verwendet werden und funktioniert dort.

Bei einem Listenfeld ist es empfehlenswert, unterschiedliche itemIDs für unterschiedliche Parameter, Gruppen und Trennlinien zu definieren.

Beispiel:

```
! Liste mit Überschrift, aber ohne Icon-Spalte

ui_listfield 1, 10, 35, 432, 220, 0, "Description Header Text", "Value Header Text"

ui_listitem 1, 1, "", 0, "", "Group Title 1" ! Group Line
ui_listitem 2, 1, "A", 1
ui_listitem 3, 1, "B", 1
ui_listitem 4, 1, "ZZYZX", 1

ui_listitem 5, 1, "" !separator
ui_listitem 6, 1, "AC_show2DHotspotsIn3D", 0, "", "Group Title 2" ! Group Parameter Line
ui_listitem 7, 1, "A", 1, "", "Custom Description A"
ui_listitem 8, 1, "B", 1, "", "Custom Description B"
ui_listitem 9, 1, "ZZYZX", 1, "", "Custom Description ZZYZX"
```



UI_CUSTOM_POPUP_LISTITEM

```
UI_CUSTOM_POPUP_LISTITEM itemID, fieldID, "name", childFlag, image, paramDesc,
storeHiddenId, treeDepth,
groupingMethod, selectedValDescription,
value1, value2, valuesArray1, .... valuen, valuesArrayn
```

UI_CUSTOM_POPUP_LISTITEM{2}

```
UI_CUSTOM_POPUP_LISTITEM{2} itemID, fieldID, name, childFlag, image, paramDesc,
storeHiddenId, treeDepth,
groupingMethod, selectedValDescription,
value1, value2, valuesArray1, .... valuen, valuesArrayn
```

Kompatibilität: eingeführt in ARCHICAD 20.

Ähnlich wie "UI_CUSTOM_POPUP_INFIELD" und "UI_CUSTOM_POPUP_INFIELD{2}"

Erzeugt ein Listenelement mit einem Popup für eine Parameter-Werteliste, welche im User Interface Script definiert wird, um die Verwendung des Parameter-Scripts zu vermeiden.

Geeignet für Listen, welche nicht im Parameter-Script abgefragt werden können. *Zu den Einschränkungen im Parameter-Script siehe „REQUEST Optionen“.*

itemID: der unique Identifizierer (ID) des Listenpunktes. Listenpunkte können in beliebiger Reihenfolge im Script abgelegt werden; sie werden sortiert durch den Wert itemID. Doppelte Listenwert-IDs innerhalb eines Listenfeldes sind nicht erlaubt.

fieldID: der unique Identifizierer des Listenfeldes, welches diesen Listenpunkt enthält.

name: Parametername als Stringausdruck für UI_CUSTOM_POPUP_LISTITEM oder Parametername mit optionalen aktuellen Indexwerten, falls Array, für UI_CUSTOM_POPUP_LISTITEM{2}.

childFlag:

`childFlag = 0`: der Listenpunkt ist ein Gruppenelement (grundeingestellter Wert ist nicht spezifiziert).

`childFlag = 1`: der Listenpunkt ist ein Tochterelement. Das Elternelement ist das erste oberhalb liegende Gruppenelement.

image: Dateiname oder Indexnummer des im Bibliothekselement gespeicherten Bildes. Falls gültig, wird es als Icon in der ersten Spalte des Listenfeldes in der dazugehörigen Zeile des Listenpunktes angezeigt.

paramDesc: der sichtbare Name des Listenpunktes in der Beschreibungs-Spalte. Falls leergelassen, wird die Beschreibung automatisch mit der Listenbeschreibung des Parameters des Bibliothekselementes ausgefüllt. Falls es dort keine Beschreibung gibt, wird stattdessen der Name des Parameters angezeigt.

storeHiddenId, treeDepth: zum Einrichten von automatischen oder manuellen Bäumen.

`storeHiddenId = 0, treeDepth = 0`: funktioniert nur bei Array-Parametern.

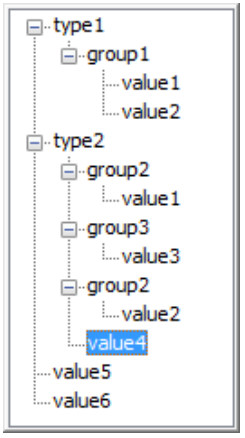
Der Parameter "treeDepth" wird automatisch von der zweiten Dimension des Arrays (Anzahl der Spalten) gesetzt..

`storeHiddenId = 1, treeDepth > 0`: funktioniert nur bei Einzelparametern.

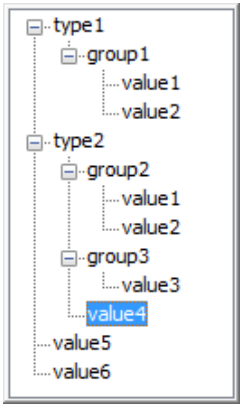
Es müssen **n * (1 + treeDepth)** Werte definiert werden (der erste für die gespeicherte ID und der Rest zur Definition des benutzerdefinierten Baumes).

`groupingMethod:` Gruppierungsmethode für die Sortierung des Baumes.

`1:` gruppiert nicht die Gruppen und Werte unterhalb des selben Elternelements.



`2:` gruppiert die Gruppen und Werte unterhalb des selben Elternelements..

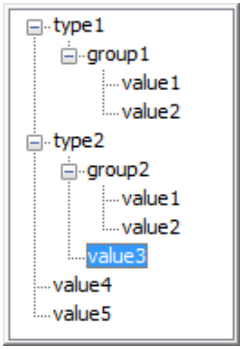


selectedValDescription: der im Feld eingetragene Text; falls ein Leerstring, wird der Text zu einer gespeicherten ID des ausgewählten Elementes.

valuei, valuesArrayi: definiert Baumwerte einen nach dem anderen und/oder mit einen 1-dimensionalen Array.

Beispiel:

```
UI_CUSTOM_POPUP_LISTITEM itemID, fieldID, "stParameterName", 0, "", "",
    1, 3, 2, "",    ! storeHiddenId, treeDepth, groupingMethod, selectedValDescription
    "hiddenID1",    "type1",    "group1",   "value1",
    "hiddenID2",    "type1",    "group1",   "value2",
    "hiddenID3",    "type2",    "group2",   "value1",
    "hiddenID4",    "type2",    "group2",   "value2",
    "hiddenID5",    "type2",    "",         "value3",
    "hiddenID6",    "",         "",         "value4",
    "hiddenID7",    "",         "",         "value5"
```



UI_TOOLTIP

```
UI_BUTTON type, text, x, y, width, height [, id [, url]] [ UI_TOOLTIP tooltiptext ]
UI_PICT_BUTTON type, text, picture_reference,
        x, y, width, height [, id [, url]] [ UI_TOOLTIP tooltiptext ]
UI_INFIELD "name", x, y, width, height [, extra parameters ... ]
        [ UI_TOOLTIP tooltiptext ]
```

```
UI_INFIELD{2} name, x, y, width, height [, extra parameters ... ]
        [ UI_TOOLTIP tooltiptext ]
UI_INFIELD{3} name, x, y, width, height [, extra parameters ... ]
        [ UI_TOOLTIP tooltiptext ]
UI_INFIELD{4} "name", x, y, width, height [, extra parameters ... ]
        [ UI_TOOLTIP tooltiptext ]
UI_CUSTOM_POPUP_INFIELD "name", x, y, width, height , extra parameters ...
        [ UI_TOOLTIP tooltiptext ]
UI_CUSTOM_POPUP_INFIELD{2} name, x, y, width, height , extra parameters ...
        [ UI_TOOLTIP tooltiptext ]
UI_RADIOBUTTON name, value, text, x, y, width, height [ UI_TOOLTIP tooltiptext ]
UI_OUTFIELD expression, x, y, width, height [, flags] [ UI_TOOLTIP tooltiptext ]
UI_PICT expression, x, y [,width, height [, mask]] [ UI_TOOLTIP tooltiptext ]
UI_LISTFIELD fieldID, x, y, width, height [, iconFlag [, description_header [, value_header]]]
        [ UI_TOOLTIP tooltiptext ]
UI_LISTITEM itemID, fieldID, "name" [, childFlag [, image [, paramDesc]]]
        [ UI_TOOLTIP tooltiptext ]
UI_LISTITEM{2} itemID, fieldID, name [, childFlag [, image [, paramDesc]]]
        [ UI_TOOLTIP tooltiptext ]
UI_CUSTOM_POPUP_LISTITEM itemID, fieldID, "name", childFlag , image , paramDesc,
        extra parameters ...
        [ UI_TOOLTIP tooltiptext ]
UI_CUSTOM_POPUP_LISTITEM{2} itemID, fieldID, name, childFlag , image , paramDesc,
        extra parameters ...
        [ UI_TOOLTIP tooltiptext ]
```

Definiert den Tooltipp für die Erläuterung der Interface Seite. Tooltipps sind verfügbar für (UI_)Buttons, Infields, Outfields, Listfields, Listitems und Pictures, falls diese nicht vom Anwender im laufenden Kontext deaktiviert wurden (z.B. im Hilfe-Menü von ARCHICAD).

Der Tooltipp des Listfields erscheint in allen enthaltenden Listitems, falls ein Listitem keinen eigenen besitzt. Der eigene Tooltipp des Listitems hat Vorrang vor dem Toolkit des inline Listfields (falls vorhanden).

tooltiptext: Der Text, der als Tooltipp-Text für das Eingabeelement angezeigt werden soll.

UI_COLORPICKER

```
UI_COLORPICKER "redParamName", "greenParamName", "blueParamName", x0, y0 [, width [, height]]
```

UI_COLORPICKER{2}

```
UI_COLORPICKER{2} redParamName, greenParamName, blueParamName, x0, y0 [, width [, height]]
```

Farbauswahldialog zum Setzen der R, G, B-Komponenten einer Farbe und zum Abspeichern in die vorgegebenen Parameter. Diese Werte kann man später für den Befehl LIGHT verwenden.

redParamName, greenParamName, blueParamName: Parameternamen als Stringausdruck für UI_COLORPICKER oder Parameternamen mit optionalen aktuellen Array-Indexwerten für UI_COLORPICKER{2}

x0, y0: Position der linken oberen Ecke der Farbauswahl.

width, height: Breite und Höhe in Pixeln.

UI_SLIDER

UI_SLIDER "name", x0, y0, width, height [, nSegments [, sliderStyle]]

UI_SLIDER{2}

UI_SLIDER{2} name, x0, y0, width, height [, nSegments [, sliderStyle]]

Erzeugt einen Schieberegler für mit RANGE definierte Ganzzahl-Parameter. Bei Ganzzahl-Parametern mit nicht durch RANGE definierten oberen und unteren Grenzwerten gelten -32768 (Minimalwert) und 32767 (Maximalwert).

name: Parametername als Stringausdruck für UI_SLIDER oder parameter mit optionalen aktuellen Array-Indexwerten für UI_SLIDER{2}.

x0, y0: Position des Schiebereglers.

width, height: Breite und Höhe des Schiebereglers in Pixeln. Falls Breite > Höhe, ist der Schieberegler waagerecht, im anderen Fall ist er senkrecht.

nSegments: optionale Anzahl an Segmenten auf dem Schieberegler. Falls 0, werden keine Segmente angezeigt, falls weggelassen oder negativ, wird die Anzahl der Segmente aus der Begrenzung der oberen und unteren Grenzwerte berechnet und der für den Parameter definierten Schrittweite.

sliderStyle: optionale Stil des Schiebereglers (grundeingestellt ist 0)

0: Punkte am Schieberegler unten (waagerechter Schieberegler) oder rechts (senkrechter Schieberegler).

1: Punkte am Schieberegler oben (waagerechter Schieberegler) oder links (senkrechter Schieberegler).

Das Vorwärts-Migration Script

Falls ein Element in einer neueren Bibliothek komplett geändert wurde, kann trotzdem eine Kompatibilität durch die Definition einer Migrationslogik aufrecht erhalten werden. Für detailliertere Informationen, lesen Sie bitte hier „Das Vorwärts-Migration Script“.

Beispiel:

```
actualGUID = FROM_GUID

! ============================================================================
! Subroutines
! ============================================================================

    _startID    = "AAAA-AAAA-...AAA"
    _endID      = "BBBB-BBBB-...BBB"
gosub "migrationstepname_FWM"

! ============================================================================
! Set Migration GUID
! ============================================================================

setmigrationguid actualGUID

! ============================================================================
end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! en
! ============================================================================

! ============================================================================
! migrationstepname
! ============================================================================
"migrationstepname_FWM":
    if actualGuid = _startID then
        newParameter = oldParameter
        parameters newParameter = newParameter
        actualGuid = _endID
    endif
return
```

FROM_GUID ist die Globale Variable, welche die Haupt-ID des originalen Objektes enthält, für welche die Migration durchgeführt werden soll. Falls das Script erfolgreich war, wird die Instanz des Objektes ersetzt durch das neue Element mit den entsprechend aktualisierten Parametern.

SETMIGRATIONGUID

SETMIGRATIONGUID guid

Dieser Befehl teilt der Laufzeitumgebung mit, welches Element das passende Migrationselement für das entsprechende Objekt ist. Falls die zurückgegebene ID zum aktuellen Element gehört, wird die Migration des Objektes abgeschlossen.

STORED_PAR_VALUE

STORED_PAR_VALUE `("oldparname", outputvalue)`

Ruft den Wert eines Parameters ab, welcher im migrierten Objekt vorhanden ist, und in der neuen Version des Objektes vorhanden ist oder gelöscht wurde. Diese Befehlsform wird für die im neuen Objekt vorhandenen Parameter ebenfalls vorgeschlagen. Um den Wert eines alten Array-Parameters zu erhalten, muss die outputvalue als Array initialisiert werden (mit dem DIM-Befehl).

oldparname: Stringausdruck, Name des Parameters in der alten Parameterliste.

outputvalue: Ausgabevariable, um den Wert des Parameters zu speichern.

Rückgabewert: 1 bei Erfolg, 0, im anderen Fall (z.B. wenn es keinen Parameter mit diesem Namen in der Parameterliste des alten Objektes gibt). Bei der Scriptüberprüfung ist der Rückgabewert immer 0, weil der alte Parameterbereich unbekannt ist.

DELETED_PAR_VALUE

DELETED_PAR_VALUE `("oldparname", outputvalue)`

Ruft den Wert eines Parameters ab, welcher im migrierten Objekt vorhanden ist, und in der neuen Version des Objektes vorhanden ist oder gelöscht wurde. Diese Befehlsform wird für die im neuen Objekt gelöschten Parameter empfohlen. Um den Wert eines alten Array-Parameters zu erhalten, muss die outputvalue als Array initialisiert werden (mit dem DIM-Befehl).

oldparname: Stringausdruck, Name des Parameters in der alten Parameterliste.

outputvalue: Ausgabevariable, um den Wert des Parameters zu speichern.

Rückgabewert: 1 bei Erfolg, 0, im anderen Fall (z.B. wenn es keinen Parameter mit diesem Namen in der Parameterliste des alten Objektes gibt). Bei der Scriptüberprüfung ist der Rückgabewert immer 0, weil der alte Parameterbereich unbekannt ist.

Das Rückwärts-Migration Script

Mit Hilfe des Rückwärts-Migration-Scriptes kann man die rückwärtsgerichtete Migrationslogik definieren, welche neue Objektinstanzen in ältere überführt. Für detailliertere Informationen, lesen Sie bitte hier „Das Rückwärts-Migration Script“.

Beispiel:

```
targetGUID = TO_GUID

! ============================================================================
! Subroutines
! ============================================================================

gosub "migrationstepname_BWM"

! ============================================================================
! Set Migration GUID
! ============================================================================

setmigrationguid targetGUID

! ============================================================================
end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! en
! ============================================================================


! ============================================================================
! migrationstepname
! ============================================================================
"migrationstepname _BWM":
    if targetGUID # "" then
        bMigrationSuccess = 1
        if bMigrationSuccess = 1 then
            oldParameter = newParameter
            parameters oldParameter = oldParameter
        else
            targetGuid = ""
        endif
    endif
return
```

TO_GUID ist die Globale Variable, welche die Haupt-ID des Zielelementes der Konvertierung enthält.
Verwenden SieSETMIGRATIONGUID zum Einstellen der targetGUID.

NEWPARAMETER

NEWPARAMETER "name", "type" [, dim1 [, dim2]]

Fügt im Rückwärts-Migration-Script einen neuen Parameter zu den vorhandenen eines Bibliothekselementes hinzu. Die Parametererzeugung geschieht nur nach der vollständigen Interpretation des Scriptes. Falls ein Parameter mit einem vorgegebenen Namen bereits in der Parameter-Liste existiert, erfolgt eine Fehlermeldung.

name: Stringausdruck, Parametername kann erzeugt werden.

type: Stringausdruck, Parametertyp. Mögliche Statuswerte:

- Integer
- Length
- Angle
- RealNum
- LightSwitch
- ColorRGB
- Intensity
- LineType
- Material
- FillPattern
- PenColor
- String
- Boolean
- BuildingMaterial*Kompatibilität: eingeführt in ARCHICAD 22.*
- Profile*Kompatibilität: eingeführt in ARCHICAD 22.*

dim1, dim2: dim1 ist die erste Dimension des Parameters, 0 falls nicht gesetzt. dim2 ist die zweite Dimension des Parameters, 0 falls nicht gesetzt.

- dim1 = 0, dim2 = 0: Ist der Parameter ein skalarer Parameter;
- dim1 > 0, dim2 = 0: ist der Parameter ein 1-dimensionales Array.
- dim1 > 0, dim2 > 0: ist der Parameter ein 2-dimensionales Array.

Einschränkung der Parameter:

Wenn dim2 > 0, dann dim1 > 0.

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 Prekompilierung überprüft.

DICT

DICT variableName1[, variableName2...]

Kompatibilität: eingeführt in ARCHICAD 23.

GDL unterstützt Dictionaries. Eine Variable wird nach der obigen Deklarationsanweisung als Dictionary deklariert (und kann nicht in ein Array oder einen einfachen Typ geändert werden oder umgekehrt).Bibliothekselementparameter können auch Dictionaries sein, indem Sie in der Parameterliste die Option Dictionary auswählen.

Nach dem DICT-Schlüsselwort können beliebig viele Variablennamen, durch Kommas getrennt, folgen. Jede Variable enthält hierarchische Schlüssel- und Wertepaare. Ein Schlüssel des Dictionaries kann mit Punktnotation referenziert werden. Der vollständige Pfad eines Schlüssels darf nicht länger als 255 Zeichen sein (Array-Indizes werden als ein Zeichen gezählt)).

Dictionaries und einfache Typwerte:

- Dictionary-Schlüsseln können einfache Werte vom Typ (Zeichenfolge, Ganzzahl, Gleitkomma) zugewiesen werden,
- es ist keine Deklaration erforderlich, der Wertetyp für den Schlüssel wird durch den aktuellen Wert festgelegt:

```
DICT myDictionary

myDictionary.element1 = 1
myDictionary.element1 = "hello"

print myDictionary
```

Dictionaries und abgeleitete Typwerte:

- in einem Dictionary können Arrays (nur mit einer Dimension) und Dictionary-Schlüssel verschachtelt werden,
- ein Array in einem Dictionary kann unbenannte Dictionaries oder einfache Typen enthalten (auf die der Index verweist)),
- ein eigenständiger Parameter / eine eigenständige Variable vom Typ Array darf jedoch keine Elemente vom Typ Dictionary enthalten,
- ein verschachtelter Array-Schlüssel kann durch sofortiges Referenzieren initialisiert werden. In diesem Fall muss er nicht mit DIM deklariert werden,

```
DICT myDictionary
myDictionary.myArray[1] = 1
myDictionary.myArray[2] = 5

print myDictionary
```

- nicht referenzierte Indizes eines verschachtelten Arrays werden automatisch entsprechend dem Typ des ersten referenzierten Elements des Arrays initialisiert (String-Schlüssel auf "", numerische Schlüssel auf 0, Dictionary-Schlüssel auf {}).

```
DICT myDictionary
DICT dictForNesting
dictForNesting.elem1 = "hello"
dictForNesting.elem2 = "world"

myDictionary.myArray[2] = dictForNesting

print myDictionary

DICT myDictionary2
myDictionary2.myArray[3] = 33
myDictionary2.myArray[4] = 44

print myDictionary2
```

- die Werte eines verschachtelten Arrays müssen vom gleichen Typ sein (alle als String, alle als Ganzzahlen, alle als Gleitkomma- oder alle als Dictionary-Typen). Dies steht im Gegensatz zur Funktionsweise von Arrays. Daher ist besondere Vorsicht geboten!

```
DICT myDictionary2
myDictionary2.myArray[1] = 1
myDictionary2.myArray[2] = 1.0 ! GDL Fehler
```

- um die Wertetypen eines verschachtelten Arrays zu ändern, muss es zuerst zurückgesetzt werden : erstellen Sie ein leeres Array und überschreiben Sie das verschachtelte Array mit diesem neuen leeren Array. Der Typ des nächsten referenzierten Werts legt den Typ für das Array nach dem Zurücksetzen automatisch fest:

```
DICT myDictionary
myDictionary.myArray[1] = "hello"
print myDictionary

DIM arrayForReset[]
myDictionary.myArray = arrayForReset
print myDictionary

myDictionary.myArray[1] = 10000
print myDictionary
```

- wenn Sie den Typ des ersten Werts eines verschachtelten Arrays ändern, das nur diesen Wert enthält, wird der Typ des Arrays nicht geändert!

```
DICT myDictionary
myDictionary.myArray[1] = "hello"
print myDictionary

myDictionary.myArray[1] = 10000 ! GDL Fehler
print myDictionary
```

Initialisierung und Kopieren:

- die erste Referenz des Dictionary muss `DICT dictName` sein, eine Subroutine, die den Namen des Dictionaries enthält, kann nicht davor stehen (wie bei DIM-Arrays),
- die Initialisierung ist vor der ersten Verwendung eines Schlüssels erforderlich, entweder explizit mit Zuweisung zum Schlüssel oder implizit mit Zuweisung eines Dictionaries, das den Schlüssel in der richtigen Tiefe enthält.

```
DICT myDictionary

myDictionary.level1.a = 1
myDictionary.level1.b = 2
myDictionary.level2 = myDictionary.level1

print myDictionary.level2.b
```

- Schreiben des Dictionary-Namens ohne die eigentlichen inneren Schlüssel, verweist auf die gesamte Dictionary-Struktur, was in einigen Fällen akzeptiert wird (CALL, PRINT, LET-Anweisungen)),

- wenn Sie einen Teil der Struktur schreiben, referenzieren Sie den Teilbaum unter diesem Schlüssel als Dictionary

```
DICT myDictionary
myDictionary.point1.x = 1
myDictionary.point1.y = 1
myDictionary.point1.type = 0
print myDictionary

DICT myPoint
myPoint = myDictionary.point1
print myPoint

myDictionary.point2 = myDictionary.point1
print myDictionary
```

- wenn Sie ein Dictionary ganz oder teilweise zuweisen, wird eine tiefe Kopie der rechten Seite auf der linken Seite erstellt

```
DICT myDictionary, tempPoint
tempPoint.x = 1
tempPoint.y = 1
myDictionary.line.point1 = tempPoint

tempPoint.x = 2
tempPoint.y = 2
myDictionary.line.point2 = tempPoint

DICT myLine
myLine = myDictionary.line
myDictionary.line.point2.x = 0
print myLine
```

Makroaufrufe und Requests:

- bei Makroaufrufen können Werte vom Typ Dictionary an das Makro übergeben werden, wenn auf der empfangenden Seite ein Parameter vom Typ Dictionary vorhanden ist,
- RETURNED_PARAMETERS funktioniert auch mit Dictionaries: ein leerer DICT muss auf der empfangenden Seite deklariert werden (Caller-Objekt).im folgenden Code ist `myDictionary` ein Parameter vom Dictionarytyp im Makro, `_myDictionary` ist eine Variable vom Dictionarytyp auf der Caller-Seite des Objekts:

```
! caller object Master script
DICT _myDictionary

_myDictionary.element1 = 1
_myDictionary.element2 = 2

DICT _dictForReceivedData

call "macroname" parameters all myDictionary = _myDictionary,
    returned_parameters _dictForReceivedData

print _dictForReceivedData

! in the macro object Master script
myDictionary.element1 = myDictionary.element1 * 2
myDictionary.element2 = myDictionary.element2 * 2

end myDictionary
```

- in den Optionen für REQUEST-Befehle gibt es derzeit keine Dictionaries, welche Requests unterstützen.. Diese Möglichkeit ist jedoch offen für die Zukunft.
- LIBRARYGLOBAL-Abfragen können keine Werte vom Typ Dictionary zurückgeben.

Sichtbarkeit und Funktionen:

- die Parameter für den Typ Dictionary sind auf der Seite "Alle Parameter" im Dialogfeld "Einstellungen" nicht sichtbar,
- für Listendarstellung oder IFC-Mapping sind keine Parameter vom Typ Dictionary verfügbar,
- eine Darstellung in Textform funktioniert nur mit PRINT ("Script überprüfen" Warnung und als gedrucktes Reportfenster im JSON-Format),
- allgemeine Textverarbeitungsbefehle wie TEXT2, RICHTEXT2 usw. unterstützen nicht das gesamte Dictionary,
- Mit ihnen können jedoch verschachtelte Werte vom Typ "Nicht-Dictionary" angezeigt werden,

```
DICT myDictionary
myDictionary.myArray[1] = "hello"
myDictionary.myArray[2] = "world"

text2 0, 0, myDictionary.myArray[1] + " " + myDictionary.myArray[2]
print myDictionary
```

- Werte für einen Dictionary-Typ-Parameter können nur über das Parameter-Script festgelegt werden (keine direkte Benutzereingabe über Parameterliste oder Benutzeroberflächen-Steuerelemente möglich).VALUES ist für diesen Typ nicht verfügbar,

- die Verwendung von Nicht-Dictionary-Parametern für Benutzereingaben kann jedoch funktionieren. Im folgenden Code ist `myDictionary` ein Parameter vom Typ Dictionary, `stTextInput` ist ein Stringtyp-Parameter (welcher im User Interface verwendet werden kann, auf der Seite "Alle Parameter" angezeigt werden kann, und welcher mit GLOB_MODPAR_NAME zusammnenarbeitet):

```
! in Parameter Script
myDictionary.text1 = stTextInput
parameters myDictionary = myDictionary

! in Master Script
print myDictionary

! in 2D script
TEXT2 0, 0, myDictionary.text1
```

- das Ersetzen von Werten mit dem LP_XMLConverter-Tool ist derzeit für Dictionary-Typ-Parameter nicht verfügbar.

HASKEY

HASKEY (dictionary.key)

Gibt als Booleschen Wert zurück, ob der Schlüssel zuvor im Dictionary definiert wurde (Schlüssel kann Unterschlüssel enthalten).

Beispiel:

```
DICT myDictionary
myDictionary.point[1].x = 1
myDictionary.point[1].y = 1

print HASKEY(myDictionary.point)          ! richtig
print HASKEY(myDictionary.point[2])       ! falsch
print HASKEY(myDictionary.point[1].z)     ! falsch
```

REMOVEKEY

REMOVEKEY (dictionary.key)

Diese Funktion entfernt den referenzierten Schlüssel zusammen mit dem (den) zugewiesenen Wert(en) aus dem Dictionary. Wenn das Entfernen erfolgreich war, ist der Rückgabewert 1, andernfalls 0 (falls der Schlüssel nicht vorhanden ist oder bereits gelöscht wurde).

Beispiel:

```
DICT myDictionary
myDictionary.myText[1] = "hello"
myDictionary.myOtherText[1] = "world"

print myDictionary

_dummy = REMOVEKEY(myDictionary.myOtherText)
print myDictionary, _dummy

_dummy2 = REMOVEKEY(myDictionary.myNonExistentText)
print myDictionary, _dummy2
```

DIM

```
DIM var1[dim_1], var2[dim_1][dim_2], var3[ ],
        var4[ ][ ], var5[dim_1][ ],
        var5[ ][dim_2]
```

GDL unterstützt ein- und zweidimensionale Arrays. Variablen werden nach der obigen Deklarationsanweisung zu Arrays, in welchen ihre Dimensionen angegeben sind. (Variablen vom Typ Dictionary können nicht als Arrays redeklariert werden oder umgekehrt.)

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 keine Werte 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 beliebigen 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. Array Variablen mit fester Länge können keine dynamischen Array-Werte in Zuordnungen erhalten. 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

VARDIM1 (expr)

VARDIM2

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.

Beispiel 1: 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
```

Beispiel 2: Beispiele für textliche Ausdrücke:

```
"Konstante Zeichenfolge" name + STR ("%m", i) + "." + ext string_param <> "Mode 1"
```

Beispiel 3: Beispiele für Ausdrücke, die Anordnungswerte verwenden:

```
DIM tab[5], tab2[3][4] ! Deklaration
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 aus 3 Elementen
v2[2][3] = 23 ! alle anderen Elemente (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
```

Beispiel 4: Beispiele für Ausdrücke mit Werten für Dictionaries:

```
DICT _exampleDict

! DICT einfache Schlüsseltypen

_exampleDict.false = (1 = 2)  ! logisch falsch (Ganzzahl intern)
_exampleDict.true = (1 = 1)   ! logisch wahr (Ganzzahl intern)
_exampleDict.int = 2    ! Ganzzahl
_exampleDict.float = 1 / 3   ! Fliesskommazahl
_exampleDict.string = "Custom text" ! String

! DICT Array-Schlüsseltypen

 ! Array on-the-fly initialisieren
_exampleDict.array[1] = _exampleDict.float
 ! Direkt an Array anhängen
_exampleDict.array[2] = _exampleDict.float * 2
 ! An Array anhängen mit automatischer Initialisierung der dazwischen liegenden Elemente
_exampleDict.array[4] = _exampleDict.float * 3

! DICT array of DICTs

DIM array[]
DICT _element
_element.a = "A"
_element.b = 1

_exampleDict.array = array   ! Ändern des vorhandenen Array in ein leeres
_exampleDict.array[2] = _element ! anderer vartype als vorher

! print DICT array of DICTs
print "\n\t",
  "Print DICT array of DICTs",
  "\n-------------------------------------------------------------------------------\n\t",
  vartype(_exampleDict.array), vardim1(_exampleDict.array),  "\n",
  _exampleDict.array,
  "\n-------------------------------------------------------------------------------"
```

PARVALUE_DESCRIPTION

PARVALUE_DESCRIPTION (parname [, ind1 [, ind2]])

Diese Funktion gibt den Beschreibungsstring des Parameterwertes eines numerischen Parameters zurück, welcher mit dem Ausdruck VALUES spezifiziert wurde. Falls keine Beschreibung angegeben war, ist der Rückgabewert ein Leerstring.

parname: Name des Parameters

ind1, ind2: aktuelle Indizes falls der Parameter ein Array ist.

OPERATOREN

Die untenstehenden Rechenoperatoren wurden in der Reihenfolge ihrer Priorität aufgelistet. Die Berechnung eines Ausdruckes 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) x MOD y = x - y * INT (x/y)	3. Priorität
+	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

Diese Operatoren können auch zwischen Strings verwendet werden (bei Vergleichen mit Strings ist die Groß-/Kleinschreibung zu beachten). 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.

Boolesche Operatoren

AND (oder &)	UND-Verknüpfung	6. Priorität
OR (oder |)	logisch einschließliche ODER-Verknüpfung	7. Priorität
EXOR (oder @)	logisch exklusive ODER-Verknüpfung	8. Priorität

Anmerkung

Boolesche 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*. Boolesche 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). (z.B., CEIL(1.23) = 2; CEIL (-1.9) = -1).

INT

INT (x)

Ergibt den ganzzahligen Teil von x (immer ganzzahlig). (z.B., INT(1.23) = 1, INT(-1.23) = -2).

FRA

FRA (x)

Ergibt den Bruchzahlenteil von x. (z.B., FRA(1.23) = 0.23, 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, Arcussinus, Arcuscosinus und Arcustangens

ACS

ACS (x)

Ergibt den Arcuscosinus von x. (-1.0 <= x <= 1.0; 0° <= ACS(x) <= 180°).

ASN

ASN (x)

Ergibt den Arcussinus von x. (-1.0 <= x <= 1.0; -90° <= ASN(x) <= 90°).

ATN

ATN (x)

Ergibt die Arcustangens von x. (-90° <= ATN(x) <= 90°).

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 Kreiszahl PI (Ludolph Konstante) . (PI = 3.1415926...).

Anmerkung: Alle Ergebniswerte sind real.

Transzendentale Funktionen

EXP

EXP (x)

Ergibt die xte Potenz von e (Eulersche Zahl) (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.

Boolesche Funktionen

NOT

NOT (x)

Ergibt falsch (=0), wenn x wahr ist (<>0), und wahr (=1), wenn x falsch (=0) ist. (Logische Verneinung).

Anmerkung: Parameterwert sollte ganzzahlig sein.

Statistische Funktionen

MIN

MIN (x1, x2, ..., xn)

Ergibt den kleinsten Wert der Argumente.

MAX

MAX (x1, x2, ..., xn)

Ergibt das größte der Argumente aus einer beliebigen Anzahl.

RND

RND (x)

Ergibt einen Zufallswert zwischen 0 und x (Bedingung: x > 0, Ergebnis: real.).

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 Ausdruck (expr)s, 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.

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.

parameter_string: Abfragestring, einer der folgenden:

"GDL_version": Versionsnummer des GDL-Compilers/Interpreters . Warnung: 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":` Fragt die gegebenen Farbkomponenten eines Materials ab Der Rückgabewert ist in RGB-Werten zwischen 0 und 1.
`"Red_of_pen index"`
`"Green_of_pen index"`
`"Blue_of_pen index":` Definiert die vorgegebenen Stiftfarben-Anteile in RGB-Farbwerten zwischen 0 und 1,
`"Pen_of_RGB r g b":` Gibt den Index des Stiftes an, der der mit r, g und b vorgegebenen 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 REQUEST dar, der zweite den Gegenstand der Abfrage (falls einer existiert) und kann entweder vom Typ Text oder ein numerischer Typ sein. (z.B. kann die Frage, "Rgb_of_material" und sein Gegenstand der Materialname, oder "Rgb_of_pen" und sein Gegenstand der Index des Stiftes sein). Die anderen Parameter sind Variablennamen, in denen die Rückgabewerte (die Antworten) gespeichert werden.

Der Ergebniswert der REQUESTs ist immer die Anzahl der erfolgreich zurückgegebenen Werte (ganzzahlig), wohingegen der Typ der zurückgegebenen Werte bei jedem Aufruf festgelegt wird. Im Fall einer schlecht formulierten Frage oder eines nicht existierenden Namens ist der Rückgabewert 0.

ARCHICAD identifiziert die Reihenfolge und die Anzahl der Eingabeparameter entweder durch die Version des Befehls oder über den genauen Namen (als String-Konstante) der REQUEST-Option.Aktuell akzeptierte Variationen:

- n = REQUEST - Standard-Request, mit 1 Eingabeparameter als String oder numerischen Typs
- n = REQUEST{2} - 2 Eingabeparameter: String oder numerisch, Stringtyp
- n = REQUEST{3} - 2 Eingabeparameter: String, String oder numerischer Arraytyp
- n = REQUEST{4} - 3 Eingabeparameter: String oder numerisch, numerisch, Stringtyp. *Kompatibilität: eingeführt in ARCHICAD 21.*

Die Liste der verfügbaren Optionen finden Sie hier „REQUEST Optionen“.

IND

IND (**MATERIAL,** name_string)
IND (**BUILDING_MATERIAL,** name_string)
IND (**FILL,** name_string)
IND (**LINE_TYPE,** name_string)
IND (**STYLE,** name_string)
IND (**TEXTURE,** name_string)
IND (**PROFILE_ATTR,** name_string, index)

Diese Funktion gibt den aktuellen Index des Materials, Baustoffs, Schraffur, Linientyp oder Stil, Textur oder Profilattribut zurück. Die Hauptverwendung der Ergebniszahl ist ihre Übertragung zu einem Makro, das dasselbe Attribut wie das Aufrufmakro benötigt.

Die Funktionen geben einen Attribut-Indexwert (ganzzahlig) zurück. Das Ergebnis ist für vorübergehende Definitionen negativ (innerhalb des Scriptes oder aus einer Master_GDL-Datei), für globale Definitionen positiv (aus den Projekt-Attributen).

Siehe auch „Inline Attributdefinition“.

APPLICATION_QUERY

REQUEST (extension_name, parameter_string, variable1, variable2, ...)

Mit Hilfe von GDL können individuelle Applikationen spezifische Anforderungsfunktionen in ihrem Kontext zur Verfügung stellen. Diese Abfrageoptionen sind nicht im GDL-Syntax definiert; für spezifische Optionen schlagen Sie in der GDL-Entwicklerdokumentation der entsprechenden Applikation nach. *Siehe auch „Application Query Optionen“.*

LIBRARYGLOBAL

LIBRARYGLOBAL (object_name, parameter, value)

Füttert einen Wert eines GDL-Objektes mit dem aktuellen Modelldarstellungs-Parameterwert des globalen Bibliotheksobjektes bestimmt durch object_name falls vorhanden. Eine globale Bibliothekseinstellung ist verfügbar, wenn das globale Objekt aktuell in der Bibliothek geöffnet ist, oder früher geladen wurde und seine Einstellung in der aktuellen Modelldarstellungs-Kombination gesichert wurde..

Gibt 1 zurück, wenn erfolgreich, andernfalls 0.

object_name: der Name des globalen Bibliotheks-Objektes muss eine String-Konstante sein. Warnung: Falls String-Variablen oder -Parameter als Objektnamen verwendet werden, werden 2D- und 3D-Ansicht von Objekten, die dieses globale Library-Objekt abfragen, nicht automatisch neu aufgebaut.

parameter: Name des abgefragten Parameters

value: mit dem abgefragten Parameterwert gefüllt

Beispiel:

```
success = LIBRARYGLOBAL ("MyGlobalOptions", "detLevel2D", det)
if success > 0 then
    text2 0, 0, det
else
    text2 0, 0, "Nicht verfügbar"
endif
```

String-Funktionen

STR

STR (numeric_expression, length, fractions)
STR (format_string, numeric_expression)

Die erste Form der Funktion erstellt einen Text vom aktuellen Wert des numerischen Ausdruckes. Die Mindestanzahl für die Zeichenanzahl des Textstrings ist length, während fraction die Ziffern nach der Fließkommazahl angibt. Umfasst der konvertierte Wert mehr als nur length Zeichen, wird er wie gewünscht verbreitert. Hat er weniger Zeichen, wird er linksbündig (length > 0) oder rechtsbündig (length < 0) ausgerichtet.

Im zweiten Form kann der format_string entweder eine Variable oder eine Konstante sein. Ist das Format leer, so wird es als Meter, mit einer Genauigkeit von drei Dezimalen interpretiert (stellt Nullen dar).

Einschränkung der Parameter:

```
length >= -100, length <= 100
fractions <= 20, fractions < length
```

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
```

STR{2}

STR{2} (format_string, numeric_expression [, extra_accuracy_string])

Erweiterung der zweiten Form von STR. Wenn die Rundungsintervall-Flags im format_string gesetzt sind, gibt die STR{2} Funktion die entsprechende Zeichenfolge der Rundung im 3. Parameter zurück.

format_string: "%[0 oder mehr flags][Feldbreite][.Präzision] conv_spec"

flags: (für m, mm, cm, dm, e, df, di, sqm, sqcm, sqf, sqi, dd, gr, rad, cum, l, cucm, cumm, cuf, cui, cuy, gal):

(none): Blocksatz rechts (standard),
-: Blocksatz links,
+: explizites Plus-Zeichen ,
(space): an Stelle eines + Zeichens,

`'*0'`: extra Genauigkeit aus (standard),
`'*1'`: extra Genauigkeit .5,
`'*2'`: extra Genauigkeit .25,
`'*3'`: extra Genauigkeit .1,
`'*4'`: extra Genauigkeit .01,
`'*5'`: runden auf 0,5 innerhalb des angezeigten Dezimalbereichs, ohne Rückgabe eines extra genauen Stringwerts (verwendet für Flächenberechnungen),
`'*6'`: runden auf 0,25 innerhalb des angezeigten Dezimalbereichs, ohne Rückgabe eines extra genauen Stringwerts (verwendet für Flächenberechnungen),
`'*7'`: füllt den Bruch-Anteil von numeric_expression in den Wert extra_accuracy_string im Fall von fi oder ffi, während der Rückgabewert der Funktion den Bruch-Anteil nicht enthält,
`'#'`: Zeigt keine Nullen für (m, mm, cm, dm, ffi, fdi, fi, df, di, sqm, sqcm, sqf, sqi, dd, fr, rad, cum, l, cucm, cumm, cuf, cui, cuy, gal),
`'0'`: Zeigt 0 Zoll (für ffi, fdi, fi),
`'~'`: blendet 0 Dezimalzahlen aus (nur wirksam, wenn das '#' flag nicht gesetzt wurde) (für m, mm, cm, dm, fdi, df, di, sqm, sqcm, sqf, sqi, dd, fr, rad, cum, l, cucm, cumm, cuf, cui, cuy, gal),
`'^'`: ändert weder dezimale Trennzeichen noch Ziffern gruppierende Zeichen (Tausenderpunkt) (falls nicht angegeben werden die aktuellen Systemeinstellungen verwendet)
`'[1*j1+2*j2+4*j3]'`: Zeigen Sie 0 feet und 0 inches vor Brüchen an, effektiv, falls das Flag '0' nicht festgelegt ist (for ffi, fdi, fi)
`j1`: Zeigen Sie 0 vor Brüchen an (1'-0 3/4")
`j2`: Zeigen Sie 0 inches an (1'-0")
`j3`: Zeigen Sie 0 feet vor Brüchen an (0 3/4")

`field_width`: unbestimmte dezimale Zahl, die minimale Anzahl der zu erstellenden Zeichen

`precision`: unbestimmte dezimale Zahl, die Anzahl der zu erstellenden Nachkommastellen (muss ein Punkt vorangestellt werden)

`conv_spec`: (Konvertierungs-Spezifizierer):
`e`: exponentiales Format (Meter)
`m`: Meter
`mm`: Millimeter
`cm`: Zentimeter
`dm`: Dezimeter,
Kompatibilität: Dezimeter: eingeführt in ARCHICAD 22.
`ffi`: Fuß & Bruchzoll
`fdi`: Fuß & dezimale Zoll

`df:` dezimale Fuß
`fi:` Bruchzoll
`di:` dezimale Zoll
`pt:` Punkt (als Schriftgröße)

für Flächen:

`sqm:` Quadratmeter
`sqcm:` Quadratzentimeter
`sqmm:` Quadratmillimeter
`sqf:` Quadratfuß
`sqi:` Quadratzoll

für Winkel:

`dd:` dezimale Grad (Altgrad)
`dms:` Grad in Stunden/Minuten/Sekunden (Altgrad)
`gr:` Grad (Neugrad)
`rad:` Bogenmaß
`surv:` Vermessungseinheit

für Raummaße:

`cum:` Kubikmeter
`l:` Liter
`cucm:` Kubikzentimeter
`cumm:` Kubikmillimeter
`cuf:` Kubikfuß
`cui:` Kubikzoll
`cuy:` Kubikyards
`gal:` Gallone

Beispiel:

```
nr = 0.345678
TEXT2 0, 23, STR ("%m", nr)       !0.346
TEXT2 0, 22, STR ("%#10.2m", nr) !35
TEXT2 0, 21, STR ("%.4cm", nr)    !34.5678
TEXT2 0, 20, STR ("%12.4cm", nr) !  34.5678
TEXT2 0, 19, STR ("%.6mm", nr)    !345.678000
TEXT2 0, 18, STR ("%+15e", nr)    !+3.456780e-01
TEXT2 0, 17, STR ("%ffi", nr)     !1'-2"
TEXT2 0, 16, STR ("%0.16ffi", nr) !1'-1 5/8"
TEXT2 0, 15, STR ("% .3fdi", nr) ! 1'-1.609"
TEXT2 0, 14, STR ("% -10.4df", nr) ! 1.1341'
TEXT2 0, 13, STR ("%0.64fi", nr) !13 39/64"
TEXT2 0, 12, STR ("%+12.4di", nr)!+13.6094"
TEXT2 0, 11, STR ("%#.3sqm", nr) !346
TEXT2 0, 10, STR ("%+sqcm", nr)  !+3,456.78
TEXT2 0,  9, STR ("% .2sqmm", nr)! 345,678.00
TEXT2 0,  8, STR ("%-12sqf", nr) !3.72
TEXT2 0,  7, STR ("%10sqi", nr)  ! 535.80
TEXT2 0,  6, STR ("%.2pt", nr)    !0.35
alpha = 88.657
TEXT2 0,  5, STR ("%+10.3dd", alpha) !+88.657°
TEXT2 0,  4, STR ("%.1dms", alpha)   !88°39'
TEXT2 0,  3, STR ("%.2dms", alpha)   !88°39'25"
TEXT2 0,  2, STR ("%10.4gr", alpha)  ! 98.5078G
TEXT2 0,  1, STR ("%rad", alpha)     !1.55R
TEXT2 0,  0, STR ("%.2surv", alpha)  !N 1°20'35" E

nr  = 1'-0 3/4"
TEXT2 0, -1, STR ("%[1].16ffi", nr) !1'-0 3/4"
nr = 1'-0"
TEXT2 0, -2, STR ("%[5].16ffi", nr) !1'
nr = 0 3/4"
TEXT2 0, -3, STR ("%#[7].16ffi", nr) !0 3/4"
```

```
nr = 0.34278
TEXT2 0, 0, STR ("%*5 .4m", nr)  !0.3430

! Aufspaltung in Ganzzahl- und Bruchwert-Teile
extra_accuracy_string = ""
nr  = 1'-0 3/4"

TEXT2 0, -3, STR{2}("%*7.16ffi", nr,  extra_accuracy_string) !1'-
TEXT2 0, -4, extra_accuracy_string        !3/4"
```

SPLIT

SPLIT (string, format, variable1 [, variable2, ..., variablen])

Teilt den Zeichenfolge-Parameter in einen oder mehrere numerische oder textliche Teile nach der Teilungsdefinition in format. Der Teilungsprozess endet, wenn der erste ungeeignete Teil vorkommt. Gibt die Anzahl der erfolgreich eingelesenen Werte zurück (Ganzzahl).

string: der zu teilende Textstring

format: beliebige Kombination der Typenzeichen, %s, %n und %^n -s. Die Teile der Zeichenfolge müssen den Typenzeichen entsprechen, %s kennzeichnet jeden Textteil, der durch Leertasten oder Tabulatoren umgrenzt wird, während %n oder %^n einen numerischen Wert kennzeichnet. Falls das Zeichen '^' vorhanden ist, werden beim Abgleich der aktuellen numerischen Werte die gegenwärtigen Systemeinstellungen für das Dezimaltrennzeichen und die Zeichen der Zifferngruppierung berücksichtigt.

variablei: die Variablennamen, die die herausgetrennten Zeichenfolgeteile als Texte oder in Zahlen umgewandelt 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 Breite des im aktuellen Schriftstil dargestellten Textes in Millimeter (reale). Die maßstäbliche Breite in Meter ist STW (string_expression) / 1000 * GLOB_SCALE.

Beispiel:

```
DEFINE STYLE "own" "Gabriola", 180000 / GLOB_SCALE, 1, 0
SET STYLE "own"
string = "abcd"
width = STW (string) / 1000 * GLOB_SCALE
n = REQUEST ("Height_of_style", "own", height)
height = height / 1000 * GLOB_SCALE
TEXT2 0,0, string
RECT2 0,0, width, -height
```

STRLEN

STRLEN (string_expression)

Übergibt die Länge (Zeichenanzahl) des Strings (Ganzzahl)

STRSTR

STRSTR (string_expression1, string_expression2[, case_insensitivity])

Übergibt die Position der ersten Erscheinung der zweiten Zeichenfolge in der ersten Zeichenfolge. Wenn die erste Zeichenfolge die zweite nicht beinhaltet, gibt die Funktion 0 zurück. (Ganzzahl). Wenn die zweite Zeichenfolge ein Leerstring ist, gibt die Funktion 1 zurück. (Ganzzahl).

Hinweis: falls string_expression2 ein Leerstring ist, gibt die Funkion 1 zurück.

case_insensitivity:

0 or not set: Case-sensitiv
1: Case-unsensitiv

Beispiel 1:

```
szFormat = ""
n = REQUEST ("Linear_dimension", "", szFormat)
unit = ""
IF STRSTR (szFormat, "m")  > 0 THEN unit = "m"
IF STRSTR (szFormat, "mm") > 0 THEN unit = "mm"
IF STRSTR (szFormat, "cm") > 0 THEN unit = "cm"
IF STRSTR (szFormat, "dm") > 0 THEN unit = "dm"
TEXT2 0, 0, STR (szFormat, a) + " " + unit !1.00 m
```

Beispiel 2:

```
STRSTR ("abcdefg", "BCdEf") = 0
STRSTR ("abcdefg", "BCdEf", 0) = 0
STRSTR ("abcdefg", "BCdEf", 1) = 2
```

STRSUB

STRSUB (string_expression, start_position, characters_number)

Übergibt den Ausschnitt des Textes string_expression, der bei Position start_position beginnt und charakter_nummer Zeichen lang ist.

Beispiel:

```
string = "Flowers.jpeg"
len = STRLEN (string)
iDotPos = STRSTR (string, ".")
TEXT2 0, -1, STRSUB (string, 1, iDotPos - 1) !Flowers
TEXT2 0, -2, STRSUB (string, len - 4, 5)     !.jpeg
```

STRTOUPPER

STRTOUPPER (string_expression)

Gibt einen String zurück, der in Großbuchstaben konvertiert wurde.

Beispiel:

```
_oldString = "flower"
_newString = STRTOUPPER (_oldString)    ! _newString liefert "FLOWER"
```

STRTOLOWER

STRTOLOWER (string_expression)

Gibt einen String zurück, der in Kleinbuchstaben konvertiert wurde.

Beispiel:

```
_oldString = "FLOWER"
_newString = STRTOLOWER (_oldString)    ! _newString liefert "flower"
```

BEFEHLE ZUR PROGRAMMSTEUERUNG

In diesem Kapitel werden die GDL-Befehle vorgestellt, die für die Steuerung von Schleifen und Subroutinen in Scripts verfügbar sind; Sie lernen auch das Konzept des Arbeitens mit dem internen Pufferspeicher kennen, der Werte für die weitere Bearbeitung speichern kann. Hier wird auch beschrieben, wie man Objekte als Makros verwenden kann und wie berechnete Ausdrücke auf dem Bildschirm angezeigt werden können.

BEFEHLE ZUR PROGRAMMSTEUERUNG

FOR - TO - NEXT

FOR variable_name = initial_value **TO** end_value [**STEP** step_value]**NEXT** variable_name

FOR Erste Anweisung einer FOR-Schleife.

NEXT 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.

Werden das Schlüsselwort STEP und der Schrittwert step_value nicht angegeben, wird automatisch der Wert 1 angenommen.

Anmerkung: Das Ändern des Schrittwertes step_value während der Erzeugung der Schleife hat keine Auswirkung.

Die Verwendung einer globalen Variable als Schleifenvariable ist nicht zulässig.

Beispiel 1:

```
FOR i=1 TO 10 STEP 2
    PRINT i
NEXT i
```

Beispiel 2:

```
! Die beiden folgenden Progammteile sind gleichbedeutend:

! Erste
a = b
1:
IF c > 0 AND a > d OR c < 0 AND a < d THEN 2
PRINT a
a = a + c
GOTO 1

! Zweite
2:
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 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 - WHILE

```
DO [statement1
    statement2
    ...
    statementn]
WHILE condition
```

Die Anweisungen zwischen den Schlüsselwörtern werden solange ausgeführt, solange die Bedingung wahr ist.

Die Bedingung wird erst nach jeder Ausführung der Anweisungen kontrolliert.

WHILE - ENDWHILE

```
WHILE condition DO
    [statement1
    statement2
    ...
    statementn]
ENDWHILE
```

Die Anweisungen zwischen den Schlüsselwörtern werden solange ausgeführt, solange die Bedingung wahr ist.

Die Bedingung wird vor jeder Ausführung der Anweisungen kontrolliert.

REPEAT - UNTIL

```
REPEAT [statement1
    statement2
    ...
    statementn]
UNTIL condition
```

Die Anweisungen zwischen den Schlüsselwörtern werden ausgeführt, bis die Bedingung wahr ist.

Die Bedingung wird erst nach jeder Ausführung der Anweisungen kontrolliert.

Beispiel: Die folgenden vier Beispiele der GDL-Befehle sind gleichbedeutend:

```
! Erste
FOR i = 1 TO 5 STEP 1
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
NEXT i

! Zweite
i = 1
DO
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
    i = i + 1
WHILE i <= 5

! Dritte
i = 1
WHILE i <= 5 DO
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
    i = i + 1
ENDWHILE

! Vierte
i = 1
REPEAT
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
    i = i + 1
UNTIL i > 5
```

IF - GOTO

```
IF condition THEN label
IF condition GOTO label
IF condition GOSUB label
```

Sprungbefehl mit Bedingung. Wenn der Wert der Bedingung (condition) 0 ist (logisch falsch), hat der Befehl keine Wirkung, andernfalls wird die Programmbearbeitung bei der angegebenen Sprungmarke fortgesetzt. THEN, GOTO oder THEN GOTO sind äquivalent in diesem Kontext.

Beispiel:

```
IF a THEN 28
IF i > j GOTO 200+i*j
IF i > 0 GOSUB 9000
```

IF - THEN - ELSE - ENDIF

```
IF condition THEN statement [ELSE statement]
IF condition THEN
    [statement1
    statement2
    ...
    statementn]
[ELSE
    statementn+1
    statementn+2
    ...
    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 (ungleich Null). 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 = "Kugel" THEN
    ADDY b
    SPHERE 1
ELSE
    ROTX 90
    TEXT 0.002, 0, name
ENDIF
```

GOTO

GOTO label

Sprungbefehl ohne Bedingung. Die Programmausführung verzweigt zu dem Befehl, der nach einem Sprungmarke (label: numerisch oder Text) folgt. Ausdrücke und Variablenwerte verlangsamen die Skriptausführung, weil zur Laufzeit die Sprungmarke ermittelt werden muss.

Beispiel:

```
GOTO K+2
```

GOSUB

GOSUB label

Aufruf eines internen Unterprogrammes. Die Sprungmarke (label) markiert den Einstieg in das Programm. Die Sprungmarke kann numerisch oder vom Typ Text sein. Ausdrücke und Variablenwerte verlangsamen die Skriptausführung, weil zur Laufzeit die Sprungmarke ermittelt werden muss.

RETURN

RETURN

Das Programm kehrt aus dem internen Unterprogramm zurück.

END / EXIT

END [v1, v2, ..., vn]
EXIT [v1, v2, ..., vn]

Ende der aktuellen GDL-Scriptausführung. Das Programm wird beendet oder kehrt zu der darüberliegenden Hierarchieebene zurück. Es können mehrere END- oder EXIT-Befehle in einem GDL-Dokument verwendet werden. Folgt eine optionale Liste von Werten, übergibt diese das aktuelle Skript an das aufrufende Objekt, welches die Werte an die dort spezifizierten Variablen und Parameter übergibt .

Hinweis: Die Anzahl der möglichen zurückgegebenen Elemente (RETURNED_PARAMETERS) ist auf 32767 begrenzt.

Siehe die Beschreibung der Rückgabeparameter im CALL.

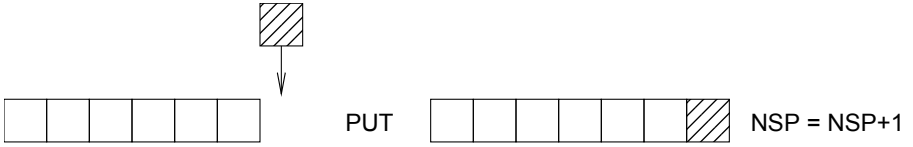
BREAKPOINT

BREAKPOINT expression

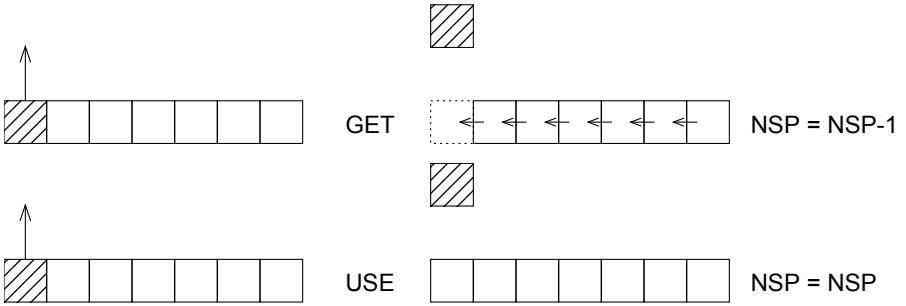
Durch diesen Befehl können Sie einen Unterbrechungspunkt im GDL-Script festlegen. Der GDL-Debugger stoppt bei diesem Befehl, wenn der Parameterwert (ein numerischer Ausdruck) wahr ist (ungleich 0) und die Einstellung des Debuggers Unterbrechungspunkte zuläßt. Im normalen Ausführungsmodus überspringt der GDL-Interpretierer 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 Pufferspeicher ist eine unendliche Reihe, in die Sie numerische Werte unter Verwendung des PUT-Befehls abspeichern können. PUT - Befehl speichert die angegebenen Werte am Ende des Puffers. Diese Werte können später (GET und USE Befehle) in der Reihenfolge benutzt werden, wie sie gespeichert wurden (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. Damit können sie in jeder GDL-Parameterliste benutzt werden, wo n Werte benötigt werden.



PUT

PUT expression [, expression, ...]

Speichert die angegebenen Werte in der angegebenen Reihenfolge in den internen Pufferspeicher.

GET

GET (n)

Holt sich die nächsten n-Werte des internen Pufferspeicher und löscht sie.

USE

USE (n)

Holt sich die nächsten n-Werte des internen Pufferspeicher ohne sie zu löschen. Nachfolgende USE- und GET-Befehle verwenden dieselbe Parametersequenz.

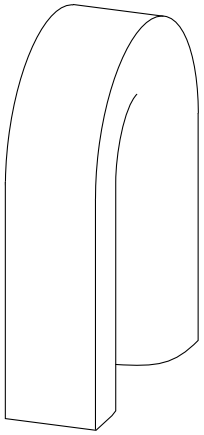
NSP

NSP

Gibt die Anzahl der im Pufferspeicher abgelegten Werte zurück.

Beispiel: 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
```

MAKRO-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 Makros. Ein GDL-Makro hat seine eigene Umgebung, die von der Reihenfolge des Aufrufs abhängig ist. Die aktuellen Werte von `MODEL`, `RADIUS`, `RESOL`, `TOLER`, `PEN`, `LINE_TYPE`, `MATERIAL`, `FILL`, `STYLE`, `SHADOW` die Optionen und die aktuelle Transformation sind auch im Makro gültig. Sie können die Werte dort benutzen oder modifizieren, dies wirkt sich jedoch nur lokal aus. Sie haben keine Auswirkungen auf diejenige Programmebene, aus welcher das Makro aufgerufen wurde. Wenn Sie Parameter einem Makro-Aufruf zuordnen, bedeutet das eine neue Wertzuweisung für das Makro. Parameter A und B werden immer zur Größenumstellung der Objekte verwendet.

CALL

```
CALL macro_name_string [,]
    PARAMETERS [ALL][name1=value1, ..., namen=valuen][[,]
    RETURNED_PARAMETERS r1, r2, ...]
```

macro_name_string: String, Name eines existierenden Bibliothekselements.

Die Namen der Makros dürfen nicht länger als 31 Zeichen sein. Makronamen können Zeichen-Konstanten, Zeichen-Variablen oder -Parameter sein. Zeichen-Operationen mit einem Macro-Aufruf können als Makronamen nicht verwendet werden. Warnung: Wenn Zeichen-Variablen oder Zeichen-Parameter als Makronamen verwendet werden, wird das aufgerufene Makro in das Archivprojekt nicht eingeschlossen werden. Um GDL diese Abhängigkeit mitzuteilen, verwenden Sie FILE_DEPENDENCE für jeden möglichen Makronamen. Der Makroname muss in Anführungszeichen (",',`,´,”,’,“,‘) gesetzt werden, damit es der Definition Identifizierern entspricht, er beginnt z.B. mit einem Buchstaben oder '_' oder dem '~' Zeichen und enthält nur Buchstaben, Zahlen und die Zeichen '_' und '~'. Außerdem müssen die vorderen und hinteren Anführungszeichen der CALL-Anweisung übereinstimmen und sollten sich von allen anderen Zeichen des Makronamens unterscheiden. Ein Makroname kann auch ohne das Schlüsselwort CALL als Befehl verwendet werden.

PARAMETERS: die konkrete Parameterliste des Makros kann folgen

Die Namen der Parameter des aufgerufenen Makros können beliebig gelistet werden, mit den beiden Anforderungen, dass ein '=' Zeichen und der konkrete Parameterwert jeweils angegeben werden. An dieser Stelle können Sie Zeichen-Typ Ausdrücke verwenden, es dürfen aber nur Stringparametern auch Textwerte zugewiesen werden. Ordnungsparametern Arrays müssen vollständige Arrays zugeordnet werden. Sollte ein Parametername in der Parameterliste des aufgerufenen Makros nicht gefunden werden, dann erscheint eine Warnung. Die nicht aufgelisteten Parameter des aufgerufenen Makros werden ihre ursprünglichen voreingestellten Werte, wie im Bibliothekselement als Makro definiert, bekommen.

ALL: alle Parameter des Aufrufers werden an den aufgerufenen Makro übergeben

Fall das Schlüsselwort vorhanden ist, brauchen die Parameter nicht einzeln spezifiziert werden. Für einen Parameter, der im aufrufenden Objekt nicht vorgegeben wird, wird der Standardwert des Objektes aus dessen Parameterliste eingesetzt. Wenn die Parameterwerte nacheinander definiert werden, werden sie die Werte, die von dem aufrufenden Objekt kommen, außer Kraft setzen oder die Parameter des aufgerufenen Objektes bleiben auf den Standardwerten.

RETURNED_PARAMETERS: Eine Variablenliste kann folgen, um die zurückgegebenen Parameter des Makros zu listen.

Auf der aufrufenden Seite können zurückgegebene Werte durch das Setzen von RETURNED_PARAMETERS, gefolgt von einer Variablenliste weiterverwendet werden. Die zurückgegebenen Werte werden in diesen Variablen in der Reihenfolge abgelegt, wie sie vom aufgerufenen Makro übergeben werden. Die Anzahl und der Typ der Variablen, wie in dem aufrufenden Objekt angegeben, und jene, die im Makro zur Übergabe gelistet sind, müssen übereinstimmen. Wenn mehr Variablen im aufrufenden Objekt vorliegen, als Werte im Makro, werden die überschüssigen Variablen auf 0 gesetzt. Die Typenkompatibilität wird nicht geprüft: Der Typ der Variablen im aufrufenden Objekt wird an die übergebenen Werte angepaßt. Wenn im aufrufenden Objekt in der Liste ein Array enthalten ist, werden dort alle zurückgegebenen Werte abgelegt. *Hinweis: Die Anzahl der möglichen zurückgegebenen Elemente (RETURNED_PARAMETERS) ist auf 32767 begrenzt. Siehe die Syntax der wiederkehrenden Parameter bei END / EXIT.*

```
CALL macro_name_string [,]PARAMETERS
   value1 or DEFAULT [, ..., valuen or DEFAULT]
```

Diese Form von Makro-Aufrufen kann aus Kompatibilitätsgründen mit den früheren Versionen benutzt werden. Verwendet man diese Syntax, müssen die aktuellen Parameterwerte einzeln spezifiziert werden und zwar in der Reihenfolge, in welcher sie im aufgerufenen Makro vorhanden sind, kein Wert darf ausgelassen werden, außer am Ende der Liste. An Stelle eines konkreten Parameterwertes kann auch das Schlüsselwort DEFAULT verwendet werden. Es übergibt den Standardwert des Objektes aus seiner gespeicherten Parameterliste. Ebenso werden für alle fehlenden Werte (die Liste der Werte kann kürzer als die Liste der Parameter sein) die Standardwerte der Parameter des aufgerufenen Objektes übergeben. Bei der Ausführung dieser Form des Makroaufrufs muss der Interpreter nicht die Parameternamen suchen, um ihnen die aktuellen Werte zuzuweisen. Auch wenn dies umständlicher zu programmieren ist als die vorher vorgestellten Arten, ist damit eine bessere Performance zu erzielen.

CALL macro_name_string [, parameter_list]

Diese Form von Makro-Aufrufen kann aus Kompatibilitätsgründen mit den früheren Versionen benutzt werden. Kann sowohl mit einfachen GDL-Textdateien als auch mit beliebigen Bibliothekselementen verwendet werden, unter der Bedingung, dass die Parameterliste nur numerische Parameter mit einem einzigem Buchstaben enthalten (A ... Z). Es sind keine String-Typ-Parameter oder Array-Parameter bei dieser Methode erlaubt . Die Parameterliste ist eine Liste mit einfachen numerischen Werten: der Wert des Parameters A wird der erste Wert in der Liste, der Wert des Parameters B wird der zweite Wert sein usw. Falls weniger als die A ... Z Werte in der Parameterliste spezifiziert sind, wird für die fehlenden Werte automatisch der Wert 0 verwendet. Enthält das Makro (Bibliothekselement) keinen Einzelbuchstaben-Parameter, der dem Wert entspricht, dann setzt die Interpretation nach dem Verwerfen dieses Wertes fort, gibt aber eine Warnmeldung aus.

Beispiel:

```
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
```

Ausgabe in einem Popup-Fenster oder in einem Report-Fenster

PRINT

PRINT expression [, expression, ...]

Schreibt sämtliche Argumente in ein Dialogfeld (Popup) oder in ein Report-Fenster, abhängig von der Arbeitsumgebung (siehe „GDL-Warnungen“). Argumente können Zeichenfolgen oder numerische Ausdrücke beliebiger Anzahl in beliebiger Reihenfolge darstellen, sie werden durch Kommas getrennt.

Beispiel:

```
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 externe Dateien zum Lesen und Schreiben öffnen sowie Werte aus /in GDL-Scripts einfügen und auslesen. Dieser Prozeß bezieht notwendigerweise spezielle Add-Ons mit ein. TEXT-Dateien können durch das „GDL Text I/O Add-On"behandelt werden. Add-Ons für andere Dateitypen können von Drittanbietern entwickelt werden.

Siehe auch „GDL Text I/O Add-On".

OPEN

OPEN (filter, filename, parameter_string)

Öffnet eine Datei wie angewiesen. Der Rückgabewert ist ein positiver Ganzzahlwert, welcher die spezifische Datei identifiziert, -2, falls das Addon fehlt, -1 falls die Datei fehlt. Falls positiv, ist dieser Wert, die Kanalnummer, die Referenznummer der Datei in den nachfolgenden Instanzen. Um diese referenzierte Datei in ein Archivprojekt mit einzubinden verwenden sie FILE_DEPENDENCEzusammen mit dem Dateinamen.

filter: Zeichenfolge, name einer vorhandenen Erweiterung.

filename: Zeichenfolge, Dateiname.

parameter_string: Zeichenfolge, enthält die spezifischen Trennungszeichen der verwendeten Erweiterung und den Modus des Öffnens. Der Inhalt wird in der Erweiterung interpretiert.

INPUT

INPUT (channel, recordID, fieldID, variable1 [, variable2,...])

Die Anzahl der angegebenen Parameter definiert die Anzahl der Werte die aus der mit der Kanalnummer (channel) identifizierten Datei ausgelesen werden. Der erste Werte wird von der definierten Anfangsposition gelesen und der ersten Variablen zugewiesen. Die Parameterliste muss wenigstens einen Wert enthalten. Die gelesenen Werte werden in die Parameter in der Reihenfolge wie ausgelesen übergeben. Diese Werte können vom numerischen- oder vom Zeichentyp sein, unabhängig von dem für die Speicherung verwendeten Parametertyp.

Der Ergebniswert ist die Anzahl der eingelesenen Werte. Wenn man ans Dateiendezeichen gelangt, dann beträgt er -1.

recordID, fieldID: Die Anfangsposition des Lesens vom textlichen oder numerischen Typ, dessen Inhalt von der Erweiterung interpretiert wird.

VARTYPE

VARTYPE (expression)

Ergibt den Variablentyp von expression:

- 1 - numerisch
- 2 - Zeichenfolge
- 3 - Gruppe (als Ergebnis von ADDGROUP und ähnliche)
- 4 - Dictionary

Nützlich beim Lesen von Werten in Variablen mit dem BefehlINPUT, der zwischen Variablentyp 1 und 2 entsprechend den ausgelesenen Werten ändern kann. Der Typ dieser Variablen wird während der Kompilierung nicht geprüft.

OUTPUT

OUTPUT channel, recordID, fieldID, expression1 [, expression2, ...]

Schreibt ebenso viele Werte ab der Anfangsposition in die Datei ein, wie Werte (expression1) aufgelistet sind. Die Zieldatei wird durch den Kanalwert bestimmt. Es muss mindestens ein Ausdruck (expression) vorhanden sein. Der Typ der Werte bestimmt sich nach dem der Ausdrücke.

recordID, fieldID: Die Anfangsposition des Lesens vom textlichen oder numerischen Typ, Der Inhalt wird in der Erweiterung interpretiert.

CLOSE

CLOSE channel

Schließt die Datei, die durch den Kanalwert identifiziert wird.

VERWENDEN VON DETERMINISTISCHEN ADD-ONS

Mit Hilfe folgender Schlüsselwörter können Sie GDL Add-Ons aufrufen, die eine determinierende Funktion anbieten, d.h. das Ergebnis einer bestimmten Operation ist nur von den angegebenen Parametern abhängig. Dieser Prozeß bezieht notwendigerweise spezielle Add-Ons mit ein. Polygon-Operationen können zum Beispiel über das Add-On PolyOperationen ausgeführt werden. Add-Ons für andere Operationen können von Drittanbietern entwickelt werden.

Siehe auch „Polygon-Operationserweiterung“.

INITADDONSCOPE

INITADDONSCOPE (extension, parameter_string1, parameter_string2)

Öffnet einen Kanal wie angewiesen. Sein Rückgabewert ist eine positive ganze Zahl, die die spezifische Verbindung identifiziert. Dieser Wert, die Kanalnummer, ist die Referenznummer einer erfolgreichen Verbindung.

extension: Zeichenfolge, name einer vorhandenen Erweiterung.

parameter_string1: Zeichenfolge, dessen Inhalt durch die Erweiterung interpretiert wird.

parameter_string2: Zeichenfolge, dessen Inhalt durch die Erweiterung interpretiert wird

PREPAREFUNCTION

PREPAREFUNCTION channel, function_name, expression1 [, expression2, ...]

Setzt einige Werte in dem Add-On als Vorbereitungsschritt zum Aufrufen einer späteren Funktion.

function_name: Zeichenfolge oder numerischer Identifizierer der aufzurufenden Funktion; Der Inhalt wird in der Erweiterung interpretiert.

expression: Parameter, die an den Vorbereitungsschritt weitergegeben werden.

CALLFUNCTION

CALLFUNCTION (channel, function_name, parameter, variable1 [, variable2, ...])

Die Funktion mit dem Namen *function_name* in dem Add-On, angegeben von *channel*, wird aufgerufen. Die Parameterliste muss wenigstens einen Wert enthalten. Die zurückgegebenen Werte werden in die Parameter in der Reihenfolge wie ausgelesen übergeben. Der Rückgabewert ist die Anzahl der erfolgreich gesetzten Werte.

channel: Kanalwert, verwendet um die Verbindung zu identifizieren.

function_name: Zeichenfolge oder numerischer Identifizierer der aufzurufenden Funktion; Der Inhalt wird in der Erweiterung interpretiert.

parameter: Eingabeparameter; Der Inhalt wird in der Erweiterung interpretiert.

variablei: Ausgabeparameter.

CLOSEADDONSCOPE

CLOSEADDONSCOPE channel

Schließt die Verbindung, die durch den Kanalwert identifiziert wird.

VERSCHIEDENES

In diesem Kapitel sind eine Reihe zusätzlicher Befehle, Techniken und Informationen zusammengefasst. Hier findet sich eine umfassende Liste aller System- und Umgebungszustände, die vom GDL-Script abgefragt werden können. Ebenso findet sich eine Anleitung, wie man graphisch GDL-Text erzeugen kann. 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 zwischengespeichert und nach dem Ende des Makroaufrufs wiederhergestellt.

Für Türen, Fenster, Etiketten und Eigenschaftenobjekte gibt es außerdem die Möglichkeit mit ARCHICAD über optionale, eindeutig standardisiert benannte Parameter in der Parameterliste des Objektes zu kommunizieren. Sind diese Parameter in der Parameterliste, werden sie mit Scriptstart von ARCHICAD mit den aktuell geltenden Werten versehen. *Siehe auch die Liste der "fix named parameters" und weiterer Details in „Festbenannte optionale Parameter“.*

Script Kompatibilität

Sicht- oder Projektabhängige Globale Variablen sollten nicht im Parameter-Script verwendet werden (oder als Parameter-Script ausgeführte Master-Scripte) um zu verhindern, dass das Parameter-Script unbeabsichtigt durchlaufen wird und die resultierenden Parameterwerte kontextabhängig werden, inkonsistent innerhalb der PLN-Datei.

Kompatibilität bis zu ARCHICAD 19: Solche versehentlich im Parameter-Script verwendeten Globalen Variablen erzeugen GDL-Warnungen.

Kompatibilität beginnend mit ARCHICAD 20: Solche im Parameter-Script verwendeten Globalen Variablen erzeugen GDL-Warnungen, und besitzen ausschließlich einen statischen Defaultwert (zum Variablentyp passend).

Kompatibilität beginnend mit ARCHICAD 22: Sichtabhängige globale Variablen sollten nicht im Eigenschaften-Script verwendet werden (oder Master-Script, das als Eigenschaften-Script ausgeführt wird). Solche Instanzen verursachen Warnungen in den Scripten. Projektabhängige Variablen sind jedoch in den meisten Fällen im Eigenschaften-Script aktiviert.

Für Kompatibilitätsdetails sehen Sie sich die Beschreibungen der Globalen Variablen an. Scripttyp-Einschränkungen gelten, wo angegeben.

Legende

✅	funktioniert ohne Einschränkungen
⚠	funktioniert (mit ergänzenden Warnungen)
⛔	besitzt einen Dummy-Defaultwert (mit ergänzender Warnung)

Allgemeine Umgebungsinformationen

GLOB_SCRIPT_TYPE Typ des aktuellen Scriptes

- *1 - Eigenschaften-Script*
- *2 - 2D-Script*
- *3 - 3D-Script*
- *4 - User Interface Script*
- *5 - Parameter-Script*
- *6 - Master-Script*
- *7 - Das Vorwärts-Migration Script*
- *8 - Das Rückwärts-Migration Script*

GLOB_VIEW_TYPE Typ der aktuellen Sicht (sichtenabhängig, nicht im Parameter/Eigenschaften-Script verwenden)

2D	✅	3D	✅	UI	⚠	Parameter	⚠	Property	⛔	Standard	-

- *2 - 2D (Grundriss)*
- *3 - 3D*
- *4 - Schnitt*
- *5 - Ansicht*
- *6 - 3D-Dokument*
- *7 - Detail*
- *8 - Layout*
- *9 - Berechnung*

Verwenden Sie die exakt benötigten Werte. Die Verwendung von Wertebereichen (RANGE) wird nicht empfohlen wegen der möglichen zukünftigen Werteerweiterungen.

GLOB_PREVIEW_MODE

Art der aktuellen Vorschau (Sichten-abhängig, nicht im Parameter/Eigenschaften-Script verwenden)

- *0 - Keine*
- *1 - Dialog*
- *2 - Liste*
- *3 - Favoritensicherung*

Verwenden Sie die exakt benötigten Werte. Die Verwendung von Wertebereichen (RANGE) wird nicht empfohlen wegen der möglichen zukünftigen Werteerweiterungen.

GLOB_FEEDBACK_MODE

zeigt eine Bearbeitung, die im Gange ist (sichtenabhängig, nicht im Parameter/Eigenschaften-Script verwenden)

0 - aus, 1 - Editier-Feedbackmodus

GLOB_SEO_TOOL_MODE

zeigt Solid Element Operationen, die im Gange sind (sichtenabhängig, nicht im Parameter/Eigenschaften-Script verwenden)

0 - aus, 1 - Solid Element Operations-Modus

GLOB_DIAGNOSTICS_MODE

Befehl im Menü "Bibliotheken-Entwickler"(59) für GDL-Diagnosen

Kompatibilität: eingeführt in ARCHICAD 23.

0 - aus, 1 - an

Verwendung in Scripten als bedingte Anweisung zur Visualisierung des Debug-Inhalts von Bibliothekselementen.

GLOB_SCALE

Zeichnungsmaßstab (sichtenabhängig, nicht im Parameter/Eigenschaften-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	100

dem aktuellen Fenster entsprechend

GLOB_DRAWING_BGD_PEN

Stift des Bildschirmhintergrundes (sichtenabhängig, nicht im Parameter/Eigenschaften-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	19

der am besten zur Hintergrundfarbe des aktuellen Fensters passende (druckbare) Stift aus der aktuellen Palette

GLOB_FILL_INDEX_SOLID Index des Schraffurtyps "Massiv" entsprechend der Projektvorlagedatei (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	Standard	16

enthält den angewendeten Index der Schraffurart "Massiv"

Kompatibilität: eingeführt in ARCHICAD 22.

GLOB_FILL_INDEX_BACKGROUND Index des Schraffurtyps "Hintergrund" entsprechend der Projektvorlagedatei (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	Standard	16

enthält den angewendeten Index der Schraffurart "Hintergrund"

Kompatibilität: eingeführt in ARCHICAD 22.

GLOB_NORTH_DIR Projekt-Nord-Richtung (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	Standard	90

mit Bezug auf das Standardprojekt, entspricht das Koordinatensystem den Einstellungen im Dialogfenster Projektstandort

GLOB_PROJECT_LONGITUDE Projekt-Längengrad (projektabhängig, nicht im Parameter-Script verwenden)

GLOB_PROJECT_LATITUDE Projekt-Breitengrad (projektabhängig, nicht im Parameter-Script verwenden)

GLOB_PROJECT_ALTITUDE Projekt-NN-Höhe (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	Standard	0

die geografischen Koordinaten des Projekt-Ursprungs bezogen auf die Einstellungen, die in dem Städtelisten-Dialog festgelegt werden.

GLOB_PROJECT_DATE Projekt-Datum (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	Standard	[0, 0, 0, 0, 0, 0]

Array mit folgenden 6 Werten: 1 - Jahr, 2 - Monat, 3 - Tag, 4 - Stunde, 5 - Minute, 6 - Sekunde. Dies Variable enthält das aktuelle Datum des Projekts und wird nur im EcoDesigner STAR® Addon gesetzt (in anderen Fällen werden alle Werte auf 0 gesetzt). der Wert dieser Variable wird vom Addon modifiziert, wenn es die Sonnenanalyse-Routinen durchläuft um bestimmten GDL-Objekten (z.B. Laubbäumen) zu erlauben, in unterschiedlichen Jahreszeiten unterschiedlich dargestellt zu werden.

GLOB_WORLD_ORIGO_OFFSET_X (projektabhängig, nicht im Parameter-Script verwenden)

GLOB_WORLD_ORIGO_OFFSET_Y (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	Standard	0

Position des Projektursprungs bezogen auf den globalen Ursprung. Siehe zur Illustration der globalen Variablen GLOB_WORLD_ORIGO_... globals.

GLOB_MODPAR_NAME Name des letzten geänderten Parameters

im Einstellungsdialog, im Editor für Bibliothekselemente oder wenn Werte, die durch editierbare Hotspots modifiziert werden.

GLOB_UI_BUTTON_ID id der Schaltfläche

die im Benutzerinterface gedrückt wurde oder 0, wenn zuletzt keine Schaltfläche mit ID gedrückt wurde.

GLOB_CUTPLANES_INFO (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	Standard	[1.,, 3.0, -0.1, -0.1]

Array mit 4 Spalten von Längenwerten: Array mit 4 Spalten von Längenwerten: 1 - Höhe der Schnittfläche, 2 - Oberste Höhe des Schnittbereichs, 3 - Unterste Höhe des Schnittbereichs, 4 - Absolute Darstellungsbegrenzung Siehe Details im Dialog der Grundriss- Schnittebenen-Einstellung Siehe Details im Dialog der Grundriss- Schnittebenen-Einstellung.

GLOB_STRUCTURE_DISPLAY Strukurdarstellung (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	Standard	0

informiert über die Option der partiellen Strukturdarstellungseinstellung (ganze Zahl): 0 - komplette Struktur, 1 - nur der Kern, 2 - ohne Bekleidungen

GLOB_ISSUE_SCHEME Liste mit benutzerdefinierten Daten, die im Ausgabeschema definiert wurden

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	Standard	-

Verfügbar in jedem Kontext. 2-zeiliger String-Array, enthält die im Ausgabeschema definierten Namensfelder (erste Zeile), mit den dazugehörigen GUIDs (zweite Zeile). Die ersten 5 Spalten sind fest vorgegeben: Revision ID, Issue ID, Issue Name, Issue Date, bearbeitet von.

Ein Beispiel:

Revision ID	***Issue ID***	***Issue Name***	***Issue Date***	***Issued By***	***Recipient***	***Status-Werte***	...
{RevIdGUID}	*{IssueIdGUID}*	*{IssueNameGUID}*	*{IssueDateGUID}*	*{IssuedByGUID}*	*{Custom1GUID}*	*{Custom2GUID}*	

LAYOUT_REVISION_HISTORY Liste der gegenwärtigen Revisionshistorie des Layouts

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	Standard	-

Nur im Layout-Kontext verfügbar. String-Array, enthält je Revision 1 Zeile, mit der selben Struktur wie GLOB_ISSUE_SCHEME. Die ersten 5 Spalten sind fest vorgegeben: Revision ID, Issue ID, Issue Name, Issue Date, bearbeitet von.

Ein Beispiel:

01	*1*	*Erste Ausgabe*	*2013-06-30*	*Zeichner 1*	*Jeder*	*SD*	...
02	*3*	*Großes Update*	*2013-07-31*	*Zeichner 2*	*Lüftung*	*DD*	
03	*5*	*Statiker-Update*	*2013-08-31*	*Zeichner 1*	*Tragend*	*DD*	
...							

GLOB_CHANGE_SCHEME Liste von benutzerdefinierten Daten, welche im Änderungsschema definiert werden.

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	Standard	-

Verfügbar in jedem Kontext. 2-zeiliger String-Array, enthält die im Ausgabeschema definierten Namensfelder (erste Zeile), mit den dazugehörigen GUIDs (zweite Zeile). Die ersten 5 Spalten sind fest vorgegeben: Revision ID, Change ID, Change Name, Last Modified Date, Last Modified by.

Ein Beispiel:

Revision ID	***Change ID***	***Change Description***	***Last Modified***	***Last Modified By***	***Created by***	***Approved by***	...
{RevIdGUID}	*{ChIdGUID}*	*{ChDescGUID}*	*{ModiTimeGUID}*	*{ModiByGUID}*	*{Custom1GUID}*	*{Custom2GUID}*	

LAYOUT_CHANGE_HISTORY Liste aller Änderungen, welche in der aktuellen Ausgabehistorie erscheint

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	Standard	-

Nur im Layout-Kontext verfügbar. String-Array, enthält 1 Zeile je Änderung, mit der selben Struktur wie GLOB_CHANGE_SCHEME. Die ersten 5 Spalten sind fest vorgegeben: Revision ID, Change ID, Change Name, Last Modified Date, Last Modified by.

Ein Beispiel:

2	*Ch-13*	*Kitchen*	*2013-07-13*	*Zeichner 1*	*Architect 1*	*Lead Architect 1*	...
2	*Ch-15*	*Ventilation*	*2013-07-16*	*Zeichner 2*	*Architect 2*	*Lead Architect 1*	
3	*Ch-18*	*Structural Col.*	*2013-08-03*	*Zeichner 2*	*Architect 1*	*Lead Architect 2*	
3	*Ch-19*	*Truss Sections*	*2013-08-12*	*Zeichner 1*	*Architect 3*	*Lead Architect 2*	
B	*Ch-23*	*Door Numbering*	*2013-10-01*	*user3*	*Architect 2*	*Lead Architect 1*	
...							

LAYOUT_CURRENTREVISION_OPEN Staus "in Bearbeitung" des aktuellen Layouts (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	Standard	0

Nur im Layout-Kontext verfügbar. 0 - aktuelles Layout besitzt keine geöffnete Revision, 1 - aktuelles Layout besitzt eine geöffnete Revision ("in Arbeit" - Layout)

Geschossinformationen

GLOB_HSTORY_ELEV Höhenlage des Ursprungsgeschosses (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	Standard	0

Ursprungsgeschoss - Geschoss, in dem das Objekt platziert wird

GLOB_HSTORY_HEIGHT Höhe des Ursprungsgeschosses (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	Standard	3.1

Ursprungsgeschoss - Geschoss, in dem das Objekt platziert wird

GLOB_CSTORY_ELEV — Höhenlage des aktuellen Geschosses (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	Standard	0.0

= *Aktuelle Geschossnummer - Ursprungsgeschossnummer*

GLOB_CSTORY_HEIGHT — Höhe des aktuellen Geschosses (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	Standard	3.1

= *Aktuelle Geschossnummer - Ursprungsgeschossnummer*

GLOB_CH_STORY_DIST — Relative Position des aktuellen Geschosses in Bezug auf das Ursprungsgeschoss (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	Standard	0.0

= *Aktuelle Geschossnummer - Ursprungsgeschossnummer*

Animationsinformationen

GLOB_FRAME_NR — aktuelle Frame-Nummer in der Animation (sichtenabhängig, nicht im Parameter/Eigenschaften-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	-1

nur für Animation gültig, -1 ist für Standbilder

GLOB_FIRST_FRAME — erster Frame-Index im Flugmodus (sichtenabhängig, nicht im Parameter/Eigenschaften-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

nur für Animation gültig, 0 ist für Standbilder

GLOB_LAST_FRAME — letzter Frame-Index im Flugmodus (sichtenabhängig, nicht im Parameter/Eigenschaften-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

nur für Animation gültig, 0 ist für Standbilder

GLOB_EYEPOS_X aktuelle Kamerapostion (x) (sichtenabhängig, nicht im Parameter/Eigenschaften-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	-5.0

gilt nur in der Perspektivprojektion für Animation und Standbilder

GLOB_EYEPOS_Y aktuelle Kamerapostion (y) (sichtenabhängig, nicht im Parameter/Eigenschaften-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	-5.0

gilt nur in der Perspektivprojektion für Animation und Standbilder

GLOB_EYEPOS_Z aktuelle Kamerapostion (z) (sichtenabhängig, nicht im Parameter/Eigenschaften-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	1.7

gilt nur in der Perspektivprojektion für Animation und Standbilder

GLOB_TARGPOS_X aktuelle Zielposition (x) (sichtenabhängig, nicht im Parameter/Eigenschaften-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	0.0

gilt nur in der Perspektivprojektion für Animation und Standbilder

GLOB_TARGPOS_Y aktuelle Zielposition (y) (sichtenabhängig, nicht im Parameter/Eigenschaften-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	0.0

gilt nur in der Perspektivprojektion für Animation und Standbilder

GLOB_TARGPOS_Z aktuelle Zielposition (z) (sichtenabhängig, nicht im Parameter/Eigenschaften-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	1.7

gilt nur in der Perspektivprojektion für Animation und Standbilder

GLOB_SUN_AZIMUTH Sonnen-Azimuth (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	Standard	240

entsprechend den Einstellungen im Dialogfenster Sonne...

GLOB_SUN_ALTITUDE Sonnenhöhe (projektabhängig, nicht im Parameter-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	Standard	35

entsprechend den Einstellungen im Dialogfenster Sonne...

Allgemeine Parameter von Elementen

GLOB_LAYER Ebene des Elementes

Name der Ebene, der das Element zugewiesen wird

GLOB_ID Benutzer-ID des Elementes

ID, wie im Dialogfenster für Einstellungen eingegeben

GLOB_INTGUID interne GUID des Elementes

die interne GUIID-Nummer, die durch das Programm generiert wird (kann von dem Benutzer nicht kontrolliert werden)

GLOB_ELEVATION Basishöhenlage des Elementes

- *Tür/Fenster - Objekte: Brüstungshöhe, entsprechend den aktuellen Einstellungen*
- *Decke: die Höhenlage der gewählten Referenz-Ebene der Decke, entsprechend den Einstellungen*
- *andere Elemente/Objekte: die Basishöhenlage, entsprechend den aktuellen Einstellungen*

GLOB_ELEM_TYPE Elementtyp für Etiketten und Eigenschaftsobjekte, enthält den Typ des assoziierten Elementes

0 - ohne (individuelles Etikett), 1 - Objekt, 2 - Leuchte, 3 - Fenster, 4 - Tür, 5 - Wand, 6 - Stütze, 7 - Decke, 8 - Dach, 9 - Schraffur, 10 - Freifläche, 11 - Raumstempel, 12 - Träger, 13 - Fassade, 14 - Fassaden-Profil, 15 - Fassaden-Paneel, 16 - Fassaden-Halterung, 17 - Fassaden-Zubehör, 18 - Schale, 19 - Dachfenster, 20 - Morph, 21 - Treppe, 22 - Treppen-Trittstufe, 23 - Treppen-Setzstufe, 24 - Treppen-Struktur, 25 - Geländer, 26 - Öffnung, 27 - Stützen-Segment, 28 - Träger-Segment.

Parameter von Objekt, Lampe, Tür, Fenster, Wandende Dachfenster

SYMB_LINETYPE Linientyp des Bibliothekselementes

enthält den grundeingestellte Linientyp des 2D-Symbols

SYMB_FILL Schraffurtyp des Bibliothekselementes

angewandt auf geschnittene Oberflächen der Bibliothekselemente in den Schnitte/Ansichten-Fenstern

SYMB_FILL_PEN Stift der Schraffur des Bibliothekselementes

angewandt auf geschnittene Oberflächen der Bibliothekselemente in den Schnitte/Ansichten-Fenstern

SYMB_FBGD_PEN	Stift des Schraffurhintergrundes des Bibliothekselementes
angewandt auf geschnittene Oberflächen der Bibliothekselemente in den Schnitte/Ansichten-Fenstern	
SYMB_SECT_PEN	Stift des Bibliothekselementes im Schnitt
angewandt auf Konturen geschnittener Oberflächen von Bibliothekselementen in den Schnitte/Ansichten-Fenstern	
SYMB_VIEW_PEN	grundeingestellter Stift des Bibliothekselementes
verwendet für alle Kanten im 3D-Fenster, alle sichtbaren Kanten in den Schnitte/Ansichten-Fenstern und der Standardwert im Grundriss.	
SYMB_MAT	Standard-Oberflächenattributindex des Bibliothekelements
SYMB_POS_X	Position des Bibliothekselementes (x)
im Verhältnis zum Projektursprung (für Türen, Fenster und Wandenden: im Verhältnis zum Startpunkt der beinhaltenden Wand)	
SYMB_POS_Y	Position des Bibliothekselementes (y)
im Verhältnis zum Projektursprung (für Türen, Fenster und Wandenden: im Verhältnis zum Startpunkt der beinhaltenden Wand) Hinweis: siehe „Türen und Fenster" zur Orientierung der Y- und Z-Achsen	
SYMB_POS_Z	Position des Bibliothekselementes (z)
im Verhältnis zum Projektursprung (für Türen, Fenster und Wandenden: im Verhältnis zum Startpunkt der beinhaltenden Wand) Hinweis: siehe „Türen und Fenster" zur Orientierung der Y- und Z-Achsen	
SYMB_ROTANGLE	Drehwinkel des Bibliothekselementes
numerische Drehung - wird vom Dialogfenster Einstellungen aus um den aktuellen Ankerpunkt ausgeführt	
SYMB_MIRRORED	Bibliothekselement gespiegelt
0-nicht, 1-gespiegelt (Spiegelung wird um den aktuellen Ankerpunkt ausgeführt). Immer 0 für Wandenden, ausgenommen, wenn der Ursprung des lokalen Koordinatensystems ein nicht-rechteckiger Scheitelpunkt eines trapezförmigen Chorpolygons ist.	

Parameter von Objekt, Lampe, Tür, Fenster, Wandende, Dachfenster, Fassadenzubehör, - nur für Listen und Etiketten verfügbar

SYMB_A_SIZE	nominale Länge/Breite des Bibliothekselementes
Länge von Objekt/Lampe, Breite von Fenster/Tür (festgesetzte Parameter) Breite des Zubehörs	
SYMB_B_SIZE	nominale Breite/Höhe des Bibliothekselementes
Breite von Objekt/Lampe, Höhe von Fenster/Tür (festgesetzte Parameter) Höhe des Zubehörs	

Parameter von Objekten, Lampen und Fassadenzubehör - nur für Listen und Etiketten verfügbar

SYMB_Z_SIZE nominale Höhe/Länge des Bibliothekselementes

Länge des Zubehörs wird ein benutzerdefinierte Parameter mit zzyzx in der Parameterliste geführt, dann wird dies für die nominale Höhe benutzt, andernfalls ist es 0

Öffnungsparameter - nur für Listen und Etiketten verfügbar

OPENING_HEIGHT Längenparameter, nominale Höhe der Öffnung

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	1

Kompatibilität: eingeführt in ARCHICAD 23.

OPENING_WIDTH Längenparameter, nominale Breite der Öffnung

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	1

Kompatibilität: eingeführt in ARCHICAD 23.

OPENING_HEADERHEIGHT_VALUES Dictionary, relative Höhe des Kopfes der Öffnung (oder des obersten Punktes, wenn er gedreht wird) von bestimmten Bezugsleveln

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	{}

Kompatibilität: eingeführt in ARCHICAD 23.

```
{
    "toHomeStory": 1.5,
    "toProjectZero": 1.5,
    "toWallBottom": 1.5,
    "toWallTop": 1.5
}
```

OPENING_CENTERHEIGHT_VALUES Dictionary, relative Höhe der Mitte der Öffnung von bestimmten Bezugsleveln

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	{}

Kompatibilität: eingeführt in ARCHICAD 23.

```
{
    "toHomeStory": 1.0,
    "toProjectZero": 1.0,
    "toWallBottom": 1.0,
    "toWallTop": 2.0
}
```

OPENING_SILLHEIGHT_VALUES Dictionary, relative Höhe der Schwelle der Öffnung (oder des niedrigsten Punkts bei Drehung) von bestimmten Bezugsleveln

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	{}

Kompatibilität: eingeführt in ARCHICAD 23.

```
{
    "toHomeStory": 0.5,
    "toProjectZero": 0.5,
    "toWallBottom": 0.5,
    "toWallTop": 0.5
}
```

Parameter des Öffnungssysmbols

OPENING_SYMBOL_DISPLAY Ganzzahl, Sichtbarkeit des Öffnungssymbols gemäß der Grundriss-Schnittebene (ansichtsabhängig, nicht in Parameter- / Eigenschaftsscripten verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	1

Kompatibilität: eingeführt in ARCHICAD 23.

1 - sichtbar, 2 - versteckt, 3 - über Kopf

OPENING_SYMBOL_GEOMETRY Dictionary, enthält die Geometrie des Symbols (sichtenabhängig, nicht in Parameter- / Eigenschaftsscripten verwenden)

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	{}

Kompatibilität: eingeführt in ARCHICAD 23.

- `.boundingBox2D{}`: *(Dictionary) Bounding-Box-Definition des Öffnungs-Polygons, immer ausgerichtet am Symbolkoordinatensystem der Öffnung.*
- `.boundingBox2D.xmin`
- `.boundingBox2D.xmax`
- `.boundingBox2D.ymin`
- `.boundingBox2D.ymax`
- `.polygon2D{}`: *(Dictionary) Projiziertes Polygon der Öffnung, eingeschnitten durch die Geometrie des übergeordneten Elements*
- `.polygon2D.isClosed`: *(Boolean) immer 1 bei einem Polygon (0 - reserviert für zukünftige Entwicklungen)*
- `.polygon2D.contour{}`: *(Dictionary) enthält Daten der Polygonkontur. Das Standardpolygon entspricht dem Rechteck in .boundingBox2D{}*
- `.polygon2D.contour.edges[n]`: *(Array) enthält ein eingebettetes Dictionary für jede Kante des Polygons*
- `.polygon2D.contour.edges[n].type`: *(Ganzzahl) 0 - gerade, 1 - gebogen (Kreisbogen)*
- `.polygon2D.contour.edges[n].begPoint{}`: *(Dictionary) ein eingebettetes Dictionary für die Koordinaten des Anfangspunkts der Kante*
- `.polygon2D.contour.edges[n].arcAngle`: *(Winkel) Zentralwinkel der Kantenkurve, positiv gegen den Uhrzeigersinn, negativ im Uhrzeigersinn (nicht für gerade Kanten eingestellt)*
- `.polygon2D.holes[m]`: *(Array) enthält Daten von inneren Öffnungen, ähnlich wie .contour, die nur festgelegt werden, wenn Öffnungen vorhanden sind*

```
OPENING_SYMBOL_GEOMETRY: {
    "boundingBox2D": {
      "xmin": 0,
      "xmax": 1,
      "ymin": 0,
      "ymax": 1
    },
    "polygon2D": {
      "isClosed": 1,
      "contour": {
        "edges": [
          {
            "type": 1,
            "begPoint": {
              "x": 1,
              "y": 0.5
            },
            "arcAngle": 90
          },
          {
            "type": 1,
            "begPoint": {
              "x": 0.5,
              "y": 1
            },
            "arcAngle": 90
          },
          {
            "type": 1,
            "begPoint": {
              "x": 0,
              "y": 0.5
            },
            "arcAngle": 90
          },
```

```
      {
        "type": 1,
        "begPoint": {
          "x": 0.5,
          "y": 0
        },
        "arcAngle": 90
      }
    ]
  },
 "holes": {
    "edges": [
      {
        "type": 0,
        "begPoint": {
          "x": 0.4,
          "y": 0.4
        },
      },
      {
        "type": 0,
        "begPoint": {
          "x": 0.4,
          "y": 0.6
        },
      },
      {
        "type": 0,
        "begPoint": {
          "x": 0.6,
          "y": 0.6
        },
      },
```

```
        {
          "type": 0,
          "begPoint": {
            "x": 0.6,
            "y": 0.4
          },
        }
      ]
    }
  }
```

Parameter von Fenstern, Türen und Wandenden

WIDO_REVEAL_ON	Fenster/Tür: eingebauter Anschlag ist aktiv
0-Anschlag ist nicht aktiv, 1-Anschlag ist aktiv	
WIDO_SILL	Brüstung von Fenster/Tür
für nicht gerade Wände: in radialer Richtung in der Öffnungswinkel von nominaler Größe	
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	Fenster/Tür: eingebauter Anschlag rechts
WIDO_LEFT_JAMB	Fenster/Tür: eingebauter Anschlag links
WIDO_THRES_DEPTH	Fenster/Tür: eingebaute Schwelle/Anschlag unten
WIDO_HEAD_DEPTH	Fenster/Tür: eingebauter Anschlag oben
WIDO_HEAD_HEIGHT	Nominale Sturzhöhe von 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	Anschlagsseite ist der Eröffnungsseite gegenüber
1-ja, 0-nein - wenn man ein Element platziert, ist der Grundeinstellungswert 0 für Fenster und 1 für Türen	

WIDO_FRAME_THICKNESS	Profilstärke von Fenster/Tür
Wird die Öffnungsseite von Türen/Fenstern geändert, werden sie gespiegelt und dann automatisch durch diesen Wert verschoben.	
WIDO_POSITION	Versatz von Tür/Fenster
Winkel oder Abstand zwischen der Öffnungsachse oder der Wandende und des normalen Vektors am Anfangspunktes der Wand	
WIDO_ORIENTATION	Orientierung der Öffnung von Fenster/Tür
links/rechts - es funktioniert nur, wenn der Tür/Fenster dem lokalen Standard entsprechend erstellt wurde	
WIDO_MARKER_TXT	Markertext von Fenster/Tür
WIDO_SUBFL_THICKNESS	Dicke des Fußbodenaufbaus (Brüstungshöhenkorrektur)
WIDO_PREFIX	Präfix der Brüstungshöhe von Fenster/Tür
WIDO_CUSTOM_MARKER	grundeingestellter Markerschalter für Fenster/Tür
1-Parameter können im 2D-Script benutzt werden, während die automatische Bemaßung nicht vorhanden ist	
WIDO_ORIG_DIST	Abstand des lokalen Ursprungs vom Ende der Wand
Abstand des lokalen Ursprungs vom Zentralpunkt der gekrümmten Wand, 0 für gerade Wände. Negativ für Wandende am Endpunkt der gekrümmten Wand.	
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	Fenster/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 Öffnungsbereich an der Anschlagsseite
WIDO_OPRSIDE_SURF	Fenster/Tür Öffnungsbereich 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	Nominale Höhe der Öffnung Fenster/Tür an der Seite gegenüber der Anschlagsseite
WIDO_N_RSIDE_SURF	Nominale Fläche der Öffnung Fenster/Tür an der Anschlagsseite
WIDO_N_OPRSIDE_SURF	Nominale Fenster/Tür Öffnungsfläche an der Anschlagsseite
WIDO_VOLUME	Fenster/Tür Öffnungvolumen
WIDO_GROSS_SURFACE	Fenster/Tür nominale Öffnungsbereich
WIDO_GROSS_VOLUME	Fenster/Tür nominales Öffnungsvolumen

Parameter von Lampen - nur zum Auflisten und für Etiketten verfügbar

LIGHT_ON	Licht ist aktiv
0-Licht ist aus, 1-Licht ist an:	
LIGHT_RED	roter Bestandteil der Lichtfarbe
LIGHT_GREEN	grüner Bestandteil der Lichtfarbe
LIGHT_BLUE	blauer Bestandteil der Lichtfarbe
LIGHT_INTENSITY	Lichtintensität

Marker-Parameter (Detail, Arbeitsblatt und Änderungswolken)

MARKER_HEAD_ROT_MODE — globaler Ganzzahltyp, eingestellt entsprechend der Funktion "Marker-Winkel: Fester Winkel zum Bildschirm / Fixierter Winkel zum Modell" im Marker-Paneel des Einstellungsdialogs

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	✓	Standard	1

kann verwendet werden, um auf die Einstellungen der Markerkopfdrehung von ARCHICAD auf dem GDL-Symbol zu reagieren

1 - Fixierter Winkel zum Bildschirm, 2 - Fixierter Winkel zum Modell

Kompatibilität: eingeführt in ARCHICAD 22.

MARKER_HEAD_ANGLE Winkeltyp global, eingestellt durch den Anwender auf dem Marker-Paneeleinstellungsdialog

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	✓	Standard	0

Markerkopf-Symbol-Rotationsdaten für GDL-Symbolteil des Markers

Kompatibilität: eingeführt in ARCHICAD 22.

In Markerobjekten von Details/Arbeitsblättern, bleibt der Wert SYMB_ROTANGLE kompatibel mit folgenden neuen Winkeldaten:

- *"Fixierter Winkel zum Bildschirm" Modus: 1 - SYMB_ROTANGLE ist das Gegenteil des Bildschirm-Rotationswinkels*
- *"Fixierter Winkel zum Modell" Modus: 2 - SYMB_ROTANGLE ist gleich wie MARKER_HEAD_ANGLE*

In Markern von Änderungswolken bleibt der Wert vom SYMB_ROTANGLE 0 wie früher auch in allen Fällen.

Bemaßungsparameter

LABEL_POSITION Position der Bemaßung

2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	Standard	0

Array[3][2] enthält die Koordinate der 3 Punkte, welche den Etikettzeiger definieren und die Startposition des Etikett-GDL-Symbols.

Kompatibilität: Parameter- und Eigenschaftenscript-Einschränkungen werden in eingeführt in ARCHICAD 22.

Sichten-abhängiger Wert bei "Fester Winkel" an. Projektabhängig, wenn "Etikett-Ausrichtung" auf "Parallel" oder "Rechtwinklig" gesetzt ist und das Elternelement verschoben wird. Dadurch wird der Wert der Variablen geändert, während das Parameterscript des Etiketts nicht ausgeführt wird (kann nicht als Parameter gespeichert werden).

LABEL_ASSOC_ELEM_ORIENTATION Ausrichtung des zugehörigen Elements

2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	Standard	0

- *gerade Elemente: die Richtung der Referenzlinie*
- *gebogene Elemente: die Orientierung der Sehne des Kreises*
- *punktförmige Elemente: der Rotationswinkel des Elements*

Kompatibilität: Parameter- und Eigenschaftenscript-Einschränkungen werden in eingeführt in ARCHICAD 22.

Projekt-abhängig: Das Elternelement kann verschoben werden. Dadurch wird der Wert der Variablen geändert, während das Parameterscript des Etiketts nicht ausgeführt wird (kann nicht als Parameter gespeichert werden).

LABEL_ROTANGLE	absolute Rotationswinkelwerte für GDL-Symbole vom Typ Etiketten

2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	Standard	0

Der Winkel wird gemäß den Einstellungen für die Etikettenausrichtung, festem Winkel und Lesbarkeit berechnet.

Kompatibilität: Parameter- und Eigenschaftenscript-Einschränkungen werden in eingeführt in ARCHICAD 22.

Sichten-abhängiger Wert bei "Fester Winkel" an. Projektabhängig, wenn "Etikett-Ausrichtung" auf "Parallel" oder "Rechtwinklig" gesetzt ist: das Elternelement kann verschoben werden, wodurch der Wert der Variablen geändert wird.

LABEL_ARROWHEAD_PEN	Stift der Pfeilspitze im Einstellungsdialogfeld
LABEL_HAS_POINTER	Boolesche

1 - "Zeiger hinzufügen/entfernen" ist gedrückt auf dem Paneel "Zeiger" der Etiketteinstellungen, andernfalls 0.

Kompatibilität: eingeführt in ARCHICAD 22. Die ähnliche umgekehrt arbeitende Variable "LABEL_CUSTOM_ARROW" ist seit ARCHICAD 22 veraltet.

Wandparameter, verfügbar für Fenster/Türen, Listen und Etiketten

WALL_ID	ID der Wand
WALL_INTGUID	interne GUID der Wand

die interne GUIID-Nummer, die durch das Programm generiert wird (kann von dem Benutzer nicht kontrolliert werden)

WALL_RESOL	3D-Auflösung einer gekrümmten Wand

nur im 3D effektiv

WALL_THICKNESS	Wandstärke

im Falle von geneigten Wänden: Wandstärke an der Öffnungsachse (lokale z-Achse)

WALL_START_THICKNESS	Wandstärke am Anfang
WALL_END_THICKNESS	Wandstärke am Ende
WALL_INCL	Neigung der Wandoberflächen

der Winkel zwischen den zwei geneigten Wandoberflächen - 0 für allgemeine gerade Wände

WALL_FLIPPED Boolescher Wert gemäß "Referenzlinienposition": Wand umklappen auf Referenzlinie "Option der Wand

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	✓	Standard	0

Information über den Umklapp-Status der aktuellen Wand

0 - Standard-Wandposition, 1 - Wand ist in Bezug zur Referenzlinie umgeklappt

Kompatibilität: eingeführt in ARCHICAD 22.

WALL_HEIGHT Wandhöhe

WALL_MAT_A Oberflächen-Attribut-Index der Wand auf der gegenüberliegenden Seite von der Öffnungsseite

WALL_MAT_B Oberflächen-Attribut-Index der Wand auf der Öffnungsseite

im Falle von Öffnungen kann es von Öffnung zu Öffnung, die in derselben Wand platziert sind, verschieden sein

WALL_MAT_EDGE Oberflächen-Attribut-Index der Wandenden

WALL_LINETYPE Linientyp der Wand

wird nur für die Konturen im Grundrissfenster verwendet

WALL_FILL Schraffurtyp der Wand

Schraffurindex, erste Oberfläche mehrschichtiger Bauteile

WALL_FILL_PEN Stift der Wandschraffur

WALL_COMPS_NAME Name der mehrschichtigen oder komplexen Struktur der Wand,

der Name des Profilattributes für komplexe Wände, der Name des mehrschichtigen Attributs für mehrschichtige Wände, andernfalls ein Leerstring.

WALL_BMAT_NAME Name des Baustoffes der Wand

Name des Baustoffes der Wand, Leerstring bei Mehrschichtwänden oder Profilwänden.

WALL_BMAT Index des Baustoffs der Wand

Kompatibilität: eingeführt in ARCHICAD 21.

Baustoff-Index der Wand, 0 bei Mehrschichtwänden oder Profilwänden.

WALL_SKINS_NUMBER Anzahl der Schichten der zusammengesetzten oder komplexen Wand

liegt zwischen 1 und 127; ist 0, falls eine einzige Schraffur verwendet wird

WALL_SKINS_PARAMS Parameter der Schichten der zusammengesetzten oder komplexen Wand

Array mit 19 Spalten und einer beliebigen Anzahl von Zeilen:

- *[1] Schraffur*
- *[2] Stärke*
- *[3] (alter Konturstift)*
- *[4] Schraffurstift*
- *[5] Stift der Hintergrundschraffur*
- *[6] Kern/Bekleidungsstatus*
- *[7] oberer Linienstift*
- *[8] oberer Linientyp*
- *[9] unterer Linienstift*
- *[10] unterer Linientyp*
- *[11] Stirnflächenstift*
- *[12] Schraffurausrichtung*
- *[13] Oberflächentyp*
- *[14] Linientyp Stirnfläche*
- *[15] fertiger Oberflächenstatus*
- *[16] Status Schraffurausrichtung*
- *[17] Status Trapezoid/doppelt geneigt*
- *[18] Baustoff-Index*
- *[19] Schichten-Oberflächen-Index (Berücksichtigung der Wandkantenoberflächen - Überlagerung). Kompatibilität: eingeführt in ARCHICAD 22.*

Kern: 0 - nicht Teil, 1 - Teil, 3 - letzte Kernoberfläche.

Schraffurausrichtung: 0 - global, 1 - lokal.

Schichtentyp: 0 - geschnitten, 1 - unter Schnittebene, 2 - über Schnittebene (einfache Wände=0).

trapezförmig / beidseitig geneigt: 0 - diese Schicht besitzt in allen Fällen parallele Sichtflächen, 1 - diese Schicht kann nicht-parallele Sichtflächen besitzen, um den Unterschied der Wandstärke trapezförmiger Wände oder beidseitig geneigter Wänden anzupassen. Auch wenn die Wandoberflächen parallel sind, kann dieses Flag eingeschaltet sein.

Fertiger Schichtstatus: 0 - nicht fertige Schicht, 1 - fertige Schicht.

Status Schraffurausrichtung: 0 - globale oder lokale Schraffurausrichtung, wie sie in der Spalte "Schraffurausrichtung" eingestellt wurde, 1 - Schraffurausrichtung und Größe stimmen mit der Wandschichtrichtung und deren Dicke überein.

Bei komplexen Wänden enthält diese Variable nur die Daten der Schichten, welche im Grundriss geschnitten sind (2D - entsprechend der Grundriss-Schnittebene), oder in der Brüstungshöhe von Fenstern/Türen/Wandenden geschnitten sind (3D).

WALL_SKINS_BMAT_NAMES	Name des Baustoffs von Mehrschichtbauteilen oder Profil-Bauteilen

Array mit 1 Spalten: Name des Baustoffs der Schicht mit beliebiger Zeilenanzahl

Für Fenster, Türen und Wandenden im 3D-Fenster enthält sie die Informationen der Oberflächen, die gerade von dem Fenster, Tür oder Wandende geschnitten werden.

WALL_SECT_PEN	Stift der Wandkontur-Schnittoberflächen

verwendet für die Konturen der geschnittenen Flächen im Grundriss-Fenster und im Schnitte/ Ansichten-Fenster

WALL_VIEW_PEN	Stift der Wandkonturen in der Ansicht

verwendet für alle Kanten im 3D-Fenster und für Umfassungskanten (Kanten von unterhalb der Schnittebene betrachtet) im Grundriss, sowie für alle Kanten im Schnitte-/Ansichten-Fenster

WALL_FBGD_PEN	Stift des Schraffurhintergrundes der Wand
WALL_DIRECTION	Ausrichtung der Wand

gerade Wände: die Orientierung der Konstruktionslinie, gekrümmte Wände: die Orientierung der Sehne des Kreises

WALL_POSITION	absolute Koordinaten der Wand

Array mit 3 Spalten: die Position des Anfangspunktes der Wand im Verhältnis zum Projektursprung

WALL_TEXTURE_WRAP	Texturmapping-Daten der Wand, zur Verwendung in den Befehlen VERT und COOR{2}, oder COOR{3}. Die Wandtextur-Koordinaten werden umgewandelt um zum lokalen Koordinatensystem des Wand-verbundenen Objektes zu passen (keine zusätzlichen Transformationen notwendig).

Array mit 14 Zeilen:

- *[1]: wrapping_method*
- *[2]: wrap_flags*
- *[3]-[4]-[5]: origin_X, origin_Y, origin_Z (nodes of vert 1)*
- *[6]-[7]-[8]: endOfX_X, endOfX_Y, endOfX_Z (nodes of vert 2)*
- *[9]-[10]-[11]: endOfY_X, endOfY_Y, endOfY_Z (nodes of vert 3)*
- *[12]-[13]-[14]: endOfZ_X, endOfZ_Y, endOfZ_Z (nodes of vert 4)*

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_LENGTH_A_CON	konditionale Wandlänge an der Seite der Konstruktionslinie

WALL_LENGTH_B_CON	konditionale Wandlänge 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	Oberflächenbereich der Wand an der Seite der Konstruktionslinie
WALL_SURFACE_B	Oberflächenbereich der Wand an der Seite gegenüber der Konstruktionslinie
WALL_SURFACE_A_CON	konditionaler Wandoberflächenbereich an der Seite der Konstruktionslinie
WALL_SURFACE_B_CON	konditionaler Wandoberflächenbereich an der Seite gegenüber der Konstruktionslinie
WALL_GROSS_SURFACE_A	Brutto-Oberflächenbereich der Wand an der Konstruktionsseite
WALL_GROSS_SURFACE_B	Brutto-Oberflächenbereich der Wand an der Seite gegenüber der Konstruktionslinie
WALL_EDGE_SURF	Oberflächenbereich der Wandkante
WALL_VOLUME	Volumen der Wand
WALL_VOLUME_CON	konditionales Volumen der Wand
WALL_GROSS_VOLUME	Brutto-Volumen der Wand
WALL_VOLUME_A	Wandschicht-Volumen an der Seite der Konstruktionslinie
WALL_VOLUME_A_CON	konditionales Wandschicht-Volumen an der Seite der Konstruktionslinie
WALL_VOLUME_B	Wandschicht-Volumen an der Seite gegenüber der Konstruktionslinie
WALL_VOLUME_B_CON	konditionales Wandschicht-Volumen an der Seite gegenüber der Konstruktionslinie
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	Oberflächenbereich der Türen in der Wand
WALL_WINDS_SURF	Oberflächenbereich der Fenster in der Wand
WALL_HOLES_SURF	Oberflächenbereich der leeren Öffnungen in der Wand
WALL_HOLES_SURF_A	Analytischer Oberflächenbereich aller Öffnungen an der Konstruktionsseite

WALL_HOLES_SURF_B	Analytischer Oberflächenbereich aller Öffnungen auf der Gegenseite
WALL_HOLES_VOLUME	Analytisches Volumen 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
WALL_CROSSSECTION_TYPE	Querschnittstyp der Wand
0 - komplex profiliert, 1 - rechteckig, 2 - geneigt, 3 - 2seitig geneigt	
WALL_MIN_HEIGHT	Minimale Höhe der Wand
WALL_MAX_HEIGHT	Maximale Höhe der Wand
WALL_SKIN_MIN_HEIGHT_A	minimale Höhe der Wandschicht an der Seite der Konstruktionslinie
WALL_SKIN_MAX_HEIGHT_A	maximale Höhe der Wandschicht an der Seite der Konstruktionslinie
WALL_SKIN_MIN_HEIGHT_B	minimale Höhe der Wandschicht an der Seite der Konstruktionslinie
WALL_SKIN_MAX_HEIGHT_B	Maximale Höhe der Wandschicht an der Seite gegenüber der Konstruktionslinie
WALL_SKIN_THICKNESS_A	Wandschicht-Dicke an der Seite der Konstruktionslinie
WALL_SKIN_THICKNESS_B	Wandschicht-Dicke an der Seite gegenüber der Konstruktionslinie
WALL_INSU_THICKNESS	Wand-Schichtdicke der Dämmung
WALL_AIR_THICKNESS	Luftschichtdicke der Wand

Parameter von Stützen - nur zum Auflisten verfügbar

Kompatibilität: Ab ARCHICAD 23 ist das Stützen-Element eine Sammlung von Stützen-Segmenten. GLOB_ELEM_TYPE besitzt einen neuen Wert für Stützen-Segmente: 27, der Wert des Stützenelements bleibt bei 6.

Die Verfügbarkeit jeder globalen Variablen (unabhängig davon, ob sie aussagekräftige Daten enthält) wird in einer Tabelle mit Symbolen angezeigt. Der Wert der globalen Variablen GLOB_ELEM_TYPE steht in Klammern.

COLU_SEGMENT_INDEX Index des Stützensegments im Array COLU_SEGMENT_INFO.segments []

✅ *Stützen-Segment (27)*	⛔ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

Kompatibilität: eingeführt in ARCHICAD 23.

COLU_SEGMENT_INFO Dictionary, enthält alle geometrischen Einstellungen der Stützensegmente, die die Verfügbarkeit anderer globaler Variablen bestimmen können. In Parameter- und Eigenschaftsscripten nicht verfügbar.

Wenn COLU_SEGMENT_INFO.segments [COLU_SEGMENT_INDEX] auf einem Stützensegment platziert wird, enthält es die Informationen zum etikettierten Segment.

- `.segments[n].tapered`: *(ganzzahlig): 0 - gleichmäßiger Querschnitt, 1 - sich verjüngender Querschnitt*
- `.segments[n].crossSection{}`: *(Dictionary) Querschnittsdaten des Segments*
- `.segments[n].crossSection.type`: *(Ganzzahl) Querschnittstyp: 1 - rechteckig, 2 - kreisförmig, 3 - komplexes Profil*
- `.segments[n].crossSection.startWidth`: *(Länge) begrenzende Querschnittsbreite des Segmentanfangs. Die Richtung des Segments wird durch seine Referenzlinie definiert.*
- `.segments[n].crossSection.startHeight`: *(Länge) begrenzende Querschnittshöhe des Segmentanfangs. Die Richtung des Segments wird durch seine Referenzlinie definiert.*
- `.segments[n].crossSection.endWidth`: *(Länge) begrenzende Querschnittsbreite des Segmentendes. Die Richtung des Segments wird durch seine Referenzlinie definiert.*
- `.segments[n].crossSection.endHeight`: *(Länge) begrenzende Querschnittshöhe des Segmentendes. Die Richtung des Segments wird durch seine Referenzlinie definiert.*

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*

Kompatibilität: eingeführt in ARCHICAD 23.

COLU_CORE Kern- / Bekleidungseigenschaften (0 bei verjüngtem Stützensegment))

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

dient der Kompatibilität: es hat nur im Eigenschaften-Script von .CPS-Dateien (Stützeneigenschaften) eine Auswirkung

COLU_HEIGHT Höhe der Stütze

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*

COLU_MIN_HEIGHT Minimale Höhe der Stütze

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*

COLU_MAX_HEIGHT Maximale Höhe der Stütze

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*

COLU_VENEER_WIDTH Dicke der Stützenummantelung

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

COLU_CORE_X Kernstärke (0 bei verjüngtem Stützensegment)

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

COLU_CORE_Y Kernstärke (0 bei verjüngtem Stützensegment)

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

COLU_DIM1 1. Abmessung der Stütze (0 bei verjüngtem Stützensegment) - nur für Etiketten

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

COLU_DIM2 2. Abmessung der Stütze (0 bei verjüngtem Stützensegment) - nur für Etiketten

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

COLU_MAT Oberflächen-Attribut-Index der Stütze

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

Die Ummantelung durch eine Wand ersetzt die Stützenoberfläche durch das Oberflächenmaterial der anschließenden Wände

COLU_LINETYPE Linientyp der Stütze

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*

wird nur für die Konturen im Grundrissfenster verwendet

COLU_CORE_FILL Schraffur des Stützenkernes

✔ *Stützen-Segment (27)*	✔ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

COLU_CORE_BMAT_NAME Name des Baustoffs des Stützenkerns

✔ *Stützen-Segment (27)*	✔ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

COLU_CORE_BMAT Baustoff-Index des Stützen-Kerns

✔ *Stützen-Segment (27)*	✔ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

Kompatibilität: eingeführt in ARCHICAD 21.

COLU_VENEER_FILL Schraffur der Stützenummantelung

✔ *Stützen-Segment (27)*	✔ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

COLU_VENEER_BMAT_NAME Name des Baustoff des Stützenmantels

✔ *Stützen-Segment (27)*	✔ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

COLU_VENEER_BMAT Baustoffindex des Stützenmantels

✔ *Stützen-Segment (27)*	✔ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

Kompatibilität: eingeführt in ARCHICAD 21.

COLU_SECT_PEN Stift der Stiftkonturen von Schnittoberflächen

✔ *Stützen-Segment (27)*	✔ *Einzel-Segment-Stütze(6)*	✔ *Multi-Segment-Stütze(6)*

verwendet auf den Konturen der geschnittenen Stützenoberflächen im Grundriss-Fenster und im Schnitte/Ansichten-Fenster

COLU_VIEW_PEN Stift der sichtbaren Stützenkanten

✔ *Stützen-Segment (27)*	✔ *Einzel-Segment-Stütze(6)*	✔ *Multi-Segment-Stütze(6)*

verwendet für alle Kanten im 3D-Fenster und für Umfassungskanten (Kanten von unterhalb der Schnittebene betrachtet) im Grundriss, sowie für alle Kanten im Schnitte-/Ansichten-Fenster

COLU_CORE_FILL_PEN Stift der Schraffur des Stützenkernes

✓ *Stützen-Segment (27)*	✓ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

COLU_CORE_FBGD_PEN Stift des Schraffurhintergrundes des Stützenkernes

✓ *Stützen-Segment (27)*	✓ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

COLU_VENEER_FILL_PEN Stift der Schraffur der Ummantelung

✓ *Stützen-Segment (27)*	✓ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

COLU_VENEER_FBGD_PEN Stift des Schraffurhintergrundes der Stützenummantelung

✓ *Stützen-Segment (27)*	✓ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

COLU_PERIMETER Umfang der Stütze

✓ *Stützen-Segment (27)*	✓ *Einzel-Segment-Stütze(6)*	✓ *Multi-Segment-Stütze(6)*

COLU_AREA Fläche der Stütze

✓ *Stützen-Segment (27)*	✓ *Einzel-Segment-Stütze(6)*	✓ *Multi-Segment-Stütze(6)*

COLU_VOLUME Volumen der Stütze

✓ *Stützen-Segment (27)*	✓ *Einzel-Segment-Stütze(6)*	✓ *Multi-Segment-Stütze(6)*

COLU_GROSS_VOLUME Brutto-Volumen der Stütze

✓ *Stützen-Segment (27)*	✓ *Einzel-Segment-Stütze(6)*	✓ *Multi-Segment-Stütze(6)*

COLU_CORE_SURF Oberflächenbereich des Stützenkerns

✓ *Stützen-Segment (27)*	✓ *Einzel-Segment-Stütze(6)*	✓ *Multi-Segment-Stütze(6)*

COLU_CORE_GROSS_SURF Brutto-Oberflächenbereich des Stützenkerns

✓ *Stützen-Segment (27)*	✓ *Einzel-Segment-Stütze(6)*	✓ *Multi-Segment-Stütze(6)*

COLU_CORE_VOL	Volumen des Stützenkerns	
✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*
COLU_CORE_GROSS_VOL	Brutto-Volumen der Stütze	
✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*
COLU_VENEER_SURF	Oberflächenbereich der Stützenummantelung	
✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*
COLU_VENEER_GROSS_SURF	Brutto-Oberflächenbereich der Stützenummantelung	
✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*
COLU_VENEER_VOL	Volumen der Stützenummantelung	
✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*
COLU_VENEER_GROSS_VOL	Brutto-Volumen der Stützenummantelung	
✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*
COLU_CORE_TOP_SURF	Oberflächenbereich der Oberseite des Stützenkernes	
✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*
COLU_CORE_BOT_SURF	Oberflächenbereich der Unterseite des Stützenkernes	
✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*
COLU_VENEER_TOP_SURF	Oberflächenbereich der Oberseite des Stützenmantels	
✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*
COLU_VENEER_BOT_SURF	Oberflächenbereich der Unterseite des Stützenmantels	
✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*

COLU_CORE_GROSS_TOPBOT_SURF Brutto-Oberflächenbereich der Ober- und Unterseite des Stützenkerns

✔ *Stützen-Segment (27)*	✔ *Einzel-Segment-Stütze(6)*	✔ *Multi-Segment-Stütze(6)*

COLU_VENEER_GROSS_TOPBOT_SURF Brutto-Oberflächenbereich der Ober- und Unterseite des Stützenmantels

✔ *Stützen-Segment (27)*	✔ *Einzel-Segment-Stütze(6)*	✔ *Multi-Segment-Stütze(6)*

COLU_PROFILE_NAME Name des Profils der Stütze, falls komplex

✔ *Stützen-Segment (27)*	✔ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

Trägerparameter - nur zur Auflistung verfügbar

Kompatibilität: Ab ARCHICAD 23 ist das Träger-Element eine Sammlung von Träger-Segmenten. GLOB_ELEM_TYPE hat einen neuen Wert für Trägersegmente: 28, der Wert des Trägerelements bleibt bei 12.

Die Verfügbarkeit jeder globalen Variablen (unabhängig davon, ob sie aussagekräftige Daten enthält) wird in einer Tabelle mit Symbolen angezeigt. Der Wert der globalen Variablen GLOB_ELEM_TYPE steht in Klammern.

BEAM_SEGMENT_INDEX Index des Träger-Segments im Array BEAM_SEGMENT_INFO.segments []

✔ *Träger-Segment (28)*	⛔ *Einzel-Segment-Träger (12)*	⛔ *Multi-Segment-Träger (12)*

Kompatibilität: eingeführt in ARCHICAD 23.

BEAM_SEGMENT_INFO Dictionary, enthält alle geometrischen Einstellungen der Trägersegmente, die die Verfügbarkeit anderer globaler Variablen bestimmen, sowie weitere geometrische Trägernsegmentdaten. In Parameter- und Eigenschaftsscripten nicht verfügbar.

BEAM_SEGMENT_INFO.segments [BEAM_SEGMENT_INDEX] enthält bei Platzierung auf einem Trägersegment die Informationen zum etikettierten Segment.

- `.segments[n].curvature`: *Achsenkrümmung (Ganzzahl): 0 - gerade, 1 - horizontal gebogen, 2 - vertikal gebogen*
- `.segments[n].tapered`: *(ganzzahlig): 0 - gleichmäßiger Querschnitt, 1 - sich verjüngender Querschnitt*
- `.segments[n].refLineLength`: *3D - Länge der Segmentreferenzlinie des Trägers (Länge)*
- `.segments[n].crossSection{}`: *(Dictionary) Querschnittsdaten des Segments*
- `.segments[n].crossSection.type`: *(integer) cross-section type: 1 - rechteckig, 2 - kreisförmig, 3 - komplexes Profil*
- `.segments[n].crossSection.startWidth`: *(Länge) Begrenzung der Querschnittsbreite des Segmentanfangs.Die Richtung des Segments wird durch seine Referenzlinie definiert.*
- `.segments[n].crossSection.startHeight`: *(Länge) Begrenzung der Querschnittshöhe des Segmentanfangs. Die Richtung des Segments wird durch seine Referenzlinie definiert.*
- `.segments[n].crossSection.endWidth`: *(Länge) Begrenzung der Querschnittsbreite des Segmentendes. Die Richtung des Segments wird durch seine Referenzlinie definiert.*
- `.segments[n].crossSection.endHeight`: *(Länge) Begrenzung der Querschnittshöhe des Segmentendes Die Richtung des Segments wird durch seine Referenzlinie definiert.*

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger (12)*	✔ *Multi-Segment-Träger (12)*

Kompatibilität: eingeführt in ARCHICAD 23.

BEAM_THICKNESS Breite des Trägers (0 bei abgeschrägtem Trägersegment)

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger (12)*	⛔ *Multi-Segment-Träger (12)*

BEAM_HEIGHT Höhe des Trägers (0 bei abgeschrägtem Trägersegment)

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger (12)*	⛔ *Multi-Segment-Träger (12)*

BEAM_REFLINE_OFFSET Abstand der Referenzachse zu den Achsen des Trägers

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger (12)*	✔ *Multi-Segment-Träger (12)*

BEAM_ELEVATION_TOP berechnete Höhe des höchstgelegenen Punktes der Träger-Geometrie

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger (12)*	✔ *Multi-Segment-Träger (12)*

Kompatibilität: eingeführt in ARCHICAD 22.

BEAM_ELEVATION_BOTTOM berechnete Höhe des tiefstgelegenen Punktes der Träger-Geometrie

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger (12)*	✔ *Multi-Segment-Träger (12)*

Kompatibilität: eingeführt in ARCHICAD 22.

BEAM_PRIORITY Indexnummer der 3D-Verschneidungspriorität

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger (12)*	⛔ *Multi-Segment-Träger (12)*

BEAM_MAT_RIGHT Oberflächen-Attribut-Index des Trägers auf der rechten Seite der Referenzlinie

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger (12)*	⛔ *Multi-Segment-Träger (12)*

BEAM_MAT_LEFT Oberflächen-Attribut-Index des Trägers auf der linken Seite der Referenzlinie

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger (12)*	⛔ *Multi-Segment-Träger (12)*

BEAM_MAT_TOP Oberflächen-Attribut-Index des Trägers auf der Oberseite

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger (12)*	⛔ *Multi-Segment-Träger (12)*

BEAM_MAT_BOTTOM Oberflächen-Attribut-Index des Trägers auf der Unterseite

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger (12)*	⛔ *Multi-Segment-Träger (12)*

BEAM_MAT_END Oberflächen-Attribut-Index des Trägers an beiden Enden

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger(12)*	⛔ *Multi-Segment-Träger(12)*

BEAM_OUTLINE_LINETYPE Linientyp der Trägerkontur

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger(12)*	✔ *Multi-Segment-Träger(12)*

BEAM_AXES_LINETYPE Linientyp der Trägerachse

Träger-Segment (28)	Einzel-Segment-Träger(12)	Multi-Segment-Träger(12)

BEAM_FILL Schraffurtyp des Trägers

Träger-Segment (28)	Einzel-Segment-Träger(12)	Multi-Segment-Träger(12)

BEAM_BMAT_NAME Name des Baustoffs des Trägers

Träger-Segment (28)	Einzel-Segment-Träger(12)	Multi-Segment-Träger(12)

BEAM_BMAT Baustoff-Index des Trägers

Träger-Segment (28)	Einzel-Segment-Träger(12)	Multi-Segment-Träger(12)

Kompatibilität: eingeführt in ARCHICAD 21.

BEAM_FILL_PEN Stift der Trägerschraffur

Träger-Segment (28)	Einzel-Segment-Träger(12)	Multi-Segment-Träger(12)

BEAM_SECT_PEN Stift der Trägerkontur-Schnittoberflächen

Träger-Segment (28)	Einzel-Segment-Träger(12)	Multi-Segment-Träger(12)

BEAM_FBGD_PEN Stift des Schraffurhintergrundes des Trägers

Träger-Segment (28)	Einzel-Segment-Träger(12)	Multi-Segment-Träger(12)

BEAM_DIRECTION Richtung der Referenzachse des Trägers

Träger-Segment (28)	Einzel-Segment-Träger(12)	Multi-Segment-Träger(12)

BEAM_POSITION absolute Koordinaten des Anfangspunkts der Trägerachsen (Array mit 3 Spalten)

Träger-Segment (28)	Einzel-Segment-Träger(12)	Multi-Segment-Träger(12)

BEAM_LENGTH_RIGHT Länge des Trägers auf der rechten Seite der Bezugslinie (0 bei verjüngten oder vertikal gekrümmten Trägern))

✓ *Träger-Segment (28)*	✓ *Einzel-Segment-Träger(12)*	⛔ *Multi-Segment-Träger(12)*

Kompatibilität: veraltet für Listen in ARCHICAD 23.

BEAM_LENGTH_LEFT Länge des Trägers auf der linken Seite der Bezugslinie (0 bei verjüngten oder vertikal gekrümmten Trägern)

✓ *Träger-Segment (28)*	✓ *Einzel-Segment-Träger(12)*	⛔ *Multi-Segment-Träger(12)*

Kompatibilität: veraltet für Listen in ARCHICAD 23.

BEAM_RIGHT_SURF Oberflächenbereich des Trägers auf der rechten Seite der Referenzachse

✓ *Träger-Segment (28)*	✓ *Einzel-Segment-Träger(12)*	✓ *Multi-Segment-Träger(12)*

BEAM_LEFT_SURF Oberflächenbereich des Trägers auf der linken Seite der Referenzachse

✓ *Träger-Segment (28)*	✓ *Einzel-Segment-Träger(12)*	✓ *Multi-Segment-Träger(12)*

BEAM_TOP_SURF Oberflächenbereich der Oberseite des Trägers

✓ *Träger-Segment (28)*	✓ *Einzel-Segment-Träger(12)*	✓ *Multi-Segment-Träger(12)*

BEAM_BOTTOM_SURF Oberflächenbereich der Unterseite des Trägers

✓ *Träger-Segment (28)*	✓ *Einzel-Segment-Träger(12)*	✓ *Multi-Segment-Träger(12)*

BEAM_END_SURF Oberflächenbereich an beiden Enden des Trägers

✓ *Träger-Segment (28)*	✓ *Einzel-Segment-Träger(12)*	✓ *Multi-Segment-Träger(12)*

BEAM_VOLUME Volumen des Träger

✓ *Träger-Segment (28)*	✓ *Einzel-Segment-Träger(12)*	✓ *Multi-Segment-Träger(12)*

BEAM_VOLUME_CON	konditionales Volumen des Trägers		
✅ *Träger-Segment (28)*	✅ *Einzel-Segment-Träger(12)*	✅ *Multi-Segment-Träger(12)*	
BEAM_HOLES_NR	Anzahl der Durchbrüche im Träger		
✅ *Träger-Segment (28)*	✅ *Einzel-Segment-Träger(12)*	✅ *Multi-Segment-Träger(12)*	
BEAM_HOLES_SURF	gesamter Oberflächenbereich der Durchbrüche im Träger		
✅ *Träger-Segment (28)*	✅ *Einzel-Segment-Träger(12)*	✅ *Multi-Segment-Träger(12)*	
BEAM_HOLE_EDGE_SURF	gesamter Oberflächenbereich der Durchbruchslaibungsflächen im Träger		
✅ *Träger-Segment (28)*	✅ *Einzel-Segment-Träger(12)*	✅ *Multi-Segment-Träger(12)*	
BEAM_HOLES_VOLUME	gesamtes Volumen der Durchbrüche im Träger		
✅ *Träger-Segment (28)*	✅ *Einzel-Segment-Träger(12)*	✅ *Multi-Segment-Träger(12)*	
BEAM_PROFILE_NAME	Name des Profils des Trägers, falls komplex		
✅ *Träger-Segment (28)*	✅ *Einzel-Segment-Träger(12)*	⛔ *Multi-Segment-Träger(12)*	

Parameter von Decken - nur zum Auflisten verfügbar

SLAB_THICKNESS	Dicke der Decke
SLAB_ELEVATION_TOP	Höhenwert Oberkante Decke
SLAB_ELEVATION_BOTTOM	Höhenwert Unterkante Decke
SLAB_MAT_TOP	Oberflächen-Attribut-Index der Decke auf der Oberseite
SLAB_MAT_EDGE	Oberflächen-Attribut-Index der Deckenkante
SLAB_MAT_BOTT	Oberflächen-Attribut-Index der Decke auf der Unterseite
SLAB_LINETYPE	Linientyp der Decke
SLAB_FILL	Schraffur der Decke

Schraffurindex - sein Wert ist negativ im Falle eines mehrschichtigen Aufbaus

SLAB_FILL_PEN	Stift der Schraffur der Decke
SLAB_FBGD_PEN	Stift des Schraffurhintergrundes der Decke
SLAB_COMPS_NAME	Name des Mehrschichtbauteils der Decke
SLAB_BMAT_NAME	Name des Baustoffs der Decke, Leerstring bei Mehrschichtdecken
SLAB_BMAT	Baustoff-Index der Decke, 0 bei Mehrschichtdecken
Kompatibilität: eingeführt in ARCHICAD 21.	
SLAB_SKINS_NUMBER	Anzahl der Schichten der Mehrschichtstruktur Decke
liegt zwischen 1 und 8; ist 0, falls eine einzige Schraffur verwendet wird	
SLAB_SKINS_PARAMS	Parameter der Schichten einer zusammengesetzten Decke

Array mit 18 Spalten und einer beliebigen Anzahl von Zeilen:

- *[1] Schraffur*
- *[2] Stärke*
- *[3] (alter Konturstift)*
- *[4] Schraffurstift*
- *[5] Stift der Hintergrundschraffur*
- *[6] Kern/Bekleidungsstatus*
- *[7] oberer Linienstift*
- *[8] oberer Linientyp*
- *[9] unterer Linienstift*
- *[10] unterer Linientyp*
- *[11] Stirnflächenstift*
- *[12] Schraffurausrichtung*
- *[13] Oberflächentyp*
- *[14] Linientyp Stirnfläche*
- *[15] fertiger Oberflächenstatus*
- *[16] Status Schraffurausrichtung*
- *[17] Status einer Kernschicht (falls keine Kernschicht existiert: die stärkste Schicht)*
- *[18] Baustoff-Index.*

Kern: 0 - nein, 1 - ja, 3 - letzte Schicht, Richtung der Schraffur: 0 - global, 1 - lokal; Schichtentyp: in aktuellem ARCHICAD immer 0 - Schnitt, kann verwendet werden wie in Wänden später; Fertiger Schichtstatus: 0: Schicht nicht beenden, 1: Beenden Schicht

SLAB_SKINS_BMAT_NAMES	Name des Baustoffs von Schichten bei Mehrschichtdecken
Array mit 1 Spalten: Name des Baustoffs der Schicht mit beliebiger Zeilenanzahl	
SLAB_SECT_PEN	Stift der Deckenkonturen im Schnitt
verwendet auf den Konturen der geschnittenen Stützenoberflächen im Grundriss-Fenster und im Schnitte/Ansichten-Fenster	
SLAB_VIEW_PEN	Stift der Decke
verwendet für alle Kanten im 3D-Fenster und für alle sichtbaren Kanten in den Schnitte/Ansichten-Fenstern	
SLAB_TOP_SURF	Oberflächenbereich der Oberseite der Decke
nicht um den Oberflächenbereich von Öffnungen reduziert	
SLAB_GROSS_TOP_SURF	Bruttooberflächenbereich der Deckenoberseite ohne Öffnungen
um den Oberflächenbereich von Öffnungen reduziert	
SLAB_TOP_SURF_CON	Konditionaler Oberflächenbereich der Deckenoberseite
um den Oberflächenbereich von Öffnungen reduziert, welche größer als der vorgegebene Wert sind	
SLAB_BOT_SURF	Oberflächenbereich einer Deckenunterseite ohne Öffnung
nicht um den Oberflächenbereich von Öffnungen reduziert	
SLAB_GROSS_BOT_SURF	Brutto-Oberflächenbereich der Unterseite der Decke
um den Oberflächenbereich von Öffnungen reduziert	
SLAB_BOT_SURF_CON	Konditionaler Oberflächenbereich der Deckenunterseite
um den Oberflächenbereich von Öffnungen reduziert, welche größer als der vorgegebene Wert sind	
SLAB_EDGE_SURF	Oberflächenbereich der Deckenkanten
nicht um den Oberflächenbereich von Öffnungen reduziert	
SLAB_GROSS_EDGE_SURF	Brutto-Oberflächenbereich der Deckenkanten ohne Öffnung
um den Oberflächenbereich von Öffnungen reduziert	
SLAB_PERIMETER	Umfang der Decke
SLAB_VOLUME	Volumen der Decke
nicht um das Volumen von Öffnungen reduziert	

SLAB_GROSS_VOLUME *um das Volumen von Öffnungen reduziert*	Bruttovolumen der Decke ohne Öffnungen
SLAB_VOLUME_CON *um die Oberfläche von Öffnungen reduziert, welche größer als der vorgegebene Wert sind*	konditionales Volumen der Decke
SLAB_SEGMENTS_NR	Anzahl der Segmenten der Deckenkante
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
SLAB_GROSS_TOP_SURF_WITH_HOLES	Brutto-Oberflächenbereich der Deckenoberseite
SLAB_GROSS_BOT_SURF_WITH_HOLES	Brutto-Oberflächenbereich der Deckenunterseite
SLAB_GROSS_EDGE_SURF_WITH_HOLES	Brutto-Oberflächenbereich der Deckenkanten
SLAB_GROSS_VOLUME_WITH_HOLES	Brutto-Volumen der Decke

Treppenkomponentenparameter

Allgemeine Treppenvariablen - verfügbar für Auswertungen und Etiketten

Kompatibilität: eingeführt in ARCHICAD 21.

STAIR_AREA — projizierte 2D-Fläche der Treppe

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

STAIR_VOLUME — Fläche der Treppe einschließlich ihrer 3D-Elemente

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

STAIR_HEIGHT — Differenz zwischen Maximum und Minimum von Z-Koordinaten

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

STAIR_WALKLINE_LENGTH — Projizierte 2D-Länge der Lauflinie der Treppe

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

STAIR_DEFAULT_WIDTH

Standardbreite der Treppe (wie in den Treppen-Standard-Einstellungen / Geometrie- und Positionierung eingestellt)

2D	✓	3D	⊖	UI	✓	Parameter	✓	Property	⊖	Standard	0

STAIR_DEFAULT_GOING_DEPTH

Standard-Auftrittsbreite (wie in den Treppen-Standardeinstellungen / Geometrie und Positionierung eingestellt)

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⊖	Standard	0

STAIR_DEFAULT_RISER_HEIGHT

Standard-Höhe der Steigung (wie in den Treppen-Standard-Einstellungen / Geometrie und Positionierung eingestellt)

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⊖	Standard	0

STAIR_DEFAULT_TREAD_THICKNESS

Standard-Trittstufen-Stärke der Treppe (wie in den Treppen-Standard-Einstellungen / Geometrie und Positionierung eingestellt)

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⊖	Standard	0

STAIR_NR_OF_TREADS_IN_FLIGHTS

Ganzzahl-Array mit einer Dimension ([n]) Anzahl der Auftritte in jedem Teil-Treppenlauf der Treppe (n = Anzahl der Teil-Treppenläufe)

2D	✓	3D	⊖	UI	✓	Parameter	✓	Property	⊖	Standard	0

STAIR_NR_OF_RISERS_IN_FLIGHTS

Ganzzahl-Array mit einer Dimension ([n]) Anzahl der Steigungen in jedem Teil-Treppenlauf der Treppe (n = Anzahl der Teil-Treppenläufe)

2D	✓	3D	⊖	UI	✓	Parameter	✓	Property	⊖	Standard	0

STAIR_NR_OF_RISERS

Anzahl der Steigungen in Bezug auf die gesamte Treppe

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⊖	Standard	0

STAIR_NR_OF_TREADS

Anzahl der Auftritte in Bezug auf die gesamte Treppe

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⊖	Standard	0

STAIR_LANDING_NUMBER

Anzahl der Zwischenpodeste in Bezug auf die gesamte Treppe

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⊖	Standard	0

STAIR_STAIR_GRADIENT Treppenneigung: Der Winkel des Verhältnisses Steigung/Auftritt im Radianten

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

STAIR_RULE_LIMITS Längen- / Winkel-Array mit zwei Dimensionen ([6] [2]), Erfassung von Minimal- und Maximalwerten, die in den Treppen-Standard-Einstellungen / Regeln und Standards / Trittstufen und Setzstufen eingestellt werden

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	[0]

Projekteinstellungen: die Einstellung der Sichtbarkeit dieser Werte wirkt sich nicht auf die Variable aus.

- *[1][1] - [1][2]: Steigungshöhe (R) Minimal- und Maximalwert*
- *[2][1] - [2][2]: Auftrittsbreite (G) Minimal- und Maximalwert*
- *[3][1] - [3][2]: 2 Steigungen + 1 Auftritt (2*R + G) Minimal- und Maximalwert*
- *[4][1] - [4][2]: Verhältnis Steigung/Auftritt (R / G) Minimal- und Maximalwert*
- *[5][1] - [5][2]: Steigung + Auftritt (R + G) Minimal- und Maximalwert*
- *[6][1] - [6][2]: Treppenneigung Minimal- und Maximalwert*

STAIR_RULE_FLAGS Boolesches Array mit zwei Dimensionen ([6] [2]), Aktivieren / Deaktivieren der Status-Erfassung von Limits gemäß STAIR_RULE_LIMITS, die in Treppen-Standardeinstellungen / Regeln und Standards / Tritt- und Setzstufen eingestellt sind

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	[0]

Wertindizes sind parallel zu STAIR_RULE_LIMITS. Mögliche Werte:

- *0 - Limit-Option des gleichen Index in STAIR_RULE_LIMITS wird derzeit nicht verwendet*
- *1 - Limit-Option des gleichen Index in STAIR_RULE_LIMITS wird derzeit verwendet*

Allgemeine Auftrittvariablen - für Auflistung und Etiketten verfügbar

Kompatibilität: eingeführt in ARCHICAD 21.

TREAD_STEP_INDEX Stufenindex der ausgewählten (aktuellen) Trittstufe

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

TREAD_GOING Lauflänge des der ausgewählten (aktuellen) Trittstufe

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

TREAD_ELEVATION				Höhe zum Projekt Null der ausgewählten (aktuellen) Trittstufe							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
TREAD_AREA				projizierte 2D-Fläche der ausgewählten (aktuellen) Trittstufe							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
TREAD_FRONT_AREA				Frontoberfläche der ausgewählten (aktuellen) Trittstufe							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
TREAD_VOLUME				Volumen der ausgewählten (aktuellen) Trittstufe							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
TREAD_BMATS				Array mit einer Dimension ([n]), Baustoffe der ausgewählten (aktuellen) Trittstufe (n = Anzahl der Baustoffe)							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

Allgemeine Steigungs-Variablen - verfügbar für Auflistung und Etiketten

Kompatibilität: eingeführt in ARCHICAD 21.

RISER_STEP_INDEX				Stufenindex der ausgewählten (aktuellen) Setzstufe							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
RISER_WIDTH				Polylinien-Länge der ausgewählten (aktuellen) Setzstufe							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
RISER_FRONT_AREA				Front-Oberfläche der ausgewählten (aktuellen) Setzstufe							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
RISER_VOLUME				Volumen der ausgewählten (aktuellen) Setzstufe							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

RISER_BMATS				Array mit einer Dimension ([n]), Baustoffe der ausgewählten (aktuellen) Setzstufe(n = Anzahl der Baustoffe)							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

Treppenstrukturvariablen - verfügbar für Auflistung und Etiketten

Kompatibilität: eingeführt in ARCHICAD 21.

STRUCTURE_WIDTH				Breite der ausgewählten (aktuellen) Strukturkomponente							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
STRUCTURE_HEIGHT				Höhe der ausgewählten (aktuellen) Strukturkomponente (Differenz von min und max z)							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
STRUCTURE_3DLENGTH				Volle 3D-Länge der ausgewählten (aktuellen) Strukturkomponente							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
STRUCTURE_VOLUME				Volumen der ausgewählten (aktuellen) Strukturkomponente							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
STRUCTURE_THICKNESS				Dicke der ausgewählten (aktuellen) Strukturkomponente							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

Modelldarstellungs-Variablen für Treppen

Zugehörige Einstellungen finden Sie im Dialogfeld Modelldarstellung / Treppen- und Geländeroptionen.

Kompatibilität: eingeführt in ARCHICAD 21.

GLOB_MVO_STAIR_FLOOR_PLAN_OPT				Modelldarstellung: Treppen-Grundriss-Symbol: 0 - Grundriss, 1 - Deckenspiegel							
2D	✓	3D	⛔	UI	✓	Parameter	⛔	Property	⛔	Standard	1

GLOB_MVO_STAIR_FLOOR_PLAN_COMP Modelldarstellung Treppe: Komponenten-Bitset

2D	✓	3D	⛔	UI	✓	Parameter	⛔	Property	⛔	Standard	1

mask: *Gibt Informationen über die sichtbaren Treppenkomponenten des Grundrisses zurück*

`mask = j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8 + 256*j9`*, hierbei kann j jeweils 0 oder 1 sein.*

`j1:` *Lauflinie*
`j2:` *Nummerierung*
`j3:` *Aufwärts-/Abwärts-Text*
`j4:` *Beschreibung*
`j5:` *Trittstufen-Zubehör*
`j6:` *Struktur - Wangen*
`j7:` *Struktur - Holm*
`j8:` *Struktur - Kragarm*
`j9:` *Struktur - monolitisch Kompatibilität: eingeführt in ARCHICAD 22.*

GLOB_MVO_RAILING_PLAN_COMP Modelldarstellung Geländer: Komponenten-Bitset

2D	✓	3D	⛔	UI	✓	Parameter	⛔	Property	⛔	Standard	127

mask: *Gibt Informationen über die sichtbaren Geländerkomponenten des Grundrisses zurück*

`mask = j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7`*, hierbei kann j jeweils 0 oder 1 sein.*

`j1:` *Pfosten*
`j2:` *Oberer Handlauf*
`j3:` *Zweite Handläufe*
`j4:` *Gurte*
`j5:` *Innenpfosten*
`j6:` *Geländerstab*
`j7:` *Paneele*

Treppen-2D-Variablen - nur für Grundrissdarstellung verfügbar

Kompatibilität: eingeführt in ARCHICAD 21.

Treppenrastervariablen

STAIR2D_FULL_TPOLYGON_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Tritttstufen-Polygon-Knoten

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

*n = Anzahl an Trittstufen * 5 Knoten für jeden Auftritt (im Allgemeinen):*

- *[n][1] - Koordinate x des Polygonknotens, gemessen vom Treppenursprung*
- *[n][2] - Koordinate y des Polygonknotens, gemessen vom Treppenursprung*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

STAIR2D_FULL_TPOLYGON_FLAGS Array mit zwei Dimensionen ([n] [4]), zusätzliche Daten der Trittstufenpolygone gemäß STAIR2D_FULL_TPOLYGON_GEOM

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

*n = Anzahl an Trittstufen * 5 Knoten für jeden Auftritt (im Allgemeinen):*

- *[n][1] - Index der Trittstufe*
- *[n][2] - Kantentyp ab Knoten (0 - führend, 1 - nachlaufend, 2 - links, 3 - rechts, 4 - Bruchlinie, -1 - Schließung)*
- *[n][3] - Sichtbarkeit der Kante ab dem Knoten (1 - sichtbar, 0 - weggelassen)*
- *[n][4] - Typ der Trittstufe (0 - Lauf, 1 - Podest)*

STAIR2D_FULL_RPOLYLINE_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Setzstufen-Polylinien-Knoten

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

*n = Anzahl der Setzstufen * 2 Knoten für jeden Auftritt (im Allgemeinen):*

- *[n][1] - Koordinate x des aus dem Treppenursprung gemessenen Polylinienknotens*
- *[n][2] - Koordinate y des aus dem Treppenursprung gemessenen Polylinienknotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

STAIR2D_FULL_RPOLYLINE_FLAGS Array mit zwei Dimensionen ([n] [1]), zusätzliche Daten der Setzstufen-Polylinien gemäß STAIR2D_FULL_RPOLYLINE_GEOM

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

*n = Anzahl der Setzstufen * 2 Knoten für jeden Auftritt (im Allgemeinen):*

- *[n][1] - Index der Setzstufe (Steigung)*

STAIR2D_FULL_BOUNDARY_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Treppengrenz-Polygon-Knoten

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

n = Anzahl der Randknoten (5 allgemein):

- *[n][1] - Koordinate x des Polygonknotens, gemessen vom Treppenursprung*
- *[n][2] - Koordinate y des Polygonknotens, gemessen vom Treppenursprung*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

Die folgenden Globals werden verwendet, um die unterhalb der ersten Bruchlinie dargestellten Teile der Treppe zu definieren.

STAIR2D_LOWER_TPOLYGON_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Trittstufen-Polygon-Knoten des unteren Teils, ähnlich wie STAIR2D_FULL_TPOLYGON_GEOM

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

STAIR2D_LOWER_TPOLYGON_FLAGS Array mit zwei Dimensionen ([n] [4]), zusätzliche Daten von Trittstufen-Polygon-Knoten des unteren Teils, ähnlich wie STAIR2D_FULL_TPOLYGON_FLAGS

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

STAIR2D_LOWER_RPOLYLINE_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Setzstufen-Polylinien-Knoten des unteren Teils, ähnlich wie STAIR2D_FULL_RPOLYLINE_GEOM

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

STAIR2D_LOWER_RPOLYLINE_FLAGS Array mit zwei Dimensionen ([n] [1]), zusätzliche Daten von Setzstufen-Polylinienknoten des unteren Teils, ähnlich wie STAIR2D_FULL_RPOLYLINE_FLAGS

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

STAIR2D_LOWER_BOUNDARY_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets der Treppengrenz-Polygon-Knoten des unteren Teils, ähnlich wie STAIR2D_FULL_BOUNDARY_GEOM

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

Die folgenden Globalen werden verwendet, um die Teile der Treppe zu definieren, die zwischen zwei Bruchlinien dargestellt sind.

STAIR2D_MIDDLE_TPOLYGON_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Trittstufen-Polygon-Knoten des Mittelteils, ähnlich wie STAIR2D_FULL_TPOLYGON_GEOM

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

STAIR2D_MIDDLE_TPOLYGON_FLAGS Array mit zwei Dimensionen ([n] [4]), zusätzliche Daten von Trittstufen-Polygon-Knoten des Mittelteils, ähnlich wie STAIR2D_FULL_TPOLYGON_FLAGS

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

STAIR2D_MIDDLE_RPOLYLINE_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Setzstufen-Polylinienknoten des Mittelteils, ähnlich wie STAIR2D_FULL_RPOLYLINE_GEOM

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

STAIR2D_MIDDLE_RPOLYLINE_FLAGS Array mit zwei Dimensionen ([n] [1]), zusätzliche Daten von Setzstufen-Polylinienknoten des Mittelteils, ähnlich wie STAIR2D_FULL_RPOLYLINE_FLAGS

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

STAIR2D_MIDDLE_BOUNDARY_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Treppengrenz-Polygon-Knoten des Mittelteils, ähnlich wie STAIR2D_FULL_BOUNDARY_GEOM

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

Die folgenden Globals werden verwendet, um die über der letzten Bruchlinie dargestellten Teile der Treppe zu definieren.

STAIR2D_UPPER_TPOLYGON_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Trittstufen-Polygon-Knoten des oberen Teils, ähnlich wie STAIR2D_FULL_TPOLYGON_GEOM

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

STAIR2D_UPPER_TPOLYGON_FLAGS Array mit zwei Dimensionen ([n] [4]), zusätzliche Daten von Profil-Polygon-Knoten des oberen Teils, ähnlich wie STAIR2D_FULL_TPOLYGON_FLAGS

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

STAIR2D_UPPER_RPOLYLINE_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Setzstufen-Polylinien-Knoten des oberen Teils, ähnlich wie STAIR2D_FULL_RPOLYLINE_GEOM

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

STAIR2D_UPPER_RPOLYLINE_FLAGS Array mit zwei Dimensionen ([n] [1]), zusätzliche Daten von Setzstufen-Polylinienknoten des Oberteils, ähnlich wie STAIR2D_FULL_RPOLYLINE_FLAGS

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

STAIR2D_UPPER_BOUNDARY_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Treppengrenz-Polygon-Knoten des oberen Teils, ähnlich wie STAIR2D_FULL_BOUNDARY_GEOM

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

Symbolvariablen der Lauflinie der Treppe

STAIR2D_FULL_WALKLINE_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Treppen-Lauflinien-Knoten, in voller Länge

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

n = Anzahl der Polylinienknoten für die volle Länge der Lauflinie

- *[n][1] - Koordinate x des aus dem Treppenursprung gemessenen Polylinienknotens*
- *[n][2] - Koordinate y des aus dem Treppenursprung gemessenen Polylinienknotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

STAIR2D_FULL_WALKLINE_FLAGS Array mit zwei Dimensionen ([n] [2]), zusätzliche Daten von Treppen-Lauflinien-Knoten, in voller Länge

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

n = Anzahl der Polylinienknoten für die volle Länge der Lauflinie

`mask:` *Gibt Informationen über die Knotenposition zurück*

`mask = j1 + 2*j2 + 4*j3 + 8*j4`, *hierbei kann j jeweils 0 oder 1 sein.*

`j1:` *Knoten befindet sich am hinteren Ende des ersten Auftritts*

`j2:` *Knoten befindet sich am vorderen Ende des Podestes*

`j3:` *Knoten befindet sich am hinteren Ende des Podestes*

`j4:` *Knoten befindet sich am vorderen Ende des letzten Auftritts*

- *[n][1] - Standort-Maskwert des Knotens*
- *[n][2] - Index der Trittstufe, bei welcher der Knoten auf dem hinteren Ende oder darüber liegt*

STAIR2D_LOWER_WALKLINE_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Lauflinien-Knoten, Unterteil (gleiche Logik wie Treppen-Polygon-Beschneidung)

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

n = Anzahl der Polylinienknoten für den unteren Teil der Lauflinie

- *[n][1] - Koordinate x des aus dem Treppenursprung gemessenen Polylinienknotens*
- *[n][2] - Koordinate y des aus dem Treppenursprung gemessenen Polylinienknotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

STAIR2D_LOWER_WALKLINE_FLAGS Ganzzahl-Array mit zwei Dimensionen ([n] [2]), zusätzliche Daten von Lauflinien-Knoten, Unterteil

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

n = Anzahl der Polylinienknoten für den unteren Teil der Lauflinie

`mask:` *Gibt Informationen über die Knotenposition zurück*

`mask = j1 + 2*j2 + 4*j3 + 8*j4`, *hierbei kann j jeweils 0 oder 1 sein.*

`j1:` *Knoten befindet sich am hinteren Ende des ersten Auftritts*

`j2:` *Knoten befindet sich am vorderen Ende des Podestes*

`j3:` *Knoten befindet sich am hinteren Ende des Podestes*

`j4:` *Knoten befindet sich am vorderen Ende des letzten Auftritts*

- *[n][1] - Standort-Maskwert des Knotens*
- *[n][2] - Index der Trittstufe, bei welcher der Knoten auf dem hinteren Ende oder darüber liegt*

STAIR2D_MIDDLE_WALKLINE_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Lauflinien-Knoten, Mittelteil

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

n = Anzahl der Polylinienknoten für den Mittelteil der Lauflinie

- *[n][1] - Koordinate x des aus dem Treppenursprung gemessenen Polylinienknotens*
- *[n][2] - Koordinate y des aus dem Treppenursprung gemessenen Polylinienknotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

STAIR2D_MIDDLE_WALKLINE_FLAGS Ganzzahl-Array mit zwei Dimensionen ([n] [2]), zusätzliche Daten von Lauflinien-Knoten, Mittelteil

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

n = Anzahl der Polylinienknoten für den Mittelteil der Lauflinie

mask: *Gibt Informationen über die Knotenposition zurück*

`mask = j1 + 2*j2 + 4*j3 + 8*j4`, *hierbei kann j jeweils 0 oder 1 sein.*

`j1:` *Knoten befindet sich am hinteren Ende des ersten Auftritts*

`j2:` *Knoten befindet sich am vorderen Ende des Podestes*

`j3:` *Knoten befindet sich am hinteren Ende des Podestes*

`j4:` *Knoten befindet sich am vorderen Ende des letzten Auftritts*

- *[n][1] - Standort-Maskwert des Knotens*
- *[n][2] - Index der Trittstufe, bei welcher der Knoten auf dem hinteren Ende oder darüber liegt*

STAIR2D_UPPER_WALKLINE_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Lauflinien-Knoten, Oberteil

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

n = Anzahl der Polylinienknoten für den oberen Teil der Lauflinie

- *[n][1] - Koordinate x des aus dem Treppenursprung gemessenen Polylinienknotens*
- *[n][2] - Koordinate y des aus dem Treppenursprung gemessenen Polylinienknotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

STAIR2D_UPPER_WALKLINE_FLAGS Array mit zwei Dimensionen ([n] [2]), zusätzliche Daten von Lauflinien-Knoten, Oberteil

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

n = Anzahl der Polylinienknoten für den oberen Teil der Lauflinie

mask: *Gibt Informationen über die Knotenposition zurück*

`mask = j1 + 2*j2 + 4*j3 + 8*j4`, *hierbei kann j jeweils 0 oder 1 sein.*

`j1:` *Knoten befindet sich am hinteren Ende des ersten Auftritts*

`j2:` *Knoten befindet sich am vorderen Ende des Podestes*

`j3:` *Knoten befindet sich am hinteren Ende des Podestes*

`j4:` *Knoten befindet sich am vorderen Ende des letzten Auftritts*

- *[n][1] - Standort-Maskwert des Knotens*
- *[n][2] - Index der Trittstufe, bei welcher der Knoten auf dem hinteren Ende oder darüber liegt*

Symbolvariablen der Bruchlinien-Symbole

STAIR2D_BREAKMARK_GEOM Array mit zwei Dimensionen ([n] [9]), Daten von Bruchlinien-Polylinienknoten

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

*n = Anzahl der Bruchlinien * 9 Knoten-Daten für jede Bruchlinie (maximale Anzahl der angezeigten Bruchlinien ist 4):*

- *[n][1] - Koordinate x des aus dem Treppenursprung gemessenen Polylinienstartpunktes*
- *[n][2] - Koordinate y des aus dem Treppenursprung gemessenen Polylinienstartpunktes*
- *[n][3] - Koordinate x des aus dem Treppenursprung gemessenen Polylinienendpunktes*
- *[n][4] - Koordinate y des aus dem Treppenursprung gemessenen Polylinienendpunktes*
- *[n][5] - Bruchlinienwinkel, berechnet aus der Senkrechten der Lauflinie, in Grad (Wertaktualisierung durch Bearbeitung). Siehe auch STAIR2D_BREAKMARK_ANGLE.*
- *[n][6] - Koordinate x des Anfangs-Überstands gemessen vom Ursprung der Treppe Kompatibilität: eingeführt in ARCHICAD 22.*
- *[n][7] - Koordinate y des Anfangs-Überstands gemessen vom Ursprung der Treppe Kompatibilität: eingeführt in ARCHICAD 22.*
- *[n][8] - Koordinate x des End-Überstands gemessen vom Ursprung der Treppe Kompatibilität: eingeführt in ARCHICAD 22.*
- *[n][9] - Koordinate y des End-Überstands gemessen vom Ursprung der Treppe Kompatibilität: eingeführt in ARCHICAD 22.*

STAIR2D_BREAKMARK_FLAGS Ganzzahl-Array mit zwei Dimensionen ([n] [1]), zusätzliche Daten zur Bruchlinien-Polylinien-Sichtbarkeit

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

n = Anzahl der in den aktuellen Einstellungen sichtbaren Bruchlinien (maximal 4)

Attribute Set Index:

1: Bruchlinie sichtbar,
2: Bruchlinie unsichtbar

STAIR2D_BREAKMARK_ANGLE Bruchlinien-Winkel in Grad (Fließkommazahl-Wertetyp), wie im Treppen-Einstellungsdialog eingestellt. Hält den voreingestellten Wert bei, auch wenn die Bruchlinie editiert wird.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

Beschreibungsvariablen von Treppenläufen und Podesten

STAIR2D_DESCRIPTION_POSITION Array mit zwei Dimensionen ([n] [4]), die Informationen über die Position und die Richtung der Treppenbeschriftung enthält

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

n: *Positionsdefinition:*

- *1: Mitte des ersten Laufes,*
- *2: Mitte des ersten Podestes,*
- *3: Mitte des letzten Laufes,*
- *4: Mitte des letzten Podestes,*
- *5: Mitte der gesamten Treppe.*

- *[n][1] - Koordinate x der Beschriftungs-Position gemessen aus Treppenursprung*
- *[n][2] - Koordinate y der Beschriftungs-Position gemessen aus Treppenursprung*
- *[n][3] - Lauflinien-Normalen-Vektor: x-Koordinate der Beschriftungsposition*
- *[n][4] - Lauflinien-Normalen-Vektor: y-Koordinate der Beschriftungsposition*

2D-Variablen der Treppen-Abläufe

In der Geometrie der folgenden Globalen Variablen ist keine Unterbrechung (Aufschneiden) mittels Bruchlinien vorhanden.

STAIR2D_EXT_TPOLYGON_GEOM Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von erweiterten Auftritts-Polygon-Knoten (einschließlich Entwässerung). Ähnliche Struktur wie STAIR2D_FULL_TPOLYGON_GEOM.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

*n = Anzahl an Trittstufen * 5 Knoten für jeden Auftritt (im Allgemeinen):*

- *[n][1] - Koordinate x des erweiterten Trittstufen-Polygonknotens, gemessen vom Treppenursprung*
- *[n][2] - Koordinate y des erweiterten Trittstufen-Polygonknotens, gemessen vom Treppenursprung*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

STAIR2D_EXT_TPOLYGON_FLAGS

Array mit zwei Dimensionen ([n] [4]), zusätzliche Daten der erweiterten Trittstufen-Polygone, in Übereinstimmung mit STAIR2D_EXT_TPOLYGON_GEOM. Ähnliche Struktur wie STAIR2D_FULL_TPOLYGON_FLAGS.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

*n = Anzahl an Trittstufen * 5 Knoten für jeden Auftritt (im Allgemeinen):*

- *[n][1] - Index der Trittstufe*
- *[n][2] - Kantentyp beginnend vom Knoten (0 - startend, 1 - nachlaufend, 2 - links, 3 - rechts, -1 - schließend)*
- *[n][3] - Sichtbarkeit der Kante ab dem Knoten (1 - sichtbar, 0 - weggelassen)*
- *[n][4] - Typ der Trittstufe (0 - Lauf, 1 - Podest)*

STAIR2D_EXT_RPOLYLINE_GEOM

Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von erweiterten Setzstufen-Polylinien-Knoten (einschließlich Entwässerung). Ähnliche Struktur wie STAIR2D_FULL_RPOLYLINE_GEOM.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

*n = Anzahl der Setzstufen * 2 Knoten für jeden Auftritt (im Allgemeinen):*

- *[n][1] - Koordinate x des aus dem Treppenursprung gemessenen Polylinienknotens*
- *[n][2] - Koordinate y des aus dem Treppenursprung gemessenen Polylinienknotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

STAIR2D_EXT_RPOLYLINE_FLAGS

Array mit zwei Dimensionen ([n] [1]), zusätzliche Daten der Setzstufen-Polylinien gemäß STAIR2D_EXT_RPOLYLINE_GEOM. Ähnliche Struktur wie STAIR2D_FULL_RPOLYLINE_FLAGS.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

*n = Anzahl der Setzstufen * 2 Knoten für jeden Auftritt (im Allgemeinen):*

- *[n][1] - Index der Setzstufe (Steigung)*

STAIR2D_DRAIN_TPOLYGON_GEOM

Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Ablauf-Polygon-Knoten. Ähnliche Struktur wie STAIR2D_FULL_TPOLYGON_GEOM.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

*n = Anzahl an Trittstufen * 5 Knoten für jeden Auftritt (im Allgemeinen):*

- *[n][1] - Koordinate x des erweiterten Trittstufen-Polygonknotens, gemessen vom Treppenursprung*
- *[n][2] - Koordinate y des erweiterten Trittstufen-Polygonknotens, gemessen vom Treppenursprung*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

STAIR2D_DRAIN_TPOLYGON_FLAGS Array mit zwei Dimensionen ([n] [4]), zusätzliche Daten der Ablauf-Polygone gemäß STAIR2D_DRAIN_TPOLYGON_GEOM. Ähnliche Struktur wie STAIR2D_FULL_TPOLYGON_FLAGS.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

*n = Anzahl an Trittstufen * 5 Knoten für jeden Auftritt (im Allgemeinen):*

- *[n][1] - Index der Trittstufe*
- *[n][2] - Kantentyp ab Knoten*
  - *0 - startend*
  - *1 - endend*
  - *2 - links (+100 - Ablauf-Seite links, +200 - Ablauf-Seite links und "gestuft")*
  - *3 - rechts (+100 - Ablauf-Seite links, +200 - Ablauf-Seite rechts und "gestuft)*
  - *-1 - schließend*
- *[n][3] - Sichtbarkeit der Kante ab dem Knoten (1 - sichtbar, 0 - weggelassen)*
- *[n][4] - Typ der Trittstufe (0 - Lauf, 1 - Podest)*

Treppenstruktur 2D-Variablen - Holm-Strukturen

STAIR2D_POLYLINES_GEOM Array mit zwei Dimensionen ([n] [3]), wobei n die Anzahl der Struktur-Polylinienknoten des aktuellen Laufes/ Podestes in 2D ist, enthält geometrische Daten der von der Treppengrenze abgeleiteten Polylinienknoten: Linke Grenzlinie, rechte Grenzlinie und Mittellinie.

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Koordinate x des Struktur-Polylinien-Knotens gemessen vom Treppenursprung*
- *[n][2] - Koordinate y des Struktur-Polylinien-Knotens gemessen vom Treppenursprung*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

STAIR2D_POLYLINES_FLAGS Array mit zwei Dimensionen ([n] [1]), wobei n die Anzahl der Struktur-Polylinien-Knoten des aktuellen Laufes/ Podestes in 2D ist, enthält Gruppendaten der Polylinienknoten.

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Positionsflags, Punkt n gehört zu: 0 - linke Grenze, 1 - rechte Grenze, 2 - Mittellinie*

STAIR2D_STRUCT_ATTRIBUTES Array mit zwei Dimensionen ([2] [7]), die Attribute-Einstellungen der sichtbaren ([1] [n]) und unsichtbaren ([2] [n]) Teile der Struktur enthalten.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0, 0, 0, 0, 0, 0, 0]

- *[n][1] - Typindex der Begrenzungslinie*
- *[n][2] - Stiftindex der Grenze*
- *[n][3] - Symbolschraffur-Ttyp*
- *[n][4] - Symbolschraffur-Stift*
- *[n][5] - Symbolschraffur-Hintergrundstift*
- *[n][6] - Symbolschraffur- ON / OFF Boolesche Steuerung nur für benutzerdefinierte Anzeige (STAIR2D_CUSTOMDISPLAY = 1).*
- *[n][7] - Symbolteil aktivieren: Boolesche Ein / Aus - Steuerung nur für benutzerdefinierte Anzeige (STAIR2D_CUSTOMDISPLAY = 1).*

Treppenstruktur 2D-Variablen - Massive Struktur

Kompatibilität: eingeführt in ARCHICAD 22.

Diese Globalen Variablen sind mit Werten befüllt mit Rücksicht auf die aktuelle Grundriss/Deckenspiegel - Sicht.

STAIR2D_FULL_SPOLYGON_GEOM Array mit 2 Dimensionen ([n][3]), welche Sub-Polygone der 2D-Projektion der Treppe enthält.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Koordinate x der Strukturpolygonknoten, gemessen vom Ursprung der Treppe*
- *[n][2] - Koordinate y der Strukturpolygonknoten, gemessen vom Ursprung der Treppe*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

STAIR2D_FULL_SPOLYGON_FLAGS Array mit 2 Dimensionen ([n][2]), bei welchem n die Anzahl der Polygonknoten ist.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Index des Sub-Polygons*
- *[n][2] - Typ des Sub-Polygons*
  - *1: Lauf-Begrenzung - ohne Verbindung*
  - *2: Lauf-Begrenzung - mit Verbindung*
  - *3: Podest-Begrenzung - ohne Verbindung*
  - *4: Podest-Begrenzung - mit Verbindung*
  - *5: Verbindung*
  - *6: Ablauf (Entwässerung)*
- *[n][3] - Typ der Kante des Subpolygons*
  - *-1: Abschlusspunkt (der nächste Punkt beginnt ein neues Sub-Polygon)*
  - *0: Unsichtbare Kante (nur im Grenzbereich)*
  - *1: sichtbare Massivkante (sichtbares oder verstecktes Attribut gesetzt durch STAIR2D_VISIBILITY)*
  - *2: sichtbare Verbindungskante (sichtbares oder verstecktes Attribut gesetzt durch STAIR2D_VISIBILITY)*
  - *3: sichtbare Verbindungsdetailkante (sichtbares oder verstecktes Attribut gesetzt durch STAIR2D_VISIBILITY)*

Ein Subpolygon vom Typ Verbindung kann gemischte Massiv- und Verbindungskanten haben.

STAIR2D_FULL_SPOLYLINE_GEOM Array mit zwei Dimensionen ([n][2]), enthält die Kanten innerhalb der Umgrenzung.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[2*i][1] - Koordinate x des Startknotens gemessen vom Treppenursprung*
- *[2*i][2] - Koordinate y des Startknotens gemessen vom Treppenursprung*
- *[2*i + 1][1] - Koordinate x des Endknotens gemessen vom Treppenursprung*
- *[2*i + 1][2] - Koordinate y des Endknotens gemessen vom Treppenursprung*

STAIR2D_FULL_SPOLYLINE_FLAGS Array mit einer Dimension [n], bei welchem n die Anzahl der Knoten in STAIR2D_FULL_SPOLYLINE_GEOM ist.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n] - Kantentyp*
  - *1: Auftritt, beschnitten mit Ablauf*
  - *2: Auftritt, vollständig*
  - *3: Auftritt Kantenprofil (geneigte Setzstufe), beschnitten mit Ablauf*
  - *4: Auftritt Kantenprofil (geneigte Setzstufe), vollständig*
  - *5: Podestlinie, beschnitten mit Ablauf*
  - *6: Podestlinie, vollständig*
  - *7: Verbindung*
  - *8: Verbindungs-Detail*
  - *9: Ablauf (abgestuft)*
  - *10: Ablauf Kantenprofil (Ablauf mit geneigte Setzstufen)*

STAIR2D_MONOLITH_ATTRIBUTES Array mit zwei Dimensionen ([2][23]), welches Einstellungen zu Attributen und Sichtbarkeit der sichtbaren ([1][n]) und unsichtbaren ([2][n]) Teile der Massivstruktur enthält.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Struktur-Sichtbarkeit (bool)*
- *[n][2] - Ablauf-Sichtbarkeit (bool)*
- *[n][3] - Kontur-Linientyp*
- *[n][4] - Kontur-Linienstift*
- *[n][5] - Struktur-Sichtbarkeit Auftritt (bool)*
- *[n][6] - Liniensichtbarkeit Podest (bool)*
- *[n][7] - Linientyp Auftritt*
- *[n][8] - Linienstift Auftritt*
- *[n][9] - Struktur-Sichtbarkeit Kantenprofil (bool)*
- *[n][10] - Linientyp Kantenprofil Auftritt*
- *[n][11] - Linienstift Kantenprofil Auftritt*
- *[n][12] - Sichtbarkeit Verbindung (bool)*
- *[n][13] - Linientyp Verbindung*
- *[n][14] - Linienstift Verbindung*
- *[n][15] - Linientyp Verbindungsdetail*
- *[n][16] - Linienstift Verbindungsdetail*
- *[n][17] - Sichtbarkeit Ablaufschraffur und Strukturschraffur (bool)*
- *[n][18] - Struktur Schraffurtyp*
- *[n][19] - Struktur Schraffurstift*
- *[n][20] - Struktur Schraffur-Hintergrundstift*
- *[n][21] - Ablauf Schraffurtyp*
- *[n][22] - Ablauf Schraffurstift*
- *[n][23] - Ablauf Schraffur-Hintergrundstift*

Allgemeine 2D zugehörige Variablen

STAIR2D_CURRSTORY_LOCATION Informationen über die aktuelle Geschoss-Sichtbarkeit der Treppe

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

values:

`1:` *unterhalb des relevanten Geschosses,*
`2:` *erstes relevantes Geschoss,*
`3:` *inter-relevantes Geschoss,*
`4:` *oberstes letzt-relevantes Geschoss,*
`5:` *oberhalb des relevanten Geschosses.*

STAIR2D_LAYOUT_TYPES[5] Array mit einer Dimension ([5]), Informationen über die Darstellungsvarianten der Treppe nach STAIR2D_CURRSTORY_LOCATION

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	[0]

- *[1] - Darstellungseinstellung: unterhalb des relevanten Geschosses*
- *[2] - Darstellungseinstellung: erstes relevantes Geschoss*
- *[3] -Darstellungseinstellung: inter-relevantes Geschoss*
- *[4] -Darstellungseinstellung: oberstes letzt-relevantes Geschoss*
- *[5] - Darstellungseinstellung: oberhalb des relevanten Geschosses*

Layout setting values:

`0:` *ungültig*
`1:` *mit Bruchlinie: Sichtbar - Unsichtbar*
`2:` *Keine Bruchlinie: Sichtbar*
`3:` *Unterhalb der Bruchlinie: Sichtbar*
`4:` *Oberhalb der Bruchlinie: Sichtbar*
`7:` *Keine Bruchlinie: Unsichtbar*
`8:` *Oberhalb der Bruchlinie: Unsichtbar*
`9:` *Unterhalb der Bruchlinie: Unsichtbar*
`13:` *Mit Bruchlinie: Alle sichtbar*
`14:` *Mit Bruchlinie: Unsichtbar- Sichtbar*
`5:` *Multistory 2D: Sichtbar zwischen Bruchlinien*
`6:` *Multistory 2D: Unsichtbar - Sichtbar - Unsichtbar*
`10:` *Multistory 2D: Alle sichtbar*
`11:` *Multistory 2D: Unsichtbar - Sichtbar - Ohne*
`12:` *Multistory 2D: Ohne - Sichtbar - Unsichtbar*

STAIR2D_VISIBILITY Typ des aktiven Attributsatzes der aktuellen Zeichnung

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

Nur verwendbar in Verbindung mit Raster und Auftritts-2D-Symbol.

values:

`1:` *'Sichtbarer' Attributesatz ist aktiv*
`0:` *'Unsichtbarer' Attributesatz ist aktiv*

STAIR2D_CUSTOMDISPLAY enthält Informationen über die Modelldarstellung der Treppe

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

values:

0: Treppe wird entsprechend den Einstellungen der Modelldarstellung dargestellt

1: Treppe wird mit benutzerdefinierten Einstellungen dargestellt

STAIR_START_WITH_RISER Boolescher Wert, ob die Treppe mit einer Steigung beginnt.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

Kompatibilität: eingeführt in ARCHICAD 22.

STAIR_END_WITH_RISER Boolescher Wert, ob die Treppe mit einer Steigung endet.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

Kompatibilität: eingeführt in ARCHICAD 22.

STAIR_TREAD_EXIST Boolesches Array ([2]), ob die Treppe eine Trittstufenkomponente besitzt.

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

Kompatibilität: eingeführt in ARCHICAD 22.

Im Editiermodus angepasste Trittstufen haben auf diese Werte keine Auswirkung.

- *[1] - im Lauf*
- *[2] - im Podest*

STAIR_RISER_EXIST Boolesches Array ([2]) ob die Treppe eine Setzstufenkomponente besitzt.

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

Kompatibilität: eingeführt in ARCHICAD 22.

Im Editiermodus angepasste Setzstufen haben auf diese Werte keine Auswirkung.

- *[1] - im Lauf*
- *[2] - im Podest*

STAIR_NOSING_EXIST Boolesches Array ([2]) ob die Treppe einen Überstand an der Trittstufe besitzt (Wert > 0).

2D	✓	3D	✗	UI	✓	Parameter	✓	Property	✗	Standard	0

Kompatibilität: eingeführt in ARCHICAD 22.

Im Editiermodus angepasste Trittstufen haben auf diese Werte keine Auswirkung.

- *[1] - im Lauf*
- *[2] - im Podest*

Treppen-3D-Variablen - verfügbar nur für 3D-Darstellung (und Verbindungspunkte)

Kompatibilität: eingeführt in ARCHICAD 21.

3D-Variablen für Setzstufe der Treppen

RISER_HEIGHT 3D-Höhenwert der ausgewählten Setzstufe

2D	✗	3D	✓	UI	✗	Parameter	✗	Property	✗	Standard	0

RISER_THICKNESS 3D-Dicke der ausgewählten Setzstufe

2D	✗	3D	✓	UI	✗	Parameter	✗	Property	✗	Standard	0

STAIR_RISER_GEOMETRY Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Setzstufen-Polyline-Pfadknoten

2D	✗	3D	✓	UI	✗	Parameter	✗	Property	✗	Standard	[0]

n = Anzahl der Knoten der Setzstufen-Polylinie:

- *[n][1] - Koordinate x des aus dem Treppenursprung gemessenen Polylinienknotens*
- *[n][2] - Koordinate y des aus dem Treppenursprung gemessenen Polylinienknotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

RISER_CUT Array mit zwei Dimensionen ([2] [2]), enthält Daten der Start- und Schließpunkte des idealen TUBE (Modellierung der Setzstufe in 3D)

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

Die Display-Segmentierung des TUBEs hat keinen Einfluss auf den Wert dieser Variablen. Um in allen Fällen ein korrektes Modell zu erreichen, können mit dieser Globalen die tatsächlichen Start- und Schließpunkte des TUBEs nach dem segmentierten Bogen berechnet werden.

- *[1][1] - Koordinate x des Startknotens gemessen vom Treppenursprung*
- *[1][2] - Koordinate y des Startknotens gemessen vom Treppenursprung*
- *[2][1] - Koordinate x des Schließknotens gemessen vom Treppenursprung*
- *[2][2] - Koordinate y des Schließknotens gemessen vom Treppenursprung*

RISER_SLANT_ANGLE Neigungswinkel der gewählten Setzstufe

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

Falls der Neigungswinkel auf 0 gesetzt ist, beträgt der Wert dieser Globalen 90 Grad.

2D-3D Variablen der Trittstufe

TREAD_THICKNESS 3D-Stärkewert der gewählten Trittstufe

2D	✅	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

STAIR_TREAD_GEOMETRY Array mit zwei Dimensionen ([n] [3]), Daten-Triplets von Trittstufen-Polygon-Knoten

2D	✅	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

n = Anzahl der Knoten des Trittstufenpolygons:

- *[n][1] - Koordinate x des vom Treppenursprung gemessenen Knotens*
- *[n][2] - Koordinate y des vom Treppenursprung gemessenen Knotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

STAIR_TREAD_FLAGS — Array mit zwei Dimensionen ([n] [1]), zusätzliche Daten der Trittstufen-Polygon-Kanten (ab Knoten) entsprechend STAIR_TREAD_GEOMETRY

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

n = Anzahl der Knoten des Trittstufenpolygons:

- *[n][1] - Flag der n-ten Kante des Polygons*

flags:

0: führende Kante des Trittstufenpolygons
1: hintere Kante des Trittstufenpolygons
2: linke Kante des Trittstufenpolygons
3: rechte Kante des Trittstufenpolygons
-1: Schließknoten des Trittstufenpolygons

TREAD_LOWER_RISER_THICKNESS — Stärke der Setzstufe unterhalb der aktuellen Lauffläche (gemessen zwischen Raster und Struktur)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

TREAD_LOWER_RISER_HEIGHT — Höhe der Setzstufe unterhalb der aktuellen Lauffläche (gemessen zwischen der Unterkante der aktuellen Trittstufe und der Oberkante der vorherigen Trittstufe)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

TREAD_LOWER_RISER_SLANT_ANGLE — Neigungswinkel der Setzstufe unterhalb der aktuellen Trittstufe in Grad (falls Slanting = 0, ist der Wert 90 Grad)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

TREAD_UPPER_RISER_THICKNESS — Stärke der Setzstufe über der aktuellen Trittstufe (gemessen zwischen Raster und Struktur)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

TREAD_UPPER_RISER_HEIGHT — Höhe der Setzstufe über der aktuellen Trittstufe (gemessen zwischen der Oberkante der aktuellen Trittstufe und der Unterkante der folgenden Trittstufe)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

TREAD_UPPER_RISER_SLANT_ANGLE — Neigungswinkel der Setzstufe oberhalb der aktuellen Trittstufe in Grad (falls Slanting = 0, ist der Wert 90 Grad)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

TREAD_NOSING_METHOD

Informationen über die Abkantungsmethode der aktuellen Trittstufe, wie im Dialog Treppeneinstellungen eingestellt

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	Standard	0

values:

1: Abkantung durch Längenwert

2: Abkantung durch Neigungslänge

TREAD_NOSING

Enthält den Wert der Abkantungstiefe der ausgewählten Trittstufe (horizontaler Offset, wie im Dialog Treppeneinstellungen eingestellt), falls TREAD_NOSING_METHOD = 1 (Abkantung nach Wertlänge)

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	Standard	0

TREAD_NOSING_BY_SLANTING

Enthält den Wert der Abkantungslänge der ausgewählten Trittstufe (vertikaler Offset, wie im Dialog Treppeneinstellungen eingestellt), falls TREAD_NOSING_METHOD = 2 (Abkantung nach Neigungslänge)

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	Standard	0

Strukturvariablen der Treppe

STAIR_STRUCTURE_GEOMETRY Array mit zwei Dimensionen ([n] [14]), wobei n die Anzahl der Struktur-Polygone Punkte des aktuellen Laufes/ Podestes ist.

2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Koordinate x des Struktur-Polygonknotens n, gemessen vom Treppenursprung*
- *[n][2] - Koordinate y des Struktur-Polygonknotens n, gemessen vom Treppenursprung*
- *[n][3] - Koordinate z des Struktur-Polygonknotens n, gemessen vom Treppenursprung*
- *[n][4] - Zentralwinkel der Kante in Grad beginnend vom Knoten n (0 - gerade, <0 - gegen den Uhrzeigersinn gekrümmt, > 0 - im Uhrzeigersinn gekrümmt)*
- *[n][5] - Neigungswinkel der Setzstufen-Kante ab Knoten n. Im Falle einer Vorderkante gehört der Winkel zur Setzstufe unterhalb des Polygons. Im Falle einer Hinterkante gehört der Winkel zur Setzstufe über dem Polygon. Enthält 0 für alle anderen Randkategorien.*
- *[n][6] - Höhe der Setzstufe unterhalb des Struktur-Polygons (gibt 0 zurück, falls keine Setzstufe vorhanden ist oder wenn das Polygon ein Verbindungs-Polygon ist)*
- *[n][7] - Höhe der Setzstufe oberhalb des Struktur-Polygons (gibt 0 zurück, falls keine Setzstufe vorhanden ist oder wenn das Polygon ein Verbindungs-Polygon ist)*
- *[n][8] - Stärke der Trittstufe des Struktur-Polygons (gibt 0 zurück, wenn das Polygon ein Verbindungs-Polygon ist)*
- *[n][9] - Stärke des Abstandes unterhalb der Trittstufe des Struktur-Polygons (liefert 0, wenn das Polygon ein Verbindungs-Polygon ist)*
- *[n][10-14] - Kompatibilität: eingeführt in ARCHICAD 22.*
- *[n][10] - Höhe der freitragenden Struktur (ermittelt aus seinem Parameter ac_stairStructureThickness unterhalb des Profils an dieser Stelle. Null bei Verbindungspolygonen.*
- *[n][11] - Trittstufen-Überstand beginnt an der Kante von diesem Punkt aus.*
  - *Null bei Verbindungspolygonen*
  - *Überstand der Trittstufe über die Vorderkanten*
  - *Überstand der nächsten Trittstufe über die Hinterkanten*
  - *Null bei allen anderen Kanten*
- *[n][12] - Setzstufenstärke an der Kante ausgehend von diesem Punkt.*
  - *Null bei Verbindungspolygonen*
  - *Stärke der Setzstufe unterhalb der Vorderkanten*
  - *Stärke der Setzstufe oberhalb der Hinterkanten*
  - *Null bei allen anderen Kanten*
- *[n][13] - Spaltstärke der Setzstufe an der Kante ausgehend von diesem Punkt.*
  - *Null bei Verbindungspolygonen*
  - *Spaltstärke der Setzstufe unterhalb der Vorderkanten*
  - *Spaltstärke der Setzstufe oberhalb der Hinterkanten*
  - *Null bei allen anderen Kanten*
- *[n][14] - Horizontale Höhe der freitragenden Struktur (ermittelt aus seinem Parameter ac_stairStructureHorizThick) an der Kante, die an diesem Punkt beginnt.*
  - *Null bei Verbindungspolygonen*
  - *Horizontale Stärke der freitragenden Struktur unterhalb der Vorderkanten*
  - *Horizontal Stärke der freitragenden Struktur oberhalb der Hinterkanten*
  - *Null bei allen anderen Kanten*

STAIR_STRUCTURE_FLAGS Array mit zwei Dimensionen ([n] [3]), wobei n die Anzahl der Strukturpolygonpunkte des aktuellen Laufes/ des Podestes ist.

2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Polygon-Typ des Knotens n im Struktur-Polygon: Kern- oder Verbindungs-Polygonpolygon*
- *[n][2] - Zeigt an, wozu welcher Seitenknoten n gehört: Linkes oder rechtes Struktur-Polygon, wo die Teilung die Mittellinie der Treppe ist: 0 - links, 1 - rechts*
- *[n][3] - Typ der Kante beginnend vom Knoten n. Mögliche Werte: 0 - Vorderkante, 1 - Hinterkante, 2 - linke Kante, 3 - rechte Kante, 4 - Mittellinienkante, -1 - Endkante.*

`values:` *Indikatoren der Knotenkategorien für [n] [1]*

`-1:` *Kein Verbindungs-Polygon (Ganze Trittstufe) : Im Falle eines Podestes befinden sich alle Punkte in dieser Kategorie. Kein solcher Wert bei Laufpolygonen.*

`0:` *Kein Verbindungs-Polygon (Kern): Im Falle eines Podestes können Start-End-Verbindungen die Geometrie beeinflussen. Im Falle eines Laufes ist das Polygon immer komplett.*

`1:` *Start Verbindungs-Polygon: zeigt das übrig gebliebene Polygon oder das eingeschlossene Polygon zwischen Kern- und Verbindungspolygonen an. Knoten, die für die Definition des Holm-Pfades notwendig sind: Startpunkt oder Endpunkt mit Grenz-Polygonpunkten.*

`2:` *Position Start Verbindungs-Polygon: Zeigt die Position der Verbindung an. Dieses Polygon ist immer ein generisches Einheits-Polygon, nicht Teil des Holm-Pfades und nur für die Schnittebene und die Endplattenposition notwendig.*

`3:` *Start Verbindung Bruch-Polygon: Zeigt das Polygon an, das die Bruchpunkte enthält, im Falle einer vertikalen oder horizontalen Verbindung. Punkte, die für den Holm-Pfad notwendig sind: Startpunkt oder Endpunkt mit Grenz-Polygonpunkten abgeschlossen.*

`4:` *Start Verbindung Richtungs-Polygon: Zeigt das Polygon an, das benötigt wird, um den Gehrungsschnitt im Falle einer Gehrungsverbindung durchzuführen. Dieses Polygon ist immer ein generisches Einheits-Polygon, nicht Teil des Holm-Pfades und nur für die Schnittebene notwendig ist.*

`5:` *Start Verbindung Erweiterungs-Polygon: Zeigt die Punkte der Holmausdehnung an, zur Schnittdurchführung. Punkte, die für den Holm-Pfad notwendig sind: Startpunkt oder Endpunkt mit Grenz-Polygonpunkten abgeschlossen.*

`6:` *Start Verbindung Strukturbegrenzungs-Polygon: Zeigt die Grenzpunkte des Holms beim Start an. Dieses Polygon ist immer ein generisches Einheits-Polygon. Punkte, die für den Holm-Pfad notwendig sind: Startpunkt oder Endpunkt.*

`101:` *Ende Verbindungs-Polygon: zeigt das übrig gebliebene Polygon oder das eingeschlossene Polygon zwischen Kern- und Verbindungspolygonen an. Knoten, die für die Definition des Holm-Pfades notwendig sind: Startpunkt oder Endpunkt mit Grenz-Polygonpunkten.*

`102:` *Ende Verbindung Positions-Polygon: Zeigt die Position der Verbindung an. Dieses Polygon ist immer ein generisches Einheits-Polygon, nicht Teil des Holm-Pfades und nur für die Schnittebene und die Endplattenposition notwendig.*

`103:` *Ende Verbindung Bruch-Polygon: Zeigt das Polygon an, das die Bruchpunkte enthält, im Falle einer vertikalen oder horizontalen Verbindung. Punkte, die für den Holm-Pfad notwendig sind: Startpunkt oder Endpunkt mit Grenz-Polygonpunkten abgeschlossen.*

`104:` *Ende Verbindung Richtungs-Polygon: Zeigt das Polygon an, das benötigt wird, um den Gehrungsschnitt im Falle einer Gehrungsverbindung durchzuführen. Dieses Polygon ist immer ein generisches Einheits-Polygon, nicht Teil des Holm-Pfades und nur für die Schnittebene notwendig ist.*

`105:` *Ende Verbindung Erweiterungs-Polygon: Zeigt die Punkte der Holmausdehnung an, zur Schnittdurchführung. Punkte, die für den Holm-Pfad notwendig sind: Startpunkt oder Endpunkt mit Grenz-Polygonpunkten abgeschlossen.*

`106:` *Ende Verbindung Struktur-Grenz-Polygon: Zeigt die Holm-Begrenzung am Ende an. Dieses Polygon ist immer ein generisches Einheits-Polygon. Punkte, die für den Holm-Pfad notwendig sind: Startpunkt oder Endpunkt.*

STAIR_STRUCTURE_CONN_OFFSETS Arry mit 2 Dimensionen([2][6]). Enthält Daten von Verbindungsversatzpunkten.

2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	Standard	[0]

- *[1][1] - Start-Anschluss horizontaler Offset (dx)*
- *[1][2] - Start-Anschluss vertikaler Offset (dy)*
- *[1][3] - Start-Anschluss horizontaler Offset 2 (dx1)*
- *[1][4] - Start-Anschluss vertikaler Offset 2 (dy1)*
- *[1][5] - Start-Anschluss horizontaler Offset (cx)*
- *[1][6] - Start-Anschluss vertikaler Offset (cy)*
- *[2][1] - End-Anschluss horizontaler Offset (dx)*
- *[2][2] - End-Anschluss vertikaler Offset (dy)*
- *[2][3] - End-Anschluss horizontaler Offset 2 (dx1)*
- *[2][4] - End-Anschluss vertikaler Offset 2 (dy1)*
- *[2][5] - End-Anschluss horizontaler Offset (cx)*
- *[2][6] - End-Anschluss vertikaler Offset (cy)*

STAIR_STRUCTURE_CONN_FLAGS Array mit 2 Dimensionen([2][3]). Enthält zusätzliche Daten der Strukturverbindung.

2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	Standard	0

- *[1][1] - Startverbindungstyp: 0 - Vertikalschnitt, 1 - Horizontaler Schnitt, 2 - Ausschnitt, 3 - senkrecht und waagerecht geschnitten, 4 - waagerechter Anschluss, 5 - horizontaler Anschluss und Ausschnitt, 6 - senkrechter Anschluss, 7 - Gehrung, 8 - automatisch.*
- *[1][2] - Start-Verbindungsrolle: 0 - Lauf und Start, 1 - Lauf und Podest, 2 - Podest und Lauf, 3 - Lauf und Lauf, 4 - Lauf und Ende, 5 - Podest und Podest.*
- *[1][3] - Auftritt oder Steigung am Start: 0 - Steigung am Start, 1 - Auftritt am Start.*
- *[2][1] - Endverbindungstyp: 0 - Vertikalschnitt, 1 - Horizontaler Schnitt, 2 - Ausschnitt, 3 - senkrecht und waagerecht geschnitten, 4 - waagerechter Anschluss, 5 - horizontaler Anschluss und Ausschnitt, 6 - senkrechter Anschluss, 7 - Gehrung, 8 - automatisch.*
- *[2][2] - End-Verbindungsrolle: 0 - Lauf und Start, 1 - Lauf und Podest, 2 - Podest und Lauf, 3 - Lauf und Lauf, 4 - Lauf und Ende, 5 - Podest und Podest.*
- *[2][3] - Auftritt oder Steigung am Ende: 0 - Steigung am Ende, 1 - Auftritt am Ende.*

STAIR_STRINGER_PATH_OFFSET Enthält den Wert "Höhe über Trittstufen" im Dialog Treppeneinstellungen.

2D	⛔	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

Geländerkomponentenparameter

Allgemeine Geländervariablen - für Auswertungen und Etiketten verfügbar

Kompatibilität: eingeführt in ARCHICAD 21.

RAILING_HEIGHT				Höhe des Geländersegments (wie im Dialogfeld Geländereinstellungen / Segmenteinstellungen eingestellt)							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
RAILING_3DLENGTH				Volle 3D Länge des Geländers							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
RAILING_HORIZONTAL_LENGTH				Volle projizierte 2D Länge des Geländers							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
RAILING_VOLUME				Volumen des Geländers (einschließlich aller Unterelemente)							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
RAILING_NR_OF_SEGMENTS				Anzahl der Segmente des Geländers							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
RAILING_NR_OF_POSTS				Anzahl der Pfosten im Geländer							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
RAILING_NR_OF_BALUSTERS				Anzahl der Geländerstäbe im Geländer							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
RAILING_NR_OF_PANELS				Anzahl der Paneele im Geländer							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0
RAILING_NR_OF_RAILS				Anzahl der Gurte im Geländer							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

Geländer 3D-Variablen

Kompatibilität: eingeführt in ARCHICAD 21.

RAILING_REFLINE_DISTANCE Array mit einer Dimension ([2]), horizontale Versätze des Geländerelementes von der Geländerreferenzlinie.

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[1] - Versatz der Segmentreferenzlinie von der Geländerreferenzlinie*
- *[2] - Versatz des Paneelelements von der Segmentreferenzlinie (immer 0 bei Innenpfosten, Pfosten oder Handlauf)*

RAIL_CONNECTING_POSTS_NUM Anzahl der Pfosten und inneren Pfosten, welche den Geländerverlauf schneiden.

2D	⛔	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

RAIL_CONNECTING_POSTS Array mit zwei Dimensionen ([n] [2]), wobei n die Anzahl der Pfosten oder inneren Pfosten entlang dem Geländerverlauf verläuft. Enthält Positionsdaten dieser sich überschneidenden Elemente.

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Position der Pfosten und inneren Pfosten entlang dem Geländerverlauf. Positionswerte werden proportional zur Geländerlänge berechnet (Werte liegen zwischen 0-1)*
- *[n][2] - Verdrehungswert des Pfosten- oder Innenpfostens (ähnlich wie beim Paneel). Eckelementwerte können sich von denen der einzelnen Elementen unterscheiden.*

RAIL_TYPE enthält den Unterelementtyp, für den das aktuelle Bibliothekselement ausgewählt ist.

2D	⛔	3D	✓	UI	✓	Parameter	✓	Property	⛔	Standard	0

- *1 - Handlauf (oder Handlaufende)*
- *2 - Zweite Handläufe (oder Enden der zweiten Handläufe)*
- *3 - Gurt (oder Gurtende)*

RAIL_POLYLINE_GEOMETRY Array mit zwei Dimensionen ([n] [5]), wobei n die Anzahl der Geländerknoten ist (n > 2). Enthält geometrische Daten des aktuellen Geländers, alle Segmente.

2D	⛔	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Koordinate x des Geländerknotens*
- *[n][2] - Koordinate y des Geländerknotens*
- *[n][3] - Koordinate z des Geländerknotens*
- *[n][4] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*
- *[n][5] - Drehung des Querschnitts für TUBE-Befehl. Die Drehung der automatisch verlängerten, verdrillten Geländerverbindungen kann von der Schrägstellung abweichen.*

RAIL_SEGMENT_FLAGS Array mit einer Dimension ([n]), wobei n die Anzahl der Geländerknoten ist (n > 2). Enthält geometrische Daten des aktuellen Geländers, alle Segmente.

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

Identifiziert das Segment, zu welchem der Knoten nach Index gehört, gemäß RAIL POLYLINE GEOMETRY. Knoten im selben Segment gehören zusammen.

RAIL_CUTS Array mit zwei Dimensionen ([2] [4]), definiert die Ausrichtung der Stirnflächen des Geländers

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

Die zurückgegebenen Koordinaten können im TUBE-Befehl für erste und letzte Pfadkoordinaten verwendet werden wie sie sind. Elemente von [1] [n] gehören zum Anfangsvertex, während [2] [n] die Koordinaten des Endvertex setzt.

- *[1][1] - Koordinate x*
- *[1][2] - Koordinate y*
- *[1][3] - Koordinate z*
- *[1][4] - Boolean, Schneidemethode: 0 - durchgehende Kanten, 1 - Gehrungs-Schnitt*

RAIL_DISCONNECTED_CUTS Array mit zwei Dimensionen ([2][4]), definiert die Ausrichtung der Endoberflächen im Falle einer losgelösten Verbindung.

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

Die zurückgegebenen Koordinaten können im TUBE-Befehl für erste und letzte Pfadkoordinaten verwendet werden wie sie sind. Elemente von [1] [n] gehören zum Endknoten des ersten Segments, während [2] [n] die Koordinaten am Startknoten des zweiten Segments festlegt.

- *[1][1] - Koordinate x*
- *[1][2] - Koordinate y*
- *[1][3] - Koordinate z*
- *[1][4] - 0 (reserviert für zukünftige Entwicklungen)*

Kompatibilität: eingeführt in ARCHICAD 22.

RAIL_COMPONENTS Array mit einer Dimension ([3]), enthält zusätzliche Informationen über Längselemente.

2D	✅	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

Boolescher Typ, um die Notwendigkeit / Anwesenheit von Befestigungen, Kappen anzuzeigen.

- *[1] - Vorhandensein von Befestigungen erforderlich (keine für Verbindungen oder Erweiterungen)*
- *[2] - Vorhandensein von Startkappen erforderlich (Start des Längselements oder an ein anderes angeschlossen))*
- *[3] - Vorhandensein von Verschlusskappen erforderlich (Ende des Längselementes oder verbunden mit einem anderen)*

RAIL_SLANT_ANGLE Neigungswinkel des Längselements in Grad relativ zur senkrechten Richtung, senkrecht zur Laufrichtung.

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

`values:` *mögliche Werte (< 90 Grad)*

`positive:` *Neigung nach links in Richtung Gehrichtung*

`negative:` *Neigung nach rechts in Richtung Gehrichtung*

Gekrümmte Abschnitte: Der Winkel wird auf einer vertikalen, radialen Ebene gemessen.

RAILINGPANEL_TYPE Generischer Geometrietyp des Paneels.

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

- *1 - planar*
- *2 - Zylindrisch (gekrümmt - vertikal oder gekrümmt - verzogen)*
- *3 - Konisch (gekrümmt - horizontal - geneigt)*
- *4 - Verdreht (gekrümmt - geneigt - schräg oder gekrümmt - schräg - verzogen)*

RAILINGPANEL_UNCUT_GEOMETRY Array mit zwei Dimensionen ([n] [4]), wobei n die Anzahl der Paneel-Knoten ist (n> 3). Enthält geometrische Daten des aktuellen Geländerpaneels, komplette Rohgeometrie (ohne Schnitte).

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Koordinate x des Geländerpaneelknotens*
- *[n][2] - Koordinate y des Geländerpaneelknotens*
- *[n][3] - Koordinate z des Geländerpaneelknotens*
- *[n][4] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

RAILINGPANEL_GEOMETRY Array mit zwei Dimensionen ([n] [5]), wobei n die Anzahl der Paneel-Knoten ist (n> 3). Enthält geometrische Daten des aktuellen Geländerpaneels, Schneideebenen angewendet.

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Koordinate x des Geländerpaneelknotens*
- *[n][2] - Koordinate y des Geländerpaneelknotens*
- *[n][3] - Koordinate z des Geländerpaneelknotens*
- *[n][4] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*
- *[n][5] - Im Falle einer Schnittkante, enthält den Schnittebenenwinkel (0 - senkrecht zu Paneel). Im Falle eines gekrümmten Paneels ist die Bezugsebene tangential.*

RAILINGPANEL_FLAGS

Array mit zwei Dimensionen ([n] [3]), wobei n die Anzahl der Paneel-Knoten (n> 3) gemäß RAILINGPANEL_GEOMETRY ist. Enthält geometrische Daten der aktuellen Geländerpaneel-Kanten, Schneideebenen angewendet.

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Segment-Anzahl der Kante beginnend vom Knoten im ungeschnittenen Paneel*
- *[n][2] - Position der Kante ab Knoten im geschnittenen Paneel (Indizierung nach RAILINGPANEL_SIDE_OFFSETS)*
- *[n][3] - Status Bit der Kante*

`mask:` *[n][3] mögliche Werte*

`mask = j1 + 2*j2`*, hierbei kann j jeweils 0 oder 1 sein.*

`j1:` *Kante, beginnend vom Knoten, der mit einem anderen Paneel verbunden ist*

`j2:` *Kante, beginnend vom Knoten, ist beschnitten*

RAILINGPANEL_SIDE_OFFSETS

Array mit einer Dimension ([4]), enthält die Paneel-Offsets, wie sie im Dialog "Geländer-Einstellungen" eingestellt sind, Abstand der Nachbar-Element-Achsen in einer vertikalen Ebene, schräg und verkantet wird vernachlässigt.

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[1] - Offset des Bodens*
- *[2] -Seitenoffset des Endscheitels*
- *[3] - Offset des Deckels*
- *[4] - Seitenoffset des Startscheitels*

RAILINGPANEL_SLANT_ANGLE

Neigungswinkel in Grad, senkrecht zur Laufrichtung. Vertikal ist 0 Grad, positive Werte bedeuten linke Seite, negative Werte bedeuten rechte Seite der Gehrichtung (Winkel <90). Gebogene Segmente: gemessen auf der Ebene senkrecht zur Anfangs-Tangentialebene des Segments (vor der Verkantung).

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

RAILINGPANEL_SKEW_ANGLE

Verkantungswinkel in Grad, parallel zur Laufrichtung. Vertikal ist 0 Grad, positive Werte bedeuten Rückwärts-, negative Werte bedeuten Vorwärts-Verkantungen entsprechend der Laufrichtung (Winkel <90). Gebogene Segmente: Gemessen an der Start-Tangentialebene des Segments (vor dem Verkanten).

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

RAILINGPOST_TYPE Enthält den Unterelementtyp, für den das aktuelle Bibliothekselement ausgewählt ist.

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	Standard	0

- *1 - Pfosten*
- *2 - Innenpfosten*
- *3 - Geländerstab*

RAILINGPOST_TOP_COORD Array mit einer Dimension ([3]), Koordinaten der Oberkante des aktuellen Pfostens

2D	⛔	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0, 0, 0]

- *[1] - Koordinate x*
- *[2] - Koordinate y*
- *[3] - Koordinate z*

RAILINGPOST_CUTS Array mit zwei Dimensionen ([2] [3]), definiert die Ausrichtung der Endflächen des Pfostens

2D	⛔	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

Die zurückgegebenen Koordinaten können im TUBE-Befehl für erste und letzte Pfadkoordinaten verwendet werden wie sie sind. Elemente von [1] [n] gehören zum Anfangsvertex, während [2] [n] die Koordinaten des Endvertex setzt.

- *[1][1] - Koordinate x*
- *[1][2] - Koordinate y*
- *[1][3] - Koordinate z*

RAILINGPOST_SEGMENT_CUTS Array mit zwei Dimensionen ([2] [4]), enthält die Vektorparameter von zwei Ebenen, die verwendet werden, um innere Pfosten und Geländerstäbe an der Grenze des Segments zu schneiden, wenn das Geländer schräg steht

2D	⛔	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

Die Normalen der Ebene ist Vektor (A; B; C), Und D ist der Abstand vom Ursprung, gemessen in Richtung der Normalen. Das Modell sollte in Richtung der Normalen geschnitten werden. Im Falle aller Rückgabewerte mit 0 für eine der 2 Ebenen existiert die Ebene nicht (kein Schnitt).

- *[1][1] - A*
- *[1][2] - B*
- *[1][3] - C*
- *[1][4] - D*

RAILINGEND_DIRECTION_AND_ANGLE Array mit zwei Dimensionen ([n] [5]), Vektor-Daten, die vom Verbindungsgurt weg in eine tangentialen Richtung zeigen. Für zwei Enden eines geraden Gurtes sind diese Vektoren richtungsmäßig entgegengesetzt.

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Koordinate x des Vektors*
- *[n][2] - Koordinate y des Vektors*
- *[n][3] - Koordinate z des Vektors*
- *[n][4] - Drehung des Verbindungsgurtes um die Achse des Vektors, mit der Wirkung der Verdrehung. Das Gurtende sollte die gleiche Drehung modellieren, um sich nahtlos zu verbinden.*

2D-Variablen beim Geländer

Kompatibilität: eingeführt in ARCHICAD 21.

RAIL2D_FULL_POLYLINE_GEOM Array mit zwei Dimensionen ([n] [3]), wobei n die Anzahl der Punkte in der Endachse von Paneel / Schiene / Schiene Polylinie ist. Enthält geometrische Daten des Geländer-Paneels.

2D	✅	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Koordinate x des vom Geländer-Ursprung gemessenen Knotens*
- *[n][2] - Koordinate y des vom Geländer-Ursprung gemessenen Knotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

RAIL2D_FULL_POLYLINE_FLAGS Array mit zwei Dimensionen ([n] [1]), wobei n die Anzahl der Punkte in der Achsen-Polylinie von Paneel / Gurt/ Gurtende ist. Enthält Sichtbarkeitsdaten von Kanten.

2D	✅	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Sichtbarkeits-Flag der Kante beginnend am Knoten n (0 - nicht sichtbar, 1 - sichtbar)*

RAIL2D_FULL_POLYGON_GEOM Array mit zwei Dimensionen ([n] [3]), wobei n die Anzahl der Punkte im Symbol-Polygon von Paneel / Gurt/ Gurtende ist. Enthält geometrische Daten des Geländer-Paneels.

2D	✅	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Koordinate x des vom Geländer-Ursprung gemessenen Knotens*
- *[n][2] - Koordinate y des vom Geländer-Ursprung gemessenen Knotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

RAIL2D_FULL_POLYGON_FLAGS Array mit zwei Dimensionen ([n] [1]), wobei n die Anzahl der Punkte im Symbol-Polygon von Paneel / Gurt/ Gurt-Ende ist. Enthält Sichtbarkeitsdaten von Kanten.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Sichtbarkeits-Flag der Kante beginnend am Knoten n (0 - nicht sichtbar, 1 - sichtbar)*

RAIL2D_FULL_VISIBILITY Typ des aktiven Attributsatzes der aktuellen Geländer-Zeichnung

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

values:

1: 'Sichtbarer' Attributesatz ist aktiv
0: 'Unsichtbarer' Attributesatz ist aktiv

RAIL2D_CUSTOMDISPLAY enthält Informationen über die Modelldarstellung des Geländers

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

values:

0: Geländer wird entsprechend den Einstellungen der Modelldarstellung dargestellt
1: Geländer wird mit benutzerdefinierten Einstellungen dargestellt

Die folgenden Globals werden verwendet, um die unterhalb der ersten Bruchlinie dargestellten Teile des Geländers zu definieren.

RAIL2D_LOWER_POLYLINE_GEOM Array mit zwei Dimensionen ([n] [3]), wobei n die Anzahl der Punkte in der Achsen-Polylinie von Paneel- / Gurt- / Gurtende unterhalb der ersten Bruchlinie ist. Enthält geometrische Daten der Polylinie des Geländer-Paneels.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Koordinate x des vom Geländer-Ursprung gemessenen Knotens*
- *[n][2] - Koordinate y des vom Geländer-Ursprung gemessenen Knotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

RAIL2D_LOWER_POLYLINE_FLAGS Array mit zwei Dimensionen ([n] [1]), wobei n die Anzahl der Punkte in der Achsen-Polylinie von Paneel- / Gurt- / Gurtende unterhalb der ersten Bruchlinie ist. Enthält Sichtbarkeitsdaten von Kanten.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Sichtbarkeits-Flag der Kante beginnend am Knoten n (0 - nicht sichtbar, 1 - sichtbar)*

RAIL2D_LOWER_POLYGON_GEOM

Array mit zwei Dimensionen ([n] [3]), wobei n die Anzahl der Punkte des Symbol-Polygons von Paneel- / Gurt- / Gurtende unterhalb der ersten Bruchlinie ist. Enthält geometrische Daten Polygons des Geländer-Paneels.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Koordinate x des vom Geländer-Ursprung gemessenen Knotens*
- *[n][2] - Koordinate y des vom Geländer-Ursprung gemessenen Knotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

RAIL2D_LOWER_POLYGON_FLAGS

Array mit zwei Dimensionen ([n] [1]), wobei n die Anzahl der Punkte des Symbol-Polygons von Paneel- / Gurt- / Gurtende unterhalb der ersten Bruchlinie ist. Enthält Sichtbarkeitsdaten von Polygon-Kanten.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Sichtbarkeits-Flag der Kante beginnend am Knoten n (0 - nicht sichtbar, 1 - sichtbar)*

RAIL2D_LOWER_VISIBILITY

Typ des aktiven Attributsatzes der aktuellen Geländer-Zeichnung unterhalb der ersten Bruchlinie

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

values:

`1:` *'Sichtbarer' Attributesatz ist aktiv*
`0:` *'Unsichtbarer' Attributesatz ist aktiv*

Die folgenden Globals werden verwendet, um die Teile des Geländers zu definieren, die zwischen zwei Bruchlinien dargestellt sind.

RAIL2D_MIDDLE_POLYLINE_GEOM

Array mit zwei Dimensionen ([n] [3]), wobei n die Anzahl der Punkte in der Achsen-Polylinie von Paneel / Gurt/ Gurtende zwischen den Bruchlinien ist. Enthält geometrische Daten der Polylinie des Geländer-Paneels.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Koordinate x des vom Geländer-Ursprung gemessenen Knotens*
- *[n][2] - Koordinate y des vom Geländer-Ursprung gemessenen Knotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

RAIL2D_MIDDLE_POLYLINE_FLAGS Array mit zwei Dimensionen ([n] [1]), wobei n die Anzahl der Punkte in der Achsen-Polylinie von Paneel / Gurt/ Gurtende zwischen den Bruchlinien ist. Enthält Sichtbarkeitsdaten von Kanten.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Sichtbarkeits-Flag der Kante beginnend am Knoten n (0 - nicht sichtbar, 1 - sichtbar)*

RAIL2D_MIDDLE_POLYGON_GEOM Array mit zwei Dimensionen ([n] [3]), wobei n die Anzahl der Punkte im Symbol-Polygon von Paneel / Gurt/ Gurtende zwischen den Bruchlinien ist. Enthält geometrische Daten Polygons des Geländer-Paneels.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Koordinate x des vom Geländer-Ursprung gemessenen Knotens*
- *[n][2] - Koordinate y des vom Geländer-Ursprung gemessenen Knotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

RAIL2D_MIDDLE_POLYGON_FLAGS Array mit zwei Dimensionen ([n] [1]), wobei n die Anzahl der Punkte im Symbol-Polygon von Paneel / Gurt/ Gurte-Ende zwischen den Bruchlinien ist. Enthält Sichtbarkeitsdaten von Polygon-Kanten.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Sichtbarkeits-Flag der Kante beginnend am Knoten n (0 - nicht sichtbar, 1 - sichtbar)*

RAIL2D_MIDDLE_VISIBILITY Typ des aktiven Attributsatzes der aktuellen Geländer-Zeichnung zwischen den Bruchlinien

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

values:

1: 'Sichtbarer' Attributesatz ist aktiv
0: 'Unsichtbarer' Attributesatz ist aktiv

Die folgenden Globals werden verwendet, um die unter der letzten Bruchlinie dargestellten Teile des Geländers zu definieren.

RAIL2D_UPPER_POLYLINE_GEOM

Array mit zwei Dimensionen ([n] [3]), wobei n die Anzahl der Punkte in der Achsen-Polylinie von Paneel- / Gurt- / Gurtende oberhalb der letzten Bruchlinie ist. Enthält geometrische Daten der Polylinie des Geländer-Paneels.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Koordinate x des vom Geländer-Ursprung gemessenen Knotens*
- *[n][2] - Koordinate y des vom Geländer-Ursprung gemessenen Knotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

RAIL2D_UPPER_POLYLINE_FLAGS

Array mit zwei Dimensionen ([n] [1]), wobei n die Anzahl der Punkte in der Achsen-Polylinie von Paneel- / Gurt- / Gurtende oberhalb der letzten Bruchlinie ist. Enthält Sichtbarkeitsdaten von Kanten.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Sichtbarkeits-Flag der Kante beginnend am Knoten n (0 - nicht sichtbar, 1 - sichtbar)*

RAIL2D_UPPER_POLYGON_GEOM

Array mit zwei Dimensionen ([n] [3]), wobei n die Anzahl der Punkte des Symbol-Polygons von Paneel- / Gurt- / Gurtende oberhalb der letzten Bruchlinie ist. Enthält geometrische Daten Polygons des Geländer-Paneels.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Koordinate x des vom Geländer-Ursprung gemessenen Knotens*
- *[n][2] - Koordinate y des vom Geländer-Ursprung gemessenen Knotens*
- *[n][3] - Zentralwinkel der Kante beginnend vom Knoten (0 - gerade, > 0 - gegen den Uhrzeigersinn gekrümmt, < 0 - im Uhrzeigersinn gekrümmt)*

RAIL2D_UPPER_POLYGON_FLAGS

Array mit zwei Dimensionen ([n] [1]), wobei n die Anzahl der Punkte des Symbol-Polygons von Paneel- / Gurt- / Gurtende oberhalb der letzten Bruchlinie ist. Enthält Sichtbarkeitsdaten von Polygon-Kanten.

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0]

- *[n][1] - Sichtbarkeits-Flag der Kante beginnend am Knoten n (0 - nicht sichtbar, 1 - sichtbar)*

RAIL2D_UPPER_VISIBILITY											Typ des aktiven Attributsatzes der aktuellen Geländer-Zeichnung oberhalb der ersten Bruchlinie
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

values:

1: 'Sichtbarer' Attributesatz ist aktiv

0: 'Unsichtbarer' Attributesatz ist aktiv

Die folgende Globale gehört zur 2D-Darstellung der Geländerpfosten.

RAILPOST2D_VISIBILITY											Typ des aktiven Attributsatzes der aktuellen Pfosten-Zeichnung
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	Standard	0

values:

1: 'Sichtbarer' Attributesatz ist aktiv

0: 'Unsichtbarer' Attributesatz ist aktiv

Dachparameter - verfügbar für Dachfenster, Listen und Etiketten

ROOF_THICKNESS	Dicke des Daches
ROOF_ANGLE	Neigung des Daches
ROOF_MAT_TOP	Oberflächen-Attribut-Index des Dachs auf der Oberseite
ROOF_MAT_EDGE	Oberflächen-Attribut-Index der Dachkanten
ROOF_MAT_BOTT	Oberflächen-Attribut-Index des Dachs auf der Unterseite
ROOF_LINETYPE	Linientyp des Daches
wird nur für die Konturen im Grundrissfenster verwendet	
ROOF_FILL	Schraffur des Daches
Schraffurindex - sein Wert ist negativ im Falle eines mehrschichtigen Aufbaus	
ROOF_FILL_PEN	Stift der Schraffur des Daches
ROOF_FBGD_PEN	Stift des Schraffurhintergrundes des Daches
ROOF_COMPS_NAME	Name des Mehrschichtbauteils des Daches
ROOF_BMAT_NAME	Name des Baustoffs des Daches, Leerstring bei Mehrschichtdächern

ROOF_BMAT Baustoff-Index des Daches, 0 für Mehrschichtdächer

Kompatibilität: eingeführt in ARCHICAD 21.

ROOF_SKINS_NUMBER Anzahl der Schichten des zusammengesetzten Daches

liegt zwischen 1 und 8; ist 0, falls eine einzige Schraffur verwendet wird

ROOF_SKINS_PARAMS Parameter der Schichten des zusammengesetzten Daches

Array mit 18 Spalten und einer beliebigen Anzahl von Zeilen:

- *[1] Schraffur*
- *[2] Stärke*
- *[3] (alter Konturstift)*
- *[4] Schraffurstift*
- *[5] Stift der Hintergrundschraffur*
- *[6] Kern/Bekleidungsstatus*
- *[7] oberer Linienstift*
- *[8] oberer Linientyp*
- *[9] unterer Linienstift*
- *[10] unterer Linientyp*
- *[11] Stirnflächenstift*
- *[12] Schraffurausrichtung*
- *[13] Oberflächentyp*
- *[14] Linientyp Stirnfläche*
- *[15] fertiger Oberflächenstatus*
- *[16] Status Schraffurausrichtung*
- *[17] Status einer Kernschicht (falls keine Kernschicht existiert: die stärkste Schicht)*
- *[18] Baustoff-Index.*

Kern: 0 - nein, 1 - ja, 3 - letzte Schicht, Richtung der Schraffur: 0 - global, 1 - lokal; Schichtentyp: in aktuellem ARCHICAD immer 0 - Schnitt, kann verwendet werden wie in Wänden später; Fertiger Schichtstatus: 0: Schicht nicht beenden, 1: Beenden Schicht

ROOF_SKINS_BMAT_NAMES Name des Baustoffs von Schichten mehrschichtiger Dächer

Array mit 1 Spalten: Name des Baustoffs der Schicht mit beliebiger Zeilenanzahl

ROOF_SECT_PEN Stift der Dachkontur-Schnittoberflächen

verwendet für die Konturen der geschnittenen Flächen im Grundriss-Fenster und im Schnitte/ Ansichten-Fenster

ROOF_VIEW_PEN	Stift des Daches in der Ansicht
verwendet für alle Kanten im 3D-Fenster und für Umfassungskanten (Kanten von unterhalb der Schnittebene betrachtet) im Grundriss, sowie für alle Kanten im Schnitte-/Ansichten-Fenster	

Parameter von Dächern - nur zum Auflisten verfügbar

ROOF_BOTTOM_SURF	Oberflächenbereich an der Dachunterseite
nicht um den Oberflächenbereich von Öffnungen reduziert, welche größer als der vorgegebene Wert sind	
ROOF_GROSS_BOTTOM_SURF	Brutto-Oberflächenbereich der Dachunterseite
um den Oberflächenbereich von Öffnungen reduziert	
ROOF_BOTTOM_SURF_CON	Konditionaler Oberflächenbereich der Dachunterseite
um den Oberflächenbereich von Öffnungen reduziert, welche größer als der vorgegebene Wert sind	
ROOF_TOP_SURF	Oberflächenbereich an der Dachoberseite
nicht um den Oberflächenbereich von Öffnungen reduziert, welche größer als der vorgegebene Wert sind	
ROOF_GROSS_TOP_SURF	Brutto-Oberflächenbereich der Dachoberseite
um den Oberflächenbereich von Öffnungen reduziert	
ROOF_TOP_SURF_CON	konditionaler Oberflächenbereich des Dachs
um den Oberflächenbereich von Öffnungen reduziert, welche größer als der vorgegebene Wert sind	
ROOF_EDGE_SURF	Oberflächenbereich der Dachkante
nicht um den Oberflächenbereich von Öffnungen reduziert	
ROOF_GROSS_EDGE_SURF	Brutto-Oberflächenbereich der Dachkanten
um den Oberflächenbereich von Öffnungen reduziert	
ROOF_CONTOUR_AREA	vom Dach überdeckte Fläche
ROOF_PERIMETER	Umfang des Daches
ROOF_VOLUME	Volumen des Daches
nicht um das Volumen von Öffnungen reduziert	

ROOF_GROSS_VOLUME *um das Volumen von Öffnungen reduziert*	Brutto-Volumen des Daches
ROOF_VOLUME_CON *um die Oberfläche von Öffnungen reduziert, welche größer als der vorgegebene Wert sind*	konditionales Volumen des Dachs
ROOF_SEGMENTS_NR	Anzahl der Segmente der Dachkante
ROOF_HOLES_NR	Anzahl der Durchbrüche im Dach
ROOF_HOLES_AREA	Fläche der Durchbrüche im Dach
ROOF_HOLES_PRM	Umfang der Durchbrüche im Dach
ROOF_INSU_THICKNESS	Dicke der Dach-Dämmung
ROOF_RIDGE	Dachfirst-Länge
ROOF_VALLEY	Dachkehlen-Länge
ROOF_GABLE	Dach-Giebel-Länge
ROOF_HIP	Dach-Walm-Länge
ROOF_EAVES	Dach-Traufen-Länge
ROOF_PEAK	Dach-Spitzen-Länge
ROOF_SIDE_WALL	Dach-Seitenwand Verbindungslänge
ROOF_END_WALL	Dachende-Wand-Verbindungslänge
ROOF_TRANSITION_DOME	Dachkuppel-Verbindungslänge
ROOF_TRANSITION_HOLLOW	Dachhöhlung-Verbindungslänge

Schraffurparameter - nur zum Auflisten verfügbar

FILL_LINETYPE	Linientyp der Schraffurkante
FILL_FILL	Schraffurtyp der Schraffur
FILL_BMAT_NAME	Name des Baustoffs einer Schraffur

FILL_BMAT	Baustoff-Index der Schraffur
Kompatibilität: eingeführt in ARCHICAD 21.	
FILL_FILL_PEN	Stift des Schraffurtyps der Schraffur
FILL_PEN	Stift der Schraffur
FILL_FBGD_PEN	Hintergrundstift der Schraffur
FILL_SURF	Fläche der Schraffur
FILL_PERIMETER	Umfang der Schraffur
FILL_SEGMENT_NR	Anzahl der Segmente des Schraffurrandes
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
FILL_FILL_CATEGORY	Schraffurkategorie der Schraffur
0 - Zeichnungsschraffur, 1 - Bauteilschraffur, 2 - Deckschraffur	

Parameter der Freiflächenform - nur zum Auflisten verfügbar

MESH_TYPE	Typ der Freiflächenform
1- geschlossener Körper 2 - Decke +Kante, 3 - nur Deckfläche	
MESH_BASE_OFFSET	Versatz der Bodenfläche zum Basisniveau
MESH_USEREDGE_PEN	Stift der benutzerdefinierten Firste der Freiflächenform
MESH_TRIEDGE_PEN	Stift der durch Triangulation gekrümmten Kanten der Freiflächenform
MESH_SECT_PEN	Stift der Konturen der Freiflächenform im Schnitt
Wird auf Konturen der geschnittenen Wandoberflächen sowohl im Grundriss-Fenster als auch im Schnitte/Ansichten-Fenster verwendet	
MESH_VIEW_PEN	Stift der Konturen in der Ansicht
verwendet für alle Kanten im 3D-Fenster, alle sichtbaren Kanten in den Schnitte/Ansichten-Fenstern und der Standardwert im Grundriss.	
MESH_MAT_TOP	Oberflächen-Attribut-Index der Freifläche auf der Oberseite

MESH_MAT_EDGE	Oberflächen-Attribut-Index der Freifläche an den Kanten
MESH_MAT_BOTT	Oberflächen-Attribut-Index der Freifläche an der Unterseite
MESH_LINETYPE *wird nur für die Konturen im Grundrissfenster verwendet*	Linientyp der Freiflächenkanten
MESH_FILL	Schraffurtyp der Freifläche
MESH_BMAT_NAME	Name des Baustoffs einer Freifläche
MESH_BMAT *Kompatibilität: eingeführt in ARCHICAD 21.*	Baustoff-Index der Freifläche
MESH_FILL_PEN	Stift der Schraffur der Freifläche
MESH_FBGD_PEN	Stift des Schraffurhintergrund der Freifläche
MESH_BOTTOM_SURF	Oberflächenbereich der Unterseite der Freifläche
MESH_TOP_SURF	Oberflächenbereich der Oberseite der Freifläche
MESH_EDGE_SURF	Oberflächenbereich der Seitenfläche der Freifläche
MESH_PERIMETER	Umfang der Freifläche
MESH_VOLUME	Volumen der Freifläche
MESH_SEGMENTS_NR	Anzahl von Segmente der Freiflächenkante
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

Komponenten-Parameter des Fassadenwerkzeugs

CW_BOUNDARY_PLACEMENT Platzierung des Fassadenrahmens

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	✓	Standard	0

Kompatibilität: eingeführt in ARCHICAD 22.

Enthält die the Platzierungseinstellungen der Begrenzungsrahmen, die im Paneel Fassadensystem / Element-Platzierung definiert sind.

`values:`

`0:` *Mitte auf Umrandung oder der aktuelle Rahmen ist kein Begrenzungsrahmen.*
`-1:` *Innerhalb der Umrandung*
`1:` *Außerhalb der Umrandung*

GLOB_MVO_CWFRAME_DETLEVEL Detaillierungsgrad von Fassaden-Profilen, die in Modelldarstellung / Fassadenoptionen festgelegt wurden.

2D	✓	3D	✓	UI	⚠	Parameter	⛔	Property	⛔	Standard	4

Kompatibilität: eingeführt in ARCHICAD 22.

`values:`

`1:` *Nur Achse*
`2:` *Schematisch*
`3:` *Vereinfacht*
`4:` *Komplett*

GLOB_MVO_CWPANEL_DETLEVEL Detaillierungsgrad von Fassaden-Paneelen, die in Modelldarstellung / Fassadenoptionen festgelegt wurden..

2D	✓	3D	✓	UI	⚠	Parameter	⛔	Property	⛔	Standard	4

Kompatibilität: eingeführt in ARCHICAD 22.

`values:`

`2:` *Schematisch*
`3:` *Vereinfacht*
`4:` *Komplett*

GLOB_MVO_CWJUNCT_DETLEVEL Detaillierungsgrad von Fassadenhalterungen, die in Modelldarstellung / Fassadenoptionen festgelegt wurden.

2D	✓	3D	✓	UI	⚠	Parameter	⛔	Property	⛔	Standard	4

Kompatibilität: eingeführt in ARCHICAD 22.

`values:`

`2:` *Schematisch*
`3:` *Vereinfacht*
`4:` *Komplett*

GLOB_MVO_CWACC_DETLEVEL Detaillierungsgrad von Fassadenzubehör, die in Modelldarstellung / Fassadenoptionen festgelegt wurden..

2D	✓	3D	✓	UI	⚠	Parameter	⛔	Property	⛔	Standard	4

Kompatibilität: eingeführt in ARCHICAD 22.

`values:`

`2:` *Schematisch*
`3:` *Vereinfacht*
`4:` *Komplett*

Parameter von Fassaden - nur zum Auflisten und für Etiketten verfügbar

CWALL_ID	ID der Fassade
CWALL_FRAMES_LENGTH	Länge der Profile in der Fassade
CWALL_CONTOUR_FRAMES_LENGTH	Länge der Rand-Profile in der Fassade
CWALL_MAINAXIS_FRAMES_LENGTH	Länge der Pfosten-Profile in der Fassade
CWALL_SECAXIS_FRAMES_LENGTH	Länge der Riegel-Profile in der Fassade
CWALL_CUSTOM_FRAMES_LENGTH	Länge eigener Profile in der Fassade
CWALL_PANELS_SURF	Oberflächenbereich der Paneele in der Fassade
CWALL_PANELS_SURF_N	Oberflächenbereich der Nordpaneele in der Fassade
CWALL_PANELS_SURF_S	Oberflächenbereich der Südpaneele in der Fassade
CWALL_PANELS_SURF_E	Oberflächenbereich der Ostpaneele in der Fassade

CWALL_PANELS_SURF_W	Oberflächenbereich der Westpaneele in der Fassade
CWALL_PANELS_SURF_NE	Oberflächenbereich der Nordostpaneele in der Fassade
CWALL_PANELS_SURF_NW	Oberflächenbereich der Nordwestpaneele in der Fassade
CWALL_PANELS_SURF_SE	Oberflächenbereich der Südostpaneele in der Fassade
CWALL_PANELS_SURF_SW	Oberflächenbereich der Südwestpaneele in der Fassade
CWALL_SURF	Oberflächenbereich der Fassade
CWALL_SURF_BOUNDARY	Oberflächenbereich der Fassade umrandet vom Rand-Profil
CWALL_LENGTH	Länge der Fassade
CWALL_HEIGHT	Höhe der Fassade
CWALL_SLANT_ANGLE	Neigungswinkel der Fassade
CWALL_THICKNESS	Stärke der Fassade
CWALL_PANELS_NR	Anzahl der Paneele in der Fassade
CWALL_PATTERN_ANGLE	Drehwinkel der Riegel in der Fassade

Profilparameter der Fassade

Allgemeine Profilvariablen der Fassade - nur für Listen und Etiketten verfügbar

CWFRAME_TYPE	Typ des Profils
'Unsichtbar' oder Name des GDL-Objektes *Kompatibilität: bis ARCHICAD 21 'Generisch', 'Bündig Verglast', 'Unsichtbar'' oder Name des GDL-Objektes*	
CWFRAME_POSITION	Position des Profils. Ab ARCHICAD 22 basiert die Definition der Profilposition auf den relativen Koordinatenwerten der Profile.
0 - vertikal, 1 - horizontal, 2 - Kontur, 3 - diagonal *Kompatibilität: bis ARCHICAD 21 basierte die Lage der Profile auf den Rasterlinien. Die Werte beziehen sich auf: 0 - Primäre Rasterlinie, 1 - Sekundäre Rasterlinie, 2 - Begrenzungskante, 3 - andere*	
CWFRAME_DIRECTION	Neigungswinkel des Profils
Grad zwischen 0 und 90	

CWFRAME_WIDTH	Breite des Profils
CWFRAME_DEPTH	Tiefe des Profils
CWFRAME_LENGTH	Länge des Profils
CWFRAME_MAT	Oberflächen-Attribut-Index des Rahmenprofils

3D-Variablen von Fassadenprofilen

CWFRAME_TOP_CUTTYPE Definiert die Schnittmethode an der Oberkante der Verbindung der Fassdenprofile.

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

Kompatibilität: eingeführt in ARCHICAD 22.

`values:`

`0:` *Ebene - Das Profil sollte von einer Ebene geschnitten werden, die in CWFRAME_TOP_CUTPLANE global definiert ist*

`1:` *Polylinie - Das Profil sollte an einer Polylinie entlang geschnitten werden, die in CWFRAME_TOP_CUTPOLYLINE global definiert ist*

CWFRAME_TOP_CUTPLANE Array mit einer Dimension ([4]), enthält die obere Position des Cutplanes des Profils, definiert im lokalen Koordinatensystem des Bibliothekselementes.

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0, 0, 0, 0]

Kompatibilität: eingeführt in ARCHICAD 22.

`values:`

`[1]:` *X Komponente des Normalvektors des oberen Cutplanes*

`[2]:` *Y Komponente des Normalvektors des oberen Cutplanes*

`[3]:` *Z Komponente des Normalvektors des oberen Cutplanes*

`[4]:` *Abstand des oberen Cutplanes vom Ursprung des Profils*

CWFRAME_TOP_CUTPOLYLINE

Array mit zwei Dimensionen ([n][2]), wobei n die Knotenanzahl der beschneidenden Polylinie ist. Enthält die beschneidenden Polylinienkoordinaten für die obere Verbindung des Profils, die in der Y-Z-Ebene des lokalen Koordinatensystems des Bibliothekselementes definiert sind. Der erste Knoten der Polylinie befindet sich auf dem minimalen Y-Koordinatenwert, der letzte Knoten auf dem maximalen Y-Koordinatenwert des Fassadenprofils.

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0, 0]

Kompatibilität: eingeführt in ARCHICAD 22.

values:

`[n][1]:` *Koordinate X des Knotens gemessen vom Ursprung des Fassadenprofils*

`[n][2]:` *Koordinate Y des Knotens gemessen vom Ursprung des Fassadenprofils*

CWFRAME_BOTTOM_CUTTYPE

Definiert die Beschneidungsmethode an der Verbindung der Profile am Fußpunkt der Fassade.

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	0

Kompatibilität: eingeführt in ARCHICAD 22.

values:

`0:` *Ebene - Das Profil sollte von einer Ebene geschnitten werden, die in CWFRAME_BOTTOM_CUTPLANE global definiert ist*

`1:` *Polylinie - Das Profil sollte an einer Polylinie entlang geschnitten werden, die in CWFRAME_BOTTOM_CUTPOLYLINE global definiert ist*

CWFRAME_BOTTOM_CUTPLANE

Array mit einer Dimension ([4]), enthält die untere Position des Cutplanes des Profils, definiert im lokalen Koordinatensystem des Bibliothekselementes

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0, 0, 0, 0]

Kompatibilität: eingeführt in ARCHICAD 22.

values:

`[1]:` *X Komponente des Normalvektors des unteren Cutplanes*

`[2]:` *Y Komponente des Normalvektors des unteren Cutplanes*

`[3]:` *Z Komponente des Normalvektors des unteren Cutplanes*

`[4]:` *Abstand des unteren Cutplanes vom Ursprung des Profils*

CWFRAME_BOTTOM_CUTPOLYLINE Array mit zwei Dimensionen ([n][2]), wobei n die Knotenanzahl der beschneidenden Polylinie ist. Enthält die beschneidenden Polylinienkoordinaten für die untere Verbindung des Profils, die in der Y-Z-Ebene des lokalen Koordinatensystems des Bibliothekselementes definiert sind. Der erste Knoten der Polylinie befindet sich auf dem minimalen Y-Koordinatenwert, der letzte Knoten auf dem maximalen Y-Koordinatenwert des Fassadenprofils.

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[0, 0]

Kompatibilität: eingeführt in ARCHICAD 22.

`values:`

`[n][1]:` *Koordinate X des Knotens gemessen vom Ursprung des Fassadenprofils*

`[n][2]:` *Koordinate Y des Knotens gemessen vom Ursprung des Fassadenprofils*

Paneelvariablen der Fassade

CWPANEL_HORIZONTAL_DIRECTION Winkel der äußeren Oberfläche des Paneels in Bezug zu Projekt Nord.

2D	✅	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[180]

Kompatibilität: eingeführt in ARCHICAD 22 für 2D- und 3D-Scripte. In früheren Versionen funktionierte das nur für Etiketten und Listen.

Werte: Wert zwischen -180 und 180

CWPANEL_VERTICAL_DIRECTION Neigungswinkel der äußeren Oberfläche des Paneels.

2D	✅	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	Standard	[90]

Kompatibilität: eingeführt in ARCHICAD 22 für 2D- und 3D-Scripte. In früheren Versionen funktionierte das nur für Etiketten und Listen.

Werte: Grad zwischen -90 und 90

Paneelparameter von Fassaden - nur zum Auflisten und für Etiketten verfügbar

CWPANEL_TYPE Typ des Paneels

"Allgemein" oder Name des GDL Objektes

CWPANEL_CLASS Klasse des Paneels

0 - haupt, 1 - neben, 2 - Sondertyp

CWPANEL_WIDTH Breite des Paneels

CWPANEL_NOMINAL_WIDTH Nominalbreite des Paneels

CWPANEL_HEIGHT	Höhe des Paneels
CWPANEL_NOMINAL_HEIGHT	Nominalhöhe des Paneels
CWPANEL_THICKNESS	Stärke des Paneels
CWPANEL_SURF	Oberflächenbereich des Paneels
CWPANEL_GROSS_SURF	Brutto-Oberflächenbereich des Paneels
CWPANEL_NOMINAL_SURF	Nominaler Oberflächenbereich des Paneels
CWPANEL_PERIMETER	Umfang des Paneels
CWPANEL_MAT_OUTER	Oberflächen-Attribut-Index der Außenoberfläche der Paneele
CWPANEL_MAT_INNER	Oberflächen-Attribut-Index der Innenoberfläche der Paneele
CWPANEL_MAT_CUT	Oberflächen-Attribut-Index der Kantenoberflächen der Paneele
CWPANEL_FUNCTION *0 - fixiert, 1 - Tür, 2 - Fenster*	Funktion des Paneels
CWPANEL_ORIENTATION *links/rechts*	Ausrichtung der Öffnung des Tür/Fenster-Paneels

Halterungsparameter von Fassaden - nur zum Auflisten und für Etiketten verfügbar

CWJUNC_TYPE *Name des GDL Objekts*	Typ der Halterung

Zubehör-Parameter von Fassaden - nur zum Auflisten und für Etiketten verfügbar

CWACC_TYPE *Name des GDL Objekts*	Typ des Zubehörs

Migrationsparameter - nur für die Migrations-Scripte verfügbar

FROM_GUID	Haupt-GUID des ursprünglich platzierten Bibliothekselements
TO_GUID	Haupt-GUID des Bibliothekselements, auf welches die Migration angewendet wird.

Dachfenster-Parameter- nur für Etiketten und Listen verfügbar

SKYL_MARKER_TXT	Dachfenster Markertext
SKYL_OPENING_SURF	Dachfenster Öffnungsoberfläche
SKYL_OPENING_VOLUME	Volumen der vom Dachfenster eingeschnittenen Öffnung
SKYL_OPENING_HEIGHT	Dachfenster: Öffnungshöhe
SKYL_OPENING_WIDTH	Dachfenster: Öffnungsbreite
SKYL_HEADER_HEIGHT	Dachfenster: Sturzhöhe
SKYL_SILL_HEIGHT	Dachfenster: Brüstungshöhe

Allgemeine Parameter von Schalen und Dachflächen - nur verfügbar für Listen und Etiketten

SHELLBASE_THICKNESS *gleichwertig zu ROOF_THICKNESS bei Dächern*	Dicke von Schale/Dach/Decke
SHELLBASE_MAT_REFERENCE *gleichwertig zu ROOF_MAT_BOTT bei Dächern*	Oberflächen-Attribut-Index des Dachs/der Schale auf der Unterseite
SHELLBASE_MAT_EDGE *gleichwertig zu ROOF_MAT_EDGE bei Dächern*	Oberflächen-Attribut-Index der Dachkanten/Schalenkanten
SHELLBASE_MAT_OPPOSITE *gleichwertig zu ROOF_MAT_TOP bei Dächern*	Oberflächen-Attribut-Index des Dachs/der Schale auf der Oberseite
SHELLBASE_LINETYPE *nur anwendbar auf die Konturen im Grundrissfenster gleichwertig zu ROOF_LINETYPE bei Dächern*	Linientyp von Schale/Dach
SHELLBASE_FILL *Schraffurindex- sein Wert ist negativ im Fall eines Mehrschichtbauteils, gleichwertig zu ROOF_FILL bei Dächern*	Schraffur von Schale/Dach
SHELLBASE_FILL_PEN *gleichwertig zu ROOF_FILL_PEN bei Dächern*	Schraffurstift bei Schale/Dach

SHELLBASE_FBGD_PEN	Schraffur-Hintergrundstift bei Schale/Dach
gleichwertig zu ROOF_FBGD_PEN bei Dächern	
SHELLBASE_COMPS_NAME	Name des Mehrschichtbauteils bei Schale/Dach
gleichwertig zu ROOF_COMPS_NAME bei Dächern	
SHELLBASE_BMAT_NAME	Name des Baustoffs einer Schale/eines Daches
gleichwertig zu ROOF_BMAT_NAME bei Dächern	
SHELLBASE_BMAT	Index des Baustoffs einer Schale/eines Daches
Kompatibilität: eingeführt in ARCHICAD 21. *gleichwertig zu ROOF_BMAT bei Dächern*	
SHELLBASE_SKINS_NUMBER	Anzahl der Schichtenzahl bei Schale/Dach
Bereich von 1 bis 8, 0 falls nur eine einzige Schraffur verwendet wird, gleichwertig zu ROOF_SKINS_NR bei Dächern	

SHELLBASE_SKINS_PARAMS Parameter des Mehrschichtbauteils der Dachhaut bei Schale/Dach

Array mit 18 Spalten und einer beliebigen Anzahl von Zeilen:

- *[1] Schraffur*
- *[2] Stärke*
- *[3] (alter Konturstift)*
- *[4] Schraffurstift*
- *[5] Stift der Hintergrundschraffur*
- *[6] Kern/Bekleidungsstatus*
- *[7] oberer Linienstift*
- *[8] oberer Linientyp*
- *[9] unterer Linienstift*
- *[10] unterer Linientyp*
- *[11] Stirnflächenstift*
- *[12] Schraffurausrichtung*
- *[13] Oberflächentyp*
- *[14] Linientyp Stirnfläche*
- *[15] fertiger Oberflächenstatus*
- *[16] Status Schraffurausrichtung*
- *[17] Status einer Kernschicht (falls keine Kernschicht existiert: die stärkste Schicht)*
- *[18] Baustoff-Index.*

Kern: 0 - nein, 1 - ja, 3 - letzte Schicht, Richtung der Schraffur: 0 - global, 1 - lokal; Schichtentyp: in aktuellem ARCHICAD immer 0 - Schnitt, kann verwendet werden wie in Wänden später; Fertiger Schichtstatus: 0: Schicht nicht beenden, 1: Beenden Schicht

gleichwertig zu ROOF_SKINS_PARAMS bei Dächern

SHELLBASE_SKINS_BMAT_NAMES Name des Baustoffs einer Schicht bei mehrschichtigen Schalen/Dächern

Array mit 1 Spalten: Name des Baustoffs der Schicht mit beliebiger Zeilenanzahl

gleichwertig mit ROOF_SKINS_BMAT_NAMES bei Dächern

SHELLBASE_SECT_PEN Stiftfarbe der Konturen der Dachschnittfläche bei Schale/Dach

anzuwenden bei Konturen von Schnittflächen sowohl im Grundriss als auch in Schnitt/Ansicht, gleichwertig zu ROOF_SECT_PEN bei Dächern

SHELLBASE_VIEW_PEN Stiftfarbe der Dachaufsicht bei Schale/Dach

anzuwenden auf alle Kanten im 3D-Fenster und auf Außenkanten im Grundriss (Kantenaufsicht unterhalb der Schnittebene) im Grundriss und in Schnitt/Ansicht, gleichwertig zu ROOF_VIEW_PEN bei Dächern

SHELLBASE_REFERENCE_SURF	Referenzseiten-Fläche von Schale/Dach
nicht um Oberflächenöffnungen reduziert, gleichwertig zu ROOF_BOTTOM_SURF bei Dächern	
SHELLBASE_COND_REFERENCE_SURF	bedingte Referenzseiten-Fläche von Schale/Dach
gleichwertig zu ROOF_BOTTOM_SURF_CON bei Dächern	
SHELLBASE_GROSS_REFERENCE_SURF	Bruttooberfläche von Schale/Dach Referenzseite
reduziert um die Fläche der Öffnungen, gleichwertig zu ROOF_GROSS_BOTTOM_SURF bei Dächern	
SHELLBASE_OPPOSITE_SURF	Fläche gegenüberliegend zur Referenzseite von Schale/Dach
nicht reduziert um die Fläche der Öffnungen, gleichwertig zu ROOF_TOP_SURF bei Dächern	
SHELLBASE_COND_OPPOSITE_SURF	bedingte Oberfläche gegenüberliegend zur Referenzseite von Schale/Dach
reduziert um die Oberfläche von Öffnungen, welche größer als ein vorgegebener Wert sind; gleichwertig zu ROOF_TOP_SURF_CON bei Dächern	
SHELLBASE_GROSS_OPPOSITE_SURF	Bruttooberfläche gegenüberliegend zur Referenzseite von Schale/Dach
reduziert um die Fläche von Öffnungen, gleichwertig zu ROOF_GROSS_TOP_SURF bei Dächern	
SHELLBASE_EDGE_SURF	Kantenoberfläche von Schale/Dach
nicht reduziert um die Fläche der Öffnungen, gleichwertig zu ROOF_EDGE_SURF bei Dächern	
SHELLBASE_GROSS_EDGE_SURF	Bruttooberfläche von Schale/Dach-Kanten
reduziert um die Fläche der Öffnungen, gleichwertig zu ROOF_GROSS_EDGE_SURF bei Dächern	
SHELLBASE_PERIMETER	Perimeter von Schale/Dach
gleichwertig zu ROOF_PERIMETER bei Dächern	
SHELLBASE_VOLUME	Volumen von Schale/Dach
nicht reduziert um das Volumen der Öffnungen, gleichwertig zu ROOF_VOLUME bei Dächern	
SHELLBASE_COND_VOLUME	bedingtes Volumen von Schale/Dach
reduziert um das Volumen der Öffnungen, welche größer als ein vorgegebener Wert sind; gleichwertig zu ROOF_VOLUME_CON bei Dächern	
SHELLBASE_GROSS_VOLUME	Bruttovolumen von Schale/Dach
reduziert um das Volumen der Öffnungen, gleichwertig zu ROOF_GROSS_VOLUME bei Dächern	

SHELLBASE_HOLES_NR *gleichwertig zu ROOF_HOLES_NR bei Dächern*	Anzahl von Öffnungen in Schale/Dach
SHELLBASE_HOLES_SURF *gleichwertig zu ROOF_HOLES_AREA bei Dächern*	Oberfläche der Öffnungen bei Schale/Dach
SHELLBASE_HOLES_PRM *gleichwertig zu ROOF_HOLES_PRM bei Dächern*	Umfang von Öffnungen bei Schale
SHELLBASE_OPENINGS_NR	Anzahl von Öffnungen in Schale
SHELLBASE_OPENINGS_SURF	Oberflächenöffnungen in der Schale
SHELLBASE_INSU_THICKNESS *gleichwertig zu ROOF_INSU_THICKNESS bei Dächern*	Dämmstoffstärke in Schale/Dach
SHELLBASE_RIDGE *gleichwertig zu ROOF_RIDGE bei Dächern*	Firstlänge bei Schale/Dach
SHELLBASE_VALLEY *gleichwertig zu ROOF_VALLEY bei Dächern*	Kehlenlängen bei Schale/Dach
SHELLBASE_GABLE *gleichwertig zu ROOF_GABLE bei Dächern*	Giebellängen bei Schale/Dach
SHELLBASE_HIP *gleichwertig zu ROOF_HIP bei Dächern*	Walmlänge bei Schale/Dach
SHELLBASE_EAVES *gleichwertig zu ROOF_EAVES bei Dächern*	Traufenlänge bei Schale/Dach
SHELLBASE_PEAK *gleichwertig zu ROOF_PEAK bei Dächern*	Firstlänge bei Schale/Dach
SHELLBASE_SIDE_WALL *gleichwertig zu ROOF_SIDE_WALL bei Dächern*	Seitenwand-Verbindungslänge bei Schale/Dach

SHELLBASE_END_WALL	Endwand-Verbindungslänge bei Schale/Dach
gleichwertig zu ROOF_END_WALL bei Dächern	
SHELLBASE_TRANSITION_DOME	Dachkuppel-Verbindungslänge bei Schale/Dach
gleichwertig zu ROOF_TRANSITION_DOME bei Dächern	
SHELLBASE_TRANSITION_HOLLOW	Dachhöhlungs-Verbindungslänge bei Schale/Dach
gleichwertig zu ROOF_TRANSITION_HOLLOW bei Dächern	

Parameter für Morphs - nur für Listen und Etiketten verfügbar

MORPH_LINETYPE	Linientyp des Morphs in der Aufsicht
MORPH_FILL	Schraffur der Schnittflächen des Morphs
MORPH_BMAT_NAME	Name des Baustoffs von Schnittflächen bei Morphs
MORPH_BMAT	Index des Baustoffs von Schnittflächen bei Morphs
Kompatibilität: eingeführt in ARCHICAD 21.	
MORPH_FILL_PEN	Stiftfarbe der Schnittflächen des Morphs
MORPH_FBGD_PEN	Hintergrundstift der Schnittfläche des Morphs
MORPH_SECT_LINETYPE	Konturlinienstift der Schnittflächen des Morphs
MORPH_SECT_PEN	Stiftfarbe der Kontur der Schnittflächen des Morphs
MORPH_VIEW_PEN	Stiftfarbe der Kontur des Morphs in der Aufsicht
MORPH_SOLID	Massiver Morph-Körper (an/aus)
MORPH_MAT_DEFAULT	Morph-Standard-Oberflächenattributindex
MORPH_CASTS_SHADOW	Schattenwurf (an/aus)
MORPH_RECEIVES_SHADOW	Schatten-Empfang (an/aus)
MORPH_SURFACE	Brutto-Oberfläche des Morphs
MORPH_VOLUME	Volumen des Morphs
MORPH_FLOOR_PERIMETER	Umfang des Morphs im Grundriss

Benutzerdefinierbare globale Variablen

GLOB_USER_1	
GLOB_USER_2	
GLOB_USER_3	
GLOB_USER_4	
GLOB_USER_5	
GLOB_USER_6	
GLOB_USER_7	
GLOB_USER_8	
GLOB_USER_9	
GLOB_USER_10	benutzerdefinierbare globale Variablen von 1 bis 10 werden standardmäßig als numerisch deklariert
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 standardmäßig als Textparameter deklariert

Beispielhafte Anwendung von Globalen Variablen

Beispiel: zur Illustration der globalen Variablen 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 Origin"
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 Origin"
    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
```

Veraltete Globale Variablen

Diese Globalen Variablen funktionieren zwar noch in der ARCHICAD-Umgebung aus Kompatibilitätsgründen, aber wir empfehlen, diese bei neu erzeugten Objekten zu vermeiden.

GLOB_CONTEXT Kontext des Auftretens (Sicht-abhängig, nicht im Parameter/Eigenschaften-Script verwenden)

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	Standard	2

1 - GDL-Editor, 2 - Grundriss, 3 - 3D-Modell, 4 - Schnitt/Ansicht, 5 - Einstellungsdialog, 6 - Liste, 7 - Detailzeichnung, 8 - Layout, 22 - Hotpot-Editierfeedback im Grundriss, 23 - Hotpot-Editierfeedback im 3D-Modell, 24 - Hotpot-Editierfeedback in Schnitt/Ansicht, 28 - Hotpot-Editierfeedback im Layout, 43 - fungiert als SEO-Operator im 3D-Modell, 44 - fungiert als SEO-Operator in Schnitt/Ansicht, 46 - fungiert als SEO-Operator in einer Liste. Für weitere Details, siehe „GDL-Ausführungs-Kontext“.

Veraltete Globale Träger- / Stützenvariablen - nur für Listen und Etiketten verfügbar

Ab ARCHICAD 23 sind diese Werte in den Globalen Variablen BEAM_SEGMENT_INFO und COLU_SEGMENT_INFO mit einheitlichen Wertereferenzen verfügbar. Siehe auch „Veraltete Träger- / Stützenparameter - nur für Auflistung und Etiketten verfügbar“. Aus Kompatibilitätsgründen sind sie weiterhin auf homogenen, geraden oder horizontal gekrümmten Trägern und auf homogenen Stützen (GLOB_ELEM_TYPE = 12 oder 6) verfügbar).

COLU_CROSSSECTION_TYPE Querschnittstyp der Stütze

⛔ *Stützen-Segment (27)*	✓ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

0 - komplex profiliert, 1 - rechteckig, 4 - rund

BEAM_CROSSSECTION_TYPE Querschnittstyp des Trägers

⛔ *Träger-Segment (28)*	✓ *Einzel-Segment-Träger(12)*	⛔ *Multi-Segment-Träger(12)*

0 - komplex profiliert , 1 - rechteckig

Veraltete globale Etiketten-Variablen

Diese Globalen wurden durch fest benannte optionale Parameter ersetzt, beginnend mit ARCHICAD 22 (siehe Abschnitt). Aus Kompatibilitätsgründen werden die globalen Variablen weiterhin gepflegt, so dass das Verhalten von älteren Objekten nicht beeinflusst werden sollte.

Textverarbeitungs-Gruppe der Globalen Variablen bei Etiketten:

LABEL_ALWAYS_READABLE Etiketttext ist immer lesbar

1 falls "immer lesbar" angekreuzt ist, sonst 0

LABEL_TEXT_WRAP	Etikett-Text umbrechen
1 falls Textumbruch angekreuzt ist, sonst 0	
LABEL_TEXT_ALIGN	Textausrichtung im Einstellungsdialogfeld
1 - linksbündig, 2 - zentriert, 3 - rechtsbündig, 4 - Blocksatz	
LABEL_TEXT_LEADING	Zeilenabstandsfaktor.
LABEL_TEXT_WIDTH_FACT	Laufweitenfaktor im Einstellungsdialogfeld
LABEL_TEXT_CHARSPACE_FACT	Faktor Zwischenraum
LABEL_FONT_NAME	Zeichensatzname im Einstellungsdialogfeld
LABEL_TEXT_SIZE	Textgröße im Einstellungsdialogfeld
LABEL_TEXT_PEN	Stift des Texts im Einstellungsdialogfeld
LABEL_TEXT_BG_PEN	Hintergrundstift des Textes im Einstellungsdialogfeld
0, wenn "deckend" ausgeschaltet ist, andernfalls der Index des Hintergrundstiftes	
LABEL_FONT_STYLE	Schriftschnitt im Einstellungsdialogfeld
0-normal, 1-fett, 2-kursiv, 4-unterstrichen	
LABEL_FONT_STYLE2	Schriftschnitt im Einstellungsdialogfeld
*0 - normal, andernfalls j1 + 2*j2 + 4*j3 + 32*j6 + 64*j7 + 128*j8, j1 - fett, j2 - kursiv, j3 - unterstrichen, j6 - hochgesetzt, j7 - tiefgesetzt, j8 - durchgestrichen*	
Zeiger/Rahmen-Gruppe der Globalen Variablen bei Etiketten:	
LABEL_CUSTOM_ARROW	Etikettpfeil verwenden ja/nein
1 wenn das Kontrollkästchen Etikettpfeil verwenden aktiviert ist, sonst 0	
LABEL_ARROW_LINETYPE	Linientyp der Linie des Pfeils
LABEL_ARROW_PEN	Stift der Pfeillinie im Einstellungsdialogfeld
LABEL_FRAME_ON	Etikettenrahmen ja/nein
1 wenn der Etikettenrahmen aktiviert ist, sonst 0	
LABEL_FRAME_OFFSET	Offset des Frames

LABEL_ANCHOR_POS	Postion des Ankerpunkts
0 - Mitte, 1 - oben, 2 - unten, 3 - unten rechts	

Veraltete globale Variable der Fassadenprofile- verfügbar für nur Auflistung und Etiketten

Ab ARCHICAD 22 wurden benutzerdefinierte Fassaden-Profil-Klassen eingeführt an Stelle von vordefinierten Klassen. Aus Kompatibilitätsgründen wird die Globale Variable CWFRAME_CLASS weiterhin gepflegt.

CWFRAME_CLASS	Klasse des Profils
2 - Grenze, 3 - andere	
Kompatibilität: bis ARCHICAD 21 gibt es folgende Werte 0 - Pfosten, 1 - Riegel, 2 - Randeinfassung, 3 - benutzerdefiniert	

Alte Globale Variablen

Alte globale Parameternamen sind zwar noch gültig; trotzdem empfehlen wir die Verwendung der neuen Namen. Alle alten Globalen beziehen sich auf eine neue Variable mit einem langen Namen.

```
A_      GLOB_SCALE
B_      GLOB_HSTORY_ELEV
C_      WALL_THICKNESS
D_      WALL_HEIGHT
E_      WALL_SECT_PEN
F_      WALL_FILL_PEN
G_      WALL_MAT_A
H_      WALL_MAT_B
I_      WALL_MAT_EDGE
J_      GLOB_ELEVATION
K_      WIDO_SILL
L_      SYMB_VIEW_PEN
M_      SYMB_MAT
N_      GLOB_FRAME_NR
O_      GLOB_FIRST_FRAME
P_      GLOB_LAST_FRAME
Q_      GLOB_HSTORY_HEIGHT
R_      WIDO_ORIG_DIST
S_      GLOB_USER_1
T_      GLOB_USER_2
U_      GLOB_USER_3
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
F~      WIDO_FRAME_THICKNESS
G~      GLOB_USER_9
H~      WIDO_POSITION
I~      GLOB_USER_10
J~      WALL_RESOL
K~      GLOB_EYEPOS_X
L~      GLOB_EYEPOS_Y
M~      GLOB_EYEPOS_Z
N~      GLOB_TARGPOS_X
O~      GLOB_TARGPOS_Y
P~      GLOB_TARGPOS_Z
Q~      GLOB_CSTORY_ELEV
R~      GLOB_CSTORY_HEIGHT
S~      GLOB_CH_STORY_DIST
T~      GLOB_SCRIPT_TYPE
U~      GLOB_NORTH_DIR
V~      SYMB_MIRRORED
W~      SYMB_ROTANGLE
X~      SYMB_POS_X
Y~      SYMB_POS_Y
Z~      SYMB_POS_Z
```

FESTBENANNTE OPTIONALE PARAMETER

Parameter festgelegt von ARCHICAD

Die neue Methode, mit welcher ARCHICAD Informationen liefert, ist die Methode der optionalen Fixnamen-Parameter (fixed named optional parameters). Falls ein vorhandenes Bibliothekselement einen Parameter enthält, der mit einem optionalen Fixnamen-Parameter bezüglich Namen und Typ übereinstimmt, legt ARCHICAD seinen Wert entsprechend seiner Funktion fest.

Parameter für D/W Attribute (verfügbar für Türen (D), Fenster (W), Etiketten und Listen)

Grundrissdarstellung

ac_hole_cut_linetype	Linientyp
Stift von Schnittlinien [Grundriss und Schnitt]	
ac_hole_overhead_pen	Stift
Stift der von nach oben betrachteten Kanten (Überkopf) [nur Grundriss]	
ac_hole_overhead_linetype	Linientyp
Linientyp der von nach oben betrachteten Kanten (Überkopf) [nur Grundriss]	
ac_hole_uncut_pen	Stift
Stift der unterhalb sichtbaren Kanten (ungeschnitten) [nur Grundriss]	
ac_hole_uncut_linetype	Linientyp
Linientyp der unterhalb sichtbaren Kanten (ungeschnitten) [nur Grundriss]	
ac_hole_display_option	ganzzahlig
Optionen der Grundrissdarstellung: 1 - Projiziert, 2 - Projiziert mit Untersicht, 3 - Symbolisch, 5 - Untersicht	

Richtung

ac_hole_direction_type	ganzzahlig
Richtung Öffnungsebene: 1 - An Wand angeschlossen, 2 - Senkrecht	
ac_wido_rotation	Winkel
Rotationswinkel Tür/Fenster relativ zur waagerechten Schneideebene	
ac_openingside	Zeichenfolge
Ausrichtungsparameter für Fenster/Tür für Listen (L - links, R - rechts, benutzerdefiniert) entsprechend der Einstellung der Ausrichtungsdefinition (Automatisch, benutzerdefiniert). ARCHICAD wird alle Einstellungen der Kompatibilitätsoptionen/Ausrichtungsanzeige außer Acht lassen, wenn dieser Parameter vorhanden ist.	

Daten einer Polygonalwand

ac_walltype ganzzahlig

gibt an, ob das Fenster sich in einer Polygonwand befindet oder nicht. 1 - nicht polygonal, 2 - polygonal.

ac_wallContourPolygon[][3] Länge

Polygonwand in 2D: Punkte plus ein Extrawinkel für Bogenabschnitte. [nur gesetzt wenn ***ac_walltype*** *ist gleich 2]*

ac_windowInWallContour[4] ganzzahlig

Indizes der vier Scheitel des ***ac_wallContourPolygon*** *Polygons, welches sich als Fenstereckpunkte innerhalb des Wandkonturpolygons befinden. [nur gesetzt wenn* ***ac_walltype*** *ist gleich 2]*

Position der Öffnung

ac_hole_position_angle Winkel

Im Fall von gebogenen Wänden wird der Winkel zwischen der Achse der Öffnung und dem Normalvektor am Anfangspunkt der Wand gegeben.

Ankerdaten

ac_vertAnchorPos ganzzahlig

senkrechter Anker von Fenster/Türen: 1 - Schwelle, 2 - Kopf

ac_revealAnchorPos ganzzahlig

Laibungs-Anker von Fenster/Türen: 1 - Vorderseite, 2 - Kern

ac_revealToWallCore Länge

Laibungstiefe gemessen von der Laibungsseite des Wandkernes.

Parameter für Wand-Attribute (verfügbar für Türe, Fenster, Etikett, Liste)

Grundrissdarstellung

ac_wall_overhead_pen Stift

Stift der oberhalb angezeigten Kanten der Wand (Untersicht) [nur Grundriss]

ac_wall_overhead_linetype Linientyp

Linientyp der oberhalb angezeigten Kanten der Wand (Untersicht) [nur Grundriss]

ac_wall_uncut_linetype	Linientyp
Linientyp der unterhalb sichtbaren Kanten (ungeschnitten) [nur Grundriss]]	
ac_wall_display_option	ganzzahlig
Grundrissdarstellungs-Optionen der Wand: 1 - Projiziert, 2 - Projiziert mit Untersicht, 3 - Symbolisch, 4 - Nur Konturlinie, 5 - Untersicht	
ac_wall_show_projection_to	ganzzahlig
senkrechte Tiefenbegenzung der Darstellung der Wand: 1 - Zum Grundrissbereich, 2 - Zur absoluten Darstellungsgrenze, 3 - Komplettes Element	

Geometriedaten

ac_wall_elevation	Länge
Höhenpunkt der Wand unten, bezogen auf das Ursprungsgeschoss der Wand	
ac_wall_crosssection_type	ganzzahlig
Querschnittstyp der Wand: 1 - Einfach, 2 - Komplexes Profil, 3 - geneigt, 4 - 2-seitig geneigt	
ac_wall_profile_name	Zeichenfolge
*Profilname, falls die Wand ein komplexes Profil mit einem Profilattribut ist, "Custom_Profile_i" falls komplex mit benutzerdefiniertem Profil (**i** ist die ID der platzierten Wand) oder "n/a" falls die Wand einfach, geneigt oder doppelt geneigt ist.*	
ac_wall_slant_angle1	Winkel
erster Neigungswinkel der Wand bezogen auf die Horizontale (90 Grad, falls die Wand senkrecht steht)	
ac_wall_slant_angle2	Winkel
zweiter Neigungswinkel der Wand bezogen auf die Horizontale (90 Grad, falls die Wand senkrecht steht)	
ac_wall_direction_type	ganzzahlig
Wandrichtungstyp; die Konstruktionsmethode der Wand, d.h. die Anpassung des Wandkörpers und der Referenzlinie: 0 - Rechts, 1 - Links, 2 - Zentriert(Rechts), 3 - Zentriert (Links). Zentriert bedeutet, dass die Wand auf "zentriert" im Einstellungsdialog gestellt ist, aber der Seitenverweis zeigt, wie die Wand intern reagiert.	

Parameter für Stützen-Attribute (verfügbar für Etikett, Liste)

Die Verfügbarkeit jedes Parameters (unabhängig davon, ob er aussagekräftige Daten enthält) wird in einer Tabelle mit Symbolen angezeigt, wobei der Wert der globalen Variablen GLOB_ELEM_TYPE in Klammern steht.

Grundrissdarstellung

ac_colu_overhead_pen Stift

Stift der oberhalb sichtbaren Kanten der Stütze (Untersicht) [nur Grundriss]

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*

ac_colu_overhead_linetype Linientyp

Linientyp der oberhalb sichtbaren Kanten der Stütze (Untersicht) [nur Grundriss]

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*

ac_colu_uncut_linetype Linientyp

Linientyp der unterhalb sichtbaren Kanten der Stütze (nicht geschnitten) [nur Grundriss]

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*

ac_colu_display_option ganzzahlig

Grundrissdarstellungsoptionen für die Stütze: 1 - Projiziert, 2 - Projiziert mit Untersicht, 3 - Symbolisch, 4 - Nur Konturlinie, 5 - Untersicht

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*

ac_colu_show_projection_to ganzzahlig

senkrechte Tiefenbegrenzung der Darstellung der Stütze: 1 - Zum Grundrissbereich, 2 - Zur absoluten Darstellungsgrenze, 3 - Komplettes Element

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*

Geometrische Daten

ac_colu_profile_name Zeichenfolge

*Profilname, falls die Stütze ein komplexes Profil mit einem Profilattribut ist, "Custom_Profile_i" falls komplex mit benutzerdefiniertem Profil (**i** ist die ID der platzierten Stütze) oder "n/a" falls die Stütze rechteckig oder rund ist*

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

ac_colu_inclination Winkel

Neigungswinkel der Stütze bezogen auf die Horizontale

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*

ac_colu_twist_angle Winkel

Drehwinkel des Querschnitts

✅ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	✅ *Multi-Segment-Stütze(6)*

Parameter für Träger-Atribute (verfügbar für Etiketten, Listen)

Die Verfügbarkeit jedes Parameters (unabhängig davon, ob er aussagekräftige Daten enthält) wird in einer Tabelle mit Symbolen angezeigt, wobei der Wert der globalen Variablen GLOB_ELEM_TYPE in Klammern steht.

Grundrissdarstellung

ac_beam_overhead_pen Stift

Stift der oberhalb sichtbaren Kanten des Trägers (Untersicht) [nur Grundriss]

✅ *Träger-Segment (28)*	✅ *Einzel-Segment-Träger(12)*	✅ *Multi-Segment-Träger(12)*

ac_beam_cut_linetype Linientyp

Linientyp der Schnittkanten des Trägers [gilt nicht für komplexes Profil]

✅ *Träger-Segment (28)*	✅ *Einzel-Segment-Träger(12)*	✅ *Multi-Segment-Träger(12)*

ac_beam_uncut_pen Stift

Stift der unterhalb sichtbaren Kanten des Trägers (nicht geschnitten) [nur Grundriss]

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger(12)*	✔ *Multi-Segment-Träger(12)*

ac_beam_uncut_linetype Linientyp

Linientyp der unterhalb sichtbaren Kanten des Trägers (nicht geschnitten) [nur Grundriss]

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger(12)*	✔ *Multi-Segment-Träger(12)*

ac_beam_display_option ganzzahlig

Grundrissdarstellungsoptionen des Trägers: 1 - Projiziert, 2 - Projiziert mit Untersicht, 3 - Symbolisch, 4 - Nur Konturlinie, 5 - Untersicht

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger(12)*	✔ *Multi-Segment-Träger(12)*

ac_beam_show_projection_to ganzzahlig

senkrechte Tiefenbegrenzung der Darstellung des Trägers: 1 - Zum Grundrissbereich, 2 - Zur absoluten Darstellungsgrenze, 3 - Komplettes Element

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger(12)*	✔ *Multi-Segment-Träger(12)*

Geometrische Daten

ac_beam_profile_name Zeichenfolge

*Profilname, falls der Träger ein komplexes Profil mit einem Profilattribut ist, "Custom_Profile_i" falls komplex mit benutzerdefiniertem Profil (**i** ist die ID des platzierten Trägers) oder "n/a" falls der Träger rechteckig ist*

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger(12)*	⛔ *Multi-Segment-Träger(12)*

ac_beam_inclination Winkel

Neigungswinkel des Trägers bezogen auf die Horizontale

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger(12)*	✔ *Multi-Segment-Träger(12)*

ac_beam_twist_angle Winkel

Drehung um die Achse des geschnittenen Profils (0.0 bei nicht-komplexen Trägern)

✔ *Träger-Segment (28)*	✔ *Einzel-Segment-Träger(12)*	✔ *Multi-Segment-Träger(12)*

Parameter für Dach-Attribute (verfügbar für Etikett, Liste)

Grundrissdarstellung

ac_roof_overhead_pen Stift

Stift der oberhalb sichtbaren Kanten des Daches (Untersicht) [nur Grundriss]

ac_roof_overhead_linetype Linientyp

Linientyp der oberhalb sichtbaren Kanten des Daches (Untersicht) [nur Grundriss]

ac_roof_display_option ganzzahlig

Grundrissdarstellungsoptionen des Daches: 1 - Projiziert, 2 - Projiziert mit Untersicht, 3 - Symbolisch, 4 - Nur Konturlinie, 5 - Untersicht

ac_roof_show_projection_to ganzzahlig

senkrechte Tiefenbegrenzung der Darstellung des Daches: 1 - Zum Grundrissbereich, 2 - Zur absoluten Darstellungsgrenze, 3 - Komplettes Element

Tür/Fenster-Marker Attribute

ac_wido_id Zeichenfolge

ID der Öffnung

ac_wido_a_size Länge

Öffnungsbreite

ac_wido_b_size Länge

Öffnungshöhe

ac_wido_z_size Länge

Öffnungstiefe/dicke

ac_glob_elevation	Länge
Höhenwert der Grundlinie der Öffnung	
ac_wido_subfl_thickness	Länge
Höhe des Unterboden-Wandteiles	
ac_wido_reveal_side	Boolesche
Seitenwert der geerbten Laibungsöffnung, verwenden Sie stattdessen ac_wido_reveal_side_2	
ac_wido_reveal_side_2	Boolesche
Anschlagseite, der Wert der Globalen Variable ***WIDO_REVEAL_SIDE****, welcher für die Öffnung gesetzt ist*	
ac_wido_mirrored	Boolesche
gespiegelter Status der Öffnung	
ac_wall_thickness	Länge
Wandstärke am Ursprung der Öffnung	
ac_wido_oversize_l	Länge
linke Öffnungsübergröße	
ac_wido_oversize_r	Länge
rechte Öffnungsübergröße	
ac_wido_oversize_t	Länge
obere Öffnungsübergröße	
ac_wido_oversize_b	Länge
untere Öffnungsübergröße	
ac_wido_orientation	Zeichenfolge
Markerposition: "L" - Links, "R" - Rechts, oder jeder andere Wert, der im Detailfenster des Bibliothekselement-Editors eingestellt wurde, entsprechend dem aktuellen Spiegelungs-Status	
ac_wido_type	ganzzahlig
1 - Tür, 2 - Fenster	

ac_symb_rotangle	Winkel
Öffnungsverdrehung in der Wand	
ac_sill_to_curr_story	Länge
Brüstungshöhe der Öffnung bezogen auf den Beginn des Geschosses verknüpft mit der Fensterbrüstung	
ac_sill_to_anchor_level	Länge
Brüstungshöhe der Öffnung bezogen auf das Niveau des Ankerpunktes; der Ankerpunkt kann entsprechend am Fußpunkt der Wand des ausgewählten Geschosses liegen	

Detail/Arbeitsblatt Marker Attribute

ac_showboundary	Boolesche
Marker-Status des Begrenzungspolygons. 0 - Rahmen an, 1 - Rahmen aus.	

Zeichnungstitel-Attribute

ac_drawingName	Zeichenfolge
Name der Zeichnung.	
ac_drawingNumber	Zeichenfolge
ID der Zeichnung.	
ac_sourceFileName	Zeichenfolge
Name der Quelldatei der Zeichnung (falls die Zeichnung aus einer externen Datei stammt).	
ac_sourceFilePath	Zeichenfolge
Pfad der Quelldatei der Zeichnung (falls die Zeichnung aus einer externen Datei stammt).	
ac_drawingScale	Zeichenfolge
eingestellter Zeichnungsmaßstab der Zeichnung.	
ac_magnification	reale Zahl
Vergrößerungsfaktor der Zeichnung in Prozent.	
ac_originalDrawingScale	Zeichenfolge
Zeichnungsmaßstab des ursprünglichen Ausschnitts.	

ac_enableBackReference	Boolesche
Rückreferenzierung ist in der Zeichnung aktiviert.	
ac_backReferenceGUIDList	Zeichenfolge-Array
Liste der verwiesenen Layout-GUIDs. Diese können in Autotext-Ausgaben verwendet werden.	
ac_showDrawingReferences	Boolesche
Anzeige der Rückrefrenzen.	

Allgemein laufender Kontext

ac_programVersion	ganzzahlig
Dieser Parameter enthält die ARCHICAD-Version, welche die Scripte der Bibliothekselemente ausführt.	

Raumparameter (verfügbar für Raumstempel)

Name	Typ	Standard	Beschreibung
ROOM_NAME	Zeichenfolge		Raumname
ROOM_NUMBER	Zeichenfolge		Raumnummer
ROOM_AREA	Reelle	0.0	Fläche brutto/netto Polygon
ROOM_PERIM	Länge	0.0	Umfang von brutto/netto Polygon
ROOM_HOLES_PRM	Länge	0.0	Umfang von netto Polygonöffnungen
ROOM_WALLS_PRM	Länge	0.0	Umfang von netto Polygonen (mit Öffnungen), aber nur bei Wänden als Raumbegrenzung
ROOM_CORNERS	ganzzahlig	0	Ecken der netto Polygone (mit Öffnungen)
ROOM_CONCAVES	ganzzahlig	0	Konkave Ecken der netto Polygone (mit Öffnungen)
ROOM_WALLS_SURF	Reelle	0.0	Angrenzende Wandoberflächen (angrenzende Elemente)
ROOM_DOORS_WID	Reelle	0.0	Türlängen auf der Raumgrenze
ROOM_DOORS_SURF	Reelle	0.0	Türoberflächen auf der Raumgrenze
ROOM_WINDS_WID	Länge	0.0	Fernsterlängen auf der Raumgrenze
ROOM_WINDS_SURF	Reelle	0.0	Fensteroberflächen auf der Raumgrenze
ROOM_BASELEV	Länge	0.0	Raumlevel
ROOM_FL_THICK	Länge	0.0	Stärke Bodenaufbau
ROOM_HEIGHT	Länge	0.0	Raumhöhe
ROOM_NET_AREA	Reelle	0.0	Fläche der Nettopolygone (mit Öffnungen)
ROOM_NET_PERIMETER	Länge	0.0	Umfang der Nettopolygone (mit Öffnungen))
ROOM_WALL_EXTR_AREA	Reelle	0.0	Durch Wände reduzierte Fläche innerhalb des Raumes (nicht vom Typ Raumflächenbegrenzung!)

Name	Typ	Standard	Beschreibung
ROOM_COLUMN_EXTR_AREA	Reelle	0.0	Durch Stützen reduzierte Fläche innerhalb des Raumes (nicht vom Typ Raumflächenbegrenzung!)
ROOM_FILL_EXTR_AREA	Reelle	0.0	Durch Luken reduzierte Fläche innerhalb des Raumes (Berücksichtigung Prozent!)
ROOM_LOW_EXTR_AREA	Reelle	0.0	Durch niedrige Elemente reduzierte Fläche (getrimmt) (Berücksichtigung Prozent!)
ROOM_TOTAL_EXTR_AREA	Reelle	0.0	Summe der vorherigen Werte (Gesamtausgabe)
ROOM_REDUCED_AREA	Reelle	0.0	ROOM_NET_AREA - ROOM_TOTAL_EXTR_AREA
ROOM_AREA_FACTOR	Reelle	0.0	1 - Reduced_by_in_dialog / 100
ROOM_CALC_AREA	Reelle	0.0	ROOM_REDUCED_AREA * ROOM_AREA_FACTOR
ROOM_VOLUME	Reelle	0.0	Von getrimmtem Raum bei Nettopolygon berechnet
ROOM_BOUNDARY_SURF	Reelle	0.0	Oberfläche von Raumseitenflächen
ROOM_TOP_SURFACE	Reelle	0.0	Oberfläche des Raumdeckels
ROOM_BOT_SURFACE	Reelle	0.0	Oberfläche des Raumbodens
ROOM_ROOF_TOP_SURF	Reelle	0.0	Oberfläche des Raumdeckels, wo durch ein Dach getrimmt
ROOM_ROOF_BOT_SURF	Reelle	0.0	Oberfläche des Raumbodens, wo durch ein Dach getrimmt
ROOM_SLAB_TOP_SURF	Reelle	0.0	Oberfläche des Raumdeckels, wo durch eine Decke getrimmt
ROOM_SLAB_BOT_SURF	Reelle	0.0	Oberfläche des Raumbodens, wo durch eine Decke getrimmt
ROOM_BEAM_TOP_SURF	Reelle	0.0	Oberfläche des Raumdeckels, wo durch einen Träger getrimmt
ROOM_BEAM_BOT_SURF	Reelle	0.0	Oberfläche des Raumbodens, wo durch einen Träger getrimmt
ROOM_WALL_IN_TOP_SURF	Reelle	0.0	Summe der Deckelflächen von Wandeinrückungen (oder Vertiefung oder Nische)
ROOM_WALL_IN_BACK_SURF	Reelle	0.0	Summe der of Rückseiten der Wandeinrückungen (mit Gesicht zum Fenster)
ROOM_WALL_IN_SIDE_SURF	Reelle	0.0	Summe der Seitenflächen von Wandeinrückungen
ROOM_POLY_STATUS	ganzzahlig	0	0:manuell, 1:automatisch, 2:automatisch-neuberechnet

Treppenbezogene Parameter

Unterstützte Subtypen von Lauf/Podest

ac_beamPlacement ganzzahlig

Dieser Parameter gibt an, auf welcher Seite der Treppe die Wange platziert ist. Kompatibilität: eingeführt in ARCHICAD 21.

- *0 - links (Blickrichtung treppenaufwärts)*
- *1 - rechts*

Subtype für Setzstufen-Komponenten

ac_RiserPosition ganzzahlig

Dieser Parameter spiegelt die aktuellen Einstellungen von Setzstufen- und Trittstufenverbindungen wider:

- *0 - Setzstufe AN, Trittstufe AUS*
- *1 - Setzstufe AN, Trittstufe AN*

Subtypen für 2D-Treppenkomponenten

ac_treadClassifications Ganzzahl-Array

Typ jeder Trittstufe (indiziert von unten aufwärts):

- *0 - Trittstufe*
- *1 - Podest*

Parameters, welche ARCHICAD liest und schreibt

ARCHICAD kann Werte mit Bibliothekselementen synchronisieren mittels Parameters mit vordefiniertem Namen und Funktion. Die Liste mit solchen Parametern folgt unten.

Treppenrelevante Parameters

Subtyp für Struktur

ac_stairStructureWidth Länge

Breite der tragenden Wange, wie in den Treppeneinstellungen eingestellt. Wert ist 0, wenn als Strukturtyp "Holm" gewählt wurde.

ac_stairStructureThickness	Länge
Dicke/Höhe des Holms oder der auskragenden Struktur, wie in den Treppeneinstellungen eingestellt.	

Parameter von ARCHICAD ausgelesen

ARCHICAD kann Werte aus Bibliothekselementen auslesen mittels Parametern mit vordefinierten Namen und Funktionen. Die Liste mit solchen Parametern folgt unten.

Objekte im Grundriss

Beschneiden von planaren Elementen im Grundriss (z.B. Dachfenster, Dachzubehörobjekte)

ac_special_2d_symbol	Boolesche
Dieser Parameter ermöglicht einen 2D-Schneidemechanismus im ARCHICAD-Grundriss. Wenn der Parameter auf 1 steht, schneidet ARCHICAD das 2D-Modell (erzeugt vom 2D-Script des Objektes) entsprechend den Parameterwerten in: ac_symb_display_option, ac_symb_show_projection_to und ac_plane_definition. Dieser 2D-geführte Schnitt funktioniert wie die Darstellung von einfachen Dächern mit den gleichen Einstellungen. Natürlich liefert diese Methode eine korrekte Ausgabe nur für platten-ähnliche Elemente - wie Dachfenster und Dachzubehör. Die Ebene des Flachobjektes - und die Ebenen des Schnittes - sind definiert durch den Parameter ac_plane_definition. Im Falle von Dachfenstern und Dachzubehör - if ac_special_2d_symbol is 1 -, werden die oben genannten Parameter automatisch durch das Addon gesetzt. Im Falle von anderen Elementen, sollten diese durch den GDL-Entwickler ausgefüllt werden.	
ac_plane_definition	Länge
Ebenendefinition: ([1],[2],[3]): ein Punkt der Ebene, ([4],[5],[6]): Normalvektor der Ebene.	
ac_symb_display_option	ganzzahlig
1 - Projiziert, 2 - Projiziert mit Untersicht, 3 - Symbolisch, 4 - Nur Konturlinie, 5 - Untersicht	
ac_symb_show_projection_to	ganzzahlig
1 - Zum Grundrissbereich, 2 - Zur absoluten Darstellungsgrenze, 3 - Komplettes Element	
ac_bottomlevel	Länge
Dieser Parameter zeigt den niedrigsten Punkt des Objektes an. Wenn "Auf Geschoss zeigen" auf "alle relevanten Geschossen" eingestellt ist, und falls dieser niedrigste Punkt (berechnet aus den Einstellungen des Ursprungsgeschosses des Objekts) in der vertikalen Erweiterung eines Geschosses enthalten ist, wird das Objekt in dem Geschoss angezeigt. Das Kopflevel muss sich oberhalb des Bodenlevels befinden.	
***ac_bottomlevel** startet vom Objektursprung des Objektes.*	

ac_toplevel Länge

Wenn "Auf Geschoss zeigen" eingestellt ist auf "alle relevanten Geschosse", gibt dieser Parameter den Kopf des Objektes an. Das Objekt ist dann auf einem Geschoss sichtbar, wenn sich die Geschosshöhe zwischen dem Bodenlevel und Kopflevel bewegt. Das Kopflevel muss sich oberhalb des Bodenlevels befinden.

***ac_toplevel** startet vom Objektursprung des Objektes.*

Tür/Fenster - Objekte

ac_wido_sill Länge

Dieser Parameter gestattet vollen Zugriff auf die Brüstungstiefe des Öffnungsobjektes. Der Parameter kann eine Auswahlliste bekommen, kann gesperrt und versteckt werden und sein Wert kann Parameter-Script gesetzt werden. Sein aktueller Wert wird der Globalen Variable WIDO_SILL zugewiesen, wegen der Kompatibilität mit älteren Scripten.

ac_wido_hide_options ganzzahlig

Mit Hilfe dieses Bitfeld-Parameters können Sie Optionen des Fenster/Tür-Einstellungsdialoges deaktivieren. ***ac_wido_hide_options = j1 + 2*j2****. wenn j1 gesetzt ist, wird das Brüstungstiefeneingabefeld im Standard-ARCHICAD-Einstellungsdialog versteckt.. wenn j2 gesetzt ist, werden die Anschlageinstellungen im Einstellungsdialog deaktiviert.*

ac_wido_flip_once Boolesche

Spiegelt das Fenster oder die Tür (um Wandmittelachse) einmal nach der Ausführung des Migrations-Scriptes, falls der Parameter vorhanden ist und sein Wert ***wahr*** *ist.*

ac_wido_flip_disable ganzzahlig

Dieser Parameter kann den "Spiegeln"-Button im Einstellungsdialog deaktivieren. Der grundeingestellte Wert betrifft nur das Platzieren des Objektes.

-1: Spiegeln ist aktiviert.

0: Spiegeln ist deaktiviert. Grundeinstellung ist nicht gespiegelt.

1: Spiegeln ist deaktiviert. Grundeinstellung ist gespiegelt.

ac_wido_mirror_once Boolesche

Spiegelt das Fenster oder die Tür (um Wandmittelachse) einmal nach der Ausführung des Migrations-Scriptes, falls der Parameter vorhanden ist und sein Wert ***wahr*** *ist.*

ac_hole_hotspot_control ganzzahlig

Kontrolliert, ob Öffnungen automatische Hotspots haben. 0 - keine automatischen Hotspots, 1 - nur in 2D, 2 - nur in 3D, 3 - überall

ac_holeSideMaterial Boolesche

Steuert die vererbten Oberflächen der Wandöffnungen. Wenn die Wandöffnung über ein GDL-Script definiert wurde, hat dieser Parameter keine Auswirkung.

1: Die Wandöffnung hat die gleiche Oberfläche wie die Wandkante.

0: Die Wandöffnung-Fläche wird durch die Schwellentiefenlinie unter Verwendung der Innen-Außenflächen der Wand geteilt.

ac_holeMaterialCurved Boolesche

Steuert die Trennliniengeometrie von Innen-Außenwandflächen der Wandöffnung in gekrümmten Wänden. Wenn die Wandöffnung über ein GDL-Script definiert wurde, hat dieser Parameter keine Auswirkung.

1: Oberflächen verbinden sich in einer gekrümmten Linie.

0: Oberflächen verbinden sich in einer geraden Linie.

Eigene-Komponenten-Vorlage

ac_custom_component_type_name Zeichenfolge

Dieser Parameter enthält den Namen der Eigene-Komponenten-Vorlage, welche im "Auswahl sichern als ...." Menü angezeigt wird. Dieser kann vom Objektnamen abweichen.

Treppenrelevante Parameter

Subtyp der Struktur

ac_beamProfileID ganzzahlig

Index des ausgewählten Profils für den tragenden Holm. ARCHICAD stellt die Begrenzungsboxgröße des Profils mit Hilfe von Abfrageoptionen zur Verfügung.

Subtyp für Lauf / Podest: unterstützte Struktur

ac_stairStructureHorizThick Länge

Horizontale Stärke der auskragenden Struktur, z.B. Teile hinter den Setzstufen. Kompatibilität: eingeführt in ARCHICAD 22.

ac_stairStructureBoundsRiser Boolesche

Dieser kann dazu verwendet werden, um anzuzeigen, dass die Setzstufen schmaler als die Treppenbegrenzung sein sollten. Verwenden Sie diese zusammen mit den folgenden vier Parametern. Kompatibilität: eingeführt in ARCHICAD 22.

ac_stairRiserLeftBoundaryFrom ganzzahlig

Strecken Sie die linke Kante der Setzstufen relativ zu: Kompatibilität: eingeführt in ARCHICAD 22.

- *-1 - Linke Begrenzung der Treppe*
- *0 - Mittelachse der Treppe*
- *1 - Rechte Begrenzung der Treppe*
- *Diese sind nicht notwendigerweise parallel zueinander.*

ac_stairRiserLeftBoundary Länge

Abstand der linken Setzstufenkante gemessen von der in ac_stairRiserLeftBoundaryFrom eingestellten Ankerlinie. Positive Richtung ist nach rechts. Kompatibilität: eingeführt in ARCHICAD 22.

ac_stairRiserRightBoundaryFrom ganzzahlig

Strecken Sie die rechte vertikale Kante der Setzstufen relativ zu: Kompatibilität: eingeführt in ARCHICAD 22.

- *-1 - Linke Begrenzung der Treppe*
- *0 - Mittelachse der Treppe*
- *1 - Rechte Begrenzung der Treppe*
- *Diese sind nicht notwendigerweise parallel zueinander.*

ac_stairRiserRightBoundary Länge

Abstand der rechten Setzstufenkante gemessen von der in ac_stairRiserRightBoundaryFrom eingestellten Ankerlinie. Positive Richtung ist nach rechts. Kompatibilität: eingeführt in ARCHICAD 22.

Subtypen für Treppenlauf-unterstützende Auskragungen / Podest-unterstützende Auskragungen

ac_stairWallFixingWidthLeft Länge

Die Tiefe der Unterstützung in der Wand (auf der linken Seite der Treppe). Verwendet beim Zeichnen des 2D-Symbols der Unterstützung: Jede darüber hinausgehende Darstellung wird beschnitten. Kompatibilität: eingeführt in ARCHICAD 21.

ac_stairWallFixingWidthRight Länge

Die Tiefe der Unterstützung in der Wand (auf der rechten Seite der Treppe). Verwendet beim Zeichnen des 2D-Symbols der Unterstützung: Jede darüber hinausgehende Darstellung wird beschnitten. Kompatibilität: eingeführt in ARCHICAD 21.

Geländerbezogene Parameter

Subtyp für Geländer-Paneel-Komponenten

ac_panelThickness Länge

Stärke des Paneelabschnitts im Geländer. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen.

Subtyp für Geländer-Gurt-Komponenten

ac_railWidth Länge

Absolute Breite des Geländerabschnitts gemessen auf einer Ebene senkrecht zur Geländerachse. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen.

ac_railHeight Länge

Absolute Höhe des Geländerabschnitts gemessen auf einer Ebene senkrecht zur Geländerachse. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen.

ac_rail_boundingbox_left Länge

Relative Distanz der linken Seite der Geländerabschnitts-Begrenzungsbox, gemessen von der Geländerabschnittsachse (RAIL_POLYLINE_GEOMETRY), auf einer Ebene senkrecht zur Geländerachse, welche in Richtung des Geländers verläuft. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen. Wird für den Abschnitts-Begrenzungsbox-Offset der Komponentenachse verwendet.

ac_rail_boundingbox_right Länge

Relative Distanz von der rechten Seite der Geländerabschnitts-Begrenzungsbox, gemessen von der Geländerabschnittsachse (RAIL_POLYLINE_GEOMETRY), auf einer Ebene senkrecht zur Geländerachse, welche in Richtung des Geländers verläuft. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen. Wird für den Abschnitts-Begrenzungsbox-Offset der Komponentenachse verwendet.

ac_rail_boundingbox_top Länge

Relative Distanz der Oberseite der Geländerabschnitts-Begrenzungsbox, gemessen von der Geländerabschnittsachse (RAIL_POLYLINE_GEOMETRY), auf einer Ebene senkrecht zur Geländerachse, welche in Richtung des Geländers verläuft. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen. Wird für den Abschnitts-Begrenzungsbox-Offset der Komponentenachse verwendet.

ac_rail_boundingbox_bottom Länge

Relative Distanz der Unterseite der Geländerabschnitts-Begrenzungsbox, gemessen von der Geländerabschnittsachse (RAIL_POLYLINE_GEOMETRY), auf einer Ebene senkrecht zur Geländerachse, welche in Richtung des Geländers verläuft. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen. Wird für den Abschnitts-Begrenzungsbox-Offset der Komponentenachse verwendet.

ac_railProfileID ganzzahlig

Index des ausgewählten Profils, welches für das Geländer verwendet wird. ARCHICAD stellt die Begrenzungsboxgröße des Profils mit Hilfe von Abfrageoptionen zur Verfügung.

Subtyp für Geländerpfosten-Komponenten

ac_postWidth Länge

Absolute Breite des Pfostenabschnitts gemessen auf einer Ebene senkrecht zur Pfostenachse. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen.

ac_postHeight Länge

Absolute Höhe des Pfostenabschnitts gemessen auf einer Ebene senkrecht zur Pfostenachse. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen.

ac_post_boundingbox_left Länge

Relative Distanz der linken Seite der Pfostenabschnitts-Begrenzungsbox, gemessen von der Pfosten-Schnitt-Achse, auf einer Ebene senkrecht zur Pfostenachse; links bedeutet die Richtung des Anfangs der Bezugslinie, von innen betrachtet. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen. Wird für den Abschnitts-Begrenzungsbox-Offset der Komponentenachse verwendet.

ac_post_boundingbox_right Länge

Relative Distanz der rechten Seite der Pfostenabschnitts-Begrenzungsbox, gemessen von der Pfosten-Schnitt-Achse, auf einer Ebene senkrecht zur Pfostenachse; rechts bedeutet die Richtung des Endes der Bezugslinie, von innen betrachtet. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen. Wird für den Abschnitts-Begrenzungsbox-Offset der Komponentenachse verwendet.

ac_post_boundingbox_inside Länge

Relative Distanz der Pfostenabschnitts-Begrenzungsbox, welche horizontal von der Pfostenabschnittsachse gemessen wird, auf einer Ebene senkrecht zur Pfostenachse, innen bedeutet die Seite, auf welcher ein rechtsseitiger Handlauf platziert wird. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen. Wird für den Abschnitts-Begrenzungsbox-Offset der Komponentenachse verwendet.

ac_post_boundingbox_outside Länge

Relative Distanz der Pfostenabschnitts-Begrenzungsbox, welche horizontal von der Pfostenabschnittsachse gemessen wird, auf einer Ebene senkrecht zur Pfostenachse, außen bedeutet die gegenüberliegende Seite, auf welcher ein rechtsseitiger Handlauf platziert wird. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen. Wird für den Abschnitts-Begrenzungsbox-Offset der Komponentenachse verwendet.

ac_postProfileID ganzzahlig

Index des ausgewählten Profils, welches für den Pfosten verwendet wird. ARCHICAD stellt die Begrenzungsboxgröße des Profils mit Hilfe von Abfrageoptionen zur Verfügung.

Subtyp für Geländer-Abschlusskomponente

ac_railendWidth Länge

Absolute Breite des Geländer-Ende-Abschnitts gemessen auf einer Ebene senkrecht zur Geländer-Ende-Achse. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen.

ac_railendHeight Länge

Absolute Höhe des Geländer-Ende-Abschnitts gemessen auf einer Ebene senkrecht zur Geländer-Ende-Achse. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen.

ac_railend_boundingbox_left Länge

Relative Distanz der linken Seite der Geländerabschnitts-Begrenzungsbox, gemessen von der Geländer-Ende-Abschnittsachse, auf einer Ebene senkrecht zur Geländer-Ende-Achse, welche in Richtung des Geländer-Endes verläuft. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen. Wird für den Abschnitts-Begrenzungsbox-Offset der Komponentenachse verwendet.

ac_railend_boundingbox_right Länge

Relative Distanz der rechten Seite der Geländerabschnitts-Begrenzungsbox, gemessen von der Geländer-Ende-Abschnittsachse, auf einer Ebene senkrecht zur Geländer-Ende-Achse, welche in Richtung des Geländer-Endes verläuft. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen. Wird für den Abschnitts-Begrenzungsbox-Offset der Komponentenachse verwendet.

ac_railend_boundingbox_top Länge

Relative Distanz der Oberseite der Geländerabschnitts-Begrenzungsbox, gemessen von der Geländer-Ende-Abschnittsachse, auf einer Ebene senkrecht zur Geländer-Ende-Achse, welche in Richtung des Geländer-Endes verläuft. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen. Wird für den Abschnitts-Begrenzungsbox-Offset der Komponentenachse verwendet.

ac_railend_boundingbox_bottom Länge

Relative Distanz der Unterseite der Geländerabschnitts-Begrenzungsbox, gemessen von der Geländer-Ende-Abschnittsachse, auf einer Ebene senkrecht zur Geländer-Ende-Achse, welche in Richtung des Geländer-Endes verläuft. Der Parameterwert sollte über das Parameterskript mit der Modellgröße übereinstimmen. Wird für den Abschnitts-Begrenzungsbox-Offset der Komponentenachse verwendet.

ac_railendProfileID ganzzahlig

Index des ausgewählten Profils, welches für das Geländerende verwendet wird. ARCHICAD stellt die Begrenzungsboxgröße des Profils mit Hilfe von Abfrageoptionen zur Verfügung.

Parameter für Fassaden

Bibliothekselemente können mit ARCHICAD über Werte von Parametern mit vordefiniertem Namen und Funktion zusammenarbeiten. Die Liste dieser Parameter im Zusammenhang mit dem Fassadenwerkzeug folgt unten.

Fassaden-Parameter, welche ARCHICAD schreibt und liest

Fassadenprofil-Parameter

ac_frameWidth Länge

Die Breite des Fassadenprofils.

Kompatibilität: eingeführt in ARCHICAD 22.

ac_frameDepth Länge

Die Tiefe des Fassadenprofils.

Kompatibilität: eingeführt in ARCHICAD 22.

ac_frameBackDepth Länge

Der Rahmen des Fassadenprofils ist gegenüber der Mittelachse des Paneels versetzt.

Kompatibilität: eingeführt in ARCHICAD 22.

ac_clampWidth Länge

Die Breite des Fassaden-Paneel Einbau-Schlitzes.

Kompatibilität: vor ARCHICAD 22 konnte dieser Parameter nur durch ARCHICAD gesetzt werden.

ac_clampDepth Länge

Die Tiefe des Fassaden-Paneel Einbau-Schlitzes.

Kompatibilität: vor ARCHICAD 22 konnte dieser Parameter nur durch ARCHICAD gesetzt werden.

Fassaden-Parameter, welche ARCHICAD schreibt

Fassaden-Profil-Parameters

ac_nConnectingPanels Länge

Die Anzahl der Paneele, die mit dem Rahmenprofil verbunden sind. Bei Randprofilen ist dieser Wert 1, sonst 2.

ac_clampVector[2][2] Längenwert, Array

Die Richtungsvektoren der verbindenden Fassaden-Paneele. Der (ac_clampVector[1][1]; ac_clampVector[1][2]) Vektor enthält den (X; Y) Vektor des Paneels mit dem kleineren Rotationswinkel gemessen von der lokalen Y-Achse (gegen den Uhrzeigersinn, unter Berücksichtigung der positiven Richtung der lokalen X-Achse). Im Falle des Begrenzungsrahmens der (ac_clampVector[2][1]; ac_clampVector[2][2]) Vektor enthält die (X; Y) Richtung des dazugehörigen Paneels, die anderen Vektorwerte sind 0.

Parameter des Fassaden-Paneels

ac_panelCoords[][2] Längenwert, Array

Die X- und Y-Koordinaten des Polygons der Fassaden-Paneels werden in Bezug auf dem Einsetzspalt der Verbindungsrahmen im lokalen Koordinatensystem gemessen. Wenn ein solches Polygon nicht existiert, wird die erste Dimension dieses Parameters auf 1 gesetzt und erhält die Werte 0, 0.

Kompatibilität: In ARCHICAD 22 wird ein Fallback auf einen Knoten bei ungültigen Polygongeometrien eingeführt.

ac_clampFreeRegion[][2] Längenwert, Array

Die X- und Y-Koordinaten des Polygons des Fassadenpaneels werden auf der Seite der Verbindungsrahmen im lokalen Koordinatensystem gemessen. Wenn ein solches Polygon nicht existiert, wird die erste Dimension dieses Parameters auf 1 gesetzt und erhält die Werte 0, 0.

Kompatibilität: In ARCHICAD 22 wird ein Fallback auf einen Knoten bei ungültigen Polygongeometrien eingeführt.

ac_frameAxisCoords[][2] Längenwert, Array

Kompatibilität: eingeführt in ARCHICAD 22.

Die X- und Y-Koordinaten des Polygons des Fassadenpaneels, gemessen auf der Achse der Verbindungsrahmen im lokalen Koordinatensystem.

Parameter von Fassaden-Halterungen

ac_frameDirs[][3] Längenwert, Array

Die Endkoordinaten der verbindenden Fassadenprofilachsen.

ac_panelOffsets[] Längenwert, Array

Die Klemmdicken der verbindenden Fassadenpaneele.

Hinweis: Gesamtdicke des Paneels für Einbaupaneele.

ac_panelPresences[] Boolesches Array

Vorhandensein der verbindenden Fassadenpaneele.

Zubehörparameter von Fassaden

ac_frameWidthLeft Länge

Referenzrahmenkontur Breite 1 (normalerweise A/2)

ac_frameWidthRight Länge

Referenzrahmenkontur Breite 2 (normalerweise A/2)

ac_frameWidthFront Länge

Referenzrahmenkontur Höhe 1 (normalerweise B/2)

ac_frameWidthBack Länge

Referenzrahmenkontur Höhe 2 (normalerweise B/2)

ac_accessoryFlipped	Boolesche
Zubehör: Status der Umkehrung. 0 - nicht umgedreht, 1 - umgedreht	
ac_globalZDir[1][3]	Längenwert, Array
Vektor der lokalen Z-Achse im globalen Koordinatensystem	
ac_cellAngle1	Winkel
Der Rahmen des Zubehörs kann an max. 2 Zellen angeschlossen werden. Diese Zellen sind Zelle 1 und Zelle 2. Zelle1 ist die Zelle mit dem kleineren Drehwinkel, gemessen von der lokalen Y-Achse (gegen den Uhrzeigersinn, unter Berücksichtigung der positiven Richtung der lokalen X-Achse). Der Parameter ac_cellAngle1 ist der Winkel zwischen Zelle1 und lokaler Y-Achse.	
ac_cellAngle2	Winkel
Der Rahmen des Zubehörs kann an max. 2 Zellen angeschlossen werden. Diese Zellen sind Zelle 1 und Zelle 2. Zelle2 ist die Zelle mit dem größeren Drehwinkel gemessen von der lokalen Y-Achse (gegen den Uhrzeigersinn, unter Berücksichtigung der positiven Richtung der lokalen X-Achse). Der Parameter ac_cellAngle2 ist der Winkel zwischen Zelle2 und lokaler Y-Achse.	
ac_validCellAngle1	Boolesche
Definiert ob es Zelle 1 gibt oder nicht	
ac_validCellAngle2	Boolesche
Definiert ob es Zelle 2 gibt oder nicht	

Veraltete Parameter von Fassaden-Profilen

Diese Parameters funktionieren zwar noch in der ARCHICAD-Umgebung aus Kompatibilitätsgründen, aber wir empfehlen, diese bei neu erzeugten Objekten zu vermeiden.

gs_rightOffset	Länge
Der Abstand der Innenseite des Fassadenprofils von der Umrandung. Im Falle von "Außerhalb der Umrandung" ist der Wert die Profilbreite, im Falle von "Innerhalb der Umrandung" ist der Wert 0.	
gs_originOffsetX	Länge
Der Abstand der Außenseite des Fassadenprofils von der Umrandung. Im Falle von "Außerhalb der Umrandung" ist der Wert 0, im Falle von "Innerhalb der Umrandung" ist der Wert die Profilbreite.	
gs_frontOffset	Länge
Der Abstand der Vorderseite des Fassadenprofils von der Mittelachse des Paneels.	

gs_originOffsetY Länge

Der Abstand der Rückseite des Fassadenprofils von der Mittelachse des Paneels.

ac_topConnPlane[4] Längenwert, Array

Die oberste Beschneidungsposition des Profils, die im lokalen Koordinatensystem des Bibliothekselements definiert ist.
ac_topConnPlane[1]: X-Komponente des Normalvektors des oberen Cutplanes
ac_topConnPlane[2]: Y-Komponente des Normalvektors des oberen Cutplanes
ac_topConnPlane[3]: Z-Komponente des Normalvektors des oberen Cutplanes
ac_topConnPlane[4]: Abstand des oberen Cutplanes vom Profilursprung

ac_bottomConnplane[4] Längenwert, Array

Die unterste Beschneidungsposition des Profils, die im lokalen Koordinatensystem des Bibliothekselements definiert ist.
ac_bottomConnplane[1]: X-Komponente des Normalvektors des unteren Cutplanes
ac_bottomConnplane[2]: Y-Komponente des Normalvektors des unteren Cutplanes
ac_bottomConnplane[3]: Z-Komponente des Normalvektors des unteren Cutplanes
ac_bottomConnplane[4]: Abstand des unteren Cutplanes vom Profilursprung

Von ARCHICAD ausgelesene Fassaden-Parameter

Fassadenpaneel- und Fassadenprofil-Parameter

AC_AutoSchematicModel Boolesche

Steuert, ob die schematische Darstellung des Fassadenpaneels oder -Profils, die unter Modelldarstellung / Fassadenoptionen festgelegt wurde, von ARCHICAD übernommen wird.
Kompatibilität: eingeführt in ARCHICAD 22.

Parameter von Fassadenprofilen

ac_iCWFrameType ganzzahlig

Bestimmt den Typ des Profils

- *0 - Normales Profil*
- *1 - Diagonalecke*
- *2 - Reguläre Doppelecke*
- *3 - Reguläre Blockecke*
- *4 - Profiliert*

Der Wert dieses Parameters wirkt sich auf den Profiltyp und den Geometrie-Dialog aus, und im Fall der regulären Eckentypen auf die symbolische Darstellung des Profils.

Kompatibilität: eingeführt in ARCHICAD 22.

ac_bButtGlazedFrame Boolesche

Bestimmt den Typ des Profils

- *0 - Profil mit Kappe*
- *1 - flächenbündiger Rahmen*

Der Wert dieses Parameters wirkt sich auf den Profiltyp und den Geometrie-Dialog aus.

Kompatibilität: eingeführt in ARCHICAD 22.

ac_capProfileID Profil

Index des ausgewählten Profils, das für die Rahmenabdeckung des Fassadenelements verwendet wird. Wenn dieser Parameter und der Parameter ac_beamProfileID im Bibliothekselement des Fassadenprofils vorhanden sind, wird die Berechnung der Profil- und Kappengrößen von ARCHICAD übernommen.

Hinweis: Dieser Parameter wirkt sich nur auf ARCHICAD aus, wenn der Parameter ac_beamProfileID - siehe „Treppenrelevante Parameter" - existiert.

Kompatibilität: eingeführt in ARCHICAD 22.

ac_frameOffsetSide Länge

Der Abstand der Profilseite vom Ursprung des Profils gemessen auf der Mittelachse des Paneels. Die Klemmtiefe wird von diesem Abstand auf der Mittelachse des Paneels gemessen.

Kompatibilität: eingeführt in ARCHICAD 22.

ac_CWFrameBuildMat Baustoff

Der Index des Baustoffs des Profils. Die symbolische Darstellung des Profils wird mit den Schraffur-Attributen dieses Baustoffs gezeichnet.

Kompatibilität: eingeführt in ARCHICAD 22.

ac_CWFrameCutLinePen	Stift
Der Index des Schnittlinien-Stiftes des Profils. Die Konturlinie der symbolischen Darstellung des Profil wird mit dem eingestellten Stift-Index gezeichnet. Kompatibilität: eingeführt in ARCHICAD 22.	
ac_CWFrameCutLineType	Linientyp
Der Index des Schnittlinientyps des Profils. Die Konturlinie der symbolischen Darstellung des Profils wird mit dem eingestellten Linientyp-Index gezeichnet. Kompatibilität: eingeführt in ARCHICAD 22.	

Parameter von Fassaden-Paneelen

ac_panel_type	ganzzahlig
Bestimmt den Paneeltyp für Listen. 0 - fixiert, 1 - Tür, 2 - Fenster	
ac_openingDir	ganzzahlig
Bestimmt die Öffnungsrichtung von Türen und Fenstern für Listen. 0 - Fixiert, 1 - nach innen, 2 - nach außen	
ac_width	Länge
Paneelbreite für Listen.	
ac_nominalWidth	Länge
Nominale Paneelbreite für Listen.	
ac_height	Länge
Paneelhöhe für Listen.	
ac_nominalHeight	Länge
Nominale Paneelhöhe für Listen.	
ac_thickness	Länge
Paneelstärke für Listen.	
ac_originIsFrameCenter	Boolesche
*Falls der Parameter existiert und sein Wert **true** ist, liegt der Ursprung des Paneels am beginnenden (linken) Mittelpunkt des Profils. Ansonsten befindet sich der Ursprung am Startpunkt der linken Halterung.*	

ac_aSizeIsWithClamp	Boolesche
*Wenn der Parameter vorhanden ist und sein Wert **wahr**ist, setzt ARCHICAD die Größe **A** als den Abstand zwischen den Pfosten plus Haltergröße fest. Andernfalls wir die Größe **A** zwischen den Pfosten gemessen.*	

Parameter für Addons

Addons können Werte aus Bibliothekselementen auslesen mittels Parametern mit vordefinierten Namen und Funktionen. Die Liste mit solchen Parametern bezogen auf ARCHICAD package Addons folgt unten.

Parameter des Dachfenster-Addons

Lochkanten-Schneidemanipulation

ac_edge_lower_type	ganzzahlig
Schnitttyp der unteren Kante: 0 - Vertikal, 1 - Rechtwinklig, 2 - Horizontal, 3 - benutzerdefiniert	
ac_edge_lower_angle	Winkel
Schnittwinkel der unteren Kante, falls ac_edge_lower_type 3 ist. Der Wertebereich ist [1-179] Grad, 90 ist der rechtwinklige Fall.	
ac_edge_upper_type	ganzzahlig
Schnitttyp der oberen Kante: 0 - Vertikal, 1 - Rechtwinklig, 2 - Horizontal, 3 - benutzerdefiniert	
ac_edge_upper_angle	Winkel
Schnittwinkel der oberen Kante, falls ac_edge_upper_type 3 ist. Der Wertebereich ist [1-179] Grad, 90 ist der rechtwinklige Fall.	

Parameter des Eckfenster-Addons

Grundparameter des Eckfenster-Objektes

ac_cw_function	Boolesche
Fensterplatzierungs-Modus, kontrolliert vom Addon. 0 - Fenster, 1 - Eckfenster	
ac_corner_window	Boolesche
Eckfenster-Modus Auswahlschalter, kontrolliert vom Objekt. 0 - Deaktiviere Eckfenster-Modus, 1 - Aktiviere Eckfenster-Modus	

ac_corner_angle	Winkel
Winkel zwischen den verbundenen Wänden.	
ac_diff_con_wall_thk	Boolesche
Immer wahr (1). Das ist ein historisches Feature, welches anzeigt, ob die eingebundene Wand eine andere Stärke besitzt als die aufnehmende Wand.	
ac_con_wall_thk	Länge
Stärke der eingebundenen Wand.	
ac_cw_debug	Boolesche
Nur für internen Gebrauch. Aspekt, an dem GDL-Programmierer kein Interesse haben.	

Datenparameter von Wandschichten bei Eckfensterobjekten (verfügbar ab ARCHICAD 12)

ac_con_wall_skins_number	ganzzahlig
Anzahl der Schichten der angeschlossenen Wand. Im Fall von Massivwänden ist der Wert Null.	
ac_con_wall_skins_params	Länge
Parameter der angeschlossenen Schichten einer Mehrschichtwand. Gleich wie der globale Parameter WALL_SKINS_PARAMS der "Eigentümer"-Wand.	
ac_con_wall_direction_type	ganzzahlig
Spiegelstatus der angeschlossenen Wand; der gespiegelte Status der Wand, welches die Einstellung des Wandkörpers und der Referenzlinie bedeutet: 0 - nicht gespiegelt, 1 - gespiegelt. (alte Bedeutung: 0 - Rechts, 1 - Links, 2 - Mittig(Recht), 3 - Mittig(Links).)	

Parameter des IFC Addons

Gemeinsame Grundparameter der Tür-und Fensterobjekte

ifc_LiningDepth	Länge
Stärke des Tür/Fenster-Rahmens.	
ifc_LiningThickness	Länge
Breite des Tür/Fenster-Rahmens.	
ifc_TransomThickness	Länge
Breite des Kämpfers.	

IFC2x_ConstEnum Ganzzahl / String

Dieser Parameter definiert die Grundtypen der Konstruktion von Türen/Fenstern.

ifc_ConstEnum (Ganzzahl) parameter value	***IFC2x_ConstEnum (string) parameter value***	***IfcDoorStyleConstructionEnum category*** ***IfcWindowStyleConstructionEnum category***
0	*Nicht definiert*	*NOTDEFINED*
1	*Aluminium*	*ALUMINIUM*
2	*Edelstahl*	*HIGH_GRADE_STEEL*
3	*Stahl*	*STEEL*
4	*Holz*	*WOOD*
5	*Aluminium-Holz*	*ALUMINIUM_WOOD*
6	*Aluminium-Kunststoff*	*ALUMINIUM_PLASTIC*
7	*Kunststoff*	*PLASTIC*
8	*Nutzerdefiniert*	*USERDEFINED*

Grundparameter von Türobjekten

ifc_optype - Doors Ganzzahl / String

Türöffnungstyp, kontrolliert vom Makro IFC_optype_door.gsm.

ifc_optype (Ganzzahl) Parameter-Wert	***ifc_optypestr (String) Parameter-Wert***	***IfcDoorStyleOperationEnum Kategorie***
0	*Nicht definiert*	*NOTDEFINED*
1	*Einzel-Drehtür*	*SINGLE_SWING_LEFT* *SINGLE_SWING_RIGHT*
2	*Doppel-Drehtür*	*DOUBLE_DOOR_SINGLE_SWING*
3	*Einzel-Schwingtür*	*DOUBLE_SWING_LEFT* *DOUBLE_SWING_RIGHT*
4	*Doppel-Schwingtür*	*DOUBLE_DOOR_DOUBLE_SWING*
5	*Doppelschwingtür gegenläufig*	*DOUBLE_DOOR_SINGLE_SWING_OPPOSITE_LEFT* *DOUBLE_DOOR_SINGLE_SWING_OPPOSITE_RIGHT*
6	*Einzel-Schiebetür*	*SLIDING_TO_LEFT* *SLIDING_TO_RIGHT*
7	*Doppel-Schiebetür*	*DOUBLE_DOOR_SLIDING*
8	*Einzel-Falttür*	*FOLDING_TO_LEFT* *FOLDING_TO_RIGHT*
9	*Doppel-Falttür*	*DOUBLE_DOOR_FOLDING*
10	*Drehtür*	*REVOLVING*
11	*Rolltür/Rolltor*	*ROLLINGUP*
12	*Nutzerdefiniert*	*USERDEFINED*
13	*Fester Flügel*	*SWING_FIXED Kompatibilität: eingeführt in ARCHICAD 23.*

ifc_LiningOffset Länge

Offset des Türrahmens.

ifc_CasingDepth *Stärke des Türrahmens.*	Länge
ifc_CasingThickness *Breite des Türrahmens.*	Länge
ifc_ThresholdDepth *Tiefe der Türschwelle.*	Länge
ifc_ThresholdThickness *Stärke der Türschwelle.*	Länge
ifc_ThresholdOffset *Offset der Türschwelle.*	Länge
ifc_TransomOffset *Offset des Kämpfers.*	Länge

ifc_DoorPanel Länge - Array

ifc_DoorPanel[x][1] - Stärke des Türflügels.
ifc_DoorPanel[x][2] - Breite des Türflügels.

ifc_DoorPanel[x][3] Parameter-Wert	***IfcDoorPanelOperationEnum Kategorie***
0	*NOTDEFINED*
1	*SWINGING*
2	*DOUBLE_ACTING*
3	*SLIDING*
4	*FOLDING*
5	*REVOLVING*
6	*ROLLINGUP*
7	*USERDEFINED*

ifc_DoorPanel[x][4] Parameter-Wert	***IfcDoorPanelPositionEnum Kategorie***
0	*NOTDEFINED*
1	*LEFT*
2	*MIDDLE*
3	*RIGHT*

Grundparameter von Fenster-Objekten

ifc_optype - Windows Ganzzahl / String

Fensteröffnungstyp, kontrolliert vom Makro IFC_optype_window.gsm.

ifc_optype (Ganzzahl) Parameter-Wert	***ifc_optypestr (string) Parameter-Wert***	***IfcWindowStyleOperationEnum Kategorie***
0	*Nicht definiert*	*NOTDEFINED*
1	*Einflügelig*	*SINGLE_PANEL*
2	*Doppelflügel nebeneinander*	*DOUBLE_PANEL_VERTICAL*
3	*Doppelflügel übereinander*	*DOUBLE_PANEL_HORIZONTAL*
4	*Dreifachflügel nebeneinander*	*TRIPLE_PANEL_VERTICAL*
5	*Dreifachflügel übereinander*	*TRIPLE_PANEL_HORIZONTAL*
6	*2 Flügel nebeneinander, 3. Flügel darunter*	*TRIPLE_PANEL_BOTTOM*
7	*2 Flügel nebeneinander, 3. Flügel darüber*	*TRIPLE_PANEL_TOP*
8	*2 Flügel nebeneinander, 3. Flügel links*	*TRIPLE_PANEL_LEFT* *TRIPLE_PANEL_RIGHT*
9	*2 Flügel nebeneinander, 3. Flügel rechts*	*TRIPLE_PANEL_RIGHT* *TRIPLE_PANEL_RIGHT*
10	*Nutzerdefiniert*	*USERDEFINED*

ifc_MullionThickness ifc_MullionThickness - Länge

Sprossenbreite.

ifc_FirstMullionOffset ifc_FirstMullionOffset - Länge

Offset der Sprossenachse.

ifc_SecondMullionOffset ifc_SecondMullionOffset - Länge

Offset der Mittelachse der 2. Sprosse.

ifc_FirstTransomOffset *Offset der Kämpferachse.*	ifc_FirstTransomOffset - Länge
ifc_SecondTransomOffset *Offset der Kämpferachse von der 2. Sprosse.*	ifc_SecondTransomOffset - Länge

ifc_WindowPanel Länge - Array

ifc_WindowPanel[x][1] - Stärke des Fensterflügels.
ifc_WindowPanel[x][2] - Breite des Fensterflügels.

ifc_WindowPanel[x][3] Parameter-Wert	***IfcWindowPanelOperationEnum Kategorie***
0	*NOTDEFINED*
1	*SIDEHUNGRIGHTHAND*
2	*SIDEHUNGLEFTHAND*
3	*TILTANDTURNRIGHTHAND*
4	*TILTANDTURNLEFTHAND*
5	*TOPHUNG*
6	*BOTTOMHUNG*
7	*PIVOTHORIZONTAL*
8	*PIVOTVERTICAL*
9	*SLIDINGHORIZONTAL*
10	*SLIDINGVERTICAL*
11	*REMOVABLECASEMENT*
12	*FIXEDCASEMENT*
13	*OTHEROPERATION*

ifc_WindowPanel[x][4] Parameter-Wert	***IfcWindowPanelPositionEnum Kategorie***
0	*NOTDEFINED*
1	*LEFT*
2	*MIDDLE*
3	*RIGHT*
4	*BOTTOM*
5	*TOP*

Grundparameter von Transportelementen

ifc_optype - Transport Elements ganzzahlig

Typenauswahl Transportelemente. Kompatibilität: veraltet seit ARCHICAD 23.

ifc_optype (Ganzzahl) Parameter-Wert	*IfcTransportElementTypeEnum Kategorie*
0	*NOTDEFINED*
1	*ELEVATOR (Aufzug)*
2	*ESCALATOR (Roltreppe)*
3	*MOVINGWALKWAY (Rollband)*
4	*USERDEFINED*

Grundparameter von Aufzugs-Objekten

ifc_CapacityByWeight realnum

Kapazität des Transportelementes, nach Gewicht bemessen. Kompatibilität: veraltet seit ARCHICAD 23.

ifc_CapacityByNumber ganzzahlig

Kapazität des Transportelement in Anzahl der Personen gemessen. Kompatibilität: veraltet seit ARCHICAD 23.

Grundparameters von Treppenobjekten

ifc_StairType ganzzahlig

Die Grundkonfiguration des Treppentyps in Bezug auf die Anzahl der Treppenläufe und die Anzahl der Podeste, kontrolliert vom StairMaker-Addon für die eingebauten Treppen.

0 *Nicht definiert*

1 *StraightRunStair = gerader Lauf*

2 *TwoStraightRunStair = gerader Lauf mit 1 Zwischenpodest*

3 *QuarterWindingStair = 1/4 gewendelte Treppe mit Verzug*

4 *QuarterTurnStair = 1/4 gedrehte Treppe mit 1 Zwischenpodest*

5 *HalfWindingStair = 1/2 gewendelte Treppe (180°) mit Verzug*

6 *HalfTurnStair = 2-Läufige Treppe (180°) mit 1 Zwischenpodest*

7 *TwoQuarterWindingStair = 2 mal 1/4 gewendelt mit Verzug*

8 *TwoQuarterTurnStair = 2 mal 1/4 gedreht mit 2 Zwischenpodesten (3- läufig)*

9 *ThreeQuarterWindingStair = 3 mal 1/4 gewendelt (270°) mit Verzug*

10 *ThreeQuarterTurnStair = 3 mal 1/4 gedreht mit 3 Zwischenpodesten (4-läufig)*

11 *SpiralStair = gewendelte Treppe*

12 *DoubleReturnStair = T-förmiger Grundriss mit 1 Zwischenpodest und 3 Läufen*

13 *CurvedRunStair = gleichmäßig geschwungene Treppe (1 Lauf) ohne Knick*

14 *TwoCurvedRunStair = gleichmäßig geschwungene Treppe (2 Läufe) ohne Knick mit 1 Zwischenpodest*

15 *OtherOperation*

ifc_NumberOfRiser ganzzahlig

Gesamtanzahl der Steigungen der Treppe.

ifc_NumberOfTreads ganzzahlig

Gesamtanzahl der Laufflächen der Treppe.

ifc_RiserHeight Länge

Vertikaler Abstand von Auftritt zu Auftritt. Die Steigungshöhe sollte bei allen Stufen einer Treppe bzw. eines Treppenlaufes gleich sein.

ifc_TreadLength	Länge

Horizontaler Abstand von der Vorderkante eines Auftritts zur Vorderkante des nächsten. Die Auftrittslänge sollte bei allen Stufen einer Treppe bzw. eines Treppenlaufes auf der Lauflinie gleich sein.

Grundparameter von MEP-Elementen

ifc_subtype ganzzahlig

1 *IfcAirTerminalBoxType*
2 *IfcAirTerminalType*
3 *IfcAirToAirHeatRecoveryType*
4 *IfcBoilerType*
5 *IfcChillerType*
6 *IfcCoilType*
7 *IfcCompressorType*
8 *IfcCondenserType*
9 *IfcCooledBeamType*
10 *IfcCoolingTowerType*
11 *IfcDamperType*
12 *IfcDuctFittingType*
13 *IfcDuctSegmentType*
14 *IfcDuctSilencerType*
15 *IfcEvaporativeCoolerType*
16 *IfcEvaporatorType*
17 *IfcFanType*
18 *IfcFilterType*
19 *IfcFlowMeterType*
20 *IfcSpaceHeaterType*
21 *IfcHeatExchangerType*
22 *IfcHumidifierType*
23 *IfcPipeFittingType*
24 *IfcPipeSegmentType*
25 *IfcPumpType*
26 *IfcSpaceHeaterType*
27 *IfcTankType*
28 *IfcTubeBundleType*
29 *IfcUnitaryEquipmentType*
30 *IfcValveType*
31 *IfcVibrationIsolatorType*
32 *IfcFireSuppressionTerminalType*
33 *IfcSanitaryTerminalType*
34 *IfcStackTerminalType*
35 *IfcWasteTerminalType*
36 *IfcCableCarrierFittingType*
37 *IfcCableCarrierSegmentType*
38 *IfcCableSegmentType*
39 *IfcElectricApplianceType*
40 *dieser Wert wird übersprungen*
41 *IfcElectricFlowStorageDeviceType*
42 *IfcElectricGeneratorType*
43 *IfcSpaceHeaterType*
44 *IfcElectricMotorType*
45 *IfcElectricTimeControlType*
46 *dieser Wert wird übersprungen*
47 *IfcJunctionBoxType*
48 *IfcLampType*
49 *IfcLightFixtureType*
50 *IfcMotorConnectionType*
51 *IfcOutletType*
52 *IfcProtectiveDeviceType*
53 *IfcSwitchingDeviceType*
54 *IfcTransformerType*
55 *IfcActuatorType*
56 *IfcAlarmType*
57 *IfcControllerType*
58 *IfcFlowInstrumentType*
59 *IfcSensorType*
60 *IfcProtectiveDeviceTrippingUnitType*
61 *IfcUnitaryControlElementType*
62 *IfcBurnerType*
63 *IfcEngineType*
64 *IfcSolarDeviceType*
65 *IfcElectricDistributionBoardType*
66 *IfcCableFittingType*
67 *IfcAudioVisualApplianceType*
68 *IfcCommunicationsApplianceType*
69 *IfcMedicalDeviceType*
70 *IfcInterceptorType*

Kompatibilität: Änderungen in ARCHICAD 23

- *neue Typen 60 bis 70*
- *20 IfcGasTerminalType umbenannt in IfcSpaceHeaterType*
- *43 IfcElectricHeaterType umbenannt in IfcSpaceHeaterType*

Parameter für die Textverarbeitung

Vor ARCHICAD 22 waren ausgewählte Steuerelemente für Textstile für Beschriftungswerkzeuge (also Bibliothekselemente) auf verschiedene Einstellungsdialogfenster verteilt. Beginnend mit ARCHICAD 22 wurde ein Standardsatz von Textverarbeitungssteuerelementen in jedem Anmerkungsdialog in Form eines dedizierten "Text Stil" -Panels integriert.

Bibliothekselemente, die zu solchen Tools gehören, haben Zugriff auf diese Funktionen, indem sie die folgenden spezifische Parameter mit festem Namen verwenden.

Ab ARCHICAD 22 ist das Ausblenden/Sperren der entsprechenden ARCHICAD-Schnittstellensteuerelemente (falls vorhanden) eine Option unter Verwendung von den Befehlen `LOCK` und `HIDEPARAMETER` über das Parameter-Script des Bibliothekselementes, kombiniert mit der Einstellung "Ausblenden/Sperren spezifischer Parameter mit festem Namen aktivieren" (siehe Dialogfeld "Details / Kompatibilitätsoptionen" des Objekts im Editor für Bibliothekselemente)

Informationen zur Werkzeug- / Versionshistorie finden Sie in den Kompatibilitätshinweisen für die einzelnen Parameter.

Standardmäßig werden die folgenden Parameter vor ARCHICAD 22 als verfügbar und kompatibel betrachtet, während Ausnahmen separat aufgeführt werden.

AC_TextFont_1	Zeichenfolge
Name des aktuell ausgewählten / gespeicherten Schriftarttyps.	
Kompatibilität: eingeführt in ARCHICAD 22 für Etikett-Objekte. Die ähnliche Globale Variable "LABEL_FONT_NAME" gilt seither als veraltet.	
AC_TextSize_1	Natürliche Zahl
Aktuell festgelegter / gespeicherter Schriftgrößenwert in mm.	
Kompatibilität: eingeführt in ARCHICAD 22 für Etikett und Raumstempel-Objekte. Die ähnliche Globale Variable "LABEL_TEXT_SIZE" gilt in Etiketten als veraltet. In Raumstempelobjekten wird der spezifische Parameter mit festem Namen "ROOM_LSIZE" ebenfalls als veraltet betrachtet.	

AC_TextStyle_1 ganzzahlig

Schriftstil Wertemaskierung. Kann als direkter Eingabeparameter verwendet werden für `DEFINE STYLE{2}` *or* `UI_TEXTSTYLE_INFIELD`*.*

values:

`values = j1 + 2*j2 + 4*j3 + 128*j8`*, hierbei kann j jeweils 0 oder 1 sein.*

`j1:` *fett*

`j2:` *kursiv*

`j3:` *unterstrichen*

`j8:` *durchgestrichen*

Wenn AC_TextStyle_1 = 0, dann ist der Stil normal.

Kompatibilität: eingeführt in ARCHICAD 22 für Raumstempel- und Etikettenobjekte. Die ähnliche Globale Variable "LABEL_FONT_STYLE2" wird in Etiketten als veraltet angesehen.

AC_TextPen_1 ganzzahlig

Index des aktuell ausgewählten / gespeicherten Textstiftes.

Kompatibilität: eingeführt in ARCHICAD 22 für Raumstempel- und Etikettenobjekte. Die ähnliche Globale Variable "LABEL_TEXT_PEN" wird in Etiketten als veraltet angesehen.

Die folgenden Textverarbeitungsparameter sind für alle Bibliothekselemente vom Anmerkungstyp verfügbar. In Etiketten werden die entsprechenden Globalen Variablen als veraltet betrachtet. Siehe „Veraltete globale Etiketten-Variablen“ für die vollständige Liste der veralteten Etiketten-Globals.

AC_TextAlignment_1 ganzzahlig

Textausrichtung.

1 - linksbündig, 2 - zentriert, 3 - rechtsbündig, 4 - Blocksatz

Kompatibilität: eingeführt in ARCHICAD 22. In Etiketten wird die ähnliche Globale Variable "LABEL_TEXT_ALIGN" seit ARCHICAD 22 als veraltet angesehen.

AC_TextLeading_1 Natürliche Zahl

Zeilenabstands-Faktor des Textes.

Kompatibilität: eingeführt in ARCHICAD 22. In Etiketten wird die ähnliche Globale Variable "LABEL_TEXT_LEADING" seit ARCHICAD 22 als veraltet betrachtet.

AC_TextCharWidthFactor_1 Natürliche Zahl

Zeichenbreitenfaktor des Textes.

Kompatibilität: eingeführt in ARCHICAD 22. In Etikette wird die ähnliche Globale Variable "LABEL_TEXT_WIDTH_FACT" seit ARCHICAD 22 als veraltet angesehen.

AC_TextCharSpaceFactor_1	Natürliche Zahl

Zeichenabstands-Faktor des Textes.

Kompatibilität: eingeführt in ARCHICAD 22. In Etiketten wird die ähnliche Globale Variable "LABEL_TEXT_CHARSPACE_FACT" seit ARCHICAD 22 als veraltet betrachtet.

Parameter für Etiketten

Parameter, welche von ARCHICAD gelesen oder geschrieben werden

Der folgende Set von Etiketten-Parametern wurde komplett in ARCHICAD 22 neu eingeführt, so dass die parallelen Globalen Variablen als veraltet betrachtet werden. Siehe „Veraltete globale Etiketten-Variablen“ für die vollständige Liste der veralteten Etiketten-Globals.

Ab ARCHICAD 22 ist das Ausblenden/Sperren der entsprechenden ARCHICAD-Schnittstellensteuerelemente (falls vorhanden) eine Option unter Verwendung von den Befehlen `LOCK` und `HIDEPARAMETER` über das Parameter-Script des Bibliothekselementes, kombiniert mit der Einstellung "Ausblenden/Sperren spezifischer Parameter mit festem Namen aktivieren" (siehe Dialogfeld "Details / Kompatibilitätsoptionen" des Objekts im Editor für Bibliothekselemente)

AC_bLabelAlwaysReadable	Boolesche

1 - "Immer lesbar" ist im Paneel "Etiketteneinstellungen/Textstil" aktiviert, ansonsten 0

Kompatibilität: eingeführt in ARCHICAD 22. Die ähnliche Globale Variable "LABEL_ALWAYS_READABLE" gilt seit ARCHICAD 22 als veraltet.

AC_bLabelTextWrap	Boolesche

1 - "Zeilenumbruch" ist im Paneel "Etiketteneinstellungen/Textstil" markiert, ansonsten 0

Kompatibilität: eingeführt in ARCHICAD 22. Die ähnliche Globale Variable "LABEL_TEXT_WRAP" gilt seit ARCHICAD 22 als veraltet.

AC_bLabelOpaqueFill	Boolesche

1 - "Deckend" ist im Paneel "Etiketteneinstellungen/Textstil" markiert, ansonsten 0

Kompatibilität: eingeführt in ARCHICAD 22. Die ähnliche Globale Variable "LABEL_TEXT_BG_PEN" gilt seit ARCHICAD 22 als veraltet.

AC_LabelTextBgrPen	ganzzahlig

Enthält den Index des ausgewählten Text-Schraffurstiftes.

Kompatibilität: eingeführt in ARCHICAD 22. Die ähnliche Globale Variable "LABEL_TEXT_BG_PEN" gilt seit ARCHICAD 22 als veraltet.

AC_bLabelFrame	Boolesche

1 - "Frame" ist angekreuzt, 0 - "Frame" ist nicht angekreuzt.

Kompatibilität: eingeführt in ARCHICAD 22. Die ähnliche Globale Variable "LABEL_FRAME_ON" gilt seit ARCHICAD 22 als veraltet.

AC_LabelFrameOffset Länge

Rahmen-Abstand: Offset-Wert entsprechend dem Paneel "Etiketteneinstellungen/Textstil"

Kompatibilität: eingeführt in ARCHICAD 22. Die ähnliche Globale Variable "LABEL_FRAME_OFFSET" gilt seit ARCHICAD 22 als veraltet.

AC_LabelPointerPen Stift

Enthält den Index des ausgewählten Zeigerlinienstiftes.

Kompatibilität: eingeführt in ARCHICAD 22. Die ähnliche Globale Variable "LABEL_ARROW_PEN" gilt seit ARCHICAD 22 als veraltet.

AC_LabelPointerLineType Linientyp

Enthält den Index des ausgewählten Zeiger-Linientyps.

Kompatibilität: eingeführt in ARCHICAD 22. Die ähnliche Globale Variable "LABEL_ARROW_LINETYPE" gilt seit ARCHICAD 22 als veraltet.

AC_LabelPointerConnection ganzzahlig

Zeiger: Anbindungspostion.

0 - Mitte, 1 - oben, 2 - unten, 3 - unten rechts

Kompatibilität: eingeführt in ARCHICAD 22. Die ähnliche Globale Variable "LABEL_ANCHOR_POS" gilt seit ARCHICAD 22 als veraltet.

AC_LabelOrientation ganzzahlig

Der Wert für den Etikettenausrichtungstyp der "Etikettenausrichtung" -Einstellungen im Paneel "Symboletikett". Der Wert "LABEL_ROTANGLE" kann sich je nach den Einstellungen für die Ausrichtung ändern.

1 - parallel, 2 - senkrecht, 3 - vertikal, 4 - horizontal, 5 - eigener Winkel

Kompatibilität: eingeführt in ARCHICAD 22.

Die erweiterte LOCK / HIDEPARAMETER-Funktion ist für diesen Parameter nicht verfügbar. Die Maskierung der entsprechenden Interface-Steuerwerte wird durch "AC_DisableLabelOrientationVal" gesetzt. Der Parameter definiert den Standardwert des Bibliothekselementes.

Parameter von ARCHICAD ausgelesen

AC_DisableLabelOrientationVal ganzzahlig

Etikettenausrichtung ("AC_LabelOrientation") Typ Mask-Wert.

`bitset:` *Deaktivieren Sie die folgenden Optionen im Dialogfeld "Symbolbezeichnung" für "Etikettenausrichtung"":*

`bitset = j1 + 2*j2 + 4*j3 + 8*j4`*, hierbei kann j jeweils 0 oder 1 sein.*

`j1:` *"Parallel",*

`j2:` *"Senkrecht",*

`j3:` *"Vertikal",*

`j4:` *"Horizontal".*

"Eigener WInkel" kann nicht deaktiviert werden.

Kompatibilität: eingeführt in ARCHICAD 22.

---

ac_bDisableLabelFrameDisplay Boolesche

Kompatibilität: eingeführt in ARCHICAD 20.

Verbirgt den eingebauten rechteckigen Rahmen, welcher um das Etikett-Symbol herum angeordnet wird, für den Fall, dass der eingebaute Zeiger und Rahmen aktiviert sind, ermöglicht dem Anwender, einen benutzerdefinierten Rahmen zu programmieren.

---

ac_bCustomPointerConnection Boolesche

Kompatibilität: eingeführt in ARCHICAD 20.

Kontrolliert die automatische Zeiger-Verbindung des Etikett-Symbols, für den Fall, dass der eingebaute Zeiger aktiviert ist. Falls dieser Parameter auf EIN gestellt ist, können 6 Hotspots im 2D-Script für die benutzerdefinierte Zeigerverbindung in Übereinstimmung mit den integrierten Typen definiert werden. Diese Hotspots sollten feste IDs von 1 bis 6 besitzen. Die IDs zeigen folgende Verbindungspositionen an:

Falls der Zeiger auf der linken Seite des Etikettsymbols liegt:

- *1: linke obere Verbindung*
- *3: linke mittlere Verbindung*
- *5: linke untere Verbindung*
- *6: rechte untere Verbindung*

Falls der Zeiger auf der rechten Seite des Etikettsymbols liegt:

- *2: rechte obere Verbindung*
- *4: rechte mittlere Verbindung*
- *6: rechte untere Verbindung*
- *5: linke untere Verbindung*

Veraltete Parameter

Veraltete Träger- / Stützenparameter - nur für Auflistung und Etiketten verfügbar

Ab ARCHICAD 23 sind diese Werte in den Globalen Variablen BEAM_SEGMENT_INFO und COLU_SEGMENT_INFO mit einheitlichen Wertereferenzen verfügbar. Siehe auch „Veraltete Globale Träger- / Stützenvariablen - nur für Listen und Etiketten verfügbar“. Aus Kompatibilitätsgründen sind sie weiterhin auf homogenen, geraden oder horizontal gekrümmten Trägern und auf homogenen Stützen (GLOB_ELEM_TYPE = 12 oder 6) verfügbar.

ac_colu_crosssection_type Querschnittstyp der der Stütze (Ganzzahl)

⛔ *Stützen-Segment (27)*	✅ *Einzel-Segment-Stütze(6)*	⛔ *Multi-Segment-Stütze(6)*

1 - rechteckig, 2 - rund, 3 - komplexes Profil

ac_beam_crosssection_type Querschnittstyp des Trägers (Ganzzahl)

⛔ *Träger-Segment (28)*	✅ *Einzel-Segment-Träger(12)*	⛔ *Multi-Segment-Träger (12)*

1 - rechteckig, 2 - komplexes Profil

Veraltete Raumstempel-Parameter

ROOM_LSIZE Natürliche Zahl

Schriftgröße des Textes in mm.

Kompatibilität: veraltet seit ARCHICAD 22. Verwenden Sie stattdessen `AC_TextSize_1`.

ac_disable_controls ganzzahlig

Dieser Parameter kann die Sichtbarkeit des Eingabefeldes für die Schriftgröße ("ROOM_LSIZE") des Dialogfelds "Raum-Grundeinstellung" steuern: 0 oder das Objekt besitzt diesen Parameter nicht - Textgröße wird angezeigt, 1 - Textgröße wird nicht angezeigt (ermöglicht so mehr Platz für die Parameterliste)

Kompatibilität: inaktiv seit ARCHICAD 22. Das Eingabesteuerelement "Zeichensatz-Größe" wurde in das Paneel "Stempel Textstil" verschoben, seine Sichtbarkeit wird durch die Befehle LOCK / HIDEPARAMETER gesteuert.

REQUEST Optionen

```
n = REQUEST (question_name, name_or_index, variable1 [, variable2, ...])
n = REQUEST{2} (question_name, name_or_index, name, variable1 [, variable2, ...])
```

```
n = REQUEST{3} (question_name, name, name_or_index_array, variable1 [, variable2, ...])
n = REQUEST{4} (question_name, name_or_index, index, name, variable1 [, variable2, ...])
```

Der erste Parameter stellt den REQUEST-Abfrage-String dar, während der zweite (oder weitere) das Inhalt der Abfrage repräsentiert (falls vorhanden). Die anderen Parameter sind Variablennamen, in denen die Rückgabewerte (die Antworten) gespeichert werden. Das Ergebnis der Funktion ist die Anzahl der Antworten (eine falsch gestellte Frage oder ein nicht existierender Name, ergibt den Wert 0).

ARCHICAD identifiziert die Reihenfolge und die Anzahl der Eingabeparameter entweder durch die Version des `REQUEST`Befehls, oder über den genauen Namen (als String-Konstante) der REQUEST-Option. Dies bedeutet, dass die Verwendung der ersten oder beider der folgenden Optionen die sicherste ist:

- Name der REQUEST-Abfrage ist immer eine konstante Zeichenfolge
- die Version wird dem Befehl hinzugefügt

Request Parameter-Script Kompatibilität

Die Verwendung der meisten REQUEST-Befehle in Parameter-Scripten (oder als Parameter-Scripten ausgeführten Master-Scripten) kann zu instabil zurückgegebenen Werten führen, weshalb dies vermieden werden sollte.

Kompatibilität bis zu ARCHICAD 19: Die Verwendung der meisten REQUESTs im Parameter-Scripten (oder als Parameter-Script ausgeführten Master-Scripten) kann zu unzuverlässigen Rückgabewerten führen.

Kompatibilität beginnend mit ARCHICAD 20: das folgende gilt im Falle von Parameter-Scripten:

- *der REQUEST-Ausdruck hat immer den Wert 0 als erfolgreichen Rückgabewert*
- *die abgefragten Werte enthalten ausschließlich einen zum Typ passenden Defaultwert (Leerstring oder 0)*

Bei eingeschränkten Anforderungen im Parameter-Script wird es auch GDL-Warnungen geben, beginnend mit ARCHICAD 19.

Zum Überprüfen der Kompatibilität des Parameter-Scripts wird auf die folgende Tabelle hingewiesen.:

Legende:

✅	funktioniert ohne Einschränkungen
⚠	funktioniert (mit ergänzenden Warnungen)
⛔	funktioniert nicht: Ausdruck liefert 0 zurück, wobei zum Typ passende Dummy-Standardwerte in Rückgabewerten (Leerstring oder 0) enthalten sind - mit ergänzenden Warnungen

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
ANCESTRY_INFO	✓	⚠	⛔
ANGULAR_DIMENSION	✓	⚠	⛔
ANGULAR_LENGTH_DIMENSION	✓	⚠	⛔
AREA_DIMENSION	✓	✓	✓
ASSOCEL_PROPERTIES	✓	⚠	⛔
ASSOCLP_NAME	✓	⚠	⛔
ASSOCLP_PARVALUE	✓	✓	✓
ASSOCLP_PARVALUE_WITH_DESCRIPTION	✓	✓	✓
AUTOTEXT_LIST	-	-	⛔

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
BUILDING_MATERIAL_INFO	✓	⚠	⛔

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
CALC_ANGLE_UNIT	✅	⚠	⛔
CALC_AREA_UNIT	✅	⚠	⛔
CALC_LENGTH_UNIT	✅	✅	✅
CALC_VOLUME_UNIT	✅	✅	✅
CLASS_OF_FILL	✅	✅	✅
CLEAN_INTERSECTIONS	✅	⚠	⛔
COMPONENT_PROJECTED_AREA	✅	⚠	⛔
COMPONENT_VOLUME	✅	⚠	⛔
CONFIGURATION_NUMBER	-	-	⛔
CONSTR_FILLS_DISPLAY	✅	⚠	⛔
CUSTOM_AUTO_LABEL	✅	⚠	⛔

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
DATETIME	✅	⚠	⚠
DOOR_SHOW_DIM	✅	⚠	⛔

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
ELEVATION_DIMENSION	✅	⚠	⛔

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
FLOOR_PLAN_OPTION	✅	⚠️	⛔
FONTNAMES_LIST	✅	✅	✅
FULL_ID_OF_PARENT	-	⚠️	⛔

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
HEIGHT_OF_STYLE	✅	✅	✅
HOME_STORY	✅	✅	✅
HOME_STORY_OF_OPENING	✅	✅	✅
HOMEDB_INFO	✅	✅	✅

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
ID_OF_MAIN	✅	⚠️	⛔
INTERNAL_ID	✅	✅	✅

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
LAYOUT_LENGTH_UNIT	✅	⚠️	⚠️
LAYOUT_TEXT_SIZE_UNIT	✅	✅	✅
LEVEL_DIMENSION	✅	⚠️	⛔
LINEAR_DIMENSION	✅	✅	✅

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
MATCHING_PROPERTIES	✅	⚠️	⛔
MATERIAL_INFO	✅	⚠️	⛔
MODEL_LENGTH_UNIT	✅	⚠️	⚠️
MODEL_TEXT_SIZE_UNIT	✅	⚠️	⛔

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
NAME_OF_FILL	✅	⚠️	⛔
NAME_OF_LINE_TYPE	✅	⚠️	⛔
NAME_OF_LISTED	✅	⚠️	⛔
NAME_OF_MACRO	✅	✅	✅
NAME_OF_MAIN	✅	✅	✅
NAME_OF_MATERIAL	✅	⚠️	⛔
NAME_OF_PLAN	✅	⚠️	⛔
NAME_OF_PROGRAM	✅	⚠️	⛔
NAME_OF_STYLE	✅	⚠️	⛔

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
PEN_OF_RGB	✅	⚠️	⛔
PROGRAM_INFO	✅	✅	✅
PROPERTIES_OF_PARENT	-	-	⛔
PROPERTY_NAME	-	-	⛔
PROPERTY_VALUE_OF_PARENT	-	-	⛔

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
RADIAL_DIMENSION	✅	⚠️	⛔
REFERENCE_LEVEL_DATA	✅	✅	✅
RGB_OF_MATERIAL	✅	⚠️	⛔
RGB_OF_PEN	✅	⚠️	⛔

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
SILL_HEIGHT_DIMENSION	✅	✅	✅
STORY	✅	⚠️	⛔
STORY_INFO	✅	✅	✅
STYLE_INFO	✅	⚠️	⚠️
SUM_WITH_ROUNDING	-	-	✅

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
TEXTBLOCK_INFO	✅	⚠	⛔

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
VIEW_ROTANGLE	✅	✅	✅
WINDOW_DOOR_DIMENSION	✅	✅	✅
WINDOW_DOOR_SHOW_DIM	✅	⚠	⛔
WINDOW_DOOR_ZONE_RELEV	✅	⚠	⚠
WINDOW_DOOR_ZONE_RELEV_OF_OWNER	✅	⚠	⚠
WINDOW_SHOW_DIM	✅	⚠	⛔
WORKING_ANGLE_UNIT	✅	⚠	⚠
WORKING_LENGTH_UNIT	✅	✅	✅

Kompatibilität im Parameter-Script	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
ZONE_CATEGORY	✅	✅	✅
ZONE_COLUS_AREA	✅	⚠	⛔
ZONE_RELATIONS	✅	✅	✅
ZONE_RELATIONS_OF_OWNER	✅	⚠	⚠

Details der Requests

```
n = REQUEST ("Name_of_program", "", program_name)
```

Gibt in der ausgegebenen Variable den Namen des Programms zurück, z.B. "ARCHICAD". *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

Beispiel 1: Drucken des Programmnamens

```
n=REQUEST(Name_of_program, "", program_name) PRINT program_name
```

```
n = REQUEST ("Name_of_macro", "", my_name)
n = REQUEST ("Name_of_main", "", main_name)
```

Nach der Ausführung dieser Funktionsaufrufe wird die Variable my_name den Namen des Makros enthalten, während die Variable main_name den Namen des übergerodeten Makros enthält (falls keines existiert wird ein leerer Text übergeben).

```
n = REQUEST ("ID_of_main", "", id_string)
```

Für Bibliothekselemente, die im Grundriss platziert wurden, wird in der Variablen id_string die ID aus dem Einstellungsdialog des übergeordneten Objektes übergeben (anderenfalls leere Zeichenfolge). Funktioniert nicht bei Auszeichnungselementen (z.B. Etiketten, Tür/Fenstermarker, Raumstempel). *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

```
n = REQUEST ("Full_ID_of_parent", "", id_string)
```

Bei Auszeichnungselementen, welche im Grundriss verlinkt oder verhotlinkt sind, werden alle Identifizierer (Master ID) der verlinkten Module und ihrer Eltern-Bibliothekselemente in der Werkzeugeinstellungs-Dialogbox in der id_string-Variable eingestellt (andernfalls Leerstring). *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

```
n = REQUEST ("Name_of_plan", "", name)
```

Gibt den Namen des aktuellen Projektes in der Variablen name zurück. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

```
n = REQUEST ("Story", "", index, story_name)
```

Ergibt den Index und Namen des aktuellen Geschosses in der Variable index und story_name zurück. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

```
n = REQUEST ("Home_story", "", index, story_name)
```

Gibt den Index und Namen des Einsetzgeschosses in der Variable index und story_name zurück.

```
n = REQUEST ("Home_story_of_opening", "", index, story_name)
```

Gibt den Index und Namen des Einsetzgeschosses der Öffnung zurück an die Variablen index und story_name. Das Einsetzgeschoss ist das erste Geschoss, auf welchem die Öffnung sichtbar ist. Kann in den Scripten von Fenstern, Türen, Wandenden,Eckfenstern und Dachfenstern verwendet werden und in den Scripten der dazugehörigen Marker und Etiketten. Verursacht Warnung wenn im Parameter-Script verwendet.

```
n = 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 2:

```
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
```

Mit den folgenden REQUESTs können Sie die in den Dialogfeldern Optionen / Projektpräferenzen / Bemaßungseinstellungen und Berechnungseinheiten und -regeln festgelegten Dimensionsformate kennenlernen. Diese REQUESTs ergeben einen Formattext, der als erster Parameter in der STR ()- Funktion verwendet werden kann.

```
n = REQUEST ("Linear_dimension", "", format_string)
n = REQUEST ("Angular_dimension", "", format_string)
```

Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.

```
n = REQUEST ("Angular_length_dimension", "", format_string)
```

Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.

```
n = REQUEST ("Radial_dimension", "", format_string)
```

Verursacht Warnung wenn im Parameter-Script verwendet.

```
n = REQUEST ("Level_dimension", "", format_string)
```

Verursacht Warnung wenn im Parameter-Script verwendet.

```
n = REQUEST ("Elevation_dimension", "", format_string)
```

Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.

```
n = REQUEST ("Window_door_dimension", "", format_string)
```

```
n = REQUEST ("Sill_height_dimension", "", format_string)
n = REQUEST ("Area_dimension", "", format_string)
n = REQUEST ("Calc_length_unit", "", format_string)
n = REQUEST ("Calc_area_unit", "", format_string)
```

Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.

```
n = REQUEST ("Calc_volume_unit", "", format_string)
n = REQUEST ("Calc_angle_unit", "", format_string)
```

Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.

Beispiel 3:

```
format = "" num = 60.55
n = REQUEST ("Angular_dimension", "",format)!"%.2dd"
TEXT2 0, 0, STR (format, num)!60.55
```

```
n = REQUEST ("Clean_intersections", "", state)
```

Gibt den Status der sauber verschnittenen Wand&Trägers-Verschneidungsfeatures zurück (1 falls eingeschaltet, 0 falls ausgeschaltet) *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

```
n = REQUEST ("Zone_category", "", name, code)
```

Für Räume, ergibt den Namen und den Code der aktuellen Raum-Kategorie.

```
n = 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 platziert wird. Für Türen und Fenster können maximal zwei Räume bestimmt werden. Ergibt die Anzahl der erfolgreich eingelesenen Werte (0, wenn das Bibliothekselement nicht innerhalb eines Raumes ist).

```
n = REQUEST ("Zone_relations_of_owner", "",
         category_name, code, name, number
         [, category_name2, code2, name2, number2])
```

Ergibt den Namen & Code der Raumkategorie, und Namen & Nummer des Raumes, in dem der Eigentümer des Objektes platziert wird. Es ist also dann wichtig, wenn der Bibliothekselement einen Besitzer hat (Tür-Fenster Etiketten und Tür-Fenster-Marker- usw). Im Falle eines Türetikettes ist die Tür der Eigentümer. Für Türen und Fenster können maximal zwei zusammenhängende Räume bestimmt werden. Der Ergebniswert der REQUEST-Funktion ist die Anzahl der erfolgreich abgerufenen Werte (0 wenn das Objekt keinen Eigentümer hat oder sich sein Eigentümer innerhalb keines Raumes befindet.) Verursacht Warnung wenn im Parameter-Script verwendet.

n = REQUEST ("Zone_colus_area", "", area)

Ergibt die gesamte Fläche der im aktuellen Raum platzierten Stützen in der Variable area. Nur bei Raumstempeln wirksam. Nur aus Gründen der Kompatibilität verfügbar. Es wird empfohlen, dass bei den Fix-Parametern der Raumstempel die Mengen eingestellt sind. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

n = REQUEST ("Custom_auto_label", "", name)

Ergibt den Namen des benutzerdefinierten, automatischen Etiketts des Bibliothekselementes oder eine leere Zeichenfolge, falls es nicht zugewiesen wurde. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

n = REQUEST ("Rgb_of_material", name, r, g, b)
n = REQUEST ("Rgb_of_pen", penindex, r, g, b)
n = REQUEST ("Pen_of_RGB", "r g b", penindex)

Wie die REQ()-Funktion, ergibt der Wert die r, g, b Bestandteile des Materials oder Stiftes oder den Index des Stiftes (in nur einem einzigen Aufruf) mit den angegebenen r, g, b Werten. *Alle 3 Ausdrücke geben 0 zurück und enthalten Dummyrückgabewerte (Leerstring oder 0) falls im Parameter-Script verwendet, verursachen außerdem einen Warnmeldung.*

n = REQUEST ("Height_of_style", name, height [, descent, leading])

Ergibt in den gegebenen Variablen die Gesamthöhe (height) des gemessenen Schriftstils in Millimetern (Höhe in Metern entspricht der Höhe / 1000 * GLOB_SCALE); die Unterlänge (descent) (Abstand in Millimetern von der Text-Grundlinie bis zur Unterlängenlinie) und den Zeilenabstand (leading) (Abstand in Millimetern von der Unterlängenlinie bis zur Oberlängenlinie).

n = 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*). Kann in Makros hilfreich sein zum Erfassen von Informationen zu dem im Haupt-Script definierten Stil. Verursacht Warnung wenn im Parameter-Script verwendet.

n = REQUEST ("Name_of_material", index, name)

Ergibt den Namen des durch index identifizierten Materials in der Variablen name. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

n = REQUEST ("Name_of_building_material", index, name)

Gibt in der Variablen den durch den Index identifizierten Baustoff-Namen zurück. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

n = REQUEST ("Name_of_fill", index, name)

Ergibt den Namen der durch Index identifizierten Schraffur in der Variablen name. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

n = REQUEST ("Name_of_line_type", index, name)

Ergibt den Durch den Index angegebenen Liniennamen in der Variablen name. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

n = REQUEST ("Name_of_style", index, name)

Ergibt den Namen der durch Index identifizierten Schriftstil in der Variablen name. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

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 Fehler auftrat oder 0 im Fall eines Fehlers, wenn der Index ungültig ist.

n = REQUEST ("WINDOW_DOOR_SHOW_DIM", "", show)

Vor ARCHICAD 9.0 wird 1 in der Variable show übergeben, wenn unter Optionen/Reinzeichnungeinstellungen/Türen & Fenster die Option "Mit Bemaßungen anzeigen" gewählt ist, andernfalls ist es Null. Seit ARCHICAD 9.0 sind die Darstellungsoptionen für Fenster und Türen aufgeteilt. Aus Kompatibilitätsgründen prüft das Skript ob der REQUEST in einem Fenster oder Fenstermarker oder aber in einer Tür oder Türmarker verwendet wird und gibt automatisch den korrekten Wert zurück. In anderen Objekten (Symbol, Leuchte, Etikett) wird die Einstellung für Fenster zurückgegeben. Es kann zum Verbergen/Anzeigen von benutzerdefinierten Bemaßungen verwendet werden, entsprechend den aktuellen Reinzeichnungseinstellungen. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

Seit ARCHICAD 9.0 gibt es separate Funktionen "window_show_dim", und "door_show_dim".

n = REQUEST ("window_show_dim", "", show)

Es wird eine 1 in der Variable show übergeben, wenn unter Darstellung (Modell) /Fensteroptionen "Mit Bemaßungen anzeigen" gewählt ist, andernfalls ist es Null. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

n = REQUEST ("door_show_dim", "", show)

Es wird eine 1 in der Variable show übergeben, wenn unter Darstellung (Modell) /Türoptionen "Mit Bemaßungen anzeigen" gewählt ist, andernfalls ist es Null. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

n = REQUEST ("name_of_listed", "", name)

Ergibt in der Variablen name den Namen des Bibliothekselementes, das mit demjenigen Bibliothekselement vom Eigenschaftentyp verbunden ist, das diesen REQUEST enthält. Für Elemente (Wände, Decken, etc.) ist der Name eine leere Zeichenfolge. Verursacht Warnung wenn im Parameter-Script verwendet.

n = 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". Gibt den Wert 1 in der Variable out_direction zurück, wenn die Öffnungsrichtung von Tür/Fenster in den ersten Raum zeigt, welcher von der "zone_relations" Abfrage identifiziert wird, 2, wenn die Öffnungsrichtung in den zweiten Raum zeigt. Dieser Wert wird auch dann 2, falls nur ein Raum vorhanden ist und die Öffnungsrichtung nach außen zeigt. Verursacht Warnung wenn im Parameter-Script verwendet.

```
n = REQUEST ("window_door_zone_relev_of_owner", "", out_direction)
```

Nur dann wirksam, wenn das Mutterelement des Bibliothekselementes eine Tür oder ein Fenster (Marker, Etikett) ist. Verwenden Sie es als Ergänzung zur "zone_relations_of_owner"-Abfrage. Übergibt 1 an die Variable out_direction, falls sich die Richtung der Tür-/Fensteröffnung in dem ersten Raum befindet, der durch die REQUESTs des Typs "zone_relations" identifiziert wird, der Wert ist 2 falls die Öffnungsrichtung in den zweiten Raum zeigt. Dieser Wert wird auch dann 2, falls nur ein Raum vorhanden ist und die Öffnungsrichtung nach außen zeigt. Verursacht Warnung wenn im Parameter-Script verwendet.

```
n = REQUEST ("matching_properties", type, name1, name2, ...)
```

Wenn typ = 1, werden in den vorgegebenen Variablen individuell assoziierte Namen von Eigenschafts-Bibliothekselementen übergeben, sonst Namen von Eigenschafts-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. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

```
n = REQUEST ("Working_length_unit", "", format_string)
n = REQUEST ("Working_angle_unit", "", format_string)
```

Durch diese REQUESTs ermittelt der Nutzer die Arbeitseinheiten der Dialogfelder u.ä., wie sie unter Optionen > Projekt-Präferenzen > Arbeitseinheiten im Projekt eingestellt wurden. Es wird ein Formattext zurückgegeben, der als erster Parameter in der STR () Funktion verwendet werden kann. Beide REQUESTs funktionieren wenn das Interface-Script interpretiert wird, allerdings verursacht "Working_angle_unit" Warnungen bei Benutzung im Parameter-Script.

```
n = REQUEST ("Model_length_unit", "", format_string)
n = REQUEST ("Layout_length_unit", "", format_string)
```

Durch diese REQUESTs ermittelt der Nutzer die Layout- und Modell-Einheiten der Dialogfelder u.ä., wie sie unter Optionen > Projekt-Präferenzen > Arbeitseinheiten im Projekt eingestellt wurden. Es wird ein Formattext zurückgegeben, der als erster Parameter in der STR () Funktion verwendet werden kann. *Beide Ausdrücke geben 0 zurück und enthalten Dummyrückgabewerte (Leerstring oder 0) falls im Parameter-Script verwendet, verursachen außerdem einen Warnmeldung. Beide funktionieren ausschließlich im User Interface Script.*

```
n = REQUEST ("Model_text_size_unit", "", format_string)
n = REQUEST ("Layout_text_size_unit", "", format_string)
```

Durch diese REQUESTs ermittelt der Nutzer die Formate des Layouts und der Textgröße des Modells. Es wird ein Formattext zurückgegeben, der als erster Parameter in der STR () Funktion verwendet werden kann. Die Requests verursachen eine Warnung, falls im Parameterscript verwendet.

n = REQUEST (**"Properties_Of_Parent"**, propertyType, parentProperties)

Gibt die Eigenschaften des Elternelements zurück. Alle Eigenschaften werden in einem Array in der folgenden Form zurückgegeben: ID, type, group, name. Nur in Etiketten verwendbar. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

```
Kern-Eigenschaft:     [id,    "",      "",              PropertyName]
AC property:        [guid,  "",     "GroupName",    PropertyName]
IFC property:       [id,    "IFC",            "GroupName", PropertyName]
Classification:     [guid,  "Classification",  "",            ClassificationSystemName]
Profile parameter: [guid,  "",        "Profile Parameters",  ParameterName]
```

propertyType: Schlüsselwort, das den Typ der abgefragten Eigenschaften definiert. Leerstring gibt alle Arten von Eigenschaften zurück. Mögliche Werte:

```
"COREPROPERTY"
"ACPROPERTY"
"IFCPROPERTY"
"CLASSIFICATION"
"PROFILEPARAMETER"
```

Kompatibilität: dieser REQUEST-Befehl wurde in ARCHICAD 20 eingeführt. In ARCHICAD 21 wurden die Eigenschaftstyp-Optionen und der Eigenschaftstyp "Klassifizierung" eingeführt, in ARCHICAD 22 der Eigenschaftstyp "Profilparameter".

Beispiel 4:

```
DIM parentProperties[]
n = REQUEST ("Properties_Of_Parent", "", parentProperties)
! parentProperties =   [Id1, TypeName1, GroupName1, PropertyName1,
                         Id2, TypeName2, GroupName2, PropertyName2,
                         ...
                         Idn, TypeNamen, GroupNamen, PropertyNamen]
```

n = REQUEST (**"Property_Value_Of_Parent"**, "id", type, dim1, dim2, propertyValues)

Gibt ein Werte-Array der ausgewählten Eigenschaft zurück. Nur in Etiketten verwendbar. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

Kompatibilität: eingeführt in ARCHICAD 20.

id: die ID der ausgewählten Eigenschaft.

type: der Typ des ausgewählten Eigenschaftswertes.
- 1: Boolesche
- 2: ganzzahlig
- 3: reale Zahl
- 4: Zeichenfolge
- 5: Länge
- 6: Fläche
- 7: Volumen
- 8: Winkel

Kompatibilität: Die Typen Länge, Fläche, Volumen und Winkel wurden in ARCHICAD 22 eingeführt.

dim1, dim2: die Dimensionen des **propertyValues** -Arrays.
- dim1 = 0, dim2 = 0: einfach, skalarer Wert.
- dim1 > 0, dim2 > 0: Werteliste.

Beispiel 5:

```
DIM propertyValues[]
n = REQUEST ("Property_Value_Of_Parent", "ExampleId", type, dim1, dim2, propertyValues)
```

n = REQUEST ("Property_Values_Of_Parent", propInputIds, propOutputVals)

Gibt ein Werte-Dictionary für das angegebene Eigenschafts-ID-Dictionary zurück. Nur in Etiketten verwendbar. *Der Ausdruck gibt 0 zurück und enthält Dummy-Rückgabewerte (Leerstring oder 0), wenn er im Parameterscript verwendet oder mit einem unbekannten Eingabeschlüssel eines Dictionaries verwendet wird, was zu einer zusätzlichen Warnung führt.*

Kompatibilität: eingeführt in ARCHICAD 23.

propInputIds: (Dictionary) Definieren der ausgewählten Eigenschafts-IDs.

propInputIds.propertyIds[n]: (Array) enthält ein Dictionary für jede ausgewählte Eigenschafts-ID.

propInputIds.propertyIds[n].id: (String) die ID der ausgewählten Eigenschaft.

propOutputVals: (Dictionary) die Eigenschaftswertdaten für die ausgewählten Eigenschafts-IDs.

propOutputVals.propertyValues[n]: (Array) enthält Dictionaries für jeden Eigenschaftswert.

propOutputVals.propertyValues[n].value_status: der Status des ausgewählten Eigenschaftswerts
- 1: die ausgewählte Eigenschaft ist verfügbar und hat einen Wert
- 2: die ausgewählte Eigenschaft ist verfügbar, es wurde jedoch kein Wert dafür definiert

3: die Eigenschaft ist nicht verfügbar oder die ausgewählte ID ist ungültig
4: der Eigenschaftswert kann nicht ausgewertet werden

propOutputVals.propertyValues[n].type: der Typ des ausgewählten Eigenschaftswertes. Dieser Schlüssel ist nur vorhanden, wenn propOutputVals.propertyValues[n].value_status 1 oder 2 ist.

1: Boolesche
2: ganzzahlig
3: reale Zahl
4: Zeichenfolge
5: Länge
6: Fläche
7: Volumen
8: Winkel

propOutputVals.propertyValues[n].value[]: (Array) enthält die Liste der ausgewählten Eigenschaftswerte.

Beispiel 6:

```
dict propInputIds
    propInputIds.propertyIds[1].id = "ExampleId1"
    propInputIds.propertyIds[2].id = "ExampleId2"
    ...
    propInputIds.propertyIds[n].id = "ExampleIdn"
dict propOutputVals
n = REQUEST ("Property_Values_Of_Parent", propInputIds, propOutputVals)
! propOutputVals
    .propertyValues[1].value_status
    .propertyValues[1].type
    .propertyValues[1].value[]
    .propertyValues[2].value_status
    .propertyValues[2].type
    .propertyValues[2].value[]
    ...
    .propertyValues[n].value_status
    .propertyValues[n].type
    .propertyValues[n].value[]
```

n = REQUEST ("**Component_Properties_Of_Parent**", propertyType, parentComponentProperties)

Gibt die Komponenteneigenschaften zurück, die für mindestens eine der Baustoffkomponenten des übergeordneten Objekts verfügbar sind. Alle Eigenschaften werden in einem Array in der folgenden Form zurückgegeben: ID, type, group, name. Nur in Etiketten verwendbar. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

```
AC property:    [guid,  "",               "GroupName", PropertyName]
Classification: [guid,  "Classification", "",          ClassificationSystemName]
```

propertyType: Schlüsselwort, das den Typ der abgefragten Eigenschaften definiert. Leere Zeichenfolge gibt alle verfügbaren Arten von Eigenschaften zurück. Mögliche Werte:

```
"ACPROPERTY"
"CLASSIFICATION"
```

Kompatibilität: dieser REQUEST-Befehl wurde in ARCHICAD 23 eingeführt.

Beispiel 7:

```
DIM parentComponentProperties[]
n = REQUEST ("Component_Properties_Of_Parent", "", parentComponentProperties)
! parentComponentProperties =   [Id1, TypeName1, GroupName1, PropertyName1,
                                 Id2, TypeName2, GroupName2, PropertyName2,
                                 ...
                                 Idn, TypeNamen, GroupNamen, PropertyNamen]
```

n = REQUEST (**"Component_IDs_Of_Parent"**, collectComponents, outputCompIds)

Gibt die IDs der Baustoffkomponenten des übergeordneten Objekts in Form eines Dictionary zurück. Nur in Etiketten verwendbar. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

Kompatibilität: eingeführt in ARCHICAD 23.

collectComponents: (Dictionary) Definieren der Methode zum Sammeln der Baustoffkomponenten des übergeordneten Objekts.

collectComponents.collectMode: (Ganzzahl) die Methode zum Sammeln der Baustoffkomponenten. Dieser Schlüssel ist optional. Wenn er nicht vorhanden ist, verwendet der Request den Standard-CollectMode 1.

1: (Standardwert) gibt alle IDs der Baustoffkomponenten des übergeordneten Elements zurück

2: gibt die gleichen IDs von Baustoffkomponenten in der gleichen Reihenfolge wie in WALL_SKINS_PARAMS, SHELLBASE_SKINS_PARAMS, SLAB_SKINS_PARAMS oder ROOF_SKINS_PARAMS zurück - abhängig vom Elementtyp des übergeordneten Elements.

outputCompIds: (Dictionary) die IDs der Baustoffkomponenten des übergeordneten Objekts.

outputCompIds.componentIds[n]: (Array) Enthält Dictionaries für jede Baustoffkomponenten-ID.

outputCompIds.componentIds[n].id: (Ganzzahl) die ID der Baustoffkomponente des übergeordneten Elements.

Beispiel 8:

```
dict collectComponents
       collectComponents.collectMode = 1
dict outputCompIds
n = REQUEST ("Component_IDs_Of_Parent", collectComponents, outputCompIds)
! outputCompIds
       .componentIds[1].id
       .componentIds[2].id
       ...
       .componentIds[n].id
```

n = REQUEST **("Component_Property_Values_Of_Parent",** compPropInput, compPropVals)

Gibt ein Werte-Dictionary für die angegebene Komponenten-ID und Eigenschafts-ID zurück. Nur in Etiketten verwendbar. *Der Ausdruck gibt 0 zurück und enthält Dummy-Rückgabewerte (Leerstring oder 0), wenn er im Parameterscript verwendet oder mit einem unbekannten Eingabeschlüssel eines Dictionaries verwendet wird, was zu einer zusätzlichen Warnung führt.*

Kompatibilität: eingeführt in ARCHICAD 23.

compPropInput: (Dictionary) definiert die ausgewählten Baustoffkomponenten- und Eigenschafts-IDs.

compPropInput.componentId: (Dictionary) enthält ein Dictionary für die ausgewählte ID der Baustoffkomponente.

compPropInput.componentId.id: (Ganzzahl) die ID der ausgewählten Baustoffkomponente, die über den Request "Component_IDs_Of_Parent" verfügbar ist.

compPropInput.propertyIds[n]: (Array) enthält ein Dictionary für jede ausgewählte Eigenschafts-ID.

compPropInput.propertyIds[n].id: (String) die ID der ausgewähten Eigenschaft.

compPropVals: (Dictionary) die Eigenschaftswertdaten der Baustoffkomponente für die ausgewählten Eigenschafts-IDs.

compPropVals.propertyValues[n]: (Array) enthält ein Dictionary für jeden Eigenschaftswert.

compPropVals.propertyValues[n].value_status: der Status des ausgewählten Eigenschaftswerts

- 1: die ausgewählte Eigenschaft ist verfügbar und hat einen Wert
- 2: die ausgewählte Eigenschaft ist verfügbar, es wurde jedoch kein Wert dafür definiert
- 3: die Eigenschaft ist nicht verfügbar oder die ausgewählte ID ist ungültig

4: der Eigenschaftswert kann nicht ausgewertet werden

compPropVals.propertyValues[n].type: der Typ des ausgewählten Eigenschaftswertes. Dieser Schlüssel ist nur vorhanden, wenn compPropVals.propertyValues [n] .value_status 1 oder 2 ist.

1: Boolesche
2: ganzzahlig
3: reale Zahl
4: Zeichenfolge
5: Länge
6: Fläche
7: Volumen
8: Winkel

compPropVals.propertyValues[n].value[]: (Array) enthält die Liste der ausgewählten Eigenschaftswerte.

Beispiel 9:

```
dict compPropInput
     compPropInput.componentId.id = iActualID ! From "Component_IDs_Of_Parent" request
     compPropInput.propertyIds[1].id = "ExampleId1"
     compPropInput.propertyIds[2].id = "ExampleId2"
     ...
     compPropInput.propertyIds[n].id = "ExampleIdn"
dict compPropVals
n = REQUEST ("Component_Property_Values_Of_Parent", compPropInput, compPropVals)
! compPropVals =  propertyValues[1].value_status
    .propertyValues[1].value_status
    .propertyValues[1].type
    .propertyValues[1].value[]
    .propertyValues[2].value_status
    .propertyValues[2].type
    .propertyValues[2].value[]
    ...
    .propertyValues[n].value_status
    .propertyValues[n].type
    .propertyValues[n].value[]


n = REQUEST ("Property_Name", "id", typeName, groupName, propertyName)
```

Gibt **Typ, Gruppe und Name** der ausgewählten Eigenschaft zurück. Nur in Etiketten verwendbar. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

Kompatibilität: eingeführt in ARCHICAD 20.

id: die ID der ausgewählten Eigenschaft.

typeName: der Typ der ausgewählten Eigenschaft (string).
- **"IFC":** für IFC-Eigenschaften
- **"":** andere Eigenschaften

groupName: die Gruppe der ausgewählten Eigenschaft (String).

Leerstring ("") für Kern-Eigenschaften.

propertyName: der Name der ausgewählten Eigenschaften (String).

n = REQUEST ("AUTOTEXT_LIST", "", autoTextListArray)

Gibt ein AUTOTEXT-Array der im Projekt verwendeten Autotexte zurück mit folgenden Dreierwerten ["ID", "Category", "Name"]. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung. Nur im UI-Script verwendbar.* Die ID wird mittels der Befehle **UI_CUSTOM_POPUP...** im Parameter gespeichert.

Enthält sämtliche Autotexte aus Projekt-Info und Autotext-Dialog (Textwerkzeug - Autotext einfügen).

Kompatibilität: eingeführt in ARCHICAD 20.

Beispiel 10:

```
DIM autoTextListArray[]
n = REQUEST ("AUTOTEXT_LIST", "", autoTextListArray)
! autoTextListArray =   [ID1, CategoryName1, TextName1,
                          ID2, CategoryName2, TextName2,
                          ...
                          IDn, CategoryNamen, TextNamen]
```

n = REQUEST{3} ("Sum_with_rounding", req_name, addends_array, result)

Gibt die Summe der Zahlen zurück in **addends_array**, mit Rundungen entsprechend den "Summe berechnen nach ..." in den Projektpräferenzen. Diese Präferenz findet man unter Optionen / Projektpräferenzen / Berechnungseinheiten und -regeln.

Mögliche Einstellungen in den Projektpräferenzen :

- "angezeigte Werte": der Request rundet zunächst die Summanden entsprechend **req_name**, und summiert diese anschließend.
- "exakte Werte": der Request summiert die Summanden einfach nur.

Verursacht Warnung wenn im Parameter-Script verwendet.

Kompatibilität: eingeführt in ARCHICAD 20.

Rückgabewerte:

- 0, falls **req_name** ungültig ist.
- 1, wenn der Aufruf erfolgreich war.

req_name: der Name des Formatierungs-Requests spezifiziert, wie die Summanden gerundet werden müssen, falls "Summe berechnen nach" auf "den angezeigte Werten" eingestellt ist.

Falls zum Beispiel **req_name = "Area_dimension"**, und die Projektpräferenzen / Berechnungseinheiten und -regeln / Flächeneinheit auf "Quadratzentimeter" mit 3 Dezimalstellen eingestellt ist, die Rundungsintervalle auf 0.025, dann werden die Summanden auf ein Vielfaches von 0.025 cm² gerundet, das wären 0.0000025 m².

Gültige Request-Namen:

Linear_dimension, Angular_dimension, Radial_dimension, Level_dimension, Elevation_dimension, Window_door_dimension, Sill_height_dimension, Area_dimension, Calc_length_unit, Calc_area_unit, Calc_volume_unit, Calc_angle_unit.

addends_array: Das Array der zu addierenden Zahlen. Ob diese behandelt werden sollen als m, m², m³ oder Grad, wird festgelegt durch **req_name**.

result: eine Zahl, welche bei der Rückgabe auf die Summe der Summanden gesetzt wird, entsprechend den "Summe berechnen nach"-Präferenzen. Bitte beachten, dass **result** in den gleichen Einheiten ist wie die Summanden. Es wird nicht in die Zieleinheit, welche festgelegt wird durch **req_name**, umgewandelt.

```
n = REQUEST ("ASSOCLP_PARVALUE", expr, name_or_index, type, flags, dim1, dim2, p_values)
n = REQUEST ("ASSOCLP_PARVALUE_WITH_DESCRIPTION", expr, name_or_index, type,
    flags, dim1, dim2, p_values_and_descriptions)
```

Übergibt die Information in den gegebenen Variablen über den Bibliothekselementparameter, mit dem das Bibliothekselement mit diesem REQUEST verbunden ist. 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.

expr: Objekt des REQUESTs , 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
16: Baustoff
17: Profil-*Kompatibilität: eingeführt in ARCHICAD 23*

flags:
flags = j_1 + 2*j_2 + 64*j_7 + 128*j_8, hierbei kann j jeweils 0 oder 1 sein.
j_1: untergeordnet/eingezogen in Parameterliste
j_2: mit fettem Text in Parameterliste
j_7: deaktiviert (geschützt in allen Kontexten)
j_8: verborgen in der Parameterliste

dim1, dim2: dim1 ist die Anzahl der Reihen, dim2 die Anzahl der Spalten.
dim1 = 0, dim2 = 0: einfach, skalarer Wert
dim1 > 0, dim2 = 0: eindimensionalen Array
dim1 > 0, dim2 > 0: zweidimensionalen Array
Wenn dim2 > 0, dann dim1 > 0.

p_values: für ASSOCLP_PARVALUE: gibt den Parameterwert oder Array der Werte zurücke. 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 "data" ein zweidimensionaler dynamischer Array, werden alle Elemente in der ersten Reihe gespeichert.

p_values_and_descriptions: für ASSOCLP_PARVALUE_WITH_DESCRIPTION: gibt den Parameterwert gefolgt vom Parameterbeschreibungsstring (wie im Befehl VALUES festgelegt) oder eines Arrays von diesen Paaren. Bei String-Typ-Prametern ist der

Beschreibungsstring immer leer. Die Array-Elemente (bzw. Arrayelement/Beschreibungsstring-Paare) 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 "data" ein zweidimensionaler dynamischer Array, werden alle Elemente in der ersten Reihe gespeichert.

n = 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. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

n = 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. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

parameter_string: Eine Kombination von Kurzbeschreibungen, durch Kommas getrennt, für die angeforderten Felder der Eigenschaftsdatensätze. 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 "data" ein zweidimensionaler dynamischer Array, werden alle Elemente in der ersten Reihe gespeichert.

Beispiel 11:

```
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
```

```
n = REQUEST ("REFERENCE_LEVEL_DATA", "", name1, elev1, name2, elev2,
          name3, elev3, name4, elev4)
```

Gibt in den gegebenen Variablen die Namen und Höhenwerte der Referenzhöhen zurück, wie sie in Optionen/ Projektpräferenzen/ Referenzhöhen eingestellt sind. Der Rückgabewert der Funktion ist die Anzahl der erfolgreich abgerufenen Werte bzw. 0, wenn ein Fehler aufgetreten ist.

```
n = REQUEST ("ANCESTRY_INFO", expr, name [, guid,
          parent_name1, parent_guid1,
          ...
          parent_namen, parent_guidn)
```

Informationen zur "Abstammung" bei einem Bibliothekselement. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

Wenn expr = 0, wird in den gegebenen Variablen der Name und die globale eindeutige ID-Nr. des Bibliothekselements, das diese REQUEST-Funktion enthält, zurückgegeben. Optional gibt die Funktion die Namen und die globalen eindeutige 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 12:

```
DIM strings[]
n = REQUEST ("ANCESTRY_INFO", 1, name, guid, strings)
IF n > 2 THEN
    ! Daten des ersetzten Bibliothekselementes
    TEXT2 0, -1, "replacing: " + name + ' ' + guid
    ! parents
    l = -2
    FOR i = 1 TO n - 2 STEP 2
        TEXT2 0, l, strings [i]
        l = l - 1
    NEXT i
ENDIF
```

n = REQUEST (**"TEXTBLOCK_INFO"**, textblock_name, width, height)

Übergibt den gegebenen Variablen die 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. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

n = REQUEST{2} (**"Material_info"**, name_or_index, param_name, value_or_values)

Stellt die Informationen in der gegebenen Variablen eines Parameters eines spezifizierten Materials wieder her (oder Zusatzparameter, siehe „Zusätzliche Daten“). RGB Informationen werden in drei separaten 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. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.* Mögliche Parameternamen, die sich auf Parameter der Materialdefinition beziehen:

param_name:

- "gs_mat_surface_rgb": surface R, G, B [0.0..1.0]
- "gs_mat_surface_r": surface R [0.0..1.0]
- "gs_mat_surface_g": surface G [0.0..1.0]
- "gs_mat_surface_b": surface B [0.0..1.0]
- "gs_mat_ambient": Umgebungskoeffizient [0.0..1.0]

"gs_mat_diffuse": Diffuser Koeffizient [0.0..1.0]
"gs_mat_specular": Glanzkoeffizient [0.0..1.0]
"gs_mat_transparent": Transparenzkoeffizient [0.0..1.0]
"gs_mat_shining": Glanzlicht [0.0..100.0]
"gs_mat_transp_att": Transparenzabschwächung [0.0..4.0]
"gs_mat_specular_rgb": Gespiegelte Farbe R, G, B [0.0..1.0]
"gs_mat_specular_r": specular color R [0.0..1.0]
"gs_mat_specular_g": specular color G [0.0..1.0]
"gs_mat_specular_b": specular color B [0.0..1.0]
"gs_mat_emission_rgb": Abstrahl-Farbe R, G, B [0.0..1.0]
"gs_mat_emission_r": emission color R [0.0..1.0]
"gs_mat_emission_g": emission color G [0.0..1.0]
"gs_mat_emission_b": emission color B [0.0..1.0]
"gs_mat_emission_att": Abstrahlungsabschwächung [0.0..65.5]
"gs_mat_fill_ind": Schraffur-Index
"gs_mat_fillcolor_ind": Schraffurfarben-Index
"gs_mat_texture": Textur-Index

Beispiel 13:

```
n = REQUEST{2} ("Material_info", "Brick-Face", "gs_mat_ambient", a)
n = REQUEST{2} ("Material_info", 1, "gs_mat_surface_rgb", r, g, b)
n = REQUEST{2} ("Material_info", "Brick-Face", "gs_mat_texture",
            file_name, w, h, mask, alpha)
n = REQUEST{2} ("Material_info", "My-Material", "my_extra_parameter", e)
```

n = REQUEST{2} (**"Building_Material_info"**, name_or_index, param_name, value_or_values)

Gibt Informationen in die vorgegebene(n) Variable(n) eines Parameters des spezifizierten Baustoffes. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.* Mögliche Parameternamen von Baustoffen entsprechend den Parametern der Baustoff-Definition:

param_name:

"gs_bmat_id": ID des Baustoffs
"gs_bmat_surface": Oberflächenmaterial-Index des Baustoffs
"gs_bmat_description": Beschreibung des Baustoffs
"gs_bmat_manufacturer": Hersteller des Baustoffs

`"gs_bmat_collisiondetection":` Baustoff beteiligt an der Kollisionserkennung (0 oder 1)
`"gs_bmat_intersectionpriority":` Verschneidungspriorität des Baustoffs
`"gs_bmat_cutFill_properties":` Schnittschraffureigenschaften des Baustoffs (Schnittschraffur-Indexnummer, Schnittschraffur-Vordergrundstift, Schnittschraffur-Hintergrundstift)
`"gs_bmat_physical_properties":` Baustoff: physikalischen Eigenschaften (Wärmeleitfähigkeit, Dichte, Wärmekapazität, gebundene Energie, gebundenes CO2)

Beispiel 14:

```
n = REQUEST{2} ("Building_Material_info", "Brick", "gs_bmat_id", id)
n = REQUEST{2} ("Building_Material_info", "Brick", "gs_bmat_surface", index)
n = REQUEST{2} ("Building_Material_info", "Brick", "gs_bmat_physical_properties",
             thermalConductivity, density, heatCapacity, embodiedEnergy, embodiedCarbon)
```

`n = REQUEST ("FONTNAMES_LIST", "", fontnames)`

Übergibt der Variablen fontnames alle verfügbaren Zeichensatznamen im aktuellen System (einschl. Zeichenkodierung). Diese Liste oder Teile davon können mit dem VALUES-Befehl für eine Zeichensatzauswahl verwendet werden. Der Rückgabewert der Funktion ist die Anzahl der erfolgreich abgerufenen Werte bzw. 0, wenn ein Fehler aufgetreten ist.

Beispiel 15:

```
dim fontnames[]
n = REQUEST ("FONTNAMES_LIST", "", fontnames)
VALUES "f" fontnames, CUSTOM
```

Diese Form des VALUES-Befehls erzeugt ein Pop-up für die Zeichensatznamen im Parameter "f", vom Typ string. Die Variable "fontnames" einhält alle Zeichensatznamen, die manuell zugewiesen oder eben automatisch mit REQUEST ("FONTNAMES_LIST", ...) gelesen werden können. Der Befehlszusatz CUSTOM ist notwendig, um nicht aufgeführte Zeichensatznamen eingeben zu können. Ist er enthalten, stellt er sicher, dass auf einem anderen Computer, auf dem der gewählte Zeichensatz nicht enthalten ist, der gewählte Zeichensatz nicht überschrieben wird. Der VALUES -Befehl nimmt, wenn ein Wert in einer pop-up Liste nicht mehr auftaucht, automatisch den ersten aus der Liste. Es wird empfohlen, diese Funktion in die Datei ARCHICAD_Library_Master einzubauen.

`n = REQUEST ("HomeDB_info", "", homeDBIntId, homeDBUserId, homeDBName, homeContext)`

Übergibt in den aufgeführten Variablen die interne ID (Typ integer), die User-ID und den Namen (Typ string) der Standarddatenbank, wo das Bibliothekselement, das diesen REQUEST enthält, abgelegt wurde.

- Platziert im Grundriss: Interne ID des Geschosses, index als Text und Name, homeContext = 1
- Schnitt/Ansicht: Interne ID des Schnittes, Referenz-ID und Name, homeContext = 2
- Detail: Interne ID des Details, Referenz-Id und Name, homeContext = 3

- Master-Layout: Interne id des Layouts, Leerer Text und Name, homeContext = 4
- Layout: Interne ID des Layouts, Nummer und Name, homeContext = 5

Bei Etiketten werden die Daten des Elementes zurückgegeben, mit dem das Etikett assoziiert ist. Die gesammelten Daten können zum einzelnen Identifizieren der Elemente in unterschiedlichen ARCHICAD-Datenbanken einer Plandatei verwendet werden. Verursacht Warnung wenn im Parameter-Script verwendet.

n = REQUEST ("**floor_plan_option**", "", storyViewpointType)

Ergibt den Blickpunkttyp des Geschosses, der unter Modelldarstellung eingestellt ist. 0 steht für "Grundriss", 1 steht für "Deckenspiegel". *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

n = REQUEST ("**class_of_fill**", index, class)

Ergibt die Klasse der durch Index identifizierten Schraffur in der Klassen-Variable. Verursacht Warnung wenn im Parameter-Script verwendet.

class: Mögliche Werte:

1: Vektorschraffur
2: Symbolschraffur
3: Durchscheinende Schraffur
4: Farbverlauf linear
5: Farbverlauf Kreisförmig
6: Bildschraffur

n = REQUEST ("**view_rotangle**", "", angleViewRotation)

Gibt den Drehwinkel des aktuellen Ausschnitts zurück. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

n = REQUEST ("**program_info**", "", name[, version[, keySerialNumber[, isCommercial]]])

Gibt Informationen zum gerade laufenden Programm zurück. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

name: Name des Programms

version: Versionsnummer des Programms

keySerialNumber: Seriennummer des Schutzschlüssels

isCommercial: Gibt 1 zurück, wenn eine Vollversion (kommerziell) des Programms läuft

n = REQUEST ("**Configuration_number**", "", stConfigurationNumber)

Gibt die Konfigurationsnummer (als Stringausdruck), welche zur aktuellen ARCHICAD-Lizenz gehört, zurück (im Falle einer Softlizenz oder eines Dongels). Gibt im Falle einer Studentenversion, einer Trial- oder Demoversion einen Leerstring zurück. Jede Konfigurationsnummer ist einmalig und ändert sich nicht.

Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.
Kompatibilität: eingeführt in ARCHICAD 20.

n = REQUEST (extension_name, parameter_string, variable1, variable2, ...)

Falls die Frage mit keiner der aufgelisteten Fragen übereinstimmt, versucht die REQUEST () Funktion diese als erweiterungsspezifischen Name zu verwenden. Befindet sich diese Erweiterung im Erweiterungen-Ordner, so wird diese zum Übernehmen sämtlicher Werte für alle genau angegebenen Variablennamen verwendet. Der Parameter -Text wird durch die Erweiterung interpretiert.

n = REQUEST (**"COMPONENT_PROJECTED_AREA"**, idxSkin, projectedArea)

Gibt die projizierte Fläche der indizierten Schicht zurück. Nur verfügbar im Eigenschaften-Script (andere Scripte geben 0 zurück). *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

idxSkin: Mögliche Werte:

0: für Grundelemente
1- : Index der Schicht in Mehrschichtbauteilen
1- : Index der Komponente eines Profils

Beispiel 16:

```
n = request ("COMPONENT_PROJECTED_AREA", 0, a)
COMPONENT "Projected Area", a, "m2"
```

Verwendung im Eigenschaften-Script, zunächst Abfrage der Fläche der Schicht, dann Erzeugung einer Komponente unter Verwendung des Rückgabewertes.

n = REQUEST (**"COMPONENT_VOLUME"**, idxSkin, skinVolume)

Gibt das Volumen der indizierten Schicht/Komponente zurück. Nur verfügbar im Eigenschaften-Script (andere Scripte geben 0 zurück). *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

idxSkin: Mögliche Werte:

0: für Grundelemente
1- : Index der Schicht in Mehrschichtbauteilen
1- : Index der Komponente eines Profils

Beispiel 17:

```
n = request ("COMPONENT_VOLUME", 0, v)
COMPONENT "Volume", v, "m3"
```

Verwendung im Eigenschaften-Script, zunächst Abfrage der Fläche der Schicht, dann Erzeugung einer Komponente unter Verwendung des Rückgabewertes.

n = REQUEST ("DateTime", format_string, datetimestring)

Gibt das aktuelle Datum und die Uhrzeit als formatierte Zeichenfolge in datetimestring zurück. Verwendet das DateTime Add-On, sowie das Öffnen und Schließen des gewünschten Kanals.

format_string: Format-String, beschrieben beim *paramString* Parameter von „Kanal öffnen“ .

datetimestring: Die formatierte Zeichenfolge wird in dieser Variablen zurückgegeben

Die Requests verursachen eine Warnung, falls im Parameterscript verwendet.

Profil-Requests

n = REQUEST ("Name_of_Profile", index, name)

Gibt in der name Variablen den Profilnamen zurück, welcher durch den index identifiziert wird. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

Kompatibilität: eingeführt in ARCHICAD 21.

n = REQUEST ("Profile_components", name_or_index, nComponents,
compType1, compType2, ..., compTypen)

Gibt die Anzahl (nComponents) und Komponent-Typen (compTypen) des Profils zurück, welches über name oder index identifiziert wird. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

compTypei: Mögliche Werte von Profilkomponententypen:

- 0: Kern
- 1: Bekleidung
- 2: andere

Kompatibilität: eingeführt in ARCHICAD 21.

Beispiel 1:

```
_nComponents = 0
dim _componentTypes[]
n = REQUEST ("Profile_components", myProfileIdx, _nComponents, _componentTypes
```

n = REQUEST ("Profile_default_boundingbox", name_or_index, xmin, ymin, xmax, ymax)

Gibt die 2 definierenden Koordinatenpunkte des ursprünglichen Begrenzungsrechtecks relativ zum Ursprung des mit dem Namen oder Index gekennzeichneten Profils zurück. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

Kompatibilität: eingeführt in ARCHICAD 21.

```
n = REQUEST ("Profile_default_geometry", name_or_index, n1, n2, ..., nm,
             x11, y11, edgeVisible11, vertEdgeVisible11, additionalStatus11, ...,
             x1n1, y1n1, edgeVisible1n1, vertEdgeVisible1n1, additionalStatus1n1,
             x21, y21, edgeVisible21, vertEdgeVisible21, additionalStatus21, ...,
             x2n2, y2n2, edgeVisible2n2, vertEdgeVisible2n2, additionalStatus2n2, ...,
             xm1, ym1, edgeVisiblem1, vertEdgeVisiblem1, additionalStatusm1, ...,
             xmnm, ymnm, edgeVisiblemnm, vertEdgeVisiblemnm, additionalStatusmnm)
```

Gibt die ursprünglichen geometrischen Daten des Profils zurück, die mit Namen oder Index gekennzeichnet sind. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

n1...ni: dann die Anzahl der Konturknoten in jeder Profilkomponente. Die Gesamtzahl der Profilkomponenten (m) kann vom REQUEST "Profile_components" zurückgegeben werden.

edgeVisiblei: Kontur ab Knoten i ist sichtbar.

vertEdgeVisiblei: vertikale Kante ab Knoten i ist sichtbar, in 3D verwendbar (0 im Falle von segmentiertem Polygon).

additionalStatusi: verwendet für Segmente und Bögen der Polylinie (Mittelpunkt setzen = 900, Bogen mit Mittelpunkt und Winkel = 4000, etc.), oder zur Markierung des Kontrollpunkt des Konturendes (-1, in diesem Fall werden die`vertEdgeVisiblei` und`edgeVisiblei` automatisch auf 0 gesetzt).

Die in dieser Struktur zurückgegebenen Statusparameter unterstützen unterschiedliche Statustypdefinitionen von poly2, cprism, tube. Jedes Format kann mit folgender Methode berechnet werden:

Beispiel 2:

```
poly2Status = edgeVisible + additionalStatus
prismStatus = additionalStatus
tubeStatus = additionalStatus
if additionalStatus >= 0 then         ! not contour end
    if edgeVisible then
        prismStatus = prismStatus + 15    ! j1, j2, j3, j4
    endif
    if verticalEdgeVisible = 0 then
        prismStatus = prismStatus+ 64     ! j7
        ! In dem TUBE werden seitlich von dem Knoten ausgehende Seitenkanten
  ! zur Darstellung der Kontur verwendet
        tubeStatus = tubeStatus + 1
    endif
endif
```

Kompatibilität: eingeführt in ARCHICAD 21.

n = REQUEST{4} ("Profile_component_info", name_or_index, component_ind, param_name, value)

Gibt einen angeforderten Attributwert einer dedizierten Komponente (durch component_ind) des mit dem Namen oder Index gekennzeichneten Profils zurück. *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

Die component_ind Komponente muss sich im gültigen Bereich von nComponents befinden (definiert durch "profile_components" Request).

param_name: adressiert Attribut-Einstellungen vom Profil Manager, zurückgegeben in value

"gs_profile_bmat": Baustoff-Index der Komponente

"gs_profile_surface": Überschreibungs-Oberflächenindex der Komponente (bei aktiven Überschreibungs-Einstellungen); Gibt andernfalls die Oberfläche des Baustoffs zurück)

"gs_profile_showoutline": "Anzeige Außenkontur" Einstellung der Komponente

"gs_profile_outlinetype": "Typ Außenkontur" Einstellung der Komponente

"gs_profile_outlinepen": "Stift Außenkontur" Einstellung der Komponente

Rückgabeattributwerte können in jedem Attributbezogenen Befehl verwendet werden, z. B POLY2_B{6}, wo Konturabschnitte des Polygons individuell angepasst werden können.

Kompatibilität: eingeführt in ARCHICAD 21.

n = REQUEST{4} ("Profile_component_info", name_or_index, component_ind, param_name, value1, value2, ..., valuen)

Gibt die abgefragten Attribute aller Kanten in der dedizierten Komponente des Profils (durch component_ind) identifiziert durch Namen oder Index zurück . *Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.*

Die component_ind Komponente muss sich im gültigen Bereich von nComponents befinden (definiert durch "profile_components" Request).

param_name: adressiert Attribut-Einstellungen vom Profil Manager, zurückgegeben in value

"gs_profile_comp_surfaces": individuelle Oberflächenindizes der Kanten der Komponente

"gs_profile_comp_pens": individuelle Linientypindizes der Kanten der Komponente

"gs_profile_comp_linetypes": individuelle Stiftindizes der Kanten der Komponente

Rückgabeattributwerte können in jedem Attributbezogenen Befehl verwendet werden, z. B POLY2_B{6}, wo Konturabschnitte des Polygons individuell angepasst werden können.

Kompatibilität: eingeführt in ARCHICAD 21.

Veraltete Requests

n = REQUEST ("**Constr_Fills_display**", "", optionVal)

Ausdruck, der 0 zurückgibt und Dummyrückgabewerte enthält (Leerstring oder 0) falls im Parameter-Script verwendet, verursacht außerdem einen Warnmeldung.

Kompatibilität bis zu ARCHICAD 19: gibt in der gegebenen Variable den Wert der Bauteilschraffurdarstellungsoption zurück, wie dies in der Modelldarstellung eingestellt ist. (frühere Bauteil-Schraffuren).

Kompatibilität beginnend mit ARCHICAD 20: der Rückgabewert ist immer standardmäßig 6 (Bauteilschraffur-Schema: wie in den Einstellungen).

optionVal: Anzeigecode der Schnittschraffur.

1: Nur Schnittflächenkontur zeigen (früher Leer)
2: Nur Schnittflächenkontur zeigen mit Schichttrennlinie (früher Keine Schraffuren)
4: Schnittschraffur: Vollinie (früher Massiv)
6: Schnittschraffur: Wie in Einstellungen (früher Vektorschraffur)

n = REQUEST ("**internal_ID**", "", id)

Gibt immer 1 zurück. Verwenden Sie stattdessen die Globale Variable GLOB_INTGUID.

Application Query Optionen

n = **APPLICATION_QUERY** (extension_name, parameter_string, variable1, variable2, ...)

Folgend finden Sie eine Liste der Abfrage-Funktionen, welche ARCHICAD als Hilfe für den APPLICATION_QUERY Befehl vorhält. Diese gewünschte Optionen werden in den Parametern **extension_name** und **parameter_string** des Befehl gesetzt. Beachten Sie, dass die Abfrage-Optionen und Rückgabewerte von einem APPLICATION_QUERY variieren können, abhängig vom Ausführungskontext.

Die Verwendung folgender Typen von Application Queries im Parameter-Script wird nicht unterstützt. Diese Queries verursachen GDL-Warnungen mit Einführung von ARCHICAD 19 und werden in darauf folgenden Versionen entweder 0 oder ein Leerstring zurückgeben. Die Einschränkung gilt für:

- "document_feature"

Dokumenteigenschaften

Dieser Befehl kann Eigenschaften des aktuellen Dokumentes/bzw. der Sicht zurückgeben. Zur Zeit gibt es nur eine Eigenschaft, die zurückgegeben werden kann - die Blickrichtung des Dokumentes. Diese Typen von Queries sind im Parameter-Script eingeschränkt und verursachen GDL-Warnungen.

Blickrichtung

n = **APPLICATION_QUERY** ("**document_feature**", "view_direction", type)

Dieser Befehl gibt die Blickrichtung des aktuellen Dokument-Types, in welchem das Objekt visualisiert wird, zurück. Dieser Befehl besitzt keine zusätzlichen Parameter.

type: Typen der Rückgabewerte:
- `"vertical_only":` für den Grundriss
- `"horizontal_only":` für Schnitte und Ansichten, welche aus 3D generiert wurden (nicht wenn das Objekt in S/A platziert wurde)
- `"free":` für 3D und 3D-Dokumente
- `"none"`
- `"unset"`

MEP-Systeme

Dieser Befehl gibt MEP-Systemtypen und Informationen über MEP-Systeme zurück. Es gibt weitere Funktionen, welche mittels des Parameters **parameter_string** abgefragt werden können:

Erfrage die MEP-Systeme

```
DIM d[2][]
n = APPLICATION_QUERY ("MEPSYSTEM", "GetMEPSystems(domain)", d)
```

domain: MEP-Klassenindex (Kanalarbeiten– 1, Rohrleitungen– 2) (GDL definiert die MEP-Klasse basierend auf der Anschluss-Klasse)

d: Array der Werte:
- `[2*k-1]:` MEP Systemindex
- `[2*k]:` MEP Systemname

n: Anzahl der MEP-Systeme multipliziert mit 2.

Abfrage der Domain

```
n = APPLICATION_QUERY ("MEPSYSTEM", "GetDomain(idx)", d)
```

idx: MEP Systemindex

d: Domains (Ganzzahl)
- `1:` Kanalarbeiten
- `2:` Rohrleitungen
- `3:` Kanal- und Rohrleitungen
- `4:` Kabel
- `5:` Kanal und Kabel

6: Rohrleitungen und Kabel
7: Kanalarbeiten, Rohrleitungen und Kabel

n: 1 falls erfolgreich, andernfalls 0

Abfrage Konturstift

```
n = APPLICATION_QUERY ("MEPSYSTEM", "GetContourPen(idx)", pen)
```

idx: MEP Systemindex

pen: Konturstift-Index (Ganzzahl)

n: 1 falls erfolgreich, andernfalls 0

Abfrage Schraffurstift

```
n = APPLICATION_QUERY ("MEPSYSTEM", "GetFillPen(idx)", pen)
```

idx: MEP Systemindex

pen: Schraffurstift-Index (Ganzzahl)

n: 1 falls erfolgreich, andernfalls 0

Abfrage Hintergrundstift

```
n = APPLICATION_QUERY ("MEPSYSTEM", "GetBgPen(idx)", pen)
```

idx: MEP Systemindex

pen: Hintergrundstift-Index (Ganzzahl)

n: 1 falls erfolgreich, andernfalls 0

Abfrage Schraffurtyp

```
n = APPLICATION_QUERY ("MEPSYSTEM", "GetFillType(idx)", filltype)
```

idx: MEP Systemindex

filltype: Schraffurtyp-Index (Ganzzahl)

n: 1 falls erfolgreich, andernfalls 0

Abfrage Linentyp Achse

```
n = APPLICATION_QUERY ("MEPSYSTEM", "GetCenterLineType(idx)", line)
```

idx: MEP Systemindex

line: Index Linientyp Achse (Ganzzahl)

n: 1 falls erfolgreich, andernfalls 0

Abfrage Stift Achse

```
n = APPLICATION_QUERY ("MEPSYSTEM", "GetCenterLinePen(idx)", pen)
```

idx: MEP Systemindex

pen: Stiftindex Achse (Ganzzahl)

n: 1 falls erfolgreich, andernfalls 0

Abfrage System-Material

```
n = APPLICATION_QUERY ("MEPSYSTEM", "GetSystemMaterial(idx)", material)
```

idx: MEP Systemindex

material: Index System-Material (Ganzzahl)

n: 1 falls erfolgreich, andernfalls 0

Abfrage Dämmmaterial

```
n = APPLICATION_QUERY ("MEPSYSTEM", "GetInsulationMaterial(idx)", material)
```

idx: MEP Systemindex

material: Index Dämmmaterial (Ganzzahl)

n: 1 falls erfolgreich, andernfalls 0

MEP Modeler

Dieser Befehl gibt zurück, ob der MEP Modeler aktiv ist. Er besitzt eine Funktion, welche durch den Parameter**parameter_string** ausgelesen werden kann:

Ist verfügbar

```
n = APPLICATION_QUERY ("MEPMODELER", "IsAvailable()", isavailable)
```

isavailable: MEP Modeler ist aktiv (Ganzzahl)

n: 1 falls erfolgreich, andernfalls 0

MEP Verbindungstyp

Dieser Befehl gibt die Verbindungstypen und Stile der Verbindungstypen zurück. Er besitzt 2 Funktionen, welche über den Befehl **parameter_string** abgerufen werden können:

Abfrage der Verbindungstypen

```
DIM d[2][]
n = APPLICATION_QUERY ("MEPCONNECTIONTYPE", "GetConnectionTypes(connectorClass)", d)
```

connectorClass: Verbindungsklasse (Abfluss– 1, Leitung– 2, Kabelträger– 3)

d: Array der Werte:
- `[2*k-1]:` GUID Verbindungstyp
- `[2*k]:` Name Verbindungstyp

n: Anzahl der Verbindungstypen multipliziert mit 2.

Abfrage des Stiles des Verbindungstypes

```
DIM d[]
n = APPLICATION_QUERY ("MEPCONNECTIONTYPE", "GetConnectionTypeStyle(connectorClass)", d)
```

connectorClass: Verbindungsklasse (Abfluss– 1, Leitung– 2, Kabelträger– 3)

d: Array der Werte:
- `[]:` Stile der Verbindungstypen

n: Anzahl der Verbindungstypen.

MEP Flexibles Segment

Dieser Befehl gibt die Geometrie von flexiblen Segmenten zurück. Er besitzt 4 Funktionen, welche mit dem Parameter **parameter_string** abgerufen werden können:

Start des Abschnitts

```
n = APPLICATION_QUERY ("MEPFLEXIBLESEGMENT", "StartSectioning()", r)
```

Zeigt an, dass die Segmentierung begonnen hat.

r: nicht verwendet

n: 1 falls erfolgreich, andernfalls 0

Füge Kontrollpunkt hinzu

```
n = APPLICATION_QUERY ("MEPFLEXIBLESEGMENT", "AddControlPoint(x; y; z)", r)
```

Übergibt einen Kontrollpunkt an das Addon.

AddControlPoint:

- x: X Koordinate des Kontrollpunkts
- y: Y Koordinate des Kontrollpunkts
- z: Z Koordinate des Kontrollpunkts

r: nicht verwendet

n: 1 falls erfolgreich, andernfalls 0

Füge Richtungs- und Breiten-Vektor hinzu

```
n = APPLICATION_QUERY ("MEPFLEXIBLESEGMENT",
                       "AddDirectionAndWidthVector(i; dx; dy; dz; wx; wy; wz)", r)
```

Übergabe der Richtungs- und Breiten-Vektoren der Splineenden an das Addon. Es wird 2 Mal aufgerufen.

AddDirectionAndWidthVector:

- i: ID des Anschlusses (1: 0. Anschluss, 2: 1. Anschluss etc.)
- dx: X Komponente des Richtungsvektors des Anschlusses
- dy: Y Komponente des Richtungsvektors des Anschlusses
- dz: Z Komponente des Richtungsvektors des Anschlusses
- wx: X Komponente des Seitenvektors des Anschlusses
- wy: Y Komponente des Seitenvektors des Anschlusses
- wz: Z Komponente des Seitenvektors des Anschlusses

r: nicht verwendet

n: 1 falls erfolgreich, andernfalls 0

Endverteiler

```
DIM d[]
n = APPLICATION_QUERY ("MEPFLEXIBLESEGMENT", "EndSectioning(res)", d)
```

Übergibt das Ergebnis der Endverteilung.

res: Auflösung der Abschnitte

d: Array der Werte:
- `[9*k-8]:` X Position des K-Segmentes
- `[9*k-7]:` Y Position des K-Segmentes
- `[9*k-6]:` Z Position des K-Segmentes
- `[9*k-5]:` X Komponente des Tangentenvektors des K-Segmentes
- `[9*k-4]:` Y Komponente des Tangentenvektors des K-Segmentes
- `[9*k-3]:` Z Komponente des Tangentenvektors des K-Segmentes
- `[9*k-2]:` X Komponente des Normalvektors des K-Segmentes
- `[9*k-1]:` Y Komponente des Normalvektors des K-Segmentes
- `[9*k]:` Z Komponente des Normalvektors des K-Segmentes

n: Anzahl an Segmenten

MEP Bögen

Dieser Befehl gibt die Geometrie von flexiblen Segmenten zurück. Er besitzt 4 Funktionen, welche mit dem Parameter **parameter_string** abgerufen werden können:

Start des Abschnitts

```
n = APPLICATION_QUERY ("MEPBEND", "GetBendTypeNames()", d)
```

d: Typennamen der Bögen (Beispiele aus der INT-Version)

```
"Radius"
"Square Throat"
"Mitered"
"45° Throat with 45° Heel"
"45° Throat with 90° Heel"
"45° Throat with Radius Heel"
"Radius Throat with 90° Heel"
"Pleated"
"Stamped"
"Segmented"
"Segmented Standing Seam"
```

n: 1 falls erfolgreich, andernfalls 0

Parameter-Script

Dieser Befehl kann unterschiedliche Bedingungen des Parameter-Scripts zurückgeben. Derzeit gibt es nur eine Eigenschaft, die er zurückgeben kann - die Unterscheidung des ersten Durchlaufs.

Erster Durchlauf ist im Gange

```
n = APPLICATION_QUERY ("parameter_script", "firstoccasion_in_progress", isFirstRun)
```

Der Rückgabewert dieses Befehls zeigt an, ob der aktuelle Scriptdurchlauf der erste Durchlauf ist oder eine Folge einer vorhergehenden Ausführung des Parameter-Scriptes, welcher einige Parameter verändert hat. Dieser Befehl besitzt keine zusätzlichen Parameter.

Diese Unterscheidung kann wichtig sein, wenn ein Teil des Parameter-Scriptes ein bestimmtes Ereignis - z.B. steuert es das Drücken eines UI_FUNCTION-Buttons.

isFirstRun: Der Rückgabewert zeigt an, ob der aktuelle Durchlauf der erste ist

Tags und Kategorien

Diese Befehle geben die Ordnernamen, Parameternamen und Parameterwerte der "Kennzeichen und Kategorien"-Tabseite zurück. Die Reihenfolge der Parameter ist die selbe wie auf der Tabseite. Es gibt zwei mögliche **extension_names** in diesem Befehlen:

- **"OwnCustomParameters"** gibt die Parameter des Objektes zurück
- **"ParentCustomParameters"** gibt die Parameter des Elternteils des Objektes zurück

Abfrage Parameter-Ordnernamen

```
DIM folderNamesArray[] ! [idString 1],[shortNameString 1],[longNameString 1],
                       ! ...
                       ! [idString n],[shortNameString n],[longNameString n]
n = APPLICATION_QUERY (extension_name, "GetParameterFolderNames()", folderNamesArray)
```

Gibt die Ordnernamen "Kennzeichen und Kategorien"-Parameter zurück.

folderNamesArray: String-Array welche die Ordnernamen der "Kennzeichen und Kategorien" enthält

n+1: Anzahl der Ordner, welche im Wurzelverzeichnis enthalten sind

Abfrage der Parameternamen

```
DIM parNamesArray[] ! [idString 1],[shortNameString 1],[longNameString 1],
                    ! ...
                    ! [idString n],[shortNameString n],[longNameString n]
n = APPLICATION_QUERY (extension_name,
                  "GetParameterNames(folderNamesArray[i][1])", parNamesArray)
```

Gibt die Namen der "Kennzeichen und Kategorien"-Parameter zurück.

Die erste Spalte des Arrays zurückgegeben in „Abfrage Parameter-Ordnernamen“.

parNamesArray: String-Array welcher die Namen aus "Kennzeichen und Kategorien" enthält

n: Anzahl de Parameter

Abfrage der Parameter

```
n = APPLICATION_QUERY (extension_name, "GetParameter(parNamesArray[i][2])", parValue)
```

Gibt die Werte der Parameter aus "Kennzeichen und Kategorien" zurück.

Die zweite Spalte des Arrays zurückgegeben in „Abfrage der Parameternamen“.

parValue: String-Array welcher die Parameterwerte aus "Kennzeichen und Kategorien" enthält

n: 1 falls erfolgreich, andernfalls 0

Bibliotheken-Manager

Dieser Befehl kann unterschiedliche Eigenschaften des Bibliotheken-Managers zurückgeben.

IES-Dateien

```
n = APPLICATION_QUERY ("LIBRARY_MANAGER", "IES_FILES", ies_files_list)
```

Dieser Befehl gibt eine Liste der .ies Dateinamen zurück, die in der aktiven Bibliothek geladen sind.

Benutzerbilddateien

```
n = APPLICATION_QUERY ("LIBRARY_MANAGER", "USER_IMAGE_FILES", image_files_list)
```

Dieser Befehl gibt eine Liste der Benutzerbilddateien zurück, die in der aktiven Bibliothek geladen sind (Bilddateien, welche nicht in den dedizierten Ordnern liegen, welche folgenden Namen enthalten: [TImg]*, [BImg]*, [UImg]* oder [HImg]*)

GDL Style Guide

Einführung

Dieses Dokument enthält den GDL Coding Standard von GRAPHISOFT, welcher im wesentlichen die formalen Anforderungen für das Schreiben von Quellcode setzt.. Es beschreibt außerdem einige Regeln und Empfehlungen bezüglich des Inhalts. Sie müssen diese Regeln befolgen, um überschaubare Scripte zu erzeugen; standardmäßig ist jeder deklarierte oder imperative Satz eine Regel, es sei denn, es ist ausdrücklich als "Empfehlung" (oder vermeidbar, optional) gekennzeichnet.

Dieses Dokument wurde erstellt, um ein einheitliches Format des GDL -Scripting zu etablieren.. Die GDL-Sprache ist bezüglich der Groß/Kleinschreibung unsensitiv und auch in Fällen der meisten Whitespace-Zeichen. Als Ergebnis davon gibt es eine große Menge an Kodierungs-Standards und -Praktiken. Das ist dann nicht tolerierbar, wenn unterschiedliche Praktiken im gleichen Projekt oder der gleichen Firma angewendet werden.. Die folgenden Abschnitte beschreiben den GRAPHISOFT-Firmenstandard, der eine reine Empfehlung für nicht mit GRAPHISOFT verwandte Entwicklern darstellt. Die angesprochene Formatierung wird auch in Zukunft nicht in die Einschränkungen der GDL-Sprache aufgenommen werden.

Namensgebung-Konvention

Allgemeine Regeln

Wegen der Subtype Hierarchie erben die untergeordneten Bibliothekselemente automatisch alle Parameter der übergeordneten Elemente. (Mehr über Subtypen 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 verantwortlich, 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 schauen Sie sich die ARCHICAD-Bibliothekselemente als Referenz an. Verwenden Sie diese Parameter in Ihren GDL-Scripts, um eine umfassende Kompatibilität mit GRAPHISOFT-Bibliotheken sicherzustellen.

FM_ ist reserviert für ArchiFM (z. B. FM_Type).

Variablennamen

Variablen- und Parameternamen sollten entsprechend der Funktion des Parameters vergeben werden.

mixedCase: beginnt mit einem Kleinbuchstaben; jedes neue Wort sollte mit einem Großbuchstaben anfangen. z.B.: `size, bRotAngle180, upperLeftCorner`

Verwenden Sie keine Variablennamen mit 1 oder 2 Buchstaben - niemand weiß später, was Sie damit meinten.

Sie sollten ein Präfix bei generell verwendeten Variablennamen benutzen, um allgemeine Kategorien zu kennzeichnen. Dies kann zeitsparend sein, wenn jemand den Typ einer Variablen oder eines Parameters herauszufinden versucht. Vergessen Sie nicht, das Präfix zu ersetzen, wenn Sie die Bedeutung einer Variable ändern.

Tabelle 6. Präfixe von Variablennamen

Präfix	Bedeutung	Beispiel
i	allgemeiner Ganzzahlwert / Ganzahlindex	`iSteigungen`
n	Ganzzahlwert - Menge von etwas	`nSteigungen`
b	Boolescher Wert	`bHandlauf`
st	Stringtyp-Wert	`stPaneelTypen`
x	X Koordinate eines Punktes	`xStabPos`
y	Y Koordinate eines Punktes	`yStabPos`
Stift	Stiftfarbe	`penKontur`
lt	Linientyp	`ltKontur`
Schraffur	Schraffurtyp	`filHauptKoerper`
mat	Materialtyp	`matDeckel`

Die Verwendung von Unterstrichen (_) wird zur Unterscheidung von Variablen und Parametern im gleichen Script empfohlen: verwenden Sie ein einfaches Unterstrich- (_) Präfix für eine Variable in einem Script, und doppelte(__) für eine Variable, welche in einer Subroutine deklariert und nur dort verwendet wird. Verwenden Sie keine Unterstriche am Anfang von Parameternamen, oder um Worte in einem Namen zu unterteilen. Historische Namen, die aus Subtypen stammen, bilden eine Ausnahme.

Beispiel

```
_iDoorTypes = iDoorTypes
! die Variable "_iDoorTypes" erhält ihren Wert vom Parameter "iDoorTypes"
gosub "beispielScript"
end

"beispielScript":
    __iDoorTypes= _iDoorTypes * 3
    ! die Subroutine "__iDoorTypes" erhält ihren Wert aus dem 3fachen Wert der Variable
 ! "_iDoorTypes"
return
```

Groß- und Kleinschreibung

- Befehle sollte durchgängig nur in Kleinbuchstaben oder Großbuchstaben geschrieben werden, ganz nach Ihrem Geschmack. **GRAPHISOFT empfiehlt Kleinschreibung.**
- GDLs Globale Variablen sollten immer in Großschreibung geschrieben werden, wegen der einfacheren Erkennbarkeit im Script.
- Folgende Schlüsselwörter sollten klein geschrieben werden: `call, goto, gosub, parameters`.

Ausdrücke (Expressions)

- Leerzeichen sollten vor und hinter den folgenden Binäroperatoren verwendet werden:
  - arithmetische: *****, **/**, **%** (mod), **+**, **-**, **^** (**)
  - logische: **&** (and), **|** (or), **@** (excluding or), **=**, **<>** (#), **<**, **<=**, **>**, **>=**
  - Zuordnung: **=**

  z.B.:

  ```
  a = (b + c % d) * e
  ```
- Keine Leerzeichen sind erlaubt vor und hinter den folgenden unären Operatoren:
  - Indizierung: `array[25]` Hinweis: vermeiden Sie Leerzeichen innerhalb von eckigen Klammern (`array[ 5]`) um die Suchfunktion besser nutzen zu können
  - logisches "not": `not(x)`
  - unäres minus, unäres plus: `-x, +x`
- Funktionsaufrufe sollten ein Leerzeichen vor der öffnenden Klammer der Parameterliste besitzen, und hinter jedem Komma, welches die Parameter trennt:

  ```
  abs (grundLaenge)
  ```

```
minimum = min (a, 25 * b, c)
```

- Wenn man mit einer Konstanten vergleicht (e.g. `i = 5`) sollte die Konstante der zweite Operand sein.
- Wenn man Werte zu Booleschen Variablen zuweist, sollte der logische Ausdruck in Klammern stehen:
  `bBoolWert = (i > j)`
- Verwenden Sie kein Boolesches Ergebnis in Form einer logischen Negation von ganzzahligen Werten oder Variablen. Verwenden Sie z.B anstelle von `if not(iIntVal) then` bitte folgendes: `if iIntVal = 0 then`. (Selbstverständlich können boolesche Variablen und Ausdrücke negiert werden, z.B. `if not(bBoolVal) then`).
- Vergleichen Sie keine Booleschen Variablen und Ausdrücke auf wahr oder unwahr; verwenden Sie den Wert von einer Booleschen oder dessen negierten Wert:

```
bBoolWert = 1
if bBoolWert then          ! anstelle von; if bBoolWert = 1
    ...
endif
if not(bBoolWert) then     ! anstelle von; if bBoolWert = 0
    ...
endif
```

- Komplexe Ausdrücke (z.B. `woand` und`or` oder beide vorhanden sind) sollten zur Klärung des Vorrangs in Klammern gesetzt werden.
- Setzten Sie auch Klammern um selten genutzte Operator-Kombinationen.
- Logische Ausdrücke, welche viele Teile enthalten, sollten auf mehrere Zeilen verteilt werden, und Sie sollten auch Sub-Ausdrücke oder logischen Operatoren untereinander ausrichten:

```
bErgebnis = (bWert1 & bWert3 & not(bWertMitLangemNamen)) | \
          (not(bWert1) & not(bWert3) & bWertMitLangemNamen) | \
          (not(bWertl2) & bWert3 & not(bWertMitLangemNamen)) | \
          (bWert2 & not(bWert3) & bWertMitLangemNamen)
```

Kontrollfluss-Ausdrücke

if - else - endif

Vermeiden Sie die Verwendung der 1-Zeilen-Form von konditionellen Ausdrücken.

Um die Codelesbarkeit zu verbessern, ist es wesentlich, die Hierarchie von verschachtelten Ausdrücken klar zum Ausdruck zu bringen. Das folgende Beispiel zeigt die empfohlenen Tab-Einzüge von Codeblöcken.

```
if bedingung1 then
    ausdruck1
    ...
    ausdruckn
else
    ausdruckn+1
    ...
    ausdruckn+m
endif

if bedingung2 then
    if bedingung3 then
        ...
    else
        ...
    endif
    if bedingung4 then
        ...
    endif
else
    ...
endif
```

for - next, do - while, while - endwhile, repeat - until

Um die Codelesbarkeit zu verbessern, ist es wesentlich, die Hierarchie von verschachtelten Ausdrücken klar zum Ausdruck zu bringen. Das folgende Beispiel zeigt die empfohlenen Tab-Einzüge von Codeblöcken.

```
for i = initialWert to endWert
    ausdruck1
    ...
    ausdruckn
next i

do
    ...
    for i = initialWert to endWert
        ausdruck1
        ...
        ausdruckn
    next i
    ...
    bKondition = ...
    ...
while bKondition
```

Subroutinen

Codeschnipsel, welche mehr als einmal benötigt werden, sollten in Subroutinen untergebracht werden. Dieses macht spätere Korrekturen weniger riskant, und der Code wird strukturierter. Das Label der Subroutine sollte mit seiner Funktion korrespondieren. Verwenden Sie keine Nummern als Namen, das macht den Code unleserlich. In Subroutinen erklärte und nur dort verwendete Variablen sollten mit einem doppelten Unterstrich beginnen.

Stil (kursiv Texte sollten implizit ersetzt werden):

```
! ============================================================================
!   Kurze Beschreibung der Funktionalität
! ----------------------------------------------------------------------------
! Input Parameter:
!   par1:    Kurze Beschreibung (typ)
!   par2:    Kurze Beschreibung (typ)
!   ...
! Output:
!   par1:    Kurze Beschreibung
!   ...
! Remark:
!   Anmerkungen vom Caller
!   Beschreibung der wichtigsten Punkte der Einbindung
! ============================================================================

subroutine_titel:
    ! body
return
```

Sie sollten den Textblock der Subroutine um ein Tabfeld eingerückt nach rechts setzen.

Sie sollten 2 leere Zeilen hinter dem abschließenden '`return`' lassen.

Sie sollten einen Ausdruck pro Zeile schreiben.

Subroutinen sollten wenn möglich nicht länger als 1-2 Bildschirmhöhen sein (um die 80 Zeilen) .

Prüfen Sie alle eingehenden Parameter auf Gültigkeit und/oder erläutern die Einschränkung im Kommentar

Die Parameter`call` und `parameters` schreiben Sie in Kleinbuchstaben.

Kommentare schreiben

Die Kommentarsprache sollte Englisch sein; vereiden Sie Schimpfworte.

Sie sollte für Kommentare folgenden Stil verwenden

Script-Kopf

Das ist nur ein Vorschlag

```
! <Initialen der Kontaktperson>
! ============================================================================
!   Ein Beschreibungssatz über den Zweck des Scripts
! ----------------------------------------------------------------------------
! Input Parameter:
!   par1:    Beschreibung des Parameters (Ganzzahl)
!   par2:    Beschreibung des Parameters (1 / -1)
!   par3:    Beschreibung des Parameters (0 / 1)
!   ...
! Output: [falls ein Makro Werte zurückgibt]
!   [1]:    Beschreibung des Wertes (typ)
!   [2]:    Beschreibung des Wertes (typ)
!   [ ... NSP]: originale Stackelemente
! Anmerkung:
!   Längere Beschreibung.
!   Hinweis für den Caller
! ============================================================================
```

Jeglicher Code darf erst hierauf folgen.

Abschnittsbegrenzung

```
! ============================================================================
! Abschnittsname
! ============================================================================
```

Die Länge der vollständigen Kommentarzeile ist 80 Schriftzeichen lang.

Bei Subroutinen sollten Sie immer die Bedeutung von nicht-trivialen Parametern erklären und deren Rückgabewerten. Zeigen Sie z.B. immer den Wertebereich an (beginnt mit 0 oder 1, jeder spezielle Wert, etc.).

Beispiel in „Subroutinen“

Sie sollten immer mit dem Schlüsselwort TODO anzeigen, falls es etwas nicht fertiggestelltes gibt, es ist dann leichter, die Stelle wiederzufinden:

```
n = 5 ! TODO: Startvariable setzen; sie wird aus der Länge berechnet
```

Sie können außerdem optionale Abschnittsbeschreibungen zwischen den Zeilen des Quellcodes einfügen, beginnend bei der aktuellen Tabeinrückungstiefe. Sie können auch kurze Erläuterungen am Ende einer Codezeile einfügen, indem Sie einen Tabsprung am Ende der Zeile einfügen; oder, falls es mehrere davon gibt, können Sie diese mit Tabs übereinander ausrichten.

Sie sollen immer Kommentare schreiben:

- Bei unüblichen Lösungen
- Falls es für andere die Lesbarkeit und das Verständnis des Codes verbessert.
- Falls etwas für andere nicht erlaubt ist oder aber empfohlen wird

Optional (andere werden sich bedanken) wenn es in irgendeiner Weise hilfreich ist.

Unterbrechen Sie nicht den Rhythmus oder die Vorzüge des Codes durch die Kommentare.

Wenn Sie dann einen zusammenhängenden Codeblock kommentieren , können Sie folgendes Format verwenden:

```
! == code block name ===[
ausdruck1
...
ausdruckn
! ]=== code block name ===
```

Dies erleichtert die optische Trennung des Blockes; außerdem unterstützen einige Editoren die Suche nach den passenden eckigen Klammern per Shortcut. (e.g.: `ctrl + ]` in Microsoft Visual Studio)

Kommentieren Sie das Ende von 'if' Ausdrücken, falls es viele Zeilen zwischen `if` und `endif` gibt, wie folgt:

```
if bedingung1 then
    ...
    if bedingung2 then
        ...
        ! many statements
        ...
    endif   ! if bedingung2

endif   ! if bedingung1
```

Einige Scripttypen (besonders Vorwärts- und Rückwärts-Migrations-Scripte) besitzen eine empfohlene Form von Trennzeichen und Struktur. Sehen Sie sich dazu ein Beispiel in der ARCHICAD-Bibliothek an. „Grundlegende Technische Standards“.

Scriptstruktur

Stellen Sie Ihren Editor auf 4 Zeichen lange Tabsprünge ein. Leerzeichen sollten niemals verwendet werden um Zeilenvorsprünge zu erzielen. Verwenden Sie stattdessen Leerzeichen zur internen Anpassung innerhalb eines zusammenhängenden Ausdrucks.

Die maximale Zeilenlänge beträgt 120 Zeichen. Ausdrücke sollten möglichst nicht so dicht an diesen Wert herankommen. Im Fall, das die Ausdrücke zu lang sind, erhalten Sie eine Warnung.

Alle Referenzen auf Dateinamen im Script sind "case sensitive" (Groß-Klein-empfindlich), ebenso die Dateierweiterungen.

Vielfach verwendete Werte sollten direkt vor dem sie verwendenden Codeblock berechnet werden wenn sie gut zu platzieren sind, andernfalls sollte man sie am Anfang des Scriptes unterbringen. Es gibt keinen Kompromiss. Berechnen Sie komplexe Werte nur ein einziges Mal - um Rechenzeit zu sparen - indem Sie diese in Variablen speichern (aber vergeuden Sie keine unnötigen Variablen) oder im Transformations-Stack (add, rot, etc.).

Die Objekt-Scripte sind linear, was sie klarer macht. Subroutinen sollten nur dann angelegt werden, wenn eine Berechnung oder ein Modellerzeugungssegment mehr als einmal benötigt wird, oder für eine bessere Lesbarkeit des Scriptes. Vermeiden von der mehrfachen Codierung einer gleichen Sache, ist ein wichtiges Prinzip in allen Programmiersprachen. Redundanz macht spätere Änderungen wesentlich schwieriger.

Versuchen Sie, keine riesigen Auswahlverzweigungen anzulegen; bereiten Sie stattdessen die Daten für eine Berechnung oder Befehlsgenerierung in kleineren Auswahlblöcken vor, wodurch Sie es einfacher haben, Redundanz zu vermeiden.

Schlechte Lösung

```
if bOnHomeStory then
    line_type ltContour
    fill gs_fill_type
    poly2_b 5, 3, gs_fill_pen, gs_back_pen,
        left,   0,      1,
        left,   -depth, 1,
        right,  -depth, 1,
        right,  0,      0,
        left,   0,     -1
endif
if (bOnUpperStory or bOnAboveUpper) and bDrawContBB then
    line_type ltBelow
    fill fillTypeBelow
    poly2_b 5, 3, fillPenBelow, fillBackBelow,
        left,   0,      1,
        left,   -depth, 1,
        right,  -depth, 1,
        right,  0,      0,
        left,   0,     -1
endif
```

Die Definition der Geometrie ist doppelt vorhanden! Es könnten noch schlimmer sein, nämlich dann, wenn der Abstand zwischen den identischen Befehlen größer wäre.

Gute Lösung

```
if bOnHomeStory then
    bPolygon = 1
    line_type ltContour
    fill gs_fill_type
    fillPen = gs_fill_pen
    fillBGPen = gs_back_pen
endif
if (bOnUpperStory or bOnAboveUpper) and bDrawContBB then
    bPolygon = 1
    line_type ltBelow
    fill fillTypeBelow
    fillPen = fillPenBelow
    fillBGPen = fillBackBelow
endif
if bPolygon then
    poly2_b 5, 3, fillPen, fillBGPen,
        left,   0,      1,
        left,   -depth, 1,
        right,  -depth, 1,
        right,  0,      0,
        left,   0,     -1
endif
```

Bereiten Sie Ihre Scripte für die Lokalisierung vor.

Verwenden Sie`"asdf"` für nicht lokalisierte Strings (z.B. Makro-Aufrufe) und `` `asdf` `` für lokalisierte (z.B. String-Konstanten, Parameter Values).

GRUNDLEGENDE TECHNISCHE STANDARDS

Einführung

Das Erscheinen neuer ARCHICAD® Länderversionen, das Wachsen der GRAPHISOFT-Produktlinie und das Portal BIMcomponents® haben die Nachfrage nach GDL-Objekten und Objektbibliotheken dramatisch gesteigert. Als Folge hiervon haben viele unabhängige oder Drittanbieter als GDL-Programmierer begonnen, Bibliotheken oder Objekte für GRAPHISOFT zu entwickeln.

Um diese Objekte untereinander kompatibel zu machen, sind grundlegende Richtlinien notwendig, welche einen garantierten Qualitätsstandard, welcher von GRAPHISOFT-Produkten erwartet wird, vorgeben. Zweck dieses Abschnitts ist es, einen GDL-Entwickler mit nützlichen Tipps und Tricks, Beispielen, Beschreibungen der bisher undokumentierten und neuen ARCHICAD-Features, in die Erstellung von Objekten zu einzuführen.

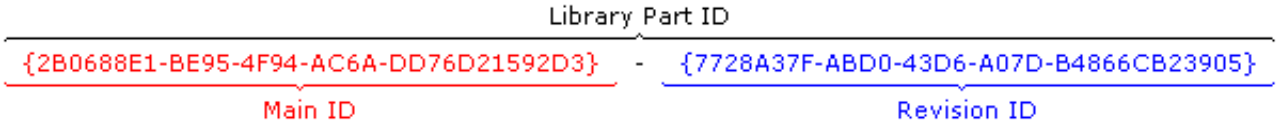
Bibliothekselement-Format

Dateierweiterung

Die meisten GDL-Bibliothekselemente werden mit der Dateiendung *.gsm gesichert und mit Hilfe ihres Subtypes in ARCHICAD unterschieden. Es gibt die spezielle Endung, *.gdl (GDL Script-Datei) für die MASTER_GDL/MASTEREND_GDL Dateien. ARCHICAD behandelt jede GDL Skript-Datei, die mit "MASTER_GDL...." oder "MASTEREND_GDL..." in ihrem Dateinamen beginnt, auf eine besondere Weise. Diese Dateien können dazu verwendet werden um Attribut-Definitionen zu laden, Linientypen und Materialien zu definieren etc. (weitere Informationen hierzu im **The GDL Script Analysis auf Seite 10**).

Identifizierung

Der Identifizierer



Die ID besteht aus zwei Teilen, jeder davon mit 36 alphanumerische Werten. Die ersten 36 Werte repräsentieren die **Main ID** und die letzten 36 Werte repräsentieren die **Revision ID**.

- Die Haupt ID wird erzeugt, wenn das Bibliothekselement das erste Mal gesichert wird. Außerdem wird sie geändert, wenn das Bibliothekselement erneut gesichert wird unter der Verwendung des Befehls “Sichern als”.
- Die Revisions-ID wird auch erzeugt wenn das Bibliothekselement zum ersten Mal gesichert wird, aber sie wird modifiziert, wenn das Bibliothekselement erneut gesichert wird unter Verwendung des Befehls „Sichern“.. Verwendet man das LP XML-Konverter Werkzeug, ändert die Kompilierung die Revisions-ID und belässt die Haupt ID natürlich unverändert.

Dies bedeutet, dass die Haupt ID ein Bibliothekselement in seiner Funktion identifiziert und die Revisions-ID dabei hilft, die Änderungen eines Objektes voneinander zu unterscheiden. Das ganze soll anhand eines praktischen Beispiels erläutert werden.

Identifizierung von Bibliothekselementen

Wenn man ein Bibliothekselement in ARCHICAD platziert, speichert das Programm die Referenz auf das Objekt mit Hilfe der ID und verwendet den Namen nur für Objekte ohne ID (Bibliothekselemente, die vor ARCHICAD 8 gespeichert wurden und GDL-Dateien). Im Falle, dass Bibliothekselemente aus Versionen vor ARCHICAD 8 kommen, gab es keinerlei Art einer GUID. Wenn also ein solches Objekt in ARCHICAD eingesetzt wird, wird die ID von ARCHICAD mit Nullen ausgefüllt.

Wenn eine Bibliothek geladen wird, verwendet ARCHICAD die folgenden hierarchischen Kriterien, um geladene Bibliothekselemente von denen, die bereits im Projekt platziert sind, voneinander zu unterscheiden:

- 1. Falls die gespeicherte ID gültig ist:
  - ARCHICAD versucht eine exakte Übereinstimmung beider Teile der ID zu erhalten
  - Schlägt dies fehl, versucht ARCHICAD den ersten Teil der ID zur Übereinstimmung zu bringen, die Haupt-ID.
  - Für den Fall, dass keine Übereinstimmung gefunden wird, beginnt ARCHICAD die Migrationstabelle anderer Elemente zu überprüfen, um einen Ersatz zu finden.
  - Falls schließlich Dateien geladen werden, die vor ARCHICAD 12 gespeichert wurden, versucht ARCHICAD, eine Übereinstimmung bei dem Bibliothekselement-Namen zu finden.
- Falls die gespeicherte ID Null ist, versucht der Identifizierungs-Vorgang eine Übereinstimmung ausschließlich durch den Namen zu erhalten.

Der selbe Vorgang wird ausgeführt, wenn nach Makros in platzierten Objekten gesucht wird, da jedes Bibliothekselement eine Nachschlage-Tabelle für die von ihr aufgerufenen Makro-GUIDs besitzt. Normalerweise wird diese Tabelle zusammengestellt, wenn ein Objekt gesichert wird, das Makroaufrufe enthält, unter Verwendung einer Namen-basierten Suche in der aktuell geladenen Bibliothek.

Wie erhält man die exakte GUID eines Bibliothekselementes?

Hierzu muss man das Subtype Hierarchie Dialogfenster eines Objektes öffnen. In diesem Dialogfenster sieht man die Subtype Hierarchie der aktuell geladenen Bibliothek als Baumdarstellung. Am unteren Rand des Dialogfensters werden folgende Hauptattribute des ausgewählten Bibliothekselements angezeigt: Name, Version, ID, Pfad und Kennzeichen, ob das Objekt Vorlage oder platzierbar ist.

Dieses Dialofeld erscheint in 3 Zusammenhängen:

- Objekt öffnen mit Subtype... (im Dateimenü)
- Subtype auswählen... (im Editor-Fenster des Bibliothekselementes)
- Alle Objekte platzieren (im Special Menü)

Normalerweise kann man die ID im XML-Format des Bibliothekselementes auslesen (location: xpointer (/Symbol/@UNID)). Um dieses Format zu erhalten, verwenden Sie das LP_XMLConverter Werkzeug.

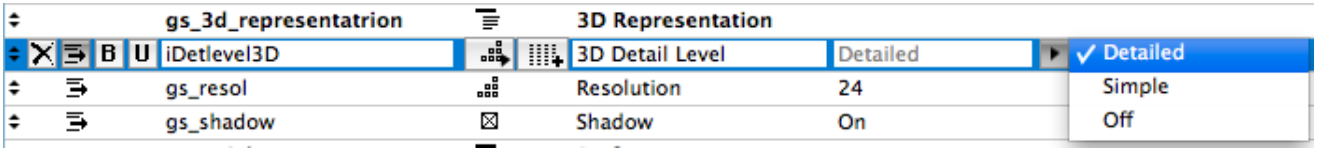
Kompatibilitäts-Probleme

Das allerwichtigste an diesem Thema ist, dass die Haupt-ID den Nutzern der Bibliothek eine gleichbleibende Funktionalität zur Verfügung stellt. Dies bedeutet folgendes: falls Sie ein neues Bibliothekselement veröffentlichen, welches eine Main ID verwendet, welche bereits von einem alten Objekt verwendet wird, dann werden alle alten platzierten Elemente durch das neue Objekt ersetzt, wenn man in ein altes Projekt die neue Bibliothek lädt. Dies widerspricht den Nutzererwartungen, wie z.B., dass es keine Änderungen in den Objektparametern und -Funktionen gibt. Falls Sie den Namen oder die Funktion von alten Parametern ändern möchten, sollten Sie unbedingt eine neue Haupt-ID erzeugen, und die Migrations-Scripte verwenden, um Mehrdeutigkeiten und unerwarteten Datenverlust zu vermeiden. Stellen Sie sicher, dass diese Main ID einzigartig ist - nicht identisch mit irgendeiner anderen ID in der Bibliothek.

Beachten Sie, dass das Umbenennen eines Objektes dieses für ARCHICAD nicht inkompatibel mit seinem alten Ego macht, solange seine Main ID identisch erhalten bleibt. Ähnliches gilt für den Fall, dass Sie den Namen eines vorhandenen Objektes an ein neues Objekt (mit neuer Main ID) übergeben: beide sind nicht kompatibel.

Dieser Aspekt betrifft auch die nationalen Varianten von Bibliotheken. Falls es bei einem Objekt Texttypparameter in Form einer Auswahlliste gibt, variieren die relevanten Werte zwischen den nationalen Versionen. Ein Beispiel: ist man sich des Problems nicht bewusst, führt das Laden einer Deutschen Plandatei mit einer Dänischen Schwesternbibliothek dazu, dass bereits platzierte Elemente sich verändern, da einige Kontrollparameter sinnlose Werte enthalten. Für das Problem gibt es zwei Lösungen. Der einfachste Weg ist zu sagen, dass die Deutsche und die Dänische Bibliothek nichts miteinander zu tun haben und die Haupt-IDs in den nationalen Varianten der Bibliothek konsequent zu ändern. Die zweite – und benutzerfreundliche – Lösung ist, Kontrollparameter vom Typ Ganzzahl, die als String funktionieren, zu erzeugen (siehe `VALUES`). Diese Ganzzahl-Parameter sind das Entscheidende, die sichtbaren Texttyp-Beschreibungen sind dann bloß eine Eingabemethode für diese (daher können Lokalisierungen unterschiedlich sein, aber die wahre Bedeutung wird die gleiche bleiben). Wenn Sie ein Script schreiben, sollten Sie Ganzzahl-Parameterwerte verwenden.

Das folgende Code-Beispiel verfügt über ein Detaillevel-Ganzzaahl-Parameter, der als String-Typ fungiert:



```
! Master script:
dim stDetlevel3DDesc [3]
stDetlevel3DDesc[1]=`detailliert`
stDetlevel3DDesc[2]=`einfach`
stDetlevel3DDesc[3]=`aus`

! iDetlevel3D Konstanten
DETLEVEL3D_DETAILED = 1
DETLEVEL3D_SIMPLE   = 2
DETLEVEL3D_OFF      = 3

! Parameter-Script:
values{2} "iDetlevel3D"  DETLEVEL3D_DETAILED, stDetlevel3DDesc[1],
                         DETLEVEL3D_SIMPLE,   stDetlevel3DDesc[2],
                         DETLEVEL3D_OFF,      stDetlevel3DDesc[3]
```

Migratings-Elemente

Es ist möglich, eine Verbindung zwischen dem alten und dem neuen Ego, eine aktualisierte Version (mit neuen Haupt-ID) eines Bibliothekselementes mithilfe der Migrationsscripte zu erhalten, („Das Vorwärts-Migration Script“, „Das Rückwärts-Migration Script“) und „Migrationstabelle“.

In diesen Scripten können Sie definieren, welches eine Bibliothekselement welche andere ersetzt (indem die alte und die neue Haupt-ID), und wie die Parameterwerte des neuen Objektes basierend auf den alten Werten aktualisiert werden können (oder vice versa). Sie können Regeln für die Migration nur unter bestimmten Bedingungen passieren lassen. Wenn das Subjekt der Migration diese Bedingungen erfüllt, ist das Upgrade oder Downgrade möglich, andernfalls steht dies Option nicht zur Verfügung.

Generell ist es möglich, dass das Subjekt der Migration einen oder mehrere Nachfahren (oder Vorfahren in der Rückwärts-Migration) in Abhängigkeit von den Parameter-Einstellungen besitzt. Allerdings verhält es ich bei der Migration von Raumstempeln etwas anders. Ein Raumstempeltyp kann mit vielen Raumkategorien verknüpft werden. Aber jede Kategorie kann nur eine Art von Raumstempel verwenden. Bei der Migration eines Raumstempels bleibt die Kategorie die gleiche. Wenn die Route der Migration divergiert ("Stempel Alt" wir zu "Stempel Neu 1" geupgradet, oder zu "Stempel Neu 2", abhängig von unterschiedlichen Parametereinstellung), ist es möglich, eine Kategorie -mit zwei unterschiedlichen Raumstempeln verknüpft- zu bekommen. Da dies ein gültiges Ergebnis in Bezug auf den Migrationsprozess ist, ergibt sich eine inkonsistente Situation für ARCHICAD. Stellen Sie sicher, dass Sie Raumstempel nur auf direktem Wege migrieren, um dies zu vermeiden.

Allgemeine Scriptingfragen

Numerische Typen - Präzision

Vor ARCHICAD 9 wurde alle numerischen Werte intern als Gleitkommazahlen gespeichert, was zu unpräzisen Werten führte. Dies bedeutete, dass Ganzzahlwerte - ein wenig - unpräzise gespeichert wurden. Seit ARCHICAD 9 sollten Ganzzahlen - und somit GDL-Parametertypen, welche am besten mit Ganzzahlen beschrieben werden können - korrekterweise intern als Ganzzahlen gespeichert werden.

Parameter types internally stored as an Integer:

```
Integer,
Boolean,
Material,
Line type,
Fillpattern,
Pencolor,
Intensity (Light)
```

Parameter types internally stored as a Floating-point number:

```
Length,
Angle,
Real,
RGB Color component (Light)
```

GDL-Variablen benötigen nach wie vor keine Typendefinition, der Typ wird während der Interpretation von dem in die Variable geladenen Wert bestimmt. Die Ausgabe eines numerischen Operators hat nun einen Typ. Schlagen Sie für Informationen hierzu in diesem GDL-Referenz-Handbuch nach.

Der Programmierer kann auf sichere Art Ganzzahltypen mit dem Gleichheitszeichen-Operator vergleichen. Tatsächlich werden seit ARCHICAD 9 Warnungen ausgegeben, wenn ein Programmierer versucht, Gleitkommazahlen direkt mit Ganzzahlwerten mit dem Gleichheitszeichen-Operator zu vergleichen. Für Gleichheit-Vergleiche von Gleitkommazahlen verwenden Sie einen kleinen Epsilon-Wert, welcher den Präzisionsgrad des Vergleichs darstellt . Für Gleichheit-Vergleiche von Gleitkommazahlen mit Ganzzahlen verwenden Sie eine `round_int` - Funktion.

Unten werden einige Beispielmethoden beschrieben, mit denen Wertegleichheit zwischen unterschiedlichen numerischen Typen geprüft werden kann:

```
iDummy = 1 * 2
if iDummy = 2 then
    ! gültiger Vergleich, dieser ist wahr, diese Ausdrücke werden ausgeführt
 ...
endif


dDummy = 1.5 + 0.5
if dDummy = 2 then
    ! Sie wissen nie, ob das Ergebnis wahr ist, trauen Sie solchen Vergleichen nicht
 ...
endif


dDummy = 1.1 * 2
if dDummy = 2.2 then
    ! Sie wissen nie, ob das Ergebnis wahr ist, trauen Sie solchen Vergleichen nicht
 ...
endif


! EPS = 0.0001 -> im Master-Script
dDummy = 1.1 * 2
if abs (dDummy - 2.2) < EPS then
    ! gültiger Vergleich, dieser ist wahr, diese Ausdrücke werden ausgeführt
 ...
endif


dDummy = 1.5 * 2
if round_int (dDummy) = 3 then
    ! gültiger Vergleich, dieser ist wahr, diese Ausdrücke werden ausgeführt
 ...
endif
```

Trigonometrische Funktionen

Während des GDL-Programmierens benötigen Sie ggf. verschiedene Trigonometrische Funktionen. Die folgenden Funktionen sind direkt aus GDL heraus verfügbar: `cos, sin, tan, acs, asn, atn`.

Alle anderen Funktionen können einfach daraus wie folgt abgeleitet werden.

```
Secant Sec(X) = 1 / cos(X)
Cosecant Cosec(X) = 1 / sin(X)
Cotangent Cotan(X) = 1 / tan(X)
Inv. Sine Arcsin(X) = atn(X / Sqr(-X * X + 1))
Inv. Cosine Arccos(X) = atn(-X / sqr(-X * X + 1)) + 2 * atn(1)
Inv. Secant Arcsec(X) = atn(X / sqr(X * X - 1)) + sgn((X) -1) * 2*atn(1)
Inv. Cosecant Arccosec(X) = atn(X / sqr(X*X - 1)) + (sgn(X) - 1) * 2*atn(1)
Inv. Cotangent Arccotan(X) = atn(X) + 2 * atn(1)
Hyp. Sine HSin(X) = (exp(X) - exp(-X)) / 2
Hyp. Cosine HCos(X) = (exp(X) + exp(-X)) / 2
Hyp. Tangent HTan(X) = (exp(X) - exp(-X)) / (exp(X) + exp(-X))
Hyp. Secant HSec(X) = 2 / (exp(X) + exp(-X))
Hyp. Cosecant HCosec(X) = 2 / (exp(X) - exp(-X))
Hyp. Cotangent HCotan(X) = (exp(X) + exp(-X)) / (exp(X) - exp(-X))
Inv. Hyp. Sine HArcsin(X) = log(X + sqr(X * X + 1))
Inv. Hyp. Cosine HArccos(X) = log(X + sqr(X * X - 1))
Inv. Hyp. Tangent HArctan(X) = log((1 + X) / (1 - X)) / 2
Inv. Hyp. Secant HArcsec(X) = log((sqr(-X * X + 1) + 1) / X)
Inv. Hyp. Cosecant HArccosec(X) = log((sgn(X) * sqr(X * X + 1) +1) / X)
Inv. Hyp. Cotangent HArccotan(X) = log((X + 1) / (X - 1)) / 2
```

Hinweis:

```
Logarithmus zur Basis N LogN(X) = log(X) / log(N)
```

GDL-Warnungen

So wie jede andere Programmiersprache, besitzt GDL eine zu befolgende Syntax und Logik. Falls es einen Fehler in der Syntax gibt, erhält der Programmierer eine Fehlermeldung. Falls es etwas verwirrendes gibt oder beim Ablauf des Scriptes unerwartete Dinge passieren, wird eine GDL-Warnmeldung ausgegeben.

Sie können wählen, WO Sie diese Hinweise sehen möchten; stellen Sie das ein unter Optionen/Arbeitsumgebung/Modellneuaufbau-Optionen:

- **Bei Fehlermeldung unterbrechen**: eine Dialogbox springt bei jedem Problem auf
- **Protokollieren**: die Hinweise werden ins Protokoll-Fenster geschrieben

Sie können entscheiden, WAS als Warnmeldung ausgegeben werden soll: diese Einstellung ist im GDL Entwickler Menü verfügbar, mit der Bezeichnung **"Unterbrechung mit Bibliothekselement-Fehlermeldungen"**. Wenn aktiviert, werden nicht nur Fehler, sondern auch Warnungen ausgegeben, entsprechenden den WO-Einstellungen.

Sie können auch entscheiden, WANN Sie Warnmeldungen sehen möchten. Dies kann ebenfalls im GDL Entwickler Menü eingestellt werden, unter**"Immer GDL-Warnungen ausgeben"**. Aktiviert man dieses Feature, werden jedesmal, wenn GDL ausgeführt wird, auf jeden Fall die Warnungen erzwungenermaßen ausgegeben. Belässt man es deaktiviert, verhalten sich die Warnmeldungen wie gewohnt.

Beachten Sie, dass eine Kombination der oben genannten Schalter zu Problemen führen kann: hat man z.B. **"Immer GDL-Warnungen ausgeben"** und **"Bei Fehlermeldung unterbrechen"** zusammen aktiviert, kann dies die Ausführung von so etwas simplem wie das Bewegen von editierbaren Hotspots verhindern, weil dann ständig Dialogfelder aufpoppen.

Drückt man im GDL Editor auf **"Script prüfen"** und es gibt ein Problem im Script, welches die aktuellen Parametereinstellungen verwendet, bekommt man immer eine Warnung oder Fehlermeldung in einem Popup-Fenster. Verwendet man den `PRINT` Befehl oder den GDL Debugger, kann dies helfen, Fehler zu finden, die anders schwer zu finden wären.

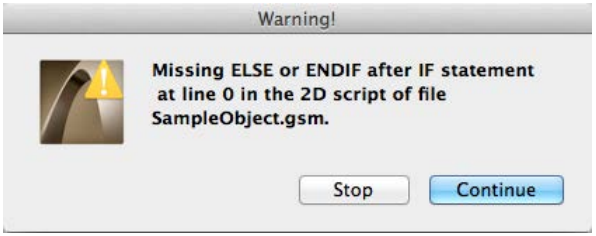
Beachten Sie, dass die Zeilennummern in den GDL-Warnungen sich auf das Script beziehen, welches das Problem enthält.

Parsing-Fehler müssen mit besonderer Sorgfalt behandelt werden. Diese bezeichnen die erste Zeile, ab der das Parsen unmöglich wird, aber die eigentliche Probleme können einige Zeilen vorher verborgen sein.

Beispiel

Der Interpreter entdeckt die fehlenden Ausdrücke erst beim `endif` und hält dort an, obwohl das Problem augenscheinlich um die Zeile 4 herum liegt, wo ein `endif` wirklich fehlt.

```
if condition1 then
    if condition2 then
        ! tu etwas


    ! tu etwas - ABER WIR VERMISSEN EIN 'endif'
else
    ! ein potenziell langer Codeblock
endif
```



Hier folgen ein paar Beispiele der zuletzt entwickelten Warnmeldungen, mit entsprechenden Erläuterungen:

Warnung	**Mögliche Erklärung**
Einfacher Parameter als Array umdeklariert	in einem Objekt einen einfachen Parameter deklariert und diesen als Array in einem Makroaufruf verwendet
Undefinierte parentId "id", die bei der Definition einer UI_PAGE verwendet wurde	fehlende parent ID in einer Tabpage-Hierarchie
Sicht- oder Projektabhängigen globalen "globalName" im Parameter-Script verwendet	siehe „Globale Variablen“
REQUEST "requestName" im Parameter-Script verwendet	siehe „REQUEST Optionen“
Application query "applicationQueryName" im Parameter-Script verwendet	siehe „Application Query Optionen“
Möglicher unerwünschter Parametertyp-Wechsel	ein Parameter erhält einem Wert, der vom Ursprungstyp nicht unterstützt wird

Hotspot und Hotline IDs

Zweck der hotspot / hotline / hotarc Identifikation

In ARCHICAD wurde die hotspot / hotline / hotarc Identifikation eingeführt, um das assoziative Bemaßen in Schnitten zu ermöglichen. Mittels dieses Features kann ein Bemaßungselement auf alle hotspots/hotlines in einem GDL-Objekt verweisen. Es ist ein ernstes Problem, wenn sich die Anzahl der hotspots/hotlines während unterschiedlicher Parametereinstellungen in einem Objekt ändert.

Problem von hotspots/hotlines der alten Schule

Wenn der Programmierer keine IDS für hotspot/hotline/hotarc festlegt - oder wenn er diese auf Null setzt - ordnet ARCHICAD diesen kontinuierlich steigende Ordnungszahlen zu. Diese Lösung ist in Ordnung für statische Objekte, bedeute jedoch Bemaßungsprobleme wenn einige hotspots/hotlines nach Parametereinstellungen auftauchen oder verschwinden. Das bedeutet, dass die IDs neu geordnet werden, so das sie sich ändern, und die assoziativen Bemaßungselemente - im Schnitt - gehen in die Irre.

Korrektes Scripting für hotspot/hotline/hotarc

Aus allen genannten Gründen sollten Sie den hotspots/hotlines in Ihren Objekte fixe IDs zuweisen. Dies kann man erreichen, indem man eine breites Intervall von einzeln steuerbaren Funktionen für hotspots/hotlines reserviert.

Nehmen wir eine Treppe als Beispiel. Die Einfassungs - hotspots/hotlines können das Intervall [1-100] verwenden, die Handläufe können das Intervall [200-299] und die Stufen [1000-) verwenden. Dies gewährleistet, dass die Bemaßung der Handläufe nicht durcheinandergebracht wird, wenn sich die Stufenanzahl ändert oder wenn nur der Fußpunktanschluss komplexer wird (und dann mehr hotspots/hotlines benötigt).

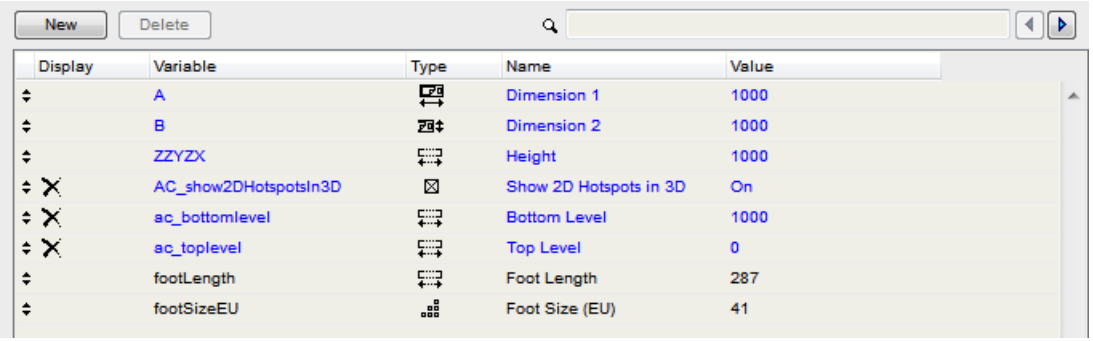
Editierbare Hotspots

Seit ARCHICAD 8 können Sie editierbare Hotspots in Ihren Bibliothekselementen verwenden. Dieses Feature wird in *Bearbeitungsbefehle auf Hotspot-Basis* beschrieben mit Ausnahme einer Möglichkeit.

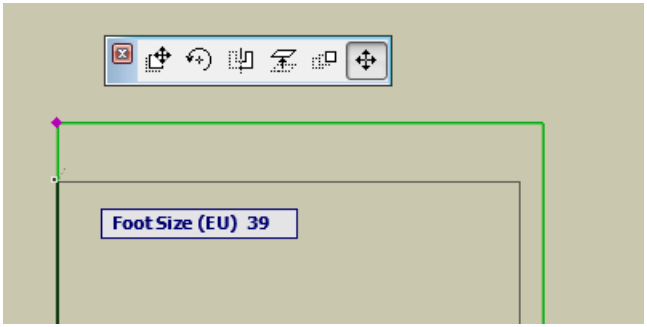
In manchen Fällen möchten Sie vielleicht einen anderen Parameter anzeigen als denjenigen, der editiert wird. Sie Beispielcode unten:

Beispiel Editierbare hotspots - Schuh / Schuhregal

Wir möchten gerne die Größe des Schuhs sowohl in Metern als auch als Schuhgröße anzeigen. Hierfür erzeugen wir 2 Parameter und verknüpfen sie im Parameter-Script. Natürlich kann der Typ des erläuternden Parameters unterschiedlich sein (z.B. Text). Wir betonen, dass der editierbare Parameter die ganze Zeit `footLength` ist, während `footSizeEU` - der angezeigte Parameter - die ganze Zeit über das Parameter-Script aktualisiert werden muss.



2D-Editierung



Parameter-Script

```
DIM lengthValues[10]
DIM sizeValues[10]
for i = 1 to 10
    sizeValues[i] = i + 35
    lengthValues[i] = (i + 35) * 0.007
next i
```

```
values "footLength" lengthValues
values "footSizeEU" sizeValues

if GLOB_MODPAR_NAME = "footLength" then
    parameters footSizeEU = round_int (footLength / 0.007)
else
    if GLOB_MODPAR_NAME = "footSizeEU" or GLOB_MODPAR_NAME = "" then
        parameters footLength = footSizeEU * 0.007
    endif
endif
```

2D-Script

```
rect2 0, 0, footLength * 0.4, footLength   ! oder ein realistischeres Schuhmodell

hotspot2 0, 0,          1, footLength, 1 + 256, footSizeEU
hotspot2 0, footLength, 2, footLength, 2, footSizeEU
hotspot2 0, -0.1,       3, footLength, 3
```

GDL-Ausführungs-Kontext

ARCHICAD lässt das GDL-Objekt über den Kontext, in welchem es angezeigt oder genutzt wird, Bescheid wissen. Die folgenden Globalen Variablen werden für diesen Zweck verwendet:

- GLOB_VIEW_TYPE zur Bestimmung der aktiven Sicht
- GLOB_PREVIEW_MODE zur Bestimmung der aktiven Vorschau
- GLOB_FEEDBACK_MODE zur Anzeige des Editier-Kontextes
- GLOB_SEO_TOOL_MODE zur Anzeige des Kontextes von Solid Element Operationen

Für mögliche Werte beziehen Sie sich bitte auf „Allgemeine Umgebungsinformationen“ und die folgende Liste:

GLOB_VIEW_TYPE = 2 - 2D, Grundriss

Das Modell wird im Standard-2D-Grundriss angezeigt. Für ein 3D-Script bedeutet das, dass das Modell mit Hilfe des Befehls `project2D` in 2D projiziert wird. Dies ist die Hauptverwendung eines Objektes - dieses 2D-Modell muss immer korrekt und effizient sein.

Falls **GLOB_FEEDBACK_MODE = 1** dann wird das Modell mittels Feedback-Linien im 2D-Grundriss während der Editierung von Hotspots des Objektes angezeigt. Dieses Modell wird während der Nutzereingabe viele Male in einer einzigen Sekunde aufgebaut. Die bedeutet,

dass das Modell nur die wesentlichen Teile des Objektes darstellt. Beachten Sie, dass Texte (erzeugt mit dem Befehl `text2`) im Feedback-Modus nicht neu aufgebaut werden - da dies den Output verlangsamen würde.

GLOB_VIEW_TYPE = 3 - 3D Ansicht

Das 3D-Modell wird im Standard-3D-Modell-Fenster angezeigt oder es ist die Quelle eines fotorealistischen Renderings. Diese Ansicht sollte interne Details des Objektes auslassen, außer diese sind nicht ohnehin sichtbar. Dies ist die zweite sehr bedeutende Anwendung eines Objektes - das 3D-Modell muss immer korrekt und effizient sein. Dieses Ziel verlangt ein korrektes äußeres Erscheinungsbild.

Falls **GLOB_FEEDBACK_MODE = 1** dann wird das 3D-Modell während dem Editieren der Hotspots mittels Feedback-Linien im 3D-Fenster angezeigt. Dieses Modell wird während der Nutzereingabe viele Male in einer einzigen Sekunde aufgebaut. Die bedeutet, dass das Modell nur die wesentlichen und sichtbaren Teile des Objektes darstellt.

GLOB_VIEW_TYPE = 4 - Schnitt oder **GLOB_VIEW_TYPE = 5 - Ansicht**

Das 3D-Modell wird in einem Schnitt/Ansicht-Fenster angezeigt. Für diese Sichten sollte das Objekt interne Details erzeugen welche für jede andere Darstellungsart überflüssig sind.

Falls **GLOB_FEEDBACK_MODE = 1** dann wird das Modell mittels Feedback-Linien im Schnitt/Ansicht-Fenster während dem Editieren von Hotspots des Objektes angezeigt. Dieses Modell wird während der Nutzereingabe viele Male in einer einzigen Sekunde aufgebaut. Die bedeutet, dass das Modell nur die wesentlichen und sichtbaren Teile des Objektes darstellt.

GLOB_VIEW_TYPE = 6 - 3D Dokument

Das 3D-Modell wird in einem Axonometrie-Fenster als Zeichnung angezeigt. Dieses verwendet man zur Dokumentation und Vermaßung in 3D.

GLOB_VIEW_TYPE = 7 - Detailzeichnung

Das Modell wird in einem Detailzeichnungs-Fenster verwendet. Das Modell kann konsequent viel detaillierter modelliert werden als in anderen Darstellungsarten. Das 2D- und 3D-Modell werden nicht unterschieden - diese Information kann aus dem Scripttyp heraus.

GLOB_VIEW_TYPE = 8 - Layout

Das Modell wird in einem Layout-Fenster verwendet, mit der Anzeige, wie es im Druck aussieht. Das Modell sollte sein Druck-Bild darstellen. Das 2D- und 3D-Modell werden nicht unterschieden - diese Information kann aus dem Scripttyp heraus.

Falls **GLOB_FEEDBACK_MODE = 1** dann wird das Modell mittels Feedback-Linien in einem Layout-Fenster während dem Editieren des Objektes mittels Hotspots angezeigt. Dieses Modell wird während der Nutzereingabe viele Male in einer einzigen Sekunde aufgebaut. Die bedeutet, dass das Modell nur die wesentlichen und sichtbaren Teile des Objektes darstellt.

GLOB_VIEW_TYPE = 9 - calculation und/oder **GLOB_PREVIEW_MODE = 2 - Listen**

Das 3D-Modell wird für Oberflächen- und Volumenberechnungen durch die Listen-Engine verwendet. Dieser Kontext ist der geeignete Ort um einige Modelländerungen für die Auflistung durchzuführen. Sie können z.B. Extra-Körper erzeugen, um die zu lackierende Oberfläche aufzubringen und die Menge der benötigten Farbe auszugeben. Verwenden Sie die Kombination der 2 Globalen Variablen, um das gewünschte Ergebnis in der Berechnung und Listenerzeugung zu erzielen.

GLOB_PREVIEW_MODE = 1 - Einstellungsdialog

Das Modell wird im Vorschaufenster des Objekteinstellungs-Dialoges angezeigt. Das 2D- und 3D-Modell werden nicht unterschieden - diese Information kann aus dem Scripttyp heraus. Das Objekt sollte eine schnelle, grobe Vorschau des Modells liefern, unter Berücksichtigung der begrenzten Größe der Vorschau.

GLOB_SEO_TOOL_MODE = 1 erzeugt als ein Operator für Solid-Element-Operationen

Das erzeugte 3D-Modell fungiert als ein Operator für Solid-Element-Operationen (SEO). Dies kann sehr nützlich sein, wenn der vom Objekt eingenommene Raum größer sein muss als das Objekt selber. Wenn Sie z.B. eine Treppe von einer Decke abziehen, erwarten Sie, dass die Treppe auch eine Öffnung für die sie benutzenden Menschen herausschneidet.. Um dies zu erreichen, sollte die Treppe in diesem Kontext ein Modell erzeugen, welches den "Begehungsfreiraum" enthält.

Werteaustausch zwischen Objekt und ARCHICAD

Es gibt zwei Richtungen, in welche ein Werteaustausch zwischen Objekt und ARCHICAD stattfinden kann. Die erste Richtung bedeutet, dass ARCHICAD das Objekt über die Attribute seine aktuellen Kontexts informiert (z.B. der Zeichnungsmaßstab des Projektes oder die Stärke einer Wand, in welche ein Fenster eingesetzt wird). Die zweite Richtung ist, wenn das Bibliothekselement etwas von sich selbst behauptet was ARCHICAD anweist, etwas im direkten Kontext des Objektes zu ändern (z.B. die Tiefe, die ein Wandende in eine Wand schneidet).

Informationsfluss von ARCHICAD zum Objekt

Es gibt 3 Kanäle, auf welchen Informationen von ARCHICAD geliefert werden: Globale Variablen, Parameter mit vordefinierten Namen und direkt aufgerufene Werte.

Globale Variablen

Globale Variablen werden von ARCHICAD entsprechend der aktuellen Projekteinstellungen und dem Kontext der Platzierung des Objektes ausgefüllt. Bite beachten: nicht alle Globalen Variablen funktionieren in jedem Kontext oder jeder Sicht.

Zur kompletten Liste aller Globalen Variablen und ihrer relevanten Einschränkungen in bestimmten Scripten, konsultieren Sie bitte „Globale Variablen“.

Festbenannte optionale Parameter

Die neuere Methode, mit welcher ARCHICAD Informationen liefert, ist die Methode der optionalen Fixnamen-Parameter (fixed named optional parameters). Falls ein bestimmtes Bibliothekselement einen Parameter mit Namen und Typ besitzt, die mit einem optionalen Parameter übereinstimmen, setzt ARCHICAD seinen Wert entsprechend seiner Funktion.

Siehe „Parameter festgelegt von ARCHICAD“ in „Festbenannte optionale Parameter“ um mehr über die von ARCHICAD definierten Bibliothekselement-Parameter zu lernen.

REQUESTs und APPLICATION QUERYs

Für selten verwendete, spezielle Informationen, verwenden GDL-Objekte REQUEST-Aufrufe oder APPLICATION QUERYs in ihren Scripten. Anders als Globale Variablen, liefern diese nur dann einen Rückgabewert, wenn die sie enthaltenden aktuellen Scripte laufen. Beachten Sie, dass die meisten REQUESTS und QUERYs in Parameter-Scripten vermieden werden sollten (oder als Parameter-Script ausgeführten Master-Scripten). Falls Sie diese trotzdem verwenden, kann die Gültigkeit der Rückgabewerte bzw. der Funktion nicht gewährleistet werden.

Schauen Se unter „REQUEST Optionen“ und„Application Query Optionen“ um mehr über die Optionen, die Parameter-Script-Kompatibilität und Syntax zu lernen.

Informationen, die aus einem Bibliothekselement kommen

ARCHICAD benötigt bestimmte Informationen, um die Bibliothekselemente korrekt zu verwenden. Diese Informationen hängen ab von der Funktion und dem Kontext, und sie werden in den integrierten ARCHICAD-Subtypen als Parameter mit vordefiniertem Namen und Funktion gespeichert. Als Ergänzung zu den eingebauten ARCHICAD-Subtypen benötigen einige Funktionen eventuell optionale Fixnamen-Parameter.

Studieren Sie die Fixen Parameter der eingebauten Subtypen und „Parameter von ARCHICAD ausgelesen“ in „Festbenannte optionale Parameter“ um einen Überblick über die Möglichkeiten zu bekommen.

Modelldarstellung (MVO), "Library Global"

Die Darstellung von Bibliothekselementen im Plan kann von der aktuellen Sicht abhängen.

Interne Modelldarstellungs-Optionen

Die internen Einstellungen sind mittels der GDL-Globalen Variablen (z.B. GLOB_SCALE, GLOB_STRUCTURE_DISPLAY) und REQUEST-Optionen (z.B. "window_show_dim", "door_show_dim", "floor_plan_option", "view_rotangle") verfügbar.

Ansichtsoptionen mittels "Library Global"

Von ARCHICAD 13 aufwärts kann man Ansichtsoptionen für seine Bibliothek definieren. Diese Optionen werden in jedem ARCHICAD-Ausschnitt und sie werden entsprechend zurückgegeben.

Folgende Eigenschaften/Parameter/Optionen sollten in sichtabhängigen Library-Global-Objekten abgelegt werden:

- anzeigen/verstecken von Öffnungslinien
- anzeigen/verstecken von minimalem Platzbedarf
- Stiftfarben und andere Attributen, welche aus Gründen der Einheitlichkeit nicht individuell geändert werden sollten (z.B. minimaler Platzbedarf)
- anzeigen/verstecken von spezifischen Elementen (z.B. Griffe und Drücker)
- einstellen von 2D-Symboltypen bei Objektgruppen

Dinge, welche NICHT in sichtabhängigen "Library Globals" gespeichert werden sollten: allgemeine Werte für das gesamte Projekt, allgemeine Werte für das gesamte Land, Werte, welche erforderlich sein könnten, um Objekte individuell einzustellen.

Um eine Tab-Seite in den Modelldarstellungs-Dialog einzubauen, müssen Sie ein Bibliothekselement erzeugen, welches aus dem Subtyp der *Library Global Settings* (GUID: {709CC5CC-6817-4C56-A74B-BED99DDB5FFA}) abgeleitet ist. Dieses Objekt muss die gewünschten globalen Optionen enthalten und es muss ein definiertes User Interface für die neue Tab-Seite der MVO besitzen. Die Breite des UI sollte auf 600 Pixel eingestellt sein, um mit den vorhandenen Tabs übereinzustimmen. Die Höhe dieses UI ist frei definierbar. Dieses Objekt kann ein Parameter-Script besitzen, um Parameter mit dem User Interface zu verbinden.

Der Befehl `LIBRARYGLOBAL` kann in Ihren platzierbaren Objekten verwendet werden um Werte Ihres eigenen "Library Globals"-Objektes abhängig von den aktuellen Ausschnitt-Einstellungen abzufragen.

Scripttyp spezifische Fragen

Master-Script

Wenn Sie ein Master-Script schreiben, müssen Sie darauf achten, dass es vor dem Durchlauf jedes anderen Scriptes von ARCHICAD ausgewertet wird. Dies bedeutet, die folgenden Dinge:

- Das Platzieren von Parameter- und Variablendefinitionen und Berechnungen, die von mehreren Scripten verwendet werden, im Master-Script ist eine gute Idee: es reduziert die Dateigröße und mach Elemente leichter veränderbar.
- Achten Sie darauf, hier nur gemeinsame Berechnungen unterzubringen, um unnötige Erhöhung der Auswertungszeit des Bibliothekselementes zu vermeiden (nicht vergessen: das Master-Script wird jedesmal vor jedem anderen Script durchlaufen).
- Vermeiden Sie aus Effektivitätsgründen die Verwendung des Parameter-Buffers im Master-Script.
- Platzieren Sie keine `end` -Befehle im Master-Script; andernfalls wird ARCHICAD den Rest der Scripte nicht ausführen.

2D-Script

Ausführungs-Kontext

Das 2D-Script wird ausgeführt, wenn ein 2D-Modell erzeugt wird:

- 2D Grundriss
- 2D Hotspot-Editierungs-Feedback
- 2D Vorschau des Objekteinstellungs-Fensters
- Layout-Zeichnung
- Layout-Zeichnungs-Feedback

Bedenken Sie, dass der größte Teil der Entwurfsarbeit in 2D durchgeführt wird, so dass dieses Modell normalerweise das wichtigste ist. Dies impliziert Anforderungen an exaktes Aussehen, schnelle Generationszeit und die richtige Funktion bei der Bearbeitung über Hotspots.

Allgemeine Empfehlung

Versuchen Sie die Verwendung von FRAGMENT-Befehlen und dem 2D-Binärformat zu vermeiden, um Objekte veränderbar zu machen.

2D-Scripte sind wesentlich besser anpassbar als 2D-Symbole, geben Sie diesen den Vorzug. In einem binären 2D-Symbol werden gebogene Schraffuren nicht korrekt gestreckt, beim 2D-Scripten werden Sie mit diesem Problem nicht konfrontiert.

Definieren von Eigenschaften von Linien und Schraffuren

Von ARCHICAD 9 an aufwärts haben Sie die Möglichkeit, aus mehreren Hauptkategorien unter den Linien und Schraffuren in GDL zu wählen. Linien- und Polygonsegmente können als Konturlinie, Schicht-Trennlinie oder Normal-Linie definiert werden; Schraffuren kann man definieren als Bauteilschraffur, Deckschraffur, Zeichnungsschraffur. Diese Kategorien werden im ARCHICAD-Benutzerhandbuch beschrieben, werfen wir eine Blick auf ihre Verwendung in GDL-Objekten.

Einstellen der richtigen Eigenschaften für Linien und Schraffuren führt dazu, dass Sie die Abhängigkeit von der Bildschirmdarstellungs-Option aus Ihren Scripten entfernen können. Früher mussten Sie eine Bedingung für die Erstellung von ein paar Schichttrennlinien entsprechend der eingestellten Bildschirmdarstellungs-Optionen einfügen. Jetzt sollten Sie eine Schichttrennlinie für diesen Zweck definieren und ARCHICAD wird diese anzeigen oder nicht, wie von den Bildschirmdarstellungs-Optionen vorgegen.

Schauen wir uns einen Ausschnitt eines 2D-Scriptes eines Fensters an, um die Definitionsarten zusammenzufassen:

```
! ===== Schwelle =====

line_property 0    ! Normal-Linie

! die Schwelle ist von oben sichtbar -> Deckschraffur
poly2_b{2} 4, 1 + 2 * (gs_fillSillCover > 0) + 4 + 64, ...
...

! ===== Wandsegment / Hohlraumverschluss =====

line_property 1    ! Schicht-Trennlinie
line2 ...
...

line_property 2    ! Bauteilkonturlinie
line2 ...
...

! Wandsegment wird geschnitten dargestellt -> Bauteilschraffur
poly2_b{2} 4, 2 + 4 + 8 + 16 + 32, ...

! ===== Fensterrahmen=====

line_property 0    ! Normal-Linie

! seitlicher Fensterrahmen ist geschnitten -> Bauteilschraffur
poly2_b{2} 4, 1 + 2 * (gs_fillFrames > 0) + 4 + 32, ...
...
```

3D-Script

Ausführungs-Kontext

Das 3D-Script wird ausgeführt, wenn ein 3D-Modell erzeugt wird:

- 3D-Fenster (Drahtgitter, versteckte Kanten, Schattierung)
- 2D-Plan, wenn`project2` verwendet wird um eine Projektion des 3D-Modells in 2D darzustellen

- 2D-Schnitt - beachten Sie die Details
- 3D Hotspot-Editierungs-Feedback- Geschwindigkeitsoptimierung
- Operator für Solid Element Operationen in 3D - fragen Sie den Designer nach der gewünschten Funktionalität
- Oberflächen- und Volumenberechnungen für Listen
- 3D- Vorschau im Objekteinstellungsdialog
- Layout-Zeichnung wenn `project2` wird für die Projektion des 3D-Modells in 2D verwendet
- Layout-Zeichnungs-Feedback

Allgemeine Empfehlung

Versuchen Sie das Binärformat zu vermeiden, um Objekte veränderbar zu machen.

Benutzen Sie Status-Werte um die Sichtbarkeit von Objekten im Verdeckte-Kanten-Modus zu kontrollieren. Machen Sie die Umrißlinien auf gebogenen Oberflächen sichtbar. Unnötige Linien sollte man, wenn möglich, verstecken.

Definieren Sie editierbare Hotspots anstelle von unveränderlichen, wann immer möglich.

Benutzen Sie nicht den Befehl `del top` , damit spätere Veränderungen einfacher sind.

Resetten Sie die globalen Koordinaten immer am Ende des 3D-Scriptes und schließen es mit einem `end` -Befehl ab damit spätere Veränderungen einfacher sind.

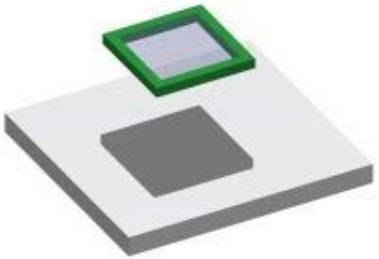
Modellieren transparenter Körper

Verwenden Sie die `body -1` -Befehle zwischen opaken und transparenten Teilen eines Objektes um korrekten Schattenwurf mit der Internen Rendering Engine zu ermöglichen (z.B. Fenster mit Sprossenraster).

Tabelle 7. Beispiele von transparenten Körpern

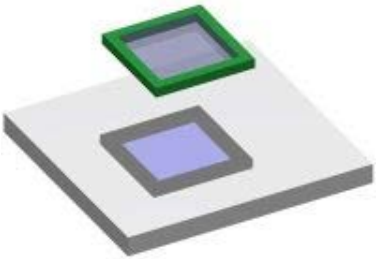
falsch

```
prism_ 10, 0.1,
    0,   0,   15,
    1,   0,   15,
    1,   1,   15,
    0,   1,   15,
    0,   0,   -1,
    0.1, 0.1, 15,
    0.9, 0.1, 15,
    0.9, 0.9, 15,
    0.1, 0.9, 15,
    0.1, 0.1, -1


material "blueglass"

prism_ 5, 0.1,
    0.1, 0.1, 15,
    0.9, 0.1, 15,
    0.9, 0.9, 15,
    0.1, 0.9, 15,
    0.1, 0.1, -1
```



korrekt

```
prism_ 10, 0.1,
    0,   0,   15,
    1,   0,   15,
    1,   1,   15,
    0,   1,   15,
    0,   0,   -1,
    0.1, 0.1, 15,
    0.9, 0.1, 15,
    0.9, 0.9, 15,
    0.1, 0.9, 15,
    0.1, 0.1, -1

body -1

material "blueglass"

prism_ 5, 0.1,
    0.1, 0.1, 15,
    0.9, 0.1, 15,
    0.9, 0.9, 15,
    0.1, 0.9, 15,
    0.1, 0.1, -1
```



Textur-Mapping

Überprüfen Sie immer, ob die Texturbelegung korrekt auf den Objekten dargestellt wird. Wenn der standardmäßige Texturbelegungsprozess von ARCHICAD kein gutes Ergebnis produziert, kann man den `coor` -Befehl benutzen um die richtige Methode einzusetzen. Unten finden Sie einen Beispielfall:

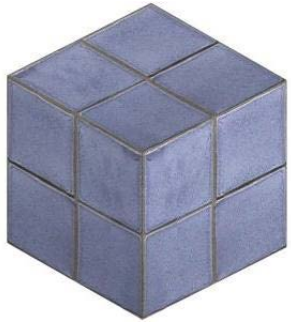
Tabelle 8. Beispielcode für zufällig und korrekt angeordnete Kachelung.

zufällige Kachelung

```
define texture "owntile" "T.jpg",
        1, 1, 128+256, 0

define material "tilemat" 21,
        0.7, 0.7, 1,
        0.15, 0.95, 0, 0.0,
        0, 0,
        ind (fill, ""), 1,
        ind (texture, "owntile")

material tilemat

block 1, 1, 1
```

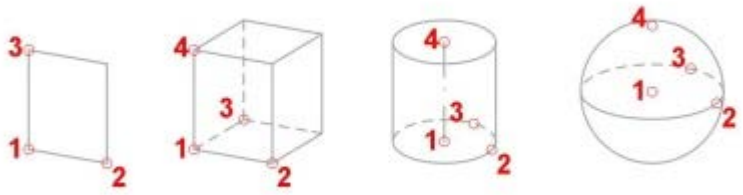
Ausgerichtete Kachelung

```
define texture "owntile" "T.jpg",
        1, 1, 128+256, 0

define material "tilemat" 21,
        0.7, 0.7, 1,
        0.15, 0.95, 0, 0.0,
        0, 0,
        ind (fill, ""), 1,
        ind (texture, "owntile")

material tilemat

block 1, 1, 1

base
vert 0, 0, 0
vert 1, 0, 0
vert 0, 1, 0
vert 0, 0, 1

coor 2 + 256, -1, -2, -3, -4
```



In der Regel sollten Sie Körper, die unterschiedliche Texturkoordinatensysteme benötigen mit einem `body -1` -Befehl voneinander trennen.

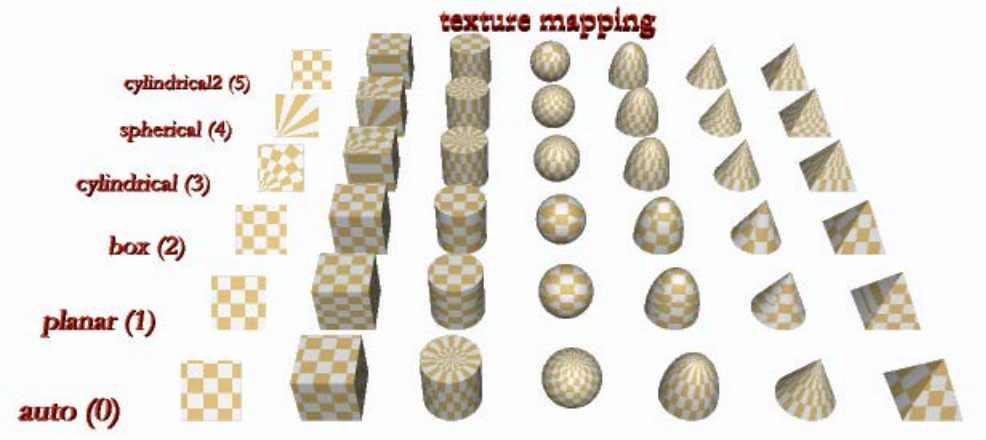
Wenn man unterschiedliche Texturbelegungs-Modi verwendet, sollten Sie auf korrekte Achsendefinitionen achten unter Verwendung der Befehle `vert` oder `teve` . Die Reihenfolge der Knotenpunkte ist unten dargestellt.



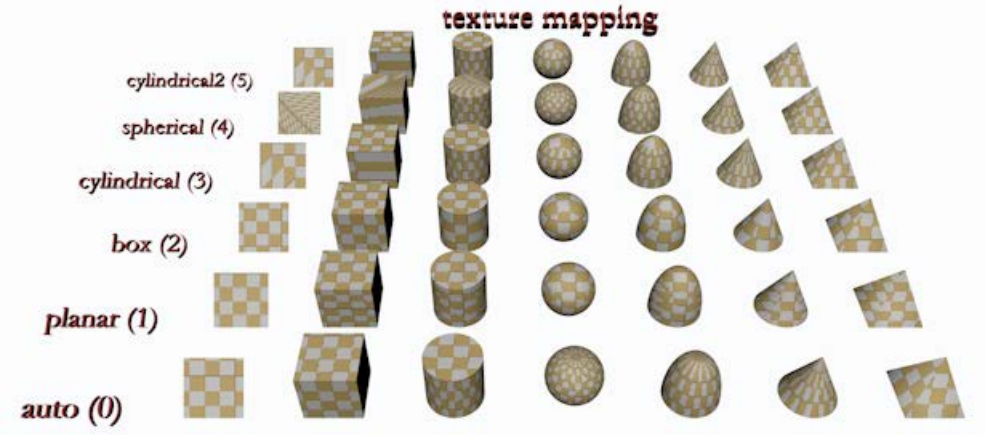
Sie können die Texturen verzerren, indem Sie unterschiedliche Abstände zwischen den Knotenpunkten setzen, indem Sie die Befehle `vert` oder `teve` verwenden.

Beachten Sie, dass das Arbeiten mit den unterschiedlichen Render-Engines zu leicht unterschiedlichen Ergebnissen führen kann, siehe Beispiele.

Interne Engine:



C4D Engine:



Korrektes Texturmapping auf komplizierten Oberflächen bzw. verzerrte Texturen kann man mit den Befehlen `coor` und `teve` modellieren. Auf diese Weise können Sie nur Oberflächen modellieren. In ARCHICAD gibt es keine direkte Texturspezifikation. Sie können eine Textur als Teil einer Materialdefinition definieren. Dies Textur wird in Rendering Engines und in OpenGL verwendet – aber in OpenGL haben wir nur eine begrenzte Einbindung unseres vollen Texturmappings zur Verfügung, und überhaupt kein Textur (Schraffur) Mapping in unserer internen 3D Engine.

Also können Sie mit dem TEVE Befehl einen planaren Texturpunkt (u,v) auf einen räumlich geometrischen Punkt (x, y, z) mappen:

- **(x, y, z)** wird im lokalen Koordinatensystem wie üblich, in Metern gemessen
- **(u, v)** wird im unendlichen Texturraum in Einheiten gemessen. Eine Einheit ist so breit wie die Textur in die entsprechende Richtung reicht.

Sie können einen negativen und auch mehr als einen Wert für u und v vergeben.

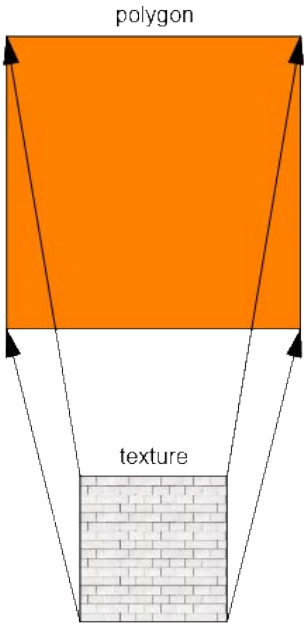
Siehe Beispiel :

Tabelle 9. Teve Beispiel 1: Mapping ohne Verzerrung

Programm

```
base

teve  0, 0, 1,     0, 0
teve  2, 0, 1,     1, 0
teve  0, 2, 1,     0, 1
teve  2, 2, 1,     1, 1
teve  0, 0, 1,     1, 1

edge  1, 2, -1, -1, 0
edge  2, 4, -1, -1, 0
edge  4, 3, -1, -1, 0
edge  3, 1, -1, -1, 0

set material 92

pgon 4, 0, 0, 1, 2, 3, 4
coor 1024, 1, 2, 3, -5

body -1
```

Logik



Ergebnis

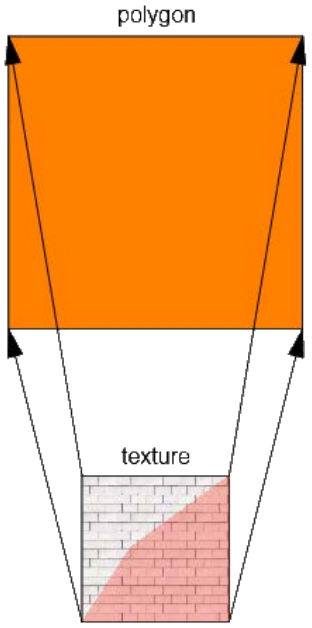
Wenn Sie ein nicht-rechteckiges Mapping machen wollen, wird die Rendering Engine die Form des Texturraumes in die Form des Modellraumes einpassen:

Tabelle 10. Teve Beispiel 1: Mapping mit Verzerrung

Programm

```
base

teve  0, 0, 1,     0,     0
teve  2, 0, 1,     1,     0
teve  0, 2, 1,     0.3,   0.5
teve  2, 2, 1,     1,     1
teve  0, 0, 1,     1,     1

edge  1, 2, -1, -1, 0
edge  2, 4, -1, -1, 0
edge  4, 3, -1, -1, 0
edge  3, 1, -1, -1, 0

set material 92

pgon 4, 0, 0, 1, 2, 3, 4
coor 1024, 1, 2, 3, -5

body -1
```

Logik



Ergebnis

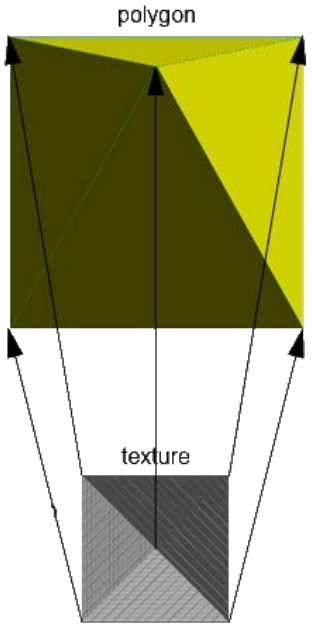
Das gleiche gilt für reale 3D-Körper, wie Sie in folgendem Beispiel sehen können:

Tabelle 11. Teve Beispiel 1: Mapping mit Verzerrung auf einer Pyramide

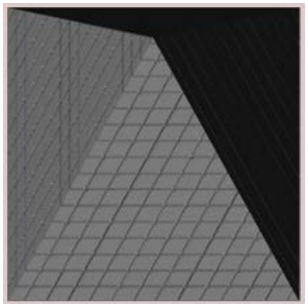
Programm

```
base

teve  0, 0, 1,     0,     0 ! 1
teve  2, 0, 1,     2,     0 ! 2
teve  2, 2, 1,     2,     2 ! 3
teve  2, 2, 1,     0,     2 ! 4
teve  1, 1, 3,     1,     1 ! 5

edge  1, 2, -1, -1, 0
edge  2, 4, -1, -1, 0
edge  4, 3, -1, -1, 0
edge  3, 1, -1, -1, 0

set material 92

pgon 3, 0, 0, 1, 6, -5
coor 1024, -6, -7, -8, -9

body -1
```

Logik



Ergebnis



Bitte beachten Sie, dass Sie nur einen Texturscheitel zu einem Modellscheitel zuweisen können. Es ist nicht möglich, die Texturscheitel pro Polygonbasis zuzuweisen. Das ist manchmal ein Vorteil, manchmal ein Nachteil.

Bildelemente

Es kann eine gute Idee sein, komplizierte Teile eines Modells durch ein einzelnes Bild zu ersetzen. Diese Methode kann man auch gut für Bäume und Büsche verwenden.

Vergessen Sie nicht die Dateiendung, wenn Sie auf externe Bilddateien referenzieren.

Wenn Sie ein Bild in einem 3D-Modell mit dem Befehl `picture` platzieren, wird ein Polygon mit dem Bild als Gesicht erzeugt. Das Material des Polygons beeinflusst das Ergebnis des Renderings. Mit diesem Wissen sollten Sie eine matte Oberfläche verwenden - die Farbe kann man in Abhängigkeit vom Bild wählen.

```
define material "pictmat" 2,
        1, 1, 1             ! RGB

material "pictmat"

picture "filename.extension", a, b, mask
```

Das erste Bild zeigt eine Grafik auf einer glänzenden Oberfläche - der unerwünschte Nebeneffekt ist nicht zu übersehen. Auf dem zweiten Bild sehen Sie eine Textur auf einem präzise eingestellten Material - das gewünschte Ergebnis.

Tabelle 12. Transparente Grafiken

Glänzende Oberfläche	**Matte Oberfläche**

Bei transparenten Grafiken - wie der Baum im Bild oben - sollten Sie eine genauere Definition des Grundmaterials berücksichtigen. Siehe folgendes Beispiel.

```
define material "pictmat" 0,
        1, 1, 1,              ! RGB
        0.5, 0.8, 0, 0,
        0, 0,
        0, 0, 0,
        0, 0, 0,
        0

material "pictmat"

picture "filename", a, b, mask
```

Gruppenoperationen

Gruppenoperationen bringen die Leistungsfähigkeit von Solid Element Operationen zu GDL. Auf der anderen Seite stellen sie einen Risikofaktor dar, wenn falsch angewendet.

Ein wichtiger Punkt ist, dass Sie keine Gruppe innerhalb einer anderen platzieren dürfen. In solchen Situationen sollten Sie eine neue Gruppe wie in folgendem Codeschnipsel definieren:

```
subtractionResult = subgroup ("sub_operand_1", "sub_operand_2")
```

Parameter-Script

Ausführungs-Kontext

Das Parameter-Script wird in folgenden Fällen durchlaufen:

- Öffnen des Objekteinstellungsdialoges
- Ändern eines Parameterwertes im Objekteinstellungsdialog
- Ändern eines Parameterwertes mit Hilfe von editierbaren Hotspots (auch bei Erzeugung des Feedbacks)
- Strecken des Objektes mit Hilfe konventioneller Hotspots
- Schritt für Schritt Laden der Migrations-Bibliotheken (beginnend mit AC18)

Das Parameter-Script kann be in folgenden Fällen durchlaufen:

- Ziehen des Objektes, für den Fall, dass es sich auf `SYMB_POS_X/SYMB_POS_Y` bezieht
- Beim *Raumstempel aktualisieren* läuft das Parameter-Script in den beteiligten Raumstempeln, falls nötig

Das Parameter-Script läuft NICHT in folgenden Fällen:

- Neuaufbau
- Maßstabsänderung
- Geschossänderung

Editieren einer Mehrfachauswahl kann zu unbeabsichtigten Parameterwerten führen.

Beachten Sie, dass das Parameter-Script bei einer einzelnen Benutzeraktion mehrmals durchlaufen werden kann. Der Grund dafür ist, dass das Parameter-Script den Wert von Parametern ändern kann und dies erfordert, dass das Parameter-Script erneut ausgeführt werden muss, und so weiter. Daher macht es keinen Sinn, einen Parameterwert um Eins im Parameter-Script zu erhöhen das Sie nicht in der Lage sind, den Vorrangigkeit der Scriptausführung zu beurteilen.

Der Durchlauf des Parameter-Scriptes ist linear und nicht notwendigerweise mehrfach. Sie können das Parameter-Script zwingen, dass es nur einmal durchlaufen wird, indem Sie die Option **"Das Parameter-Script nur einmal durchlaufen lassen"** in den Kompatibilitätsoptionen des Objektes abhaken, wenn Sie sich sicher sind, dass es nicht mehrfach durchlaufen werden muss. Dies kann das Objekt schneller reagieren lassen, was Zeit und Rechnerressourcen spart.

Allgemeine Empfehlung

Wenn Sie Parameter im Parameter-Script beeinflussen, versuchen Sie der Reihenfolge der zusätzlichen Parameter zu folgen.

Sie können Beziehungen zwischen Parametern mit Hilfe des Wertes des `GLOB_MODPAR_NAME` definieren (welcher den Namen des zuletzt geänderten Parameters enthält). Zum Beispiel können Sie einen Kreis-Objekt erstellen, für welches sowohl der Radius als auch der Durchmesser eingestellt werden kann (vielleicht einer von beiden über die Parameterliste und den anderen über editierbare Hotspots). Verwenden Sie diese Möglichkeit nicht, um den gültigen Parameterbereich festzulegen - verwenden Sie stattdessen den `values` -Befehl.

Definieren sie den gültigen Bereich aller Parameters mit Hilfe des `values` -Befehls.

Wenn Sie unter einer bestimmten Bedingung den Wert eines Parameters im Parameter-Script zurücksetzen, indem Sie den Befehl `parameters` verwenden, muss eine ähnliche Festsetzung im Master-Script erfolgen. Dies sorgt für eine korrekte Anzeige des Objektes für Fälle, wenn das Parameter-Script nicht vom System durchlaufen wird. z.B.:

```
! Parameter-Script
if bCondition then
    yy = 1
    parameters yy = yy
endif

! Master-Script
if bCondition then yy = 1
```

Schriftart Namen

Falls Sie einen Stringparameter haben möchten - im Beispiel benannt mit `stFont` - um die Schriftart für ein Textfeld festzulegen, verwenden Sie die folgende "value" - Listendefinition, damit Sie eine plattformunabhängige Lösung erhalten.

```
DIM fontNames[]
request ("FONTNAMES_LIST", "", fontNames)
values "stFont" fontNames, CUSTOM
```

Wenn Sie dies in dem Objekt ARCHICAD_Library_Master.gsm Objekt machen, wird jedes geladene Bibliothekselement mit dem gleichen Parameter "stFont" automatisch die gleiche Werteliste empfangen.

CUSTOM : dieser Wert wird benötigt, um mit fehlenden oder unerwarteten Fonttypen klar zu kommen.

Einstellungsbeschränkungen von Array-Parametern

Array-Parameter sollten für homogene Daten verwendet werden; d.h. alle Array-Elemente sollten eine ähnliche Bedeutung haben.

Beispielcode-Snippet zur Begrenzung aller Komponenten des Array-Parameters `gridXPosition` auf den Wertebereich range [1, 5] und wie man es im User Interface verwendet:

```
! parameter script
values "gridXPosition" range [1, 5]

! UI script
for i = 1 to nGridLines     ! nGridLines: Anzahl der Zeilen im Array-Parameter
    ui_infield{3}   gridXPosition[i], xPos, yPos, infieldWidth, infieldHeight

    yPos = yPos + diffY
next i
```

User Interface Script

Ausführungs-Kontext

Das Ergebnis des User Interface Scripts wird nur in einem Kontext angezeigt: nämlich der User Interface Tabseite im Objekteinstellungsdialog.

Das Script wird durch die Initialisierung des Dialogfensters ausgeführt und nach jeder Benutzerinteraktion und Parameteränderung.

Allgemeine Empfehlung

Wenn Sie möchten, dass die Seite "Individuelle Einstellungen" als Standard anstelle der Parameterliste angezeigt werden soll, drücken Sie den Button **als Grundeinst. festlegen** (oder fügen Sie den Bit "STBit_UIDefault" in die Sektion "StatBits" der XML-Datei ein. Andernfalls wird die Parameterliste als Start-Tab angezeigt. Bei Hierarchischen Seiten drücken Sie auf den Button **Hierarchische Seiten** im GDL Editor/UI Fenster (oder fügen den Bit "STBit_UIUseHierarchicalPages" in der Sektion "StatBits" der XML-Datei ein).

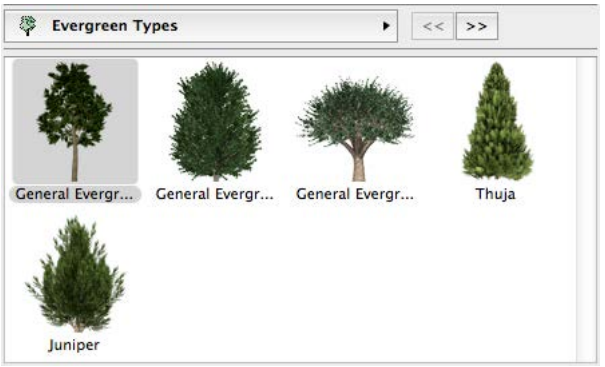
Beim Textstyling ist zu beachten, dass extra kleine Buchstaben nicht jeden Stil besitzen können, sondern nur den einfachen. Außerdem haben *Outline* und *Shadow* Stile keinen Effekt auf Windows-Systemen.

Beachten Sie, dass ARCHICAD-Nutzer versuchen, die Schriften im UI mit den Systemschriften in Übereinstimmung zu bringen. Wenn Sie grafische Benutzeroberflächen auf Windows programmieren, lassen Sie mehr Platz um die Texte herum, andernfalls sehen Mac-User abgeschnittene Texte.

Vorschau-Steuerbilder

Falls Sie den `ui_infield` -Befehl verwenden, um eine bebilderte Auswahlliste zu definieren, beachten Sie folgendes. Es sollten gleich große Vorschaubilder für alle Parameterwerte werden (einschließlich Leerwert). Vorschaubilder müssen dieselbe Größe haben, in welcher sie angezeigt werden sollen, andernfalls wird ARCHICAD sie verfälschen. Wir empfehlen Ihnen, ARCHICADs Bild-Werkzeug für die Montage der Miniaturansichten in eine Bilddatei zu verwenden.

Tabelle 13. Infield mit Bild

Input Bild	Output Bild
	

Ein User Interface-Bild, welches nur von einem einzigen Objekt verwendet wird, sollte in das Bibliothekselement selber integriert werden. Dies kann man mit Hilfe des LP_XMLConverter Tools erreichen.

Vergessen Sie nicht die Dateiendung, wenn Sie auf externe Bilddateien referenzieren. Auf diese Weise vermeiden Sie Fehler aufgrund von Bildern und Objekten mit identischen Namen.

Legen Sie alle von User Interface Scripten genutzten Bilddateien im Makros-Ordner ab, oder binden Sie sie in das Objekt selber ein. Verwendung externen Bilddateien: bauen Sie den Befehl `file_dependence` in Ihre Scripte ein, um sicherzustellen, dass die externen Bilder beim Speichern eines Projektes im Archiv-Format mitgesichert werden.

Handhabung der Tab-Seiten

Mit Beginn von ARCHICAD 18 ist eine neue hierarchische Paging-Option für die Tabseitenauswahl verfügbar. Hierauf greift man zu via UI_PAGE, indem man einige Extraparameter zufügt, und die **Hierarchischen Seiten** -Parameter im Objekt selber einstellt. Dadurch wird eine gesonderte Popup Tabpage-Kontrolle über dem benutzerdefinierte UI-Feld angezeigt. Die Reihenfolge und die Hierarchie der verfügbaren Seiten kann durch die ID der Seiten definiert werden. Die Root-ID ist immer`-1`. Die Möglichkeit, einen Tabseitenwähler innerhalb der UI-Seite nach "alter Schule" zu bauen, ist weiterhin verfügbar.

Ein Beispiel-Script:

```
! Master-Script

! TabIDs
TABID_ROOT      = -1
TABID_PAGE_1    = 50
TABID_PAGE_2    = 60

dim uiUsedPageIDs[][2]
dim uiUsedPageNames[][2]
```

```
idxPage = 1

uiUsedPageNames[idxPage][1] = "PageName_1"
uiUsedPageNames[idxPage][2] = "pageIconName_1.png"

uiUsedPageIDs[idxPage][1]   = TABID_PAGE_1
uiUsedPageIDs[idxPage][2]   = TABID_ROOT    ! Parent Page ID

idxPage = idxPage + 1

uiUsedPageNames[idxPage][1] = "PageName_2"
uiUsedPageNames[idxPage][2] = "pageIconName_2.png"

uiUsedPageIDs[idxPage][1]   = TABID_PAGE_2
uiUsedPageIDs[idxPage][2]   = TABID_PAGE_1  ! Parent Page ID



file_dependence "pageIconName_1.png"
file_dependence "pageIconName_2.png"
file_dependence "pageIconName_3.png"


! Parameter-Script

dim pageValues[]
for i = 1 to vardim1(uiUsedPageIDs)
    pageValues[i]= uiUsedPageIDs[i][1]
next i

values "gs_ui_current_page" pageValues
```

```
! UI-Script

ui_dialog "Custom Settings Title"
ui_current_page gs_ui_current_page

for i = 1 to vardim1(uiUsedPageIDs)
    if uiUsedPageIDs[i][1] = TABID_PAGE_1 then
        ui_page uiUsedPageIDs[i][1],    uiUsedPageIDs[i][2],
                uiUsedPageNames[i][1],  uiUsedPageNames[i][2]
        if gs_ui_current_page = TABID_PAGE_1 then
            gosub "pageSubroutinTitle_1"
        endif
    endif

    if uiUsedPageIDs[i][1] = TABID_PAGE_2 then
        ui_page uiUsedPageIDs[i][1],    uiUsedPageIDs[i][2],
                uiUsedPageNames[i][1],  uiUsedPageNames[i][2]
        if gs_ui_current_page = TABID_PAGE_2 then
            gosub "pageSubroutinTitle_2"
        endif
    endif
next i



! ============================================================================
! Call User Interface Macro's TabPages
! ============================================================================

call "ui_customMacro" parameters all    uiUsedPageIDs   = uiUsedPageIDs,
                                        uiUsedPageNames = uiUsedPageNames

! ============================================================================
end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! en
! ============================================================================
```

```
! ============================================================================
! UI Page Subroutines
! ============================================================================

"pageSubroutinTitle_1":
    ! UI Page 1 description
return


"pageSubroutinTitle_2":
    ! UI Page 2 description
return
```

Vorschaubild-Auswahl mit dynamischen Elementen

Seit ARCHICAD 10 aufwärts ist eine neue Methode der Verlinkung von Kontrollelementen und VALUE-Listen verfügbar. Mit dieser Methode können Sie die Logik der Verfügbarkeit von Parameterwerten für das Parameter-Script lokalisieren - die Steuerung wird die Menge der verfügbaren Werte annehmen. Diese dynamische Verlinkung ist verfügbar `ui_infield{3}` und `ui_infield{4}`. Die statische Verlinkung im alten Stil funktioniert weiterhin für statische Funktionen (unter Verwendung von `ui_infield` und `ui_infield{2}`).

Die beiden Komponenten der dynamischen Methode sind:

1. Definieren Sie ein User Interface Eingabefeld mit einer Option für jeden möglichen Wert.

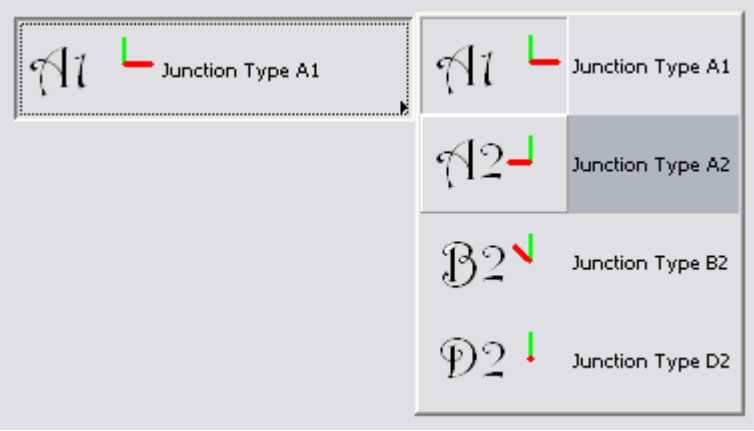
Das Beispiel zeigt eine Popup-Menüsteuerung (Methode = 2), welche ein indiziertes Bild verwendet, welches 2 Zeilen und 4 Spalten besitzt. Die Beispielsteuerung unterstützt 8 mögliche Werte.

```
ui_infield{3} iJunctionType, xColumn1-10, 44, 200, 50,
    2, 3, 8, 2,
    70, 45, 70, 45,
    1, `Junction Type A1`, 2,
    2, `Junction Type B1`, 4,
    3, `Junction Type C1`, 1,
    4, `Junction Type D1`, 3,
    5, `Junction Type A2`, 5,
    6, `Junction Type B2`, 7,
    7, `Junction Type C2`, 6,
    8, `Junction Type D2`, 8
```

2. Geben Sie die Liste der verfügbaren Parameterwerte untern den jeweils gegebenen Bedingungen vor.

```
if iLeftNeighbour = 1 then
    values "iJunctionType" 1, 3, 4, 6
else
    if iRightNeighbour = 1 then
        values "iJunctionType" 2, 5, 7, 8
    else
        values "iJunctionType" 1, 5, 7
    endif
endif
```

Das Ergebnis dieser Steuerung sehen Sie auf dem folgenden Bild. (iLeftNeighbour = 0, iRightNeighbour = 1)



Transparente UI-Grafiken

In ARCHICAD 10 wurde eine neue Methode eingeführt, welche transparente Bilder mit Alphakanälen handhaben kann. Die folgende Liste zeigt die korrekte Verwendung von Bildern mit Alphakanal:

- ui_pict
- ui_infield{3}, method = 1 (Bildauswahlliste)
- ui_infield{3}, method = 2 (Popup mit Icons und Texten)

- ui_infield{3}, method = 3 (Popup nur mit Icons)
- ui_infield{3}, method = 4 (Icon Radio Druckknopf)
- ui_infield{4}, method = 1 (Bildauswahlliste)
- ui_infield{4}, method = 2 (Popup mit Icons und Texten)
- ui_infield{4}, method = 3 (Popup nur mit Icons)
- ui_infield{4}, method = 4 (Icon Radio Druckknopf)

Fontgrößen im UI

Wenn Sie statische Texte verwenden (möglich in Kombination mit dem Befehl `ui_style`), achten Sie bitte auf folgendes.

Wegen der Unterschiede in den Betriebssystemen der Anwender, sind Schriftgrößen auf Windows und auf MAC nicht identisch. Als Nebeneffekt ist die Schriftgröße *extra klein* auf Windows ein wenig größer als die Schriftgröße *klein*. Als allgemeine Regel sollten Sie User Interfaces immer auf beiden Plattformen testen, um Überlappungen und Abschneidungen zu vermeiden.

Außerdem sind spezielle Stile wie fett, kursiv und unterstrichen nicht in Kombination mit der Schriftgröße *extra klein* erlaubt. Outline und Schatten sind alte Macintosh Stile, welche nicht länger verwendet werden

Die folgenden beiden Bilder zeigen das Aussehen von statischen Texten in unterschiedlichen Größen und Stilen.

Auf Windows:

	small	xsmall	large
normal	ui_style 0,0	**ui_style 1,0**	ui_style 2,0
bold	**ui_style 0,1**	**ui_style 1,1**	**ui_style 2,1**
italic	*ui_style 0,2*	**ui_style 1,2**	*ui_style 2,2*
underline	ui_style 0,4	**ui_style 1,4**	ui_style 2,4

Auf Mac:

	small	xsmall	large
normal	ui_style 0,0	ui_style 1,0	ui_style 2,0
bold	**ui_style 0,1**	ui_style 1,1	**ui_style 2,1**
italic	*ui_style 0,2*	ui_style 1,2	*ui_style 2,2*
underline	ui_style 0,4	ui_style 1,4	ui_style 2,4

Das Vorwärts-Migration Script

Ausführungs-Kontext

Das Vorwärtsmigrations-Script wird ausgeführt, wenn ein Projekt ein in einer früheren Version von ARCHICAD gesichertes Projekt in einer neueren Version mit der aktualisierten Bibliothek geöffnet wird (seit Version ARCHICAD 15). Diese neue Bibliothek kann entweder manuell geladen werden oder unter Verwendung der Konsolidierungsoption des Bibliotheken-Managers. Falls eine platzierte Instanz eines Objektes eine neue, geänderte Haupt-ID besitzt und ein gültiges Vorwärts-Migrations-Script in der neuen Bibliothek besitzt, kann es automatisch von ARCHICAD ersetzt werden. Falls die Ausführung des Scriptes erfolgreich ist, wird das alte Objekt vom neuen ersetzt.

Diese Script ermöglicht es dem Objekt, die neuen Parameter auf Grundlage der alten Parameter, ohne den Verlust von Eigenschaften oder größeren Änderungen im Aussehen.

Allgemeine Empfehlung

Die erste Zeile des Scriptes schreibt die Globale Variable FROM_GUID (diese enthält die Haupt-ID des originalen zu migrierenden Objektes) in die Variable "actualGuid". Sie können die folgende Struktur verwenden, um die Wartbarkeit sicherzustellen.

Der Rest des Scriptes ist unterteilt in Subroutinen-Aufrufe, einer für jede Änderung der GUID. Jeder Block muss eine entsprechende Zeile in der Migrations-Tabelle des Objektes und einen Block im Rückwärts Migrations- Script haben. Die jüngste Änderung der Haupt ID muss immer der letzte Aufruf dieses Scriptes sein. In jedem Block setzen Sie die ID, dass diese mit der (_startID) startet und mit der (_endID) endet, definieren die Migrationslogik in einer Subroutine (Details hierzu im GDL-Referenz-Handbuch), und am Ende des Blocks schreiben Sie immer die neue "_endID" (oder setzen eine leere ID, d.h. dass der Upgrade-Prozess bei der vorigen Blockversion des Objektes anhält) in die Variable "actualGuid". Beispiel:

```
actualGUID = FROM_GUID

! ============================================================================
! Subroutines
! ============================================================================

    _startID    = "AAAA-AAAA-...AAA"
    _endID      = "BBBB-BBBB-...BBB"
gosub "migrationstepname_FWM"

! ============================================================================
! Set Migration GUID
! ============================================================================

setmigrationguid actualGUID

! ============================================================================
end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! en
! ============================================================================

! ============================================================================
! migrationstepname
! ============================================================================
"migrationstepname_FWM":
    if actualGuid = _startID then
        newParameter = oldParameter
        parameters newParameter = newParameter
        actualGuid = _endID
    endif
return
```

Das Rückwärts-Migration Script

Ausführungs-Kontext

Das Rückwärtsmigrations-Script wir ausgeführt, wenn ein Projekt als vorherige Version von ARCHICAD gespeichert wird. Falls die aktuelle Version des Bibliothekselements ein andere Main ID besitzt als sein Gegenstück aus der vorigen Version, wird das Migrationsscript des Objektes ausgeführt. Als Ergebnis davon wird das Objekt entweder herabgestuft (Downgrade), falls möglich (manchmal mit weniger Kompromissbereitschaft, falls es nicht wesentliche Funktionen berührt), oder es geht vollständig verloren (in diesem Fall wird es als "fehlender"

Objekt-Punkt in der niedrigeren Projekt-Version angezeigt). Letzteres passiert, wenn ein neuer Funktionssatz in der aktuellen Version eingeführt wird, welcher eine größere Änderung verglichen mit der Vorversion bedeutet.

Ein erfolgreicher Rückwärts-Migrations-Prozess sollte die Objektparameter derart konvertieren, dass größere Verluste von Eigenschaften oder Änderungen im Aussehen vermieden werden.

Allgemeine Empfehlung

Die erste Zeile des Scriptes stellt die Steuervariable der Kontinuität auf gültig. Sie können die folgende Struktur verwenden, um die Wartbarkeit sicherzustellen.

Der Rest des Scriptes ist in Subroutinen unterteilt: eine Änderung der Haupt-ID ist eine Subroutine. Jede Subroutine muss eine entsprechende Zeile in der Migrationstabelle des Objekts und eine entsprechende Subroutine im Vorwärts-Migrations-Script haben. Der **letzte Schritt zurück** in der geänderten Haupt-ID muss immer die **erste Subroutine** dieses Scripts sein.

Zu Beginn jeder Subroutine wird die Ziel-GUID geprüft. Wenn nicht leer, läuft das Skript in aufgerufener Reihenfolge. Falls nicht leer, wird das Script in der aufgerufenen Reihenfolge durchlaufen. Rückwärts-Migration funktioniert nur eine Version zurück, so dass die Ziel-GUID nur einmal eingestellt werden muss (außer, wenn Sie bei der Migration eine Abzweigung anlegen, um frühere Objektversionen zu trennen).

Das Ende der Subroutine dient der Übernahme der (alten) Ziel- ID in die Variable "targetGuid". Wenn Sie eine leere ID in die Variable einsetzen, wird der Downgrade-Prozess abgebrochen. Wenn die "targetGuid" der Globalen Variablen TO_GUID global (welche die Haupt-ID des Zielelements in der Konvertierung enthält) entspricht ist, ist der erste Teil des Migrationsprozess abgeschlossen.

Es ist sehr zu empfehlen, einen Titel oder eine kurze Beschreibung des Migrationsschritts für jede Subroutine hinzuzufügen. Sie sollten den gleichen Titel für das Vorwärts Migrations-Script-Paar der Subroutine verwenden.

Nachdem Sie die gewünschte Stufe der Rückkonvertierung des Objekts erreicht haben, müssen Sie die platzierte Objekt-ID mit Hilfe der `setmigrationguid einstellen.`

Falls die Migration eine leere ID zurückgibt, wird das Element in dem mit der früheren Version geöffneten Projekt fehlen.

```
targetGUID = TO_GUID

! ============================================================================
! Subroutines
! ============================================================================

gosub "migrationstepname_BWM"

! ============================================================================
! Set Migration GUID
! ============================================================================

setmigrationguid targetGUID

! ============================================================================
end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! en
! ============================================================================


! ============================================================================
! migrationstepname
! ============================================================================
"migrationstepname_BWM":
    if targetGUID <> "" then
        bMigrationSuccess = 1
        if bMigrationSuccess = 1 then
            oldParameter = newParameter
            parameters oldParameter = oldParameter
        else
            targetGuid = ""
        endif
    endif
return
```

Migrationstabelle

Jedes Mal, wenn Sie die Haupt-ID eines Objektes ändern, müssen Sie die alte ID in die Migrations-Tabelle des Elements füllen. Jede Zeile enthält eine frühere ID und eine ARCHICAD Versionsnummer (oder 0, wenn Sie mehr als einmal zwischen zwei Versionen wechseln). Während der Vorwärts-Migration scannt das Programm die Liste von IDs, mit der Vorauswahl der für den Migrationsprozess verfügbaren Elemente. Während

der Rückwärts-Migration wählt das Programm beim Scannen dieser Liste nur mit einer zu früheren ARCHICAD-Versionen äquivalenten Elemente aus. Jede Zeile dieser Tabelle muss mindestens eine entsprechende Subroutine im Vorwärts Migrations-Script und im Rückwärts Migrations-Script besitzen.

Das Schreiben von Makros

Versuchen Sie, häufig verwendete Funktionalitäten in Makros sammeln. Der Aufruf eines Makro-Objekt aus vielen Objekten kann die Bibliotheksgröße reduzieren und die Solidität erhöhen, indem Redundanz verringert wird.

Vermeiden Sie jedoch, Makros mit kleinen funktionalen Ergänzungen in Bezug auf den vorherigen Abstraktionsgrad zu erzeugen. Erstellen Sie z.B. nicht ein block_1x1x1 Makro zur Erzeugung eines Blocks mit der Größe 1m x 1m x 1m. Dies erhöht die Anzahl der Makroaufrufe unnötig und kann die Transparenz verschlechtern.

Verwenden Sie niemals .gdl als Makros, verwenden Sie stattdessen Makro-Objekte.

Wenn Sie ein Makro aufrufen, verwenden Sie immer den Aufruf `call` und setzen Sie den Namen des Makros in Anführungszeichen (z. B. als `call "m_rail_wired"`). Erstellen Sie keine Makro-Anrufe, bei denen der Name des Makros ist ein Parameter ist, um fehlende Makros aus Archivdateien zu vermeiden. ARCHICAD speichert nur die Standard-Makros in eine Archivdatei. (Workaround: Aufrufen aller Parameterwerte als Makro nach dem end Ausdruck.)

Seien Sie vorsichtig im Umgang mit dem Parameter-Buffer. Speichern Sie den Inhalt des Buffers am Anfang des Skripts, wenn Sie ihn benutzen wollen. Stellen Sie sicher, dass nur die festgelegten (return) Werte sich noch am Ende des Scriptes im Buffer befinden.

Makro Rückgabeparameter

Seit ARCHICAD 10 können aus Makros Parameter an das aufrufende Objekt zurückgegeben werden. Auf der aufrufenden Seite können zurückgegebene Werte mit dem Schlüsselwort `returned_parameters` gesammelt werden, gefolgt von der Liste der Variablen. Die zurückgegebenen Werte werden in diesen Variablen in der Reihenfolge abgelegt, wie sie vom aufgerufenen Makro übergeben werden. Die Zahl und der Typ der im Aufrufer angegebenen Variablen und der, die vom Makro zurückgegebenen werden, muss nicht übereinstimmen. Falls im Aufrufer mehr Variablen angegeben sind als im Makro, werden diese auf Ganzzahl und 0 gesetzt. Die Typenkompatibilität wird nicht geprüft: Der Typ der Variablen im aufrufenden Objekt wird an die übergebenen Werte angepaßt. Falls eine der Variablen der Aufrufer ein dynamischer Array ist, werden alle folgenden Werte darin gespeichert.

Im Makroobjekt definieren die Befehle `end` und `exit` die Werte, welche zum aufrufenden Objekt zurückkehren sollen. Siehe folgendes Beispiel.

Erweitertes "parameters all"

Seit ARCHICAD 10 aufwärts können Sie nach dem Befehl `parameters all` zusätzliche Parameter angeben, welche an das Makro übergeben werden sollen. Diese werden die Werte überschreiben, die vom Aufrufer kommen oder Parameter des aufgerufenen Makros, welche als grundeingestellt belassen wurden. Das Makro kann in diesem Fall auch Parameters zurückgeben.

Schnellerer Makroaufruf

Die Geschwindigkeit der Übertragung von Parameter-Werten zwischen dem Aufrufer-Objekt und dem Makro wurde in ARCHICAD 10 verbessert. Finden Sie Tipps zu Geschwindigkeitsverbesserungen bei Nutzung von Makro-Aufrufen in„Geschwindigkeits-Betrachtungen“.

Beispiel Makroaufruf

Script im Aufruf-Objekt.

```
call "myMacro" parameters all extraParam = 1
call "myMacro" parameters returned_parameters realWidth
call "myMacro" parameters all extraParam = 1 returned_parameters realWidth
call "myMacro" parameters all returned_parameters realWidth
```

Script im Makro.

```
realWidth = 2
end realWidth
```

Probleme bei der Rückwärtskonvertierung

Mit der Einführung von ARCHICAD 19 werden alle für das Öffnen von 3D-bezogenen Sichten oder Standorten benötigten Berechnungen als Hintergrund-Prozesse durchgeführt.

Unterstütze 3D-Ansichtstypen:

- 3D Fenster
- Schnitte
- Ansichten
- Innenansichten (außer es ist folgendes aktiviert: "Umgebenen Bereich hinzufügen" oder "An Raum angepasst")
- 3D Dokumente

Falls der Hintergrund-Prozess erfolgreich war, benötigen die abgefragten Sichten nur wenige Sekunden zum Öffnen. **Es kann jedoch einige nicht thread-sichere Bibliothekselemente oder im Plan platzierte Objekte geben, welche die Hintergrundberechnung außer Kraft setzen:**

- Raumstempel
- Objekte, welche Textengine-Operationen enthalten (außer den Befehlen SET STYLE und DEFINE STYLE)
- Objekte mit den folgenden REQUESTs: "CUSTOM_AUTO_LABEL", "ZONE_COLUS_AREA", "MATCHING_PROPERTIES", "ASSOCEL_PROPERTIES", "STYLE_INFO", "TEXTBLOCK_INFO", "FONTNAMES_LIST"
- Objekte, welche Makros oder REQUESTs mit Variablennamen aufrufen oder nicht thread-sichere Makros. Project2-Befehle oder Definitionen von Symbolschraffuren werden als nicht sichere Makroaufrufe gewertet.

Die Beziehung zwischen GDL-Addons und dem Hintergrund-Prozess hängt von den Addons selber ab.

Deterministische Addons (beeinflussen den Hintergrund-Prozess nicht):

- Polygon Operationen
- Property Addon
- Falls ausschließlich im Read-Modus verwendet, und wenn die entsprechenden Dateien in der geladenen Bibliothek vorliegen: Text oder Data I/O Addons, XML Addon

Nicht-determistische Addons (unterbinden den Hintergrund-Prozess):

- DateTime Addon
- FileManager Addon
- Falls nicht im Read-Modus verwendet, oder wenn die entsprechenden Dateien in der geladenen Bibliothek nicht vorliegen: Text oder Data I/O Addons, XML Addon

Die Objektscripte werden statisch untersucht, so dass die Hintergrund-Konvertierung deaktiviert ist, selbst wenn die behindernde Funktion mit den derzeitigen Einstellungen des Objektes gar nicht ausgeführt wird.

Um die Kompatibilität der geladenen Bibliothekselemente mit den Hintergrund-Prozessen zu prüfen, benutzen Sie bitte den Befehl "Überprüfen ob Bibliothekselemente Thread sicher sind" im Bibliotheken-Entwickler-Menü.

Geschwindigkeits-Betrachtungen

Versuchen Sie die Verwendung von `project2` zu vermeiden, da dies den Plan-Neuaufbau verlangsamt.

Reduzieren Sie die Anzahl der Flächen in Ihrem Modell auf das Minimum, um die 3D-Erzeugung schneller zu machen. Verwenden Sie `RESOL`, `TOLER` und `RADIUS` um die Segmentierung von gekrümmten Oberflächen zu kontrollieren.

Beachten Sie, dass geschlossene Körper in 3D schneller aufzubauen sind als offene (beispielsweise ist ein Zylinder schneller als ein offenes Rohr).

Bedenken Sie beim Scripten im Master-Script, dass das Master-Script vor jedem anderen Script-Typ ausgeführt wird, weshalb keine Script-spezifische Berechnung hierher gehört. Dies ist der Ort für allgemeine Berechnungen, welche von mehreren Scripten benötigt werden.

Vermeiden Sie beim Programmieren von Türen und Fenstern unnötige Wandschnitte (`wallhole` und `wallniche`).

Verwenden Sie Ganzzahl-Werte und Operationen, wann immer vernünftig, da diese viel schneller als Fließkomma-Operationen sind.

Versuchen Sie die Verwendung von String-Operationen zu minimieren.

Im Fall von Makroaufrufen verwenden Sie die selbe Parameterreihenfolge nach dem `call` Befehl wie sie in der Parameterliste des Makros vorliegt. `call "myMacro" parameters all` ist schneller, wenn die Parameterreihenfolge von Makro und Aufrufer ähnlich ist. Versuchen Sie die Übertragung von String-Typ-Parameter in Makroaufrufe zu vermeiden. Verwenden Sie wenn möglich, numerische Typen.

Windows-Macintosh Kompatibilität

Obwohl GDL-Objekte und Bibliotheken von GRAPHISOFT als plattformunabhängig gedacht waren, tauchen die folgenden Schwierigkeiten auf, wenn Windows-Objekte manuell nach Macintosh transportiert werden:

- Windows Schriften werden durch Standard Macintosh Schriften in Objekten und Listenvorlagen ersetzt und vice-versa.
- Listendateien im Textformat (listset.txt, listkex.txt, Listenschablonen, etc.) können Zeilenumbrüche verlieren, weshalb Listen nicht mehr funktionieren (nicht-utf-8 codierte Texte, normalerweise)

7.2 Wechseln der Plattform mit Binär-Bibliotheken

Um die genannten Probleme zu vermeiden, speichern Sie ein .pla Archiv ihrer Bibliothek auf der ersten Plattform und entpacken das Archiv auf der zweiten Plattform. Auf diese Weise werden auch die nicht-utf-8 Dateien korrekt konvertiert.

Bilder und HDPI-Unterstützung in GDL

Ab ARCHICAD 21 ist eine echte HDPI-Unterstützung für OS X Geräte verfügbar. Um diese Funktion zur Verfügung zu stellen, werden skalierbare Vektorgrafiken (.svg) Quellbilder verwendet, um Multi-Repräsentations- .tiff-Bilder zu erstellen, die Versionen derselben Grafik mit verschiedenen Auflösungen enthalten. ARCHICAD entscheidet zur Laufzeit, welche Bildauflösung für das aktuelle Display aus dem verfügbaren Satz von 100%, 150% und 200 am besten ist. Auf Windows-Plattform ist 100% die Standardauflösung. Allerdings haben Bilder, die aus .svg-Dateien erstellt wurden, auch unter Windows einen etwas anderen Look.

Diese Option ist nur über das LP_XMLConverter-Tool verfügbar. Die .svg-Quellbilddateien werden bei der Konvertierung der Bibliothek automatisch in .tiff-Bilder konvertiert. Ein .gsm-Objekt kann keine .svg-Bilder verarbeiten. Stellen Sie daher sicher, dass die Referenzstrings der Bildnamen in Skripten die Erweiterung .tiff enthalten (oder lassen Sie die Erweiterung weg).

Testen Sie immer Vektorbilder in allen Auflösungen (die .tiff Bilder können von jedem Bildeditor, Auflösung für Auflösung überprüft werden), um verschwommene Bilder nach der Skalierung zu vermeiden (verwenden Sie Linien, die auf volle Pixel in der Quelle .svg so weit wie möglich skaliert sind. Gleiche Empfehlung wie für .png Bilder.)

Die Syntaxanforderungen eines beliebigen als .svg **eingebetteten Bildes** (wo das Bild in dem binären Bibliothekselement selbst kompiliert wird) in GDLPict-, Picture-, InfoPict- .xml Abschnitten sind die folgenden:

- MIME Attribut ist `"image/svg"`. Falls der MIME-Typ und die Quellbild-Erweiterung unterschiedlich sind, endet die Konvertierung mit einer Warnung (`MIME ist "image/svg", aber die Image-Dateierweiterung ist nicht svg`)
- SectionsFlag-Attribut ist `"1"`. Dieses Flag löst die .tiff-Konvertierung aus. Wenn das Bild ein .svg ist und das Flag unterscheidet sich davon, wird während der Konvertierung eine Warnung angezeigt(`SectionFlags sollte "1" sein im Falle eines svg-Bildes`).

Der Rest der Bildreferenz in GDLPict-, Picture-, InfoPict- .xml Abschnitten hat sich nicht geändert. Achten Sie darauf, die Erweiterung des Quellbildes direkt im Pfad zu anzugeben.

Nicht eingebettete Grafiken, welche direkt mit ihrem Namen in den Objektskripten bezeichnet werden, werden ebenfalls vom LP_XMLConverter-Tool bearbeitet:

- Das .svg-Bild sollte Teil der Bibliotheksquelle sein, aber nicht im Ordner _images liegen.
- Die Umwandlung erzeugt die .tiff-Pendants der .svg-Quellbilder an den exakt gleichen Ort. Die .svg-Quellbilder werden auch in die Binärbibliothek in einen separaten Ordner kopiert (_svg_source name extension), um den source.xml zu .gsm zu reverse.xml Vergleichs-Workflow zu unterstützen. Dieser zusätzliche Ordner kann durch die `-excludesvg` -Option des LP_XMLConverter tools vermieden werden.

Jede Art von .svg zu .tiff Umwandlungsfehler führt zu einer Warnung oder einem Fehler während des Bibliotheksaufbaus.

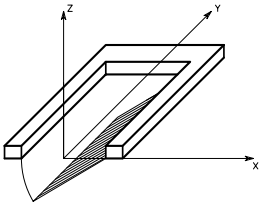
Für Tutorials und Beispiele zu diesem Thema und dem LP_XMLConverter Tool, schauen Sie sich die *GDL Center Tips and Tricks* [https://gdl.graphisoft.com/tips-and-tricks/how-to-use-the-lp_xmlconverter-tool] Anweisungen an.

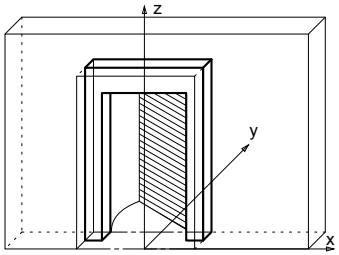
Türen und Fenster

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.

Positionierung

Eine Tür ist korrekt definiert, wenn ihre Platzierung in einer Wand wie folgt funktioniert: Mausklick auf der rechten Seite des Einsetzpunktes bedeutet, dass das Türblatt auf die selbe Seite auf der rechten Seite öffnet. Ein Fenster ist korrekt definiert, wenn die Seite, die beim Einsetzen angeklickt wird, mit der Außenseite des Fensters korrespondiert.

Eine Öffnungsposition kann eine von 8 Formen besitzen. Diese werden von drei globalen Variablen in GDL vertreten:

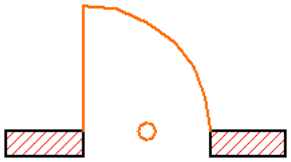
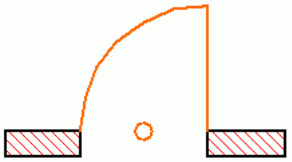
- Spiegeln zur Y-Z-Ebene in 3D oder um die Y-Achse in 2D (`SYMB_MIRRORED`)
- Spiegeln um die Längsachse der Wand (Rotation um 180 Grad: `SYMB_ROTANGLE`)
- Umklappen (in der deutschen Version ist das der Spiegeln-Schalter, aber nicht der Spiegeln-Befehl) (`WIDO_REVEAL_SIDE`)

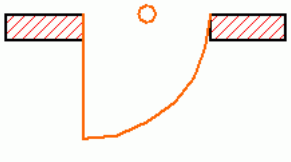
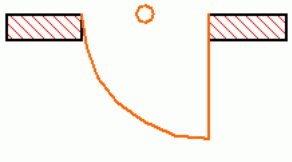
Üblicherweise sollte jede Seite des Fensters auf eine andere Weise auf diese Bedingungen reagieren. Die Spezifikation muss klar sein bezüglich der Entscheidung, wie die Teile des Objekts reagieren sollen, oder wie nicht. Z.B. bewegt sich ein Türblatt mit diesen Transformationen, aber der Hohlraum-Verschluss nicht. Um die Bibliothekselemente konsistent zu halten, sollten mehrere Transformationen für diese Kombinationen

verwendet werden. Bei Änderung der Anschlagseite (Spiegeln/Umklappen), wird das Bibliothekselement gespiegelt und um den Wert der Nennrahmenstärke zurückversetzt.

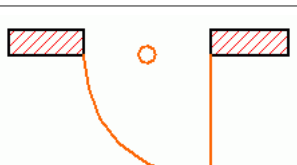
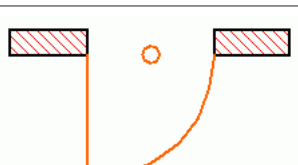
Illustration der 8 Zustände mit einer vereinfachten Tür - die kleinen Kreis markieren den Ursprung.

Globale Variablen 1.	**Beispielskizze 1.**	**Globale Variablen 2.**	**Beispielskizze 2.**

`WIDO_REVEAL_SIDE = 0`
`SYMB_MIRRORED = 0`
`SYMB_ROTANGLE = 0`
		`WIDO_REVEAL_SIDE = 0`
		`SYMB_MIRRORED = 1`
		`SYMB_ROTANGLE = 0`
			

`WIDO_REVEAL_SIDE = 1`
`SYMB_MIRRORED = 0`
`SYMB_ROTANGLE = 180`
		`WIDO_REVEAL_SIDE = 1`
		`SYMB_MIRRORED = 1`
		`SYMB_ROTANGLE = 180`
			

`WIDO_REVEAL_SIDE = 1`
`SYMB_MIRRORED = 0`
`SYMB_ROTANGLE = 0`
		`WIDO_REVEAL_SIDE = 1`
		`SYMB_MIRRORED = 1`
		`SYMB_ROTANGLE = 0`	

`WIDO_REVEAL_SIDE = 0`
`SYMB_MIRRORED = 0`
`SYMB_ROTANGLE = 180`
		`WIDO_REVEAL_SIDE = 0`
		`SYMB_MIRRORED = 1`
		`SYMB_ROTANGLE = 180`
			

Beispielcode, welcher die von ARCHICAD automatisch durchgeführten Transformationen aufhebt:

```
! 2D-Script
bRotated = round_int (SYMB_ROTANGLE) = 180
if bRotated then
 rot2 180
endif
if SYMB_MIRRORED then
 mul2 -1, 1
endif
if WIDO_REVEAL_SIDE exor bRotated then
 add2 0, WALL_THICKNESS
endif

! 3D-Script
bRotated = round_int (SYMB_ROTANGLE) = 180
if bRotated then
 roty 180
endif
if SYMB_MIRRORED then
 mulx -1
endif
if WIDO_REVEAL_SIDE exor bRotated then
 addz -WALL_THICKNESS
endif
```

Beachten Sie, dass, obwohl Spiegeln/Umklappen und Spiegeln (per Befehl) bei allen Türen und Fenstern möglich ist, es in Hersteller-Bibliotheken falsch ist, wo ein Bibliothekselement ein echtes Fenster modelliert - Was natürlich nicht umgeklappt werden kann. In diesem Fall sollte das Script die Umklapp-Spiegelung von ARCHICAD rückgängig machen.

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
- 3D-bezogene Aufgaben
  - Erstellung von nicht rechteckigen Türen/Fenstern in geraden Wänden
  - Erstellung von rechteckigen Türen/Fenstern in gebogenen Wänden
  - Erstellung von nicht rechteckigen Türen/Fenstern in gekrümmten Wänden

- 2D-bezogene Aufgaben
  - Individuelle Wandöffnung schneiden
  - WALLHOLE2
  - Das Wandpolygon erweitern
  - WALLBLOCK2
  - WALLLINE2
  - WALLARC2

Rechteckige Türen/Fenster in geraden Wänden

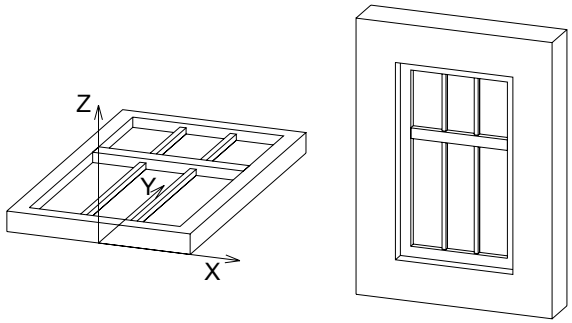
Das ist die leichteste und direkteste Methode der Erstellung von Türen und Fenstern. Die Verwendung von einfachen GDL-Befehlen, wie PRISM_ oder RECT wird empfohlen.

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 WALL_MAT_A, WALL_MAT_B und WALL_MAT_EDGE einstellen, die die Oberflächenmaterialien der Wand, in der die Türen/Fenster platziert werden, angeben. Im 2D-Script können die globalen Variablen WALL_SECT_PEN, WALL_FILL_PEN und WALL_FILL 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 für 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 allgemein benutzten Rahmen, Paneelen 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:



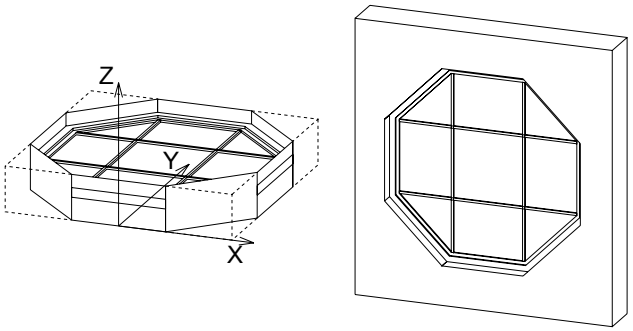
```
a=0.9: b=1.5: c=0.1: d=0.08
e=0.08: f=0.9: g=0.03: h=3
PRISM_ 10, c,
        -a/2, 0, 15, a/2, 0, 15,
        a/2, b, 15, -a/2, b, 15,
        -a/2, 0, -1,
        -a/2+d, d, 15, a/2-d, d, 15,
        a/2-d, b-d, 15, -a/2+d, b-d, 15,
        -a/2+d, d, -1
ADD -a/2+d, f, 0
BRICK a-2*d, e, c
ADD -g/2, -f+d, c/2
GOSUB 1
ADDZ -g
GOSUB 1
DEL 2
MATERIAL "Glass - Blue"
ADD 0, -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
```

3D-bezogene Aufgaben

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 polygonale 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

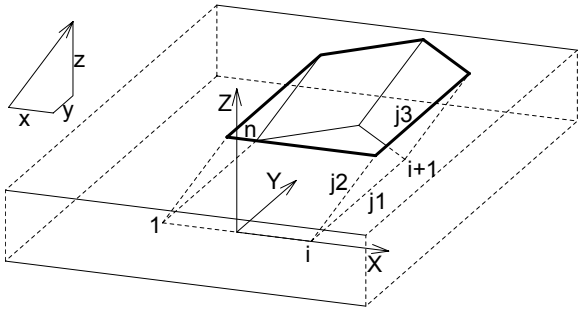
status:

`1:` verwendet die eigenen Attribute des Körpers der erzeugten Polygone und Kanten.
`2:` erzeugte Schnittflächenpolygone werden wie normale Polygone behandelt.

`xi, yi:` Koordinaten des Querschnittspolygons

`maski:` gleicht dem CUTPOLYA-Befehl.
`maski = j1 + 2*j2 + 4*j3 + 64*j7`, hierbei kann j jeweils 0 oder 1 sein.

`x, y, z:` optionaler Richtungsvektor (z-Achse von Türen/Fenstern wird grundeingestellt)

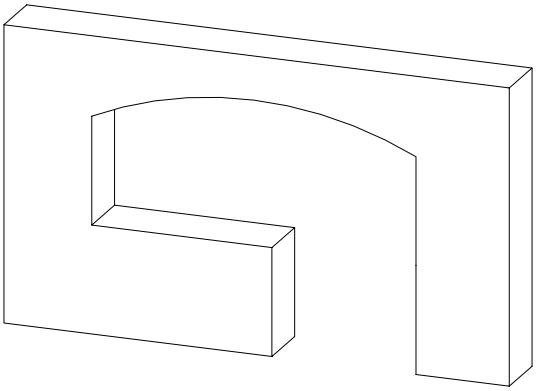


Dieser Befehl kann im 3D-Script von Türen/Fenstern zum Ausschneiden benutzerdefinierter Öffnung(en) in der Wand benutzt werden, wo sie platziert werden. Während der 3D-Generierung der aktuellen Wand wird das 3D-Script von allen ihren Türen/Fenstern ohne Generierung eines Modells interpretiert werden, 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 WALLHOLEs 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 schattierten und photorealistischen Darstellungen oder Ausschnittsfehlern führen. Konvexe Polygone können kombiniert werden, um konkave zu erstellen. Spiegel-Transformationen beeinflussen die Schnittrichtung auf unvorhersehbare Weise - um unkomplizierte Ergebnisse zu erhalten, sollten Sie WALLNICHE.

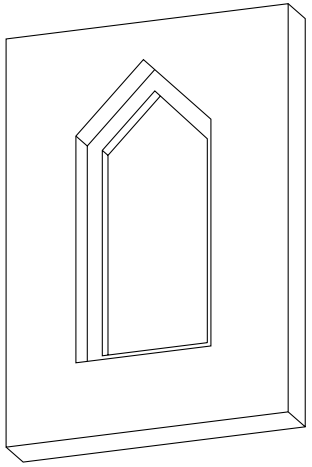
Beispiel 1:



```
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
```

Beispiel 2:

```
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, [mat1,]
...
xn, yn, maskn[, matn]

Gleicht dem CUTFORM-Befehl.

method: Steuert die Form des Schnittkörpers:

1: prismenförmig

2: pyramidenförmig

3: keilförmiger Schnittkörper. Die Richtung der Firstkante 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 + 256*j9`, hierbei kann j jeweils 0 oder 1 sein.

`j1:` verwendet die eigenen Attribute des Körpers der erzeugten Polygone und Kanten.

`j2:` erzeugte Schnittflächenpolygone werden wie normale Polygone behandelt.

`j4:` definiert die Begrenzung des Schnittes (mit j4),

`j5:` definiert die Begrenzung des Schnittes (mit j5),

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

`j7:` Kanten, generiert durch den Schnittkörperboden werden unsichtbar

`j8:` Kanten, generiert durch die Schnittkörperspitze werden unsichtbar

`j9:` Schnittflächen haben benutzerdefinierte Seitenmaterialien (mati)

`j4 = 0 und j5 = 0:` endlicher Schnitt

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

`j4 = 1 und j5 = 1:` unendlicher Schnitt

`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 Ursprung des lokalen Koordinatensystems 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.

`mati:` Seitenmaterialien der Schnittflächen (wenn Status j9 = 1)

`mask:` Definiert die Sichtbarkeit der Kanten des Schnittkörpers:

`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

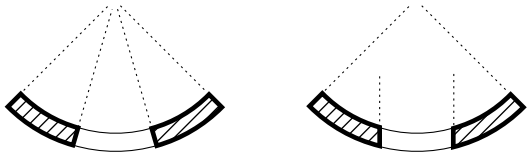
`j4:` Die untere Kante der Schnittform ist sichtbar

`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.

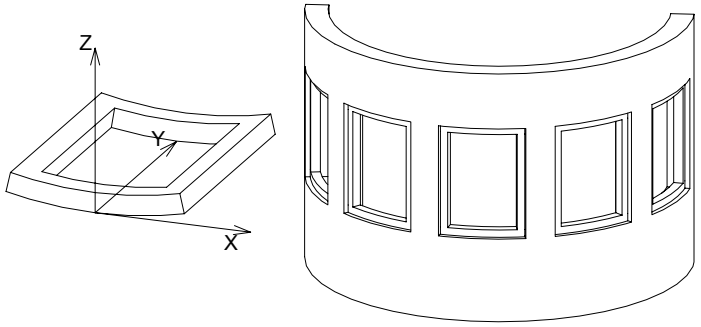


Die Öffnung in der Wand an der linken Seite wird erstellt, wenn das Programm die Öffnung für die Türen/Fenster automatisch ausschneidet. In diesem Falle werden die Seiten in radialer Richtung gestellt. Im Bild rechts wird die Öffnung durch den WALLHOLE-Befehl im 3D-Script des Objektes vom Türen-/Fenster-Typ 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: Fenster mit einem Rahmen, welcher der Krümmung der Wand folgt

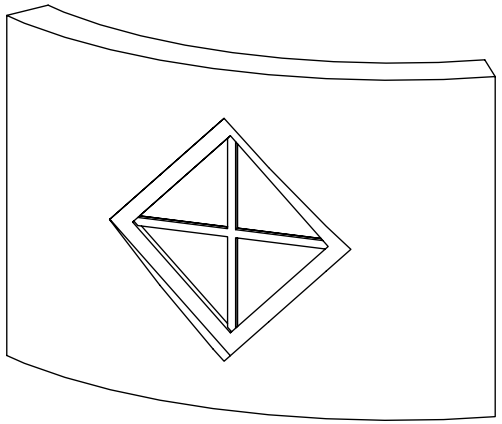


```
RESOL 72
ROTX -90 : MULY -1
C= 0.12 : Z=360*A/(2*WIDO_ORIG_DIST*PI)
Y= 360*C/(2*WIDO_ORIG_DIST*PI) : A1= 270+Z/2 : A2=270-Z/2
GOSUB "curved_horizontal_frame"
ADDZ B
MULZ -1
GOSUB "curved_horizontal_frame"
DEL 2
ADDZ C
GOSUB "vertical_frame"
MULX -1
GOSUB "vertical_frame"
END
"curved_horizontal_frame":
    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
"vertical_frame":
    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 gekrümmten Wänden

Die allgemeinen Richtlinien für rechteckige Türen/Fenster in gekrümmten Wänden gelten auch hier.

Beispiel:



```
wFrame=0.1: wDivider=0.025
Z=A/2-SQR(2)*wFrame: Y=A/2-SQR(2)*wFrame-wDivider
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 "cross_divider"
ADDZ 0.03
GOSUB "cross_divider"
ADDY -Z
SET MATERIAL "Glass-Blue"
ROTZ 45
RECT SQR(2)*Z, SQR(2)*Z
END
```

```
"cross_divider":
    PRISM_ 16, 0.03,
            0, -Z, 15,
            wDivider, -Y, 15,
            wDivider, -wDivider, 15,
            Y, -wDivider, 15,
            Z, 0, 15,
            Z, wDivider, 15,
            wDivider, wDivider, 15,
            wDivider, Y, 15,
            0, Z, 15,
            -wDivider, Y, 15,
            -wDivider, wDivider, 15,
            -Y, wDivider, 15,
            -Z, 0, 15,
            -Y, -wDivider, 15,
            -wDivider, -wDivider, 15,
            -wDivider, -Y, 15
    RETURN
```

2D-bezogene Aufgaben

Individuelle Wandöffnung schneiden

Wenn Sie Türen/Fenster platzieren, wird eine rechteckige Öffnung in der Wand als Grundeinstellung geschnitten. Die Größe dieser Öffnung in 2D wird durch die Parameter A des Bibliothekselementes vom Türen/Fenster-Typ bestimmt. Beim Implementieren von individuellen Anschlägen oder Sonderleibungen wird das Schneiden von Wandöffnungen individueller Form oder eine kleine Erweiterung in der Grundrissansicht benötigt.

Eine korrekte Lösung für diese Problematik kann erreicht werden, wenn Sie die Befehle WALLHOLE2, WALLBLOCK2, WALLLINE2 und WALLARC2 verwenden.

WALLHOLE2

```
WALLHOLE2 n, fill_control, fill_pen, fill_background_pen,
        fillOrigoX, fillOrigoY, fillAngle,
        x1, y1, s1,
        ...
        xn, yn, sn
```

Wandöffnungsdefinition für den Planausschnitt, mit einem Deckschraffurpolygon verbunden. Nur der Schnittteil der Wand wird berührt, Ansichtswandpolygone bleiben unberührt.

Das Deckschraffurpolygon hat keine Kontur.

Dieser Befehl kann nur im 2D Script der Türen/Fenster Objekte verwendet werden.

Die Parametrisierung des Befehls bestimmt sich hauptsächlich nach dem Ausdruck POLY2_B{2}.

fill_control:

fill_control = $2*j_2 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7$, hierbei kann j jeweils 0 oder 1 sein.

j_2: Zeichnen Sie Deckschraffur auf dem Polygon

j_4: lokale Schraffurausrichtung

j_5: lokale Schraffur sollte nach der Wand ausgerichtet werden (Schraffurursprung ist im Wandursprung und die Richtungen sind entsprechend),

j_6: Schraffur ist Bauteilschraffur (Grundeinstellung ist Zeichenschraffur)

j_7: Schraffur ist Deckschraffur (nur falls j6 = 0, Grundeinstellung ist Zeichenschraffur)

WALLHOLE2{2}

```
WALLHOLE2{2} n, frame_fill, fillcategory, distortion_flags,
        fill_pen, fill_background_pen,
        fillOrigoX, fillOrigoY,
        mxx, mxy, myx, myy,
        innerRadius,
        x1, y1, s1,
        ...
        xn, yn, sn
```

Erweiterte Version des Befehls WALLHOLE2, wo die Schraffurverformung auf verbesserte Weise bestimmt werden kann.

Sie entspricht POLY2_B{5} in der geometrischen Definition.

distortion_flags:

distortion_flags = $j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8$, hierbei kann j jeweils 0 oder 1 sein.

Der gültige Wert der Verformungs_flags liegt zwischen 0 und 255. Verwenden Sie keine Werte außerhalb dieses Bereiches.

j_1-j_7: gleicht dem POLY2_B{5}-Befehl.

j_8: lokale Schraffur sollte nach der Wand ausgerichtet werden (Schraffurursprung ist im Wandursprung und die Richtungen sind entsprechend); wenn j4 gesetzt ist. Verformungsmatrix (mij Parameter) sind nicht vorhanden.

Das Wandpolygon erweitern

WALLBLOCK2

```
WALLBLOCK2 n, fill_control, fill_pen, fill_background_pen,
        fillOrigoX, fillOrigoY, fillAngle,
        x1, y1, s1,...
        xn, yn, sn
```

WALLBLOCK2{2}

```
WALLBLOCK2{2} n, frame_fill, fillcategory, distortion_flags,
        fill_pen, fill_background_pen,
        fillOrigoX, fillOrigoY,
        mxx, mxy, myx, myy,
        innerRadius,
        x1, y1, s1,
        ...
        xn, yn, sn
```

Wandöffnungs(erweiterungs) definition für den Planausschnitt. Sowohl die Schnitt- als auch die Ansichtswandpolygone werden durch das bestimmte Polygon geschnitten. Wandöffnungen, die über den Befehl WALLHOLE2 in einem anderen Fenster/Tür Objekt definiert wurden, schneiden das über den Befehl WALLBLOCK2 erstellte Polygon. Wandöffnungen, die aus dem gleichen Objekt stammen benehmen sich anders.

Dieser Befehl kann nur im 2D Script der Türen/Fenster Objekte verwendet werden.

Die Parametrisierung des Befehls bestimmt sich genau nach dem Ausdruck WALLHOLE2.

WALLLINE2

```
WALLLINE2 x1, y1, x2, y2
```

Wandlinien(erweiterungs) definition zwischen zwei Punkten für den Planausschnitt. Wandöffnungen, die über den Befehl WALLHOLE2 in einem anderen Fenster/Tür Objekt definiert wurden, schneiden die von diesem Befehl erzeugte Linie ab, während WALLHOLEs aus dem selben Objekt das nicht tun.

Dieser Befehl kann nur im 2D Script der Türen/Fenster Objekte verwendet werden.

Die Parametrisierung des Befehls bestimmt sich nach dem Ausdruck LINE2.

WALLARC2

```
WALLARC2 x, y, r, alpha, beta
```

Ein Bogen mit dem Mittelpunkt bei (x, y) zwischen Winkel alpha und beta, und mit Radius r, der von der enthaltener Wand gezeichnet wird. Wandöffnungen, die über den Befehl WALLHOLE2 in einem anderen Fenster/Tür Objekt definiert wurden, schneiden den mit diesem Befehl erzeugten Bogen ab, während WALLHOLEs aus dem selben Objekt das nicht tun.

Dieser Befehl kann nur im 2D Script der Türen/Fenster Objekte verwendet werden.

Die Parametrisierung des Befehls bestimmt sich nach dem Ausdruck ARC2.

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. Sie können diese GDL-Scripts als Muster für Ihre eigenen Bibliothekselemente verwenden.

Befehle

Häufige Befehle

```
FILE_DEPENDENCE
MOD
AND
OR
EXOR
FOR
TO
STEP
NEXT
DO (bei DO - WHILE, bei WHILE - ENDWHILE)
WHILE (bei DO - WHILE, bei WHILE - ENDWHILE)
ENDWHILE
REPEAT
UNTIL
IF (bei IF - GOTO, bei IF - THEN - ELSE - ENDIF)
THEN (bei IF - GOTO, bei IF - THEN - ELSE - ENDIF)
GOTO (bei IF - GOTO, bei GOTO)
```

GOSUB (bei IF - GOTO, bei GOSUB)
ELSE
ENDIF
RETURN
END
EXIT
BREAKPOINT

FILLTYPES_MASK (bei DEFINE FILL, bei DEFINE FILLA, bei DEFINE SYMBOL_FILL, bei DEFINE SOLID_FILL, bei DEFINE EMPTY_FILL, bei DEFINE LINEAR_GRADIENT_FILL, bei DEFINE RADIAL_GRADIENT_FILL, bei DEFINE TRANSLUCENT_FILL, bei DEFINE IMAGE_FILL, bei VALUES)
PROFILETYPES_MASK
DICT
DIM
PUT
GET
USE
NSP
CALL
RETURNED_PARAMETERS
DEFAULT
PRINT

HASKEY
REMOVEKEY
VARDIM1
VARDIM2
PARVALUE_DESCRIPTION
ABS
CEIL
INT
FRA
ROUND_INT

```
SGN
SQR
ACS
ASN
ATN
COS
SIN
TAN
PI
EXP
LGT
LOG
NOT
MIN
MAX
RND
BITTEST
BITSET
REQ
REQUEST (bei REQUEST, bei APPLICATION_QUERY)
IND
LIBRARYGLOBAL
STR
STR{2}
SPLIT
STW
STRLEN
STRSTR
STRSUB
STRTOUPPER
STRTOLOWER
OPEN
INPUT
```

```
VARTYPE
OUTPUT
CLOSE
INITADDONSCOPE
PREPAREFUNCTION
CALLFUNCTION
CLOSEADDONSCOPE
```

Reservierte Befehle

Die Schlüsselwörter in dieser Liste sind aus Kompatibilitätsgründen reserviert oder wurden nicht publiziert.

```
BAS
BOX
CONT
FILTER
GDLBIN
HIP_ROOFS
LIN_
LINE
MIGRATIONWARNING
NOD
NODE
ORIGO
PARS
PAUSE
PLOTMAKER
PLOTTER
RECT_
REF
SFLINE
TET
TETRA
TRI
WALL_
VOCA_
UI_OK
UI_CANCEL
```

Befehle für den 3D-Bereich

```
ADDX
ADDY
ADDZ
ADD
MULX
MULY
MULZ
MUL
ROTX
ROTY
ROTZ
ROT
XFORM

BLOCK
BRICK
CYLIND
SPHERE
ELLIPS
CONE
PRISM
PRISM_
CPRISM_
CPRISM_{2}
CPRISM_{3}
CPRISM_{4}
BPRISM_
FPRISM_
HPRISM_
SPRISM_
SPRISM_{2}
```

```
SPRISM_{3}
SPRISM_{4}
SLAB
SLAB_
CSLAB_
CWALL_
BWALL_
XWALL_
XWALL_{2}
XWALL_{3}
BEAM
CROOF_
CROOF_{2}
CROOF_{3}
CROOF_{4}
MESH
ARMC
ARME
ELBOW
EXTRUDE
PYRAMID
REVOLVE
REVOLVE{2}
REVOLVE{3}
REVOLVE{4}
REVOLVE{5}
RULED
RULED{2}
RULEDSEGMENTED
RULEDSEGMENTED{2}
SWEEP
TUBE
TUBE{2}
```

```
TUBEA
COONS
COONS{2}
MASS
MASS{2}
POLYROOF
POLYROOF{2}
POLYROOF{3}
POLYROOF{4}
EXTRUDEDSHELL
EXTRUDEDSHELL{2}
EXTRUDEDSHELL{3}
REVOLVEDSHELL
REVOLVEDSHELL{2}
REVOLVEDSHELL{3}
REVOLVEDSHELLANGULAR
REVOLVEDSHELLANGULAR{2}
REVOLVEDSHELLANGULAR{3}
RULEDSHELL
RULEDSHELL{2}
RULEDSHELL{3}
TEXT
BODY
BASE
NURBSCURVE2D
NURBSCURVE3D
NURBSSURFACE
NURBSVERT
NURBSEDGE
NURBSTRIM
NURBSTRIMSINGULAR
NURBSFACE
NURBSFACE{2}
```

```
NURBSLUMP
NURBSBODY
POINTCLOUD
CUTPLANE
CUTEND
(bei CUTPLANE, bei CUTPLANE{2}, bei CUTPLANE{3}, bei CUTPOLY, bei CUTPOLYA, bei CUTSHAPE)
CUTPLANE{2}
CUTPLANE{3}
CUTPOLY
CUTPOLYA
CUTSHAPE
CUTFORM
CUTFORM{2}
GROUP
ENDGROUP
ADDGROUP
ADDGROUP{2}
ADDGROUP{3}
SUBGROUP
SUBGROUP{2}
SUBGROUP{3}
ISECTGROUP
ISECTGROUP{2}
ISECTGROUP{3}
ISECTLINES
PLACEGROUP
KILLGROUP
SWEEPGROUP
SWEEPGROUP{2}
SWEEPGROUP{3}
SWEEPGROUP{4}
SWEEPGROUP{5}
CREATEGROUPWITHMATERIAL
```

```
BINARY
WALLNICHE

HOTSPOT
HOTLINE
HOTARC
LIN_
RECT
POLY
POLY_
PLANE
PLANE_
CIRCLE
ARC
LIGHT
PICTURE
RICHTEXT
VERT (bei VERT, bei VERT{2})
TEVE
VECT
EDGE
PGON
PGON{2}
PGON{3}
PIPG
COOR
COOR{2}
COOR{3}
MODEL
WIRE
SURFACE
SOLID
MATERIAL (bei [SET] MATERIAL, bei IND)
```

```
BUILDING_MATERIAL (bei [SET] BUILDING_MATERIAL, bei IND)
SECT_FILL
SECT_ATTRS
SECT_ATTRS{2}
SHADOW
ON
OFF
AUTO
DEFINE MATERIAL (bei DEFINE MATERIAL, bei DEFINE MATERIAL BASED_ON)
BASED_ON
DEFINE TEXTURE
TEXTURE
WALLHOLE
```

Befehle für den 2D-Bereich

```
ADD2
MUL2
ROT2

LINE2
RECT2
POLY2
POLY2_
POLY2_A
POLY2_B
POLY2_B{2}
POLY2_B{3}
POLY2_B{4}
POLY2_B{5}
POLY2_B{6}
ARC2
CIRCLE2
SPLINE2
```

```
SPLINE2A
TEXT2
RICHTEXT2
FRAGMENT2
PROJECT2
PROJECT2{2}
PROJECT2{3}
PROJECT2{4}
DRAWING2
DRAWING3
DRAWING3{2}
DRAWING3{3}
WALLHOLE2
WALLHOLE2{2}
WALLBLOCK2
WALLBLOCK2{2}
WALLLINE2
WALLARC2

HOTSPOT2
HOTLINE2
HOTARC2
PICTURE2
PICTURE2{2}
LINE_PROPERTY
DRAWINDEX
FILL (bei [SET] FILL, bei IND)
LINE_TYPE (bei [SET] LINE_TYPE, bei IND)
DEFINE FILL
DEFINE FILLA
DEFINE SYMBOL_FILL
DEFINE SOLID_FILL
DEFINE EMPTY_FILL
```

```
DEFINE LINEAR_GRADIENT_FILL
DEFINE RADIAL_GRADIENT_FILL
DEFINE TRANSLUCENT_FILL
DEFINE IMAGE_FILL
DEFINE LINE_TYPE
DEFINE SYMBOL_LINE
```

Befehle für den 2D- und 3D-Bereich

```
DEL (bei DEL, bei DEL TOP)
TOP
NTR
ADDITIONAL_DATA (bei LIGHT, bei DEFINE MATERIAL BASED_ON)
LET
RADIUS
RESOL
TOLER
PEN
SET (bei [SET] STYLE, bei [SET] MATERIAL, bei [SET] BUILDING_MATERIAL, bei [SET]
FILL, bei [SET] LINE_TYPE)
STYLE (bei [SET] STYLE, bei IND)
DEFINE STYLE
DEFINE STYLE{2}
PARAGRAPH
ENDPARAGRAPH
TEXTBLOCK
TEXTBLOCK_
PROFILE_ATTR
```

Nicht geometrische Scripts

Eigenschaften-Script

```
DATABASE_SET
```

```
DESCRIPTOR
REF DESCRIPTOR
COMPONENT
REF COMPONENT
BINARYPROP
SURFACE3D
VOLUME3D
POSITION
WALLS
COLUMNS
BEAMS
DOORS
WINDOWS
OBJECTS
CEILS
PITCHED_ROOFS
LIGHTS
HATCHES
ROOMS
MESHES
DRAWING
```

Parameter-Script

```
VALUES
CUSTOM (bei VALUES, bei UI_INFIELD{4})
RANGE
VALUES{2}
PARAMETERS (bei PARAMETERS, bei CALL)
LOCK
ALL (bei LOCK, bei HIDEPARAMETER, bei CALL)
HIDEPARAMETER
```

Interface-Script

UI_DIALOG
UI_PAGE
UI_CURRENT_PAGE
UI_BUTTON (bei UI_BUTTON, bei UI_TOOLTIP)
UI_PREV
UI_NEXT
UI_FUNCTION
UI_LINK
UI_PICT_BUTTON (bei UI_PICT_BUTTON, bei UI_TOOLTIP)
UI_SEPARATOR
UI_GROUPBOX
UI_PICT (bei UI_PICT, bei UI_TOOLTIP)
UI_STYLE
UI_OUTFIELD (bei UI_OUTFIELD, bei UI_TOOLTIP)
UI_INFIELD (bei UI_INFIELD, bei UI_TOOLTIP)
UI_INFIELD{2} (bei UI_INFIELD{2}, bei UI_TOOLTIP)
UI_INFIELD{3} (bei UI_INFIELD{3}, bei UI_TOOLTIP)
UI_INFIELD{4} (bei UI_INFIELD{4}, bei UI_TOOLTIP)
UI_CUSTOM_POPUP_INFIELD (bei UI_CUSTOM_POPUP_INFIELD, bei UI_TOOLTIP)
UI_CUSTOM_POPUP_INFIELD{2} (bei UI_CUSTOM_POPUP_INFIELD{2}, bei UI_TOOLTIP)
UI_RADIOBUTTON (bei UI_RADIOBUTTON, bei UI_TOOLTIP)
UI_RADIOBUTTON{2}
UI_PICT_RADIOBUTTON
UI_PICT_RADIOBUTTON{2}
UI_PICT_PUSHCHECKBUTTON
UI_PICT_PUSHCHECKBUTTON{2}
UI_TEXTSTYLE_INFIELD
UI_TEXTSTYLE_INFIELD{2}
UI_LISTFIELD (bei UI_LISTFIELD, bei UI_TOOLTIP)
UI_LISTITEM (bei UI_LISTITEM, bei UI_TOOLTIP)
UI_LISTITEM{2} (bei UI_LISTITEM{2}, bei UI_TOOLTIP)

```
UI_CUSTOM_POPUP_LISTITEM (bei UI_CUSTOM_POPUP_LISTITEM, bei UI_TOOLTIP)
UI_CUSTOM_POPUP_LISTITEM{2} (bei UI_CUSTOM_POPUP_LISTITEM{2}, bei UI_TOOLTIP)
UI_TOOLTIP
UI_COLORPICKER
UI_COLORPICKER{2}
UI_SLIDER
UI_SLIDER{2}
```

Vorwärts- und Rückwärts-Migration Scripts

```
SETMIGRATIONGUID
STORED_PAR_VALUE
DELETED_PAR_VALUE
NEWPARAMETER
```

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 GDL Text 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 Schüssel und die anderen Elemente werden durch ein Zeichen (im Befehl OPENfestgelegt) 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)
```

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 Rückgabewert 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.

`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

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'). Ein Sonderfall ist das Tabulatorzeichen ('\t').

`MODE:` auf den Befehl muss die Art der Öffnung folgen. Es gibt drei Öffnungsweisen:

- RO (nur lesen)
- WA (read, modify (lesen, modifizieren))
- WO (read, modify) öffnet die Datenbank, falls vorhanden.

`DIALOG:` der Parameter 'filename' 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':

- Erstellen: erstellt eine neue Datei (Dialog speichern als).
- Browse: sucht eine vorhandene Datei (Öffnungsdialog)

`LIBRARY:` wird der Befehl LIBRARY im Parameterstring angegeben, muss sich die Datei in der geladenen Datei befinden. Das Öffnen eine data-Datei zum Lesen aus der geladenen Bibliothek ist aus allen Scripten heraus möglich, aber das Schreiben ist nur möglich im Parameter-, User-Interface- und Eigenschaften-Script.

Setzen Sie immer ein Komma (,) zwischen SEPARATOR, MODE und DIALOG.

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, ...])
```

Fragt die Datenbank aufgrund der Befehlswerte ab.

Wenn der Eintrag gefunden wird, liest das Element die Einträge ab der angegebenen Spalte und listet die gelesenen Werte der Parametersequenz entsprechend auf.

In der Parameterliste muss mindestens ein Wert angegeben sein. Diese Werte können vom numerischen- oder vom Zeichentyp sein, unabhängig von dem gespeicherten Parametertyp. Der Ergebniswert ist die Anzahl der eingelesenen 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.

`channel:` Kanalwert, der verwendet wird, um die Verbindung zu identifizieren.

`recordID:` Befehlswert (numerisch oder String)

`fieldID:` die Spaltenzahl der gegebenen Eintragung (kleinste Zahl, 1 bezieht sich auf das Element nach dem Befehlswert)

`vari:` Variablen für die Annahme der gelesenen Eintragungselemente

Beispiel:

```
! Eingabe von drei Werten ab der ersten Spalte
! (nach dem Befehl) der Eintragung mit dem
! key "key1"
nr = INPUT (ch1, "key1", 1, v1, v2, v3)

PRINT nr, v1, v2, v3
```

Eingeben von Werten in die Datenbank

```
OUTPUT channel, recordID, fieldID, expr1 [, expr2, ...]
```

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 ein Ausdruck (expression) 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.

Das Modifizieren von Data-Dateien aus der geladenen Bibliothek ist nur möglich im Parameter-, User-Interface- und Eigenschaften-Script.

`recordID:` Befehlswert (numerisch oder String)

`fieldID:` Flag: 0 angeben (oder <= 0) um eine Eintragung zu löschen oder 1 angeben 1 (oder > 0) um eine Eintragung zu erstellen oder zu ändern

`expri:` neue Elementwerte der gefundenen oder neuer Eintrag. Beim Löschen werden diese Werte ignoriert.

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 channel
```

`channel:` Kanalwert

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_string, datetimestring)
```

Der zweite Parameter der REQUEST-Funktion stimmt mit der im OPEN-Funktion paramstring Parameter beschrieben, überein.

Kanal öffnen

```
channel = OPEN (filter, filename, paramstring)
```

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.

`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

Die Spezifizierer werden durch Datum- und Zeitwerten wie folgt ersetzt:

%y	Jahr ohne Jahrhundert, als eine Dezimalzahl (00-99)
%Y	Jahr mit Jahrhundert, als eine Dezimalzahl
%b	abgekürzter Monatsname
%B	voller Monatsname
%m	Monat, als Dezimalzahl (01-12)
%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)
%M	Minute, als Dezimalzahl (00-59)
%S	Sekunde, als Dezimalzahl (00-59)
%P	AM/PM Bezeichnung für eine 12-Stunden Uhr

%c	Datum und Zeit in der Form: 01:35:56 PM Mittwoch, March 27, 1996
%x	Datum in der Form von Wednesday, March 27, 1998
%X	Zeit in der Form 01:35:56 PM
%a	abgekürzter Wochentag-Name
%A	voller Wochentag-Name
%w	Wochentag, als Dezimalzahl (0 (Sonntag)-6 (Samstag))
%j	Tag des Jahres, als Dezimalzahl (001-366)
%U	Wochennummer des Jahres (mit Sonntag als erster Tag der ersten Woche),als Dezimalzahl
%W	Wochennummer des Jahres (mit Monat als erster Tag der ersten Woche), als Dezimalzahl(00-53)
%Z	Wenn eine eingestellte Standard-Zeitzone ermittelt werden kann, druckt es es aus.
%%	das % Zeichen

Beispiel:

```
dstr = ""
ch = OPEN ("DateTime", "", "%w/%m/%d/%Y, %H:%M%P")
n = INPUT (ch, "", "", dstr)
CLOSE (ch)
PRINT dstr !it prints 3/03/27/1996, 14:36 PM
```

Lesen der Information

```
n = INPUT (channel, "", "", datetimestr)
```

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 Nullen sein).

Der Rückwert ist die Anzahl der erfolgreich eingelesenen Werte, in diesem Fall ist sie 1.

`channel:` Kanalwert, der verwendet wird, um die Verbindung zu identifizieren.

`datetimestr:` Wert des Typs Zeichenfolge

Schließen des Kanals

```
CLOSE channel
```

Schließt den Kanal, der mit dem Kanalwert identifiziert wird.

GDL File Manager I/O Add-On

Das GDL File Manager In-Out Add-On ermöglicht Ihnen, die Dateien und Unterordner eines Ordners aus einem GDL Script heraus zu lesen.

Definieren Sie den Ordner den Sie prüfen möchten mit dem Befehl OPEN.

Übernehmen Sie über den Befehl INPUTden erste/nächste Datei/Ordner Namen in den definierten Ordner.

Beenden Sie die Ordnerprüfung mit dem Befehl CLOSE.

Spezifikation des Ordners

```
channel = OPEN (filter, filename, paramstring)
```

`channel:` 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:` Add-on spezifischer Parameter, enthält das gewünschte Ausgabeformat von Datum und Zeit Die Parameter in paramString müssen durch Kommas getrennt werden (,).

`1. parameter: FILES/FOLDERS:` Was möchten Sie suchen?

`2. parameter (optional): DIALOG:` Zeigt an, dass der Ordner durch eine Variable ID-string 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.

Beispiel: Öffnet das Rootverzeichnis C (auf einem PC) zum Scannen von Dateien

```
folder = OPEN ("FileMan", "c:\", "FOLDERS")
```

Datei/Ordnernamen 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/Ordnernamen

n: Anzahl der erfolgreich ausgefüllten Variablen

Beispiel: Holt den nächsten Namen vom angegebenen Ordner

```
n = INPUT (folder, 0, 0, fileName)
```

bei Erfolg, n ist es 1. 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

```
topFolder = open ("FileMan", "MyFavouriteFolder", "files, dialog")
y = 0
n = input (topFolder, 0, 0, fileName)
while n = 1 do
    text2 0, y, fileName
    y = y - 0.6
    n = input (topFolder, 0, 0, fileName)
endwhile
close (topFolder)
```

Die folgende Befehlskette (z. B. als 2D Script-Sektion eines Objektes) listet die Dateien des Ordners, der mit "MyFavouriteFolder" spezifiziert wird. Beim ersten Gebrauch gibt der Anwender einen bestehenden Ordner zur Identifizierung an. Später wird die Eigene Dateien Id dieser Ordner sein.

GDL Text I/O Add-On

Über die nachstehenden Schlüsselwörter können Sie externe Dateien zwecks Lesens und Schreibens öffnen, sowie Werte aus /in GDL-Scripts einfügen und einholen.

Dieses Add-On interpretiert die Zeichenfolgen in der Parameterliste der Befehle OPEN, INPUT, OUTPUT des GDL-Scripts.

Die erzeugten Dateien werden in einem Unterordner des "Application data" Ordners platziert, falls diese durch einen relative Pfadangabe definiert wird. 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, filename, paramstring)`

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 Rückgabewert ist eine positive ganze Zahl, die die spezifische Datei identifizert. Dieser Wert wird die zukünftige Referenznummer der Dateien.

`filter:` interner Name des Add-Ons, hier "TEXT"

`filename:` Name der zu öffnenden Datei

`paramstring:` Add-On-spezifische Parameter, enthält Trennzeichen und Parameter zum Öffnen der Datei

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 Lesen). Sonderfälle sind die Zeichen Tabulator ('\t') und neue Zeile ('\n').
`MODE:` auf diesen Befehl muss die Art der Öffnung folgen. Es gibt nur drei Öffnungsweisen:

- RO (nur lesen)
- WA (nur schreiben, ans Dateiende angehängt)
- (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. Das Öffnen eine data-Datei zum Lesen aus der geladenen Bibliothek ist aus allen Scripten heraus möglich, aber das Schreiben ist nur möglich im Parameter-, User-Interface- und Eigenschaften-Script.

Fügen Sie zwischen den Befehlen immer ein Komma (,) ein.

`NEWLINE:` Definition von neuen Zeilensatzzeichen. Mögliche Werte:

- CR (Carriage return, 0x0D)
- LF (Line feed, 0x0A)
- CRLF (Carriage return + Line feed, 0x0D0x0A)

Für Windows-artige Zeilenenden verwenden Sie "NEWLINE = CRLF"

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, NEWLINE = LF"`

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, ...])
```

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 Lese-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, im Falle des Dateiendes 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.

`channel:` channel value, used to identify the connection.

`recordID:` Zeilennummer (numerisch oder String)

`fieldID:` Spaltennummer in der angegebenen Zeile

`var1, ...:` Variablen für die Annahme der gelesenen Eintragungselemente

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, ...]
```

Schreibt ebenso viele Werte in die durch Kanal-Wert identifizierte Datei an der angegebenen Position ein, wie es definierte Ausdrücke gibt. Es muss mindestens ein Ausdruck (expression) 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 Dateiöffnung definierten Trennzeichen. In diesem Fall wird die gegebene Position nicht interpretiert.

Das Modifizieren von Data-Dateien aus der geladenen Bibliothek ist nur möglich im Parameter-, User-Interface- und Eigenschaften-Script.

`channel:` Kanalwert

`recordID:` 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.

`fieldID:` keine Funktion, der Wert wird nicht benutzt

`expr1:` auszugebende Werte

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 channel`

Schließt die Textdatei, die durch den Kanalwert identifiziert wird.

`channel:` Kanalwert

Beispiel:

Ein GDL-Objekt, das 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 "close all"
    endif

"close all":
    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 Eigenschaften Datenbasis können Sie im "ARCHICAD Berechnungshandbuch", in dem Hilfe Menü lesen.

Öffnet die Eigenschaften-Datenbank

```
OPEN ("PROP", "database set name", "[database files]")
```

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.

database set name: ist ein beliebiger Name, der ein Set von Datenbank-Dateien in aufeinander folgenden OPEN-Aufrufen kennzeichnet.

database files: ist eine Liste von Textdateien, die Teil der Eigenschafts-Datenbasis sind. Dieser Parameter ist optional, falls Sie den einzulesenden Dateien zuvor einen `database set name` () zugeordnet haben. Die Reihenfolge der Dateien ist fest: `key file` (Schlüsseldatei), `component file` (Komponentendatei), `descriptor file` (Eigenschaftsdatei), `unit file` (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 ").

Beispiel 1:

```
channel = OPEN ("PROP", "sample",
        "'ArchiCAD_Library_KEY.txt', 'ArchiCAD_Library_COMP.txt',
         'ArchiCAD_Library_DESC.txt', 'ArchiCAD_Library_UNIT.txt'")
```

Mit diesem Befehl wird eine Datenbank geöffnet, die aus den beiden oben angegebenen Dateien besteht (den Dateien der ARCHICAD Eigenschafts-Datenbasis) und mit dem Namen "sample" versehen. Beachten Sie, dass innerhalb des dritten Parameters ein anderes Anführungszeichen verwendet werden muss (Sie können " und ' verwenden).

Beispiel 2:

```
channel = OPEN ("PROP", "sample", "")
```

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.

Schliesst die Eigenschaften-Datenbank

```
CLOSE (channel_number)
```

Rückgabewert: keiner

Schließt den zuvor geöffneten Kommunikationskanal.

Eingabe in die Eigenschaften-Datenbank

```
INPUT (channel_number, "query type", "field list", variable1 [, ...])
```

channel_number: eines gültigen Kommunikationskanals, die von einem vorangegangenen Befehl OPEN angegeben wurde.

query type: gibt die auszuführende Abfrage an. Das Add-On kennt die folgenden Schlüsselwörter:

- Einzelabfragen:
  - KEY, <keycode> – fragt den Datensatz aus der Schlüssel-Datenbasis ab, wobei <keycode> der Wert des Schlüsselcode-Attributs ist. Gültige Felder: KEYCODE, KEYNAME
  - UNIT, <unitcode> – fragt den Datensatz aus der Einheiten-Datenbasis ab, wobei <unitcode> 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 Eigenschaftscode-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
- Datensätze auflisten:
  - KEYLIST – listet alle Datensätze in der Einheiten-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 Eigenschafts-Datenbasis auf oder, wenn der <Keycode> angegeben ist, nur die Datensätze, deren <Keycode> gleich ist. Gültige Felder: KEYCODE, KEYNAME, CODE, NAME, QUANTITY, QUANTITYSTR, UNITCODE, UNITNAME, UNITFORMATSTR
  - DESCLIST[, Keycode] – listet alle Datensätze in der Eigenschafts-Datenbasis auf oder, wenn der <Keycode> angegeben ist, nur die Datensätze, deren <Keycode> gleich ist. Gültige Felder: KEYCODE, KEYNAME, CODE, NAME, NUMOFLINES, FULLNAME
  - COMPDESCLIST[, <keycode>] – listet alle Datensätze in der Komponenten-Datenbasis und der Eigenschafts-Datenbasis auf oder, wenn der <keycode> angegeben ist, nur die Datensätze, deren <keycode> gleich ist. Gültige Felder: ISCOMP, KEYCODE, KEYNAME, CODE, NAME, QUANTITY, QUANTITYSTR, UNITCODE, UNITNAME, UNITFORMATSTR, NUMOFLINES, FULLNAME

    Verwenden Sie diese Abfrage mit großer Sorgfalt! Falls entweder ein Feld in der Datenbank nicht gültig ist (z.B. FULLNAME in der Component Datenbank) wird es einfach weggelassen in der Ergebnisliste (Sie sollten daran denken)

`field list:` 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

`variables:` behalten das Ergebnis der Abfrage auch nach Abschluss. 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.

Beispiel 1:

```
INPUT (channel, "KEY, 001", "KEYNAME", Keyname)
```

Dies ist eine einfache Abfrage: Der Name des Schlüssels mit dem Code ‘001’ wird in die Variable keyname eingetragen.

Beispiel 2:

```
INPUT (channel, "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>

Beispiel 3:

```
INPUT (channel, "COMPLIST", "NAME, KEYNAME, QUANTITY", comp_list)
```

Erstellt eine Komponentenliste, sortiert nach dem Namensfeld, anschließend nach Keyname und zuletzt nach dem Mengenfeld; 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.

Beispiel 4:

```
INPUT (channel, "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 Mengenfeld)

...

x_list[(n*2)-1] = 1 (Anzahl –> es wurden n-1 Eigenschaften aufgelistet; jetzt folgen die Komponenten)

x_list[n*2] = <Keyname_n> (String) ... usw.

Ausgabe in die Eigenschaften-Datenbank

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 DOM-Schnittstelle ist über die Schnittstelle der Erweiterung nicht verfügbar.

	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 Knoten in der Baumstruktur sind ein Name und ein Wert-String zugeordnet, dessen Bedeutung vom Typ des Knotens abhängt:

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

```
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. 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 filename Parameter des Befehls OPEN angegeben wurde. (Falls die ID-Nr bereits einer Datei zugeordnet ist, wird dem Benutzer kein Dialogfenster angezeigt)

'f': Der filename Parameter enthält einen vollständigen Pfad.

'l': Die Datei befindet sich in den geladenen Bibliothekselementen. Das Öffnen eine data-Datei zum Lesen aus der geladenen Bibliothek ist aus allen Scripten heraus möglich, aber das Schreiben ist nur möglich im Parameter-, User-Interface- und Eigenschaften-Script.

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 filename als Pfad relativ zu dem Dokument-Ordner des Benutzers interpretiert.

Das XML-Dokument lesen

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.

```
n = INPUT (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:

- **GetLastError:** Ruft das Ergebnis der letzten Operation ab
  recordID: "GetLastError"
  fieldID: ignoriert
  Rückgabewerte:
  var1: Fehlercode / ok
  var2: Der Erläuterungstext zum Fehler / ok
- **NewPositionDesc:** Anforderung einer neuen Positionsbeschreibung
  recordID: "NewPositionDesc"
  fieldID: ignoriert
  Rückgabewert: var1: die neue Positionsbeschreibung (bezieht sich auf die Wurzel)
- **CopyPositionDesc:** Anforderung einer neuen Positionsbeschreibung, deren Startknoten aus einer anderen Beschreibung geholt wird.
  recordID: "CopyPositionDesc"

fieldID: eine vorhandene Positionsbeschreibung

Rückgabewert: var1: Die neue Positionsbeschreibung (verweist ursprünglich auf die Stelle, auf die die Beschreibung in fieldID verweist)

- **ReturnPositionDesc:** Wenn eine Positionsbeschreibung nicht mehr erforderlich ist.

  recordID: "ReturnPositionDesc"

  fieldID: die Positionsbeschreibung

  var1: ignoriert

  Verwenden Sie diese Anweisung, wenn eine über die Anweisungen NewPositionDesc oder CopyPositionDesc abgerufene Positionsbeschreibung nicht mehr verwendet wird.

- **MoveToNode:** Ändert die Position einer Beschreibung. (und übernimmt die Daten des neuen Knotenpunktes)

  Diese Anweisung kann für die Navigation im Hierarchiebaum 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.

  Um einen exakten Pfad zu spezifizieren, sollte der Bewegungsmodus des Pfads verwendet werden. Nach diesem Bewegungsmodus sollte nur noch der erforderliche Pfad vorhanden sein.

  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.

  Bewegungsmodi:

  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.

  Suchmodi:

  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: Schlüsselworts 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:

```
n = INPUT (ch, "MoveToNode FromPrevSibling K* ELEM+EREF 2", posDesc, name, val, type)
```

- **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

- **NumberofChildNodes:** Gibt die Anzahl der untergeordneten Knoten eines angegebenen Knotens an

  recordID: "NumberofChildNodes nodetype nodename"

  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

fieldID: Positionsbeschreibung

Rückgabewerte:

var1: Anzahl der untergeordneten Knoten

- **NumberofAttributes:** Gibt die Anzahl der Attribute eines Elementknotens zurück.

  recordID: "NumberofAttributes attrname"

  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.

  fieldID: Positionsbeschreibung (muss auf einen Elementknoten verweisen)

  Rückgabewerte:

  var1: Anzahl der Attribute

- **GetAttribute:** Gibt die Daten eines Attributs eines Elementknotens zurück

  recordID: "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

- **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

var1: 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.

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.

- **"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.
- **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

- **NewText:** Fügt einen Textknoten in das Dokument ein

  recordID: "NewText insertpos"

  fieldID: Positionsbeschreibung

  var1: Einzufügender Text

  Siehe auch die Anweisung NewElement.
- **NewComment:** Fügt einen neuen Kommentarknoten in das Dokument

  recordID: "NewComment insertpos"

  fieldID: Positionsbeschreibung

  var1: Text des einzufügenden Kommentars

  Siehe auch die Anweisung NewElement.
- **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.
- **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. Positionsbeschreiber, die auf einen bestimmten Knoten im kopierten Teilbaum zeigen, zeigen nach dem Kopieren auf denselben Knoten.
- **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.

- **Delete:** Löscht einen Knoten und seine Teil-Baumstruktur aus dem Dokument

  recordID: "Delete"

  fieldID: Positionsbeschreibung mit der Angabe des zu löschenden Knotens

  var1: ignoriert

  Alle Positionsbeschreibungen, die auf einen Knoten in der gelöschten Teil-Baumstruktur verweisen, werden ungültig.
- **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
- **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.
- **RemoveAttribute:** Entfernt ein Attribut aus einem Elementknoten

  recordID: "RemoveAttribute"

  fieldID: Positionsbeschreibung, muss auf einen Elementknoten verweisen

  var1: Name des zu entfernenden Attributs
- **Flush:** Schreibt das aktuelle Dokument in die Datei zurück

  recordID: "Flush"

  fieldID: ignoriert

  var1: ignoriert

  Wenn die Datei im Modus "Validate" geöffnet worden war, wird nur ein gültiges Dokument gesichert.
- **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 open-flags in „Das XML-Dokument öffnen“). 'l' bedeutet, dass die Datei aus den geladenen Bibliotheken verwendet wird. 'f' bedeutet, dass fieldID einen vollständigen Pfad enthält.

Tabelle 14. Fehlercodes und Meldungen (mögliche Rückgabewerte der OPEN-Funktion)

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"

POLYGON-OPERATIONSERWEITERUNG

Dieses Addon berechnet Ergebnispolygone basierend auf den eingegebenen Polygonen und den mit ihnen durchgeführten Operationen.

Kompatibilität: eingeführt in ARCHICAD 21. Es gibt auch Operationen für Polylinien.

Eingabepolygone werden identifiziert durch einen Namen, wenn sie an das Addon übergeben werden und in einem vorher definierten Container gespeichert werden. Ergebnispolygone werden automatisch vom Addon mit einem Namen versehen und werden in einem zweiten, Zielcontainer gespeichert. Eingabe- und Ergebnispolygone werden demnach in verschiedenen Containern abgelegt.

Mehrfachpolygone, möglicherweise mit einer hohen Anzahl an Konturen, können in einer einzigen Operation erzeugt werden. Diese werden als individuelle Polygone in den Zielcontainern verwaltet. Als Ergebnis, kann auf diese Polygone in späteren Polygonoperationen zugegriffen werden Das Prinzip ist das gleiche wie bei den Solid Element Befehlen (siehe auch „Solid-Element-Befehle“). Eingabepolygone müssen zusammenhängend sein.

Ein Polygon wird definiert durch verschiedene Polygonzüge (Konturen), von welchen jeder eine ununterbrochene Sequenz von zusammenhängenden Eckpunkten darstellt. Die erste Kontur ist die Außenbegrenzung. Die folgenden Konturen müssen sich alle innerhalb der ersten befinden, sie dürfen nicht überlappen, sie erzeugen dann Ausschnitte in dem ersten Polygon.

Polylinien müssen nicht geschlossen sein, können aber nicht mehrere Konturen haben.

Einen Kanal öffnen

```
ch = INITADDONSCOPE ("PolyOperations ", "", "")
```

Öffnet einen Kanal Der Ergebniswert ist die ID des geöffneten Kanals.

Container-Management

CreateContainer

Erzeugt einen neuen Container.

```
PREPAREFUNCTION ch, "CreateContainer", "myContainer", ""
```

DeleteContainer

Löscht einen existierenden Container.

```
PREPAREFUNCTION ch, "DeleteContainer", "myContainer", ""
```

EmptyContainer

Leert einen existierenden Container.

```
PREPAREFUNCTION ch, "EmptyContainer", "myContainer", ""
```

SetSourceContainer

Container als Quell-Container einstellen.

```
PREPAREFUNCTION ch, "SetSourceContainer", "mySourceContainer", ""
```

SetDestinationContainer

Container als Ziel-Container einstellen.

```
PREPAREFUNCTION ch, "SetDestinationContainer", "myDestinationContainer", ""
```

Polygon / Polylinien-Management

Die Geometrie kann in zwei Datenformaten übertragen werden: Arrays oder Dictionaries. *Kompatibilität: eingeführt in ARCHICAD 23.*

Dieselbe exakte Geometrie kann auf beide Arten gespeichert / abgerufen werden.

Array

Das Array-Format besteht aus drei separaten Arrays (sie können einen beliebigen Namen haben, sie werden durch ihre Position in der Funktionsparameterliste identifiziert):

contourArray: Wird nur für Polygone benötigt, nicht für Polylinien. Ein Array, das den Index des letzten Scheitelpunktes jeder Kontur in **vertArray** enthält . Gegeben mit *"Store"*, zurückgekommen von *"GetContourEnds"*.

vertArray: Array mit allen Scheitelpunkten, die alle Konturen des Polygons / der Polylinie beschreiben. Zwei-dimensionales Array zur Verwendung mit *"Store"* und *"StorePolyline"*, eindimensionales (abgeflachtes zweidimensionales) Array, das zurückgegeben wird von *"GetVertices"* und *"GetPolylineVertices"*.

inhEdgeInfosArray: Optionales Array mit Informationen, die an Kanten angehängt sind und vom Aufrufer definiert wurden. Es muss die gleiche Anzahl von Eckpunkten enthalten wie **vertArray**. Gegeben mit *"Store"* und *"StorePolyline"*, zurückgegeben von *"GetInhEdgeInfos"* und *"GetPolylineInhEdgeInfos"*.

Dictionary

Das Dictionary-Format ist einfacher zu handhaben (es kann einen beliebigen Namen haben, es wird durch seine Position in der Funktionsparameterliste identifiziert):

.isClosed: (Boolean) 1 - Polygon oder geschlossene Polylinie, 0 - offene Polylinie (letzter Punkt als zusätzliche Kante)

.useEdgeInfo: (Boolean, optional) wenn der Wert auf 1 gesetzt ist, erwartet PolyOperations den Schlüssel .edgeInfo, ansonsten wird er ignoriert

.defaultInhEdgeInfo: (Ganzzahl, optional) entspricht dem defaultInhEdgeInfo-Parameter von *"Store"*

.contour: (Dictionary) enthält Daten der Polygonkontur, entsprechend **vertArray**

.contour.edges[n]: (Array) enthält ein eingebettetes Dictionary für jede Kante des Polygons

.contour.edges[n].type: (Ganzzahl) 0 - gerade, 1 - gebogen (Kreisbogen)

.contour.edges[n].begPoint: (Dictionary) ein eingebettetes Dictionary für den Anfangspunkt der Kante

.contour.edges[n].begPoint.x / .y: (Fliesskommazahl) Koordinaten

.contour.edges[n].arcAngle: (Winkel) Zentralwinkel der Kantenkurve, positiv gegen den Uhrzeigersinn, negativ im Uhrzeigersinn (nicht für gerade Kanten eingestellt)

.contour.edges[n].edgeInfo: (Ganzzahl, optional) Informationen an der Kante angeordnet

.holes[m]: (Array, optional) enthält Daten von inneren Öffnungen, ähnlich wie .contour, die nur festgelegt werden, wenn ein oder mehrere Löcher vorhanden sind

Einige globale Variablen verwenden eine Dictionary-Struktur, die mit PolyOperations kompatibel ist (z. B. *OPENING_SYMBOL_GEOMETRY.polygon2D*)

Das gleiche Datenformat kann verwendet werden für *"StoreDictPolygon"* und *"StoreDictPolyline"*.

Polylinien haben niemals .holes [], Polygone sind immer .isClosed = 1

Store

Speichert das Polygon "poly1" mit den gegebenen Parametern im aktuellen Quell-Container.

```
PREPAREFUNCTION ch, "Store", "poly1", nVertices, nContours,
         vertArray, contourArray [, defaultInhEdgeInfo, inhEdgeInfosArray]
```

poly1: Name des abgelegten Polygons

nVertices: Anzahl der Eckpunkte

nContours: Anzahl der Konturen

vertArray: Array mit genau nVertices Elementen, das alle Konturen des Polygons beschreibt. Zweidimensionales Array von (x, y, Winkel) Einträgen, wo x, y, und Winkel reelle Werte sind. Der Winkelparameter ist der Blickwinkel (Ablenkung) bei gekrümmten Kanten. Es ist ein übertragener Wert, der die Ausrichtung widerspiegelt. Nullwert bedeutet gerade Kante.

contourArray: Array, das den Index des letzten Scheitelpunktes der i. Kontur beinhaltet. Es muss genau nContours Elemente beinhalten.

defaultInhEdgeInfo: Eine Information der übernommenen Kante. Bei den durchgeführten Operationen wird diese Information an die brandneuen Kanten (nicht mit Split erstellt) angehängt. Mit diesen Daten können neu erstellte Kanten nach komplexen Operationen verfolgt werden. (Optional)

inhEdgeInfosArray: Ein Array, das Informationen beinhaltet, die den Kanten beigelegt sind. Es muss genau nVertices Ganzzahl-Typ Elemente beinhalten. Wenn eine Kante während einer Operation in mehrere neue Kanten gesplittet wird, wird diese Information ohne Veränderung für alle neu erstellten Kanten übernommen. Beispielsweise können die Seitenwinkel eines Daches gespeichert werden. (Optional)

Anmerkungen:

- Polygone können Ausschnitte und gebogene Kanten besitzen obwohl diese gebogenen Kanten nur Kreisbögen sein können.
- Dieses Polygon kann auf zusätzliche Daten für jede Kante verlinken.
- Der erste Eckpunkt muss bei allen Polygonzügen immer durch den letzten Punkt wiederholt werden. Deshalb hat ein Dreieck in dieser Darstellung 4 Eckpunkte, bei denen der erste und letzte Punkt identisch sind.

- Der erste Polygonzug ist die Hauptkontur und dieser muss die anderen einschließen.

StorePolyline

Speichert die Polylinie "polyline1" mit den gegebenen Parametern im eigentlichen Quellcontainer.

```
PREPAREFUNCTION ch, "StorePolyline", "polyline1", nVertices,
         vertArray, [, defaultInhEdgeInfo, inhEdgeInfosArray]
```

polyline1: Name der gepeicherten Polylinie

nVertices: Anzahl der Eckpunkte

vertArray: Array, das genau die nVertices -Elemente enthält, die die Polylinie beschreiben. Zweidimensionales Array von (x, y, Winkel) Einträgen, wo x, y, und Winkel reelle Werte sind. Der Winkelparameter ist der Blickwinkel (Ausschlag) im Fall von gebogenen Kanten. Es ist ein übertragener Wert, der die Ausrichtung widerspiegelt. Nullwert bedeutet gerade Kante.

defaultInhEdgeInfo: Eine Information der übernommenen Kante. Bei den durchgeführten Operationen wird diese Information an die brandneuen Kanten (nicht mit Split erstellt) angehängt. Mit diesen Daten können neu erstellte Kanten nach komplexen Operationen verfolgt werden. (Optional)

inhEdgeInfosArray: Ein Array, das Informationen beinhaltet, die den Kanten beigelegt sind. Es muss genau nVertices Ganzzahl-Typ Elemente beinhalten. Wenn eine Kante während einer Operation in mehrere neue Kanten gesplittet wird, wird diese Information ohne Veränderung für alle neu erstellten Kanten übernommen. Beispielsweise können die Seitenwinkel eines Daches gespeichert werden. (Optional)

Anmerkungen:

- Die Polylinie kann mit zusätzlichen Daten für jede Kante verknüpft werden.
- Um eine geschlossene Polylinie zu definieren, müssen die letzten Eckpunktkoordinaten mit denen der ersten identisch sein. Offene und geschlossene Polylinien verhalten sich beim Versetzen oder Teilen unterschiedlich.

StoreDictPolygon

Speichert das als Dictionary beschriebene Polygon "poly1" im eigentlichen Quellcontainer. *Kompatibilität: eingeführt in ARCHICAD 23.*

```
PREPAREFUNCTION ch, "StoreDictPolygon", "poly1", PolyOpPolygon
```

poly1: Name des abgelegten Polygons

PolyOpPolygon: Polygon als Dictionary

StoreDictPolyline

Speichert die als Dictionary beschriebene Polylinie "polyline1" im eigentlichen Quellcontainer. *Kompatibilität: eingeführt in ARCHICAD 23.*

```
PREPAREFUNCTION ch, "StoreDictPolyline", "polyline1", PolyOpPolyline
```

`polyline1:` Name der gepeicherten Polylinie

`PolyOpPolyline:` polyline as dictionary

Dispose

Löscht das Polygon / die Polylinie "poly1" aus dem Container "myContainer".

```
PREPAREFUNCTION ch, "Dispose", "poly1", "myContainer"
```

Polygon / Polylinien-Operations-Einstellungen

Diese Befehle funktionieren unabhängig davon, ob Daten im Dictionary- oder Array-Format angegeben wurden.

HalfPlaneParams

Stellen Sie die in der Operation "PolyCut" verwendete Funktion "Beschreibt eine Halbebene in 2D" ein.

```
PREPAREFUNCTION ch, "HalfPlaneParams", "", ca, cb, cc
```

Definiert Unebenheiten für die Halbebene: ca * x + cb * y > cc.

`ca:` Koeffizient von x

`cb:` Koeffizient von y

`cc:` Konstant

OffsetParams

Setzt die Offset-Parameter, verwendet in den Operationen *"OffsetEdge"*, *"ResizeContour"* und *"PolylineOffsetVectors"*..

```
PREPAREFUNCTION ch, "OffsetParams", "", itemIdx, offsetValue
```

`itemIdx:` Index der Kante, welche für die Operation *"OffsetEdge"* übersetzt werden soll. Index der veränderbaren Kontur for *"ResizeContour"* operation. Immer 1 für die Operation *"PolylineOffsetVectors"*.

`offsetValue:` Abstand der Übertragung. Entsprechend den negativen und positiven Versatzwerten bewegen sich die Kanten entweder nach innen oder nach außen. Ist der Versatz groß, dann können die benachbarten Eckpunkte ausgeschnitten werden.

MultipleEdgeOffsetParams

Einstellen der Offset-Parameter, verwendet in den Operationen *"OffsetMultipleEdges"* und *"PolylineOffsetVectors"*.

```
PREPAREFUNCTION ch, "MultipleEdgeOffsetParams", "", nOffset, offsetArray
```

nOffset: Anzahl der zu versetzenden Kanten.

offsetArray: Array mit nOffset-Elementen, die die zu versetzenden Kanten beschreiben. Zweidimensionales Array mit den Datensätzen (edgeIndex, offsetValue), wobei edgeIndex der 1. Index der Kante ist, offsetValue ist der Abstand der Translation. Bei offenen Polylinien ist links positiv, rechts negativ (in Richtung der Kante gesehen)).

PolylineOffsetVectors

Legt die End-Offset-Parameter fest, verwendet in der Operation *"OffsetPolylineWithVectors"*. *"OffsetParams"* oder *"MultipleEdgeOffsetParams"* müssen separat festgelegt werden.

```
PREPAREFUNCTION ch, "PolylineOffsetVectors", "", sx, sy, ex, ey
```

sx, sy: Start Vertex Offset Richtungsvektor. *(Kompatibilität: muss bis zur Version ARCHICAD 21 ein Einheitsvektor sein.)*

ex, ey: Ende Vertex Offset Richtungsvektor. *(Kompatibilität: muss bis zur Version ARCHICAD 21 ein Einheitsvektor sein.)*

Polygon / Polylinien-Operationen

Diese Funktionen funktionieren unabhängig davon, ob Daten im Dictionary- oder Array-Format angegeben wurden.

In den folgenden Polygonoperationen befinden sich die Quell-Polygone "poly1", "poly2" und die Quell-Polylinie "polyline1" im Quellcontainer. Die resultierenden Polygone / Polylinien werden im Zielcontainer mit einem eindeutigen Namen gespeichert, der mit "resPolygonID" oder "resPolylineID" beginnt, wobei "ID" eine Nummer ist. Diese Namen werden normalerweise in einem Array zurückgegeben.

+ - /

Führt die "OP" Operation mit "poly1" und "poly2" Polygone aus und trägt die neuen Werte in die gegebenen Parameter ein. Der Ergebniswert ist die Anzahl der erstellten Polygone.

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "poly1 OP poly2", "", resPolyIDArray)
```

OP: kann folgendes sein:

- **+:** Polygon Hinzufügen
- **-:** Polygon Subtrahieren
- **/:** Polygon Verschnitten

resPolyIDArray: Array des resultierenden Polygon ID Nummern.

ClipPolyline

Beschneidet eine Polylinie mit einem Polygon

```
dim resPolyIDArray[]
nPline = CALLFUNCTION (ch, "ClipPolyline", "polyline1 poly1", resPolyIDArray)
```

polyline1: Name der Polylinie im Quellcontainer.

poly1: Name des beschnittenen Polygons im Quellcontainer.

resPolyIDArray: Array der resultierenden Polylinienbezeichner.

CopyPolygon

Kopieren eines Polygons / einer Polylinie aus dem Quellcontainer in den Zielcontainer

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "CopyPolygon", "poly1", resPolyIDArray)
```

Regularize

Regulieren eines Polygons - Es geometrisch gültig machen.

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "Regularize", "poly1", resPolyIDArray)
```

Ein Polygon ist gültig wenn:

- seine erste Begrenzungskante alle anderen beinhaltet
- Ist korrekt ausgerichtet (die Hauptkontur ist eine positiv ausgerichtete Kurve, der Rest ist negativ ausgerichtet)
- keine Selbstschneidungen besitzt
- Seine Fläche nicht Null ist
- keine Kanten mit Null-Länge besitzt

PolyCut

schneidet das Polygon mit einer Halbebene.

Die Halbebene muss mit dem Befehl *"HalfPlaneParams"* gesetzt werden. Das Ergebnis wird reguliert.

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "PolyCut", "poly1", resPolyIDArray)
```

OffsetEdge

Versetzt eine Kante des Polygons senkrecht zu ihrer Richtung.

Der Kantenindex und -versatz muss über den Befehl "OffsetParams" gesetzt werden. Das Ergebnis wird reguliert.

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "OffsetEdge", "poly1", resPolyIDArray)
```

OffsetMultipleEdges

Versetzen mehrerer Kanten eines Polygons senkrecht zu seiner Richtung.

Die Kantenindizes und Translationsversätze müssen mit dem Befehl *"MultipleEdgeOffsetParams"* gesetzt werden. Das Ergebnis wird reguliert.

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "OffsetMultipleEdges", "poly1", resPolyIDArray)
```

OffsetPolyline

Alle Kanten einer Polylinie werden senkrecht versetzt

Die Translationsversätze müssen mit dem Befehl *"OffsetParams"* durchgeführt werden.

```
dim resPolyIDArray[]
nPline = CALLFUNCTION (ch, "OffsetPolyline", "polyline1", resPolyIDArray)
```

OffsetPolylineWithVectors

Alle Kanten einer offenen Polylinie werden senkrecht versetzt und die Endpunkte werden entlang eines Vektors verschoben

Die Übersetzungsvektoren und der Offset müssen mit den Befehlen *"PolylineOffsetVectors"* und *"OffsetParams"* gesetzt werden.

```
dim resPolyIDArray[]
nPline = CALLFUNCTION (ch, "OffsetPolylineWithVectors", "polyline1", resPolyIDArray)
```

Liefert einen `Input-Argument-Fehler` für geschlossene Polylinien

ResizeContour

Vergrößert oder verkleinert die Kontur des Polygons.

Der Konturindex und -versatz muss über den Befehl "OffsetParams" gesetzt werden. Das Ergebnis wird reguliert.

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "ResizeContour", "poly1", resPolyIDArray)
```

CentreOfGravity

Berechnet den Schwerpunkt eines Polygons. *Kompatibilität: eingeführt in ARCHICAD 22.*

```
n = CALLFUNCTION (ch, "CentreOfGravity", "poly1", x, y)
```

Rückgabewerte:

n: 1 bei Erfolg, 0 wenn der Schwerpunkt nicht genau definiert ist, z.B das Polygon überschneidet sich selber.

x, y: Die Koordinaten des Schwerpunkts.

Ergibt die resultierenden Polygone / Polylinien

Array

GetSourcePolygons, GetSourcePolylines

Abrufen aller Polygon- / Polyliniennamen aus dem aktuellen Quellcontainer.

```
dim resPolyIDArray[], resPolylineIDArray[]
nPgon = CALLFUNCTION (ch, "GetSourcePolygons", "", resPolyIDArray
nPline = CALLFUNCTION (ch, "GetSourcePolylines", "", resPolylineIDArray
```

GetDestinationPolygons, GetDestinationPolylines

Abrufen aller Polygon- / Polyliniennamen aus dem aktuellen Zielcontainer.

```
dim resPolyIDArray[], resPolylineIDArray[]
nPgon = CALLFUNCTION (ch, "GetDestinationPolygons", "", resPolyIDArray)
nPline = CALLFUNCTION (ch, "GetDestinationPolylines", "", resPolylineIDArray
```

GetVertices, GetPolylineVertices

Abrufen der resultierenden Polygon- / Polylinienscheitelpunkte nach jedem Polygon- / Polylinienoperationsaufruf.

Das Polygon / die Polylinie mit dem Namen "polygonID" oder "polylineID" befindet sich im Zielcontainer.

```
dim resVertices[], resPolylineVertices[]
nVertices = CALLFUNCTION (ch, "GetVertices", polygonID, resVertices)
nVertices = CALLFUNCTION (ch, "GetPolylineVertices", polylineID, resPolylineVertices)
```

GetContourEnds

Ergibt die resultierenden Polygonkonturen-Endindizes nach jedem Polygon-Operationsaufruf.

Das Polygon mit dem Namen "polygonID" befindet sich im Zielcontainer.

```
dim contArr[]
nContours = CALLFUNCTION (ch, "GetContourEnds", polygonID, contArr)
```

GetInhEdgeInfos, GetPolylineInhEdgeInfos

Abrufen der resultierenden Polygonkontur- / Polylinieninformationen nach jedem Polygon- / Polylinienoperationsaufruf.

```
dim inhEdgeInfosArr[], polylineInhEdgeInfosArr[]
nEdgeInfos = CALLFUNCTION (ch, "GetInhEdgeInfos", polygonID, inhEdgeInfosArr)
nEdgeInfos = CALLFUNCTION (ch, "GetPolylineInhEdgeInfos",
                               polylineID, polylineInhEdgeInfosArr)
```

Das Polygon / die Polylinie mit dem Namen "polygonID" oder "polylineID" befindet sich im Zielcontainer.

Dictionary

GetSourceDictPolygon, GetSourceDictPolyline

Abrufen des Quellpolygons / der Quellpolylinie als Dictionary. *Kompatibilität: eingeführt in ARCHICAD 23.*

Das Polygon / die Polylinie mit dem Namen "polygonID" oder "polylineID" befindet sich im Quellcontainer.

```
dict PolyOperationsPolygon, PolyOperationsPolyline
CALLFUNCTION (ch, "GetSourceDictPolygon", polygonID, PolyOperationsPolygon
CALLFUNCTION (ch, "GetSourceDictPolyline", polylineID, PolyOperationsPolyline
```

GetDestinationDictPolygon, GetDestinationDictPolyline

Abrufen des resultierenden Polygons / der resultierenden Polylinie als Dictionary nach einem beliebigen Aufruf einer Polygon- / Polylinienoperation. *Kompatibilität: eingeführt in ARCHICAD 23.*

Das Polygon / die Polylinie mit dem Namen "polygonID" oder "polylineID" befindet sich im Zielcontainer.

```
dict PolyOperationsPolygon, PolyOperationsPolyline
CALLFUNCTION (ch, "GetDestinationDictPolygon", polygonID, PolyOperationsPolygon)
CALLFUNCTION (ch, "GetDestinationDictPolyline", polylineID, PolyOperationsPolyline
```

Schließen des Kanals

Schliesst Kanal "ch". Löscht alle gespeicherten Polygone / Polylinien.

```
CLOSEADDONSCOPE (ch)
```

AUTOTEXT-FÜHRER

Es ist nicht Teil von GDL selber. ARCHICAD ersetzt alle Verweise auf Autotext-Felder, egal in welchen GDL-Ausgabefeldern es sie findet. Wenn Sie z.B. <PROJECTSTATUS> in den Parameterstring eines Text2-Befehl schreiben, wird ARCHICAD es brav mit dem tatsächlichen Wert ersetzen. All dies ist für GDL unsichtbar - folglich sind die Größe und andere Eigenschaften des Textes nicht messbar.

Projectinfo Schlüsselworte

```
PROJECTNAME
PROJECTNUMBER
PROJECTSTATUS
DATEOFISSUE
SITEFULLADDRESS

SITEADDRESS1
SITEADDRESS2
SITEADDRESS3
SITECITY
SITESTATE

SITEPOSTCODE
SITECOUNTRY
KEYWORDS
NOTES
ARCHITECTNAME

ARCHITECTPOSITION
CADTECHNICIAN
ARCHITECTCOMPANY
ARCHITECTFULLADDRESS
ARCHITECTADDRESS1

ARCHITECTADDRESS2
ARCHITECTADDRESS3
ARCHITECTCITY
ARCHITECTSTATE
ARCHITECTPOSTCODE
```

```
ARCHITECTCOUNTRY
ARCHITECTEMAIL
ARCHITECTPHONE
ARCHITECTFAX
ARCHITECTWEB

CLIENTNAME
CLIENTCOMPANY
CLIENTFULLADDRESS
CLIENTADDRESS1
CLIENTADDRESS2

CLIENTADDRESS3
CLIENTCITY
CLIENTSTATE
CLIENTPOSTCODE
CLIENTCOUNTRY

CLIENTEMAIL
CLIENTPHONE
CLIENTFAX
```

Allgemeines

```
SHORTDATE
LONGDATE
TIME
FILENAME
FILEPATH
LASTSAVEDAT
LASTSAVEDBY
```

Layout Autotexte

```
LAYOUTNAME
LAYOUTID
```

```
SUBSETNAME
SUBSETID
LAYOUTNUMBER
NUMOFLAYOUTS
```

Zeichnungs-Autotexte

```
DRAWINGNAME
DRAWINGID
DRAWINGSCALE
ORIGINALSCALE
MAGNIFICATION
RENOVATIONFILTER
```

Referenztyp Autotexte

```
LAYOUTNAME_R
LAYOUTID_R
SUBSETNAME_R
SUBSETID_R
DRAWINGNAME_R
DRAWINGID_R
DRAWINGSCALE_R
ORIGINALSCALE_R
MAGNIFICATION_R
FILENAME_R
FILEPATH_R
LAYOUTNUMBER_R
RENOVATIONFILTER_R
```

Markertyp Atotexte

```
MARKERSHEETNUMBER_R
MARKERDRAWINGNUMBER_R
```

```
MARKERSHEETNUMBER90_R
MARKERDRAWINGNUMBER90_R
MARKERSHEETNUMBER110_R
MARKERDRAWINGNUMBER110_R
BACKREFSHEETNUMBER_R
```

Änderungsbedingte Autotexte

```
CHANGEID
CHANGEDESCRIPTION
REVISIONID
ISSUEID
ISSUEDESCRIPTION
ISSUEDATE
ISSUEDBY
```

Layout Revisions-bezogene Autotexte

```
CURRENTREVISIONID
CURRENTISSUEID
CURRENTISSUEDESCRIPTION
CURRENTISSUEDATE
CURRENTISSUEDBY
```

Neue GDL Features in ARCHICAD 23

Dieser Abschnitt enthält einen Abriss der neuen GDL-Funktionen in ARCHICAD 23. Für weitere Details, Tutorials und Beispiele besuchen Sie bitte *GDL Center* [https://gdl.graphisoft.com].

Neuer Datentyp: dictionary

Es ist jetzt möglich, Variablen und Parameter ähnlich wie in vielen anderen Sprachen strukturiert zu organisieren. Dictionary-Datentyp - deklariert mit dem Schlüsselwort `DICT` – ist eine hierarchische Sammlung von Schlüssel-Wert-Paaren, wobei Werte auch andere Dictionaries oder Arrays sein können.

```
dict pointA, object
pointA.x = 1
pointA.y = 2
object.parts[3].edges[4].points[5] = pointA
```

Diese Technik
- ermöglicht besser lesbaren und leichter zu pflegenden Code
- kann verwendet werden, wenn mehr als zweidimensionale Arrays benötigt werden
- reduziert die Anzahl der für den Datenaustausch zwischen ARCIHCAD- und GDL-Objekten erforderlichen Variablen (Globale Variablen und Requests)
- reduziert die Anzahl der Parameter, die für den Datenaustausch zwischen Objekten und Makros benötigt werden

Weitere Informationen finden Sie unter
- allgemeine Syntax: *Strukturierte Typen*, `DICT`
- neue Funktionen: `HASKEY`, `REMOVEKEY`, *PolyOperations* Funktionen
- neuer Rückgabewert: `VARTYPE`
- Globals, die den Datentyp dict verwenden: `OPENING_SYMBOL_GEOMETRY`, `OPENING_HEADERHEIGHT_VALUES`, `OPENING_CENTERHEIGHT_VALUES`, `OPENING_SILLHEIGHT_VALUES`, `BEAM_SEGMENT_INFO`, `COLU_SEGMENT_INFO`
- Requests, die den Datentyp dict verwenden: ”Component_Ids_Of_Parent”, “Component_Property_Values_Of_Parent”, “Property_Values_Of_Parent” requests

Neue Funktionen im PolyOperations-Addon

Bei der Kommunikation mit dem *PolyOperations* Addon können Polygone / Polylinien als Dictionary angegeben werden. Der allgemeine Arbeitsablauf ist derselbe, das Speichern / Lesen kann mit folgenden Funktionen vereinfacht werden:

- *StoreDictPolyline, StoreDictPolygon*
- *GetSourceDictPolyline, GetSourceDictPolygon*
- *GetDestinationDictPolyline, GetDestinationDictPolygon*

Die Dictionary-Struktur ist die gleiche wie `OPENING_SYMBOL_GEOMETRY.polygon2D`. Ausführlichere Informationen finden Sie unter Polygon-Operations-Erweiterung - Polygon- / Polylinienverwaltung – „Dictionary“.

Neues Werkzeug: Öffnung

Das 2D-Symbol des neuen Öffnungswerkzeugs ist ein GDL-Objekt. Mit dem Subtyp `General GDL Object / Documentation Element / Opening Symbol` können neue Symbole erzeugt werden. Eine Öffnung hat zwei voneinander unabhängige Symbole, von denen eines beim Schneiden der Öffnung (in einer Wand) und das andere beim Betrachten der Öffnung (in einer Decke) angezeigt wird).

Weitere Informationen finden Sie unter

- `OPENING_SYMBOL_DISPLAY`
- `OPENING_SYMBOL_GEOMETRY`
- `OPENING_HEIGHT`
- `OPENING_WIDTH`
- `OPENING_HEADERHEIGHT_VALUES`
- `OPENING_CENTERHEIGHT_VALUES`
- `OPENING_SILLHEIGHT_VALUES`
- `GLOB_ELEM_TYPE`

Aktualisierte Werkzeuge: Träger und Stütze

Mit den aktualisierten Träger- und Stützenwerkzeugen können multi-segmentierte Elemente modelliert werden. Neue Etiketten können ausgewählt werden, um auf den Elementen oder Segmenten platziert zu werden. Etiketten, die auf Segmenten platziert werden, erhalten einen neuen `GLOB_ELEM_TYPE`. Die meisten der Globalen Träger- und Stützen-Variablen sind für diese verfügbar, zu Einzelheiten über die Verfügbarkeit siehe Abschnitt der Globalen Variablen für *Träger* und *Stützen*.

Weitere Informationen finden Sie unter

- `GLOB_ELEM_TYPE`
- `BEAM_SEGMENT_INDEX`
- `COLU_SEGMENT_INDEX`
- `BEAM_SEGMENT_INFO`
- `COLU_SEGMENT_INFO`
- veraltet `BEAM_CROSSSECTION_TYPE`
- veraltet `COLU_CROSSSECTION_TYPE`
- veraltet `ac_beam_crosssection_type`
- veraltet `ac_colu_crosssection_type`

Neue Eigenschaften-Requests

Seit ARCHICAD 23 können Baustoffe Eigenschaften und Klassifizierungen haben. Die Eigenschaften der Baustoffe werden als Komponenteneigenschaften bezeichnet, da diese Eigenschaften zu einer Komponente des übergeordneten Elements gehören. Um diese Eigenschaften anzuzeigen, wurden neue `REQUEST`s eingeführt, welche für Etiketten verfügbar sind.

Die Eingabe- und Ausgabewerte der Requests "Component_Ids_Of_Parent" und "Component_Property_Values_Of_Parent" haben das neue Dictionary-Format. "Component_Properties_Of_Parent" hat dieselbe Eingabe- und Ausgabedatenstruktur wie der vorhandene Request "Properties_Of_Parent".

- “Component_Ids_Of_Parent”:

  Die verfügbaren Baustoffkomponenten im übergeordneten Element werden von ARCHICAD identifiziert. Diese IDs sind über den Request „Component_Ids_Of_Parent“ verfügbar.
- “Component_Property_Values_Of_Parent”:

  Um die Komponenteneigenschaftswerte eines übergeordneten Elements abzurufen, kann der neue Request "Component_Property_Values_Of_Parent" verwendet werden. Es gibt die Eigenschaftswerte für eine bestimmte Komponente mit einer beliebigen Anzahl von Eigenschafts-IDs zurück.
- “Component_Properties_Of_Parent”:

  Für die Eigenschaftsauswahl auf der Benutzeroberfläche wurde die Komponente "Component_Properties_Of_Parent" eingeführt. Dieser Request ähnelt "Properties_Of_Parent", außer dass die verfügbaren Komponenteneigenschaften des übergeordneten Elements zurückgegeben werden.
- “Property_Values_Of_Parent”:

  Eine neue Request-Version von "Property_Value_Of_Parent" ist verfügbar: “Property_Values_Of_Parent”. Die Eingabe- und Ausgabewerte dieses Requests werden im neuen Dictionary-Format definiert. Im Gegensatz zu "Property_Value_Of_Parent" können mehrere Eigenschafts-IDs als Dictionary-Schlüssel definiert werden, und es werden mehrere Eigenschaftswerte zurückgegeben.

Diagnosemodus

Es gibt einen neuen Menübefehl, der das Debuggen erleichtert, wenn der GDL-Debugger nicht funktioniert (komplexe Werkzeuge wie Fassade, Treppe, Geländer, Öffnung oder Debuggen des Inhalts von Dictionaries). Im Bibliotheken-Entwickler-Menü (kann über Arbeitsumgebung / Menüs hinzugefügt werden) schaltet der Menüpunkt Bibliotheselement-Diagnose-Modus den `GLOB_DIAGNOSTICS_MODE` global um. Es kann verwendet werden, um bedingte `PRINT` - Befehle zu schreiben und / oder verborgene Parameter auf der Benutzeroberfläche anzeigen, die den Debugging-Prozess steuern. Das Modell wird nach dem Umschalten des Menüs nicht automatisch neu erstellt. Verwenden Sie "Neu aufbauen & regenerieren" (Strg-Alt-Umschalt-R), um die Scripte erneut auszuführen.

Einige ARCHICAD-Bibliothekselemente verwenden diese Funktion. Stellen Sie sicher, dass Sie Pläne mit deaktiviertem Menüelement veröffentlichen.

Befehls-Updates und erweiterte Versionen

- `TUBE` und `TUBE{2}`:

  Die Kantenentfernung (automatisch ausgeblendete Linien, bei denen sich zwei Flächen in derselben Ebene mit derselben Oberfläche treffen) ist auch mit diesen Befehlen möglich. Similarly to `CPRISM_{3}`, masks can be given for the whole body to control base, end, longitudinal and sectional edges.
- neu `COONS{2}`:

  Es ist möglich, die Grenzen von einem `COONS` mit glatter Oberfläche zu verbergen.
- neu `RULEDSEGMENTED{2}`:

  Es ist möglich, jedem Segment unterschiedliche Materialien zuzuweisen und die Texturprojektion zu steuern.
- "ASSOCLP_PARVALUE" Request:

  "ASSOCLP_PARVALUE" und "ASSOCLP_PARVALUE_WITH_DESCRIPTION" Requests können Parameter des Profiltyps verarbeiten.
- Dokumentation:

  Wir haben im Referenzhandbuch einige Erläuterungen vorgenommen. Es wird empfohlen, diese nachzuschlagen, wenn Sie diese verwenden:
  - `AC_LabelPointerLineType` - Typ
  - `AC_LabelPointerPen` - Typ
  - `AC_bLabelFrame` - Typ
  - `ac_holeSideMaterial` – vorher existent, aber undokumentiert
  - `ac_holeMaterialCurved` – vorher existent, aber undokumentiert
  - `ac_beam_cut_linetype` – vorher existent, aber undokumentiert
  - `ac_beam_overhead_linetype` – vorher existent, aber undokumentiert
  - `VARTYPE` – undokumentierter Rückgabewert 3 für Gruppen, neuer Wert 4 für Dictionaries
  - *`"CentreOfGravity"`* – vorher existent, aber undokumentierte PolyOperations-Funktion

IFC4

Das IFC-Add-On verwendet den IFC4-Standard anstelle von IFC 2 ×3. An folgenden Fixnamen-Parametern wurden einige geringfügige Änderungen vorgenommen:

- `ifc_optype` - neue Öffnungsart für Türen
- `ifc_optype`, `ifc_CapacityByWeight` und `ifc_CapacityByNumber` - veraltet für Transport Elements

- `ifc_subtype` - neue Typen für MEP-Elemente

Details finden Sie in der „Parameter des IFC Addons“.

Neues Quellformat mit dem LP_XMLConverter

Das LP_XMLConverter-Tool kann in und aus einem neuen Typ von Quellformat namens HSF konvertieren. Bibliothekselemente werden im selben Ordner wie Bilder gespeichert, die XML-Quelle wird in kleinere XML-Dateien und separate .gdl-Skripte aufgeteilt.

LCF und GSM, erzeugt aus XML, sind in ARCHICAD voll kompatibel mit einem aus HSF erzeugten. Die Konvertierung zwischen HSF und XML kann mit LP_XMLConverter über das GSM-Format erfolgen.

Weitere Informationen zum LP_XMLConverter-Tool finden Sie in den *GDL Center Tips and Tricks* [https://gdl.graphisoft.com/tips-and-tricks/how-to-use-the-lp_xmlconverter-tool]

INDEX

SYNTAX-LISTE ALLER GDL-BEFEHLE

A


```
ABS (x)

ACS (x)

ADD dx, dy, dz

ADD2 x, y

ADDGROUP (g_expr1, g_expr2)

ADDGROUP{2} (g_expr1, g_expr2, edgeColor, materialId, materialColor [, operationStatus])

ADDGROUP{3} (g_expr1, g_expr2, edgeColor, materialId, materialColor [, operationStatus])

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

DEFINE MATERIAL name [,] BASED_ON orig_name [,] PARAMETERS name1 = expr1 [, ...]
        [[,] ADDITIONAL_DATA name1 = expr1 [, ...]]

ADDX dx

ADDY dy

ADDZ dz

LOCK ALL ["name1" [, "name2", ..., "namen"]]

HIDEPARAMETER ALL ["name1" [, "name2", ..., "namen"]]
```

CALL macro_name_string [,]
    PARAMETERS [ALL][name1=value1, ..., namen=valuen][[,]
    RETURNED_PARAMETERS r1, r2, ...]

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)

ATN (x)

B

BASE

DEFINE MATERIAL name [,] BASED_ON orig_name [,] PARAMETERS name1 = expr1 [, ...]
        [[,] ADDITIONAL_DATA name1 = expr1 [, ...]]

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, elementID]

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

[SET] BUILDING_MATERIAL name_or_index
     [, cut_fill_pen [, cut_fill_bkgd_pen, [iOverrideFlag]]]

IND (BUILDING_MATERIAL, name_string)

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_string [,]
    PARAMETERS [ALL][name1=value1, ..., namen=valuen][[,]
    RETURNED_PARAMETERS r1, r2, ...]

 CALL macro_name_string [,]PARAMETERS
    value1 or DEFAULT [, ..., valuen or DEFAULT]
```

```
CALL macro_name_string [, parameter_list]

CALLFUNCTION (channel, function_name, parameter, variable1 [, variable2, ...])

CEIL (x)

CIRCLE r

CIRCLE2 x, y, r

CLOSE channel

CLOSEADDONSCOPE channel

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

COONS{2} 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

COOR wrap, vert1, vert2, vert3, vert4

COOR{2} wrap_method, wrap_flags, vert1, vert2, vert3, vert4

COOR{3} wrapping_method, wrap_flags,
        origin_X, origin_Y, origin_Z,
        endOfX_X, endOfX_Y, endOfX_Z,
        endOfY_X, endOfY_Y, endOfY_Z,
        endOfZ_X, endOfZ_Y, endOfZ_Z

COS (x)
```

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

CPRISM_{2} top_material, bottom_material, side_material,
        n, h,
        x1, y1, alpha1, s1, mat1,
        ...
        xn, yn, alphan, sn, matn

CPRISM_{3} top_material, bottom_material, side_material, mask,
          n, h,
          x1, y1, alpha1, s1, mat1,
          ...
          xn, yn, alphan, sn, matn

CPRISM_{4} top_material, bottom_material, side_material, mask,
          n, h,
          x1, y1, alpha1, s1, mat1,
          ...
          xn, yn, alphan, sn, matn

CREATEGROUPWITHMATERIAL (g_expr, repl_directive, pen, material)

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

CROOF_{2} top_material, bottom_material, side_material,
        n, xb, yb, xe, ye, height, angle, thickness,
        x1, y1, alpha1, s1, mat1,
        ...
        xn, yn, alphan, sn, matn

CROOF_{3} top_material, bottom_material, side_material, mask,
```

```
          n, xb, yb, xe, ye, height, angle, thickness,
          x1, y1, alpha1, s1, mat1,
          ...
          xn, yn, alphan, sn, matn

CROOF_{4} top_material, bottom_material, side_material, mask,
          n, xb, yb, xe, ye, height, angle, thickness,
          x1, y1, alpha1, s1, mat1,
          ...
          xn, yn, alphan, sn, matn

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

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

CUTPLANE{2} angle [, status]
[statement1 ... statementn]
CUTEND

CUTPLANE{3} [x [, y [, z [, side [, status]]]]]
[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]
CUTEND

CUTSHAPE d [, status]
[statement1 statement2 ... statementn]
CUTEND

CUTFORM n, method, status,
        rx, ry, rz, d,
        x1, y1, mask1 [, mat1],
        ...
        xn, yn, maskn [, matn]

CUTFORM{2} n, method, status,
        rx, ry, rz, d,
        x1, y1, mask1 [, mat1],
        ...
        xn, yn, maskn [, matn]

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

CUTPLANE{2} angle [, status]
[statement1 ... statementn]
CUTEND

CUTPLANE{3} [x [, y [, z [, side [, status]]]]]
[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]
CUTEND

CUTSHAPE d [, status]
[statement1 statement2 ... statementn]
CUTEND

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]

 CALL macro_name_string [,]PARAMETERS
    value1 or DEFAULT [, ..., valuen or DEFAULT]


 CALL macro_name_string [,]PARAMETERS
    value1 or DEFAULT [, ..., valuen or DEFAULT]


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 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
```

```
DEFINE IMAGE_FILL name image_name [[,] FILLTYPES_MASK fill_types]
        part1, part2, part3, part4, part5, part6, part7, part8,
        image_vert_size, image_hor_size, image_mask, image_rotangle

DEFINE LINEAR_GRADIENT_FILL name [[,] FILLTYPES_MASK fill_types]]

DEFINE LINE_TYPE name spacing, n,
        length1, ..., lengthn

DEFINE MATERIAL name type,
        surface_red, surface_green, surface_blue
        [, ambient_ce, diffuse_ce, specular_ce, transparent_ce,
         shining, transparency_attenuation
         [, specular_red, specular_green, specular_blue,
          emission_red, emission_green, emission_blue, emission_att]]
        [, fill_index [, fillcolor_index, texture_index]]

DEFINE MATERIAL name [,] BASED_ON orig_name [,] PARAMETERS name1 = expr1 [, ...]
        [[,] ADDITIONAL_DATA name1 = expr1 [, ...]]

DEFINE RADIAL_GRADIENT_FILL name [[,] FILLTYPES_MASK fill_types]]

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_LINE name dash, gap, macro_name PARAMETERS [name1 = value1,
        ...
        namen = valuen]

DEFINE TEXTURE name expression, x, y, mask, angle
```

```
DEFINE TRANSLUCENT_FILL name [[,] FILLTYPES_MASK fill_types]
        pat1, pat2, pat3, pat4, pat5, pat6, pat7, pat8,
        percentage

DEL n [, begin_with]

DEL TOP

DELETED_PAR_VALUE ("oldparname", outputvalue)

DESCRIPTOR name [, code, keycode]

DICT variableName1[, variableName2...]

DIM var1[dim_1], var2[dim_1][dim_2], var3[ ],
        var4[ ][ ], var5[dim_1][ ],
        var5[ ][dim_2]

DO [statement1
    statement2
    ...
    statementn]
WHILE condition

WHILE condition DO
    [statement1
    statement2
    ...
    statementn]
ENDWHILE

DRAWINDEX number

DRAWING

DRAWING2 [expression]

DRAWING3 projection_code, angle, method

DRAWING3{2} projection_code, angle, method [,backgroundColor,
```

```
        fillOrigoX, fillOrigoY, filldirection]
DRAWING3{3} projection_code, angle, method, parts [, backgroundColor,
        fillOrigoX, fillOrigoY, filldirection][[,]
        PARAMETERS name1=value1, ..., namen=valuen]
```

E

```
EDGE vert1, vert2, pgon1, pgon2, status
```

```
ELBOW r1, alpha, r2
```

```
ELLIPS h, r
```

```
IF condition THEN statement [ELSE statement]
```

```
IF condition THEN
    [statement1
    statement2
    ...
    statementn]
[ELSE
    statementn+1
    statementn+2
    ...
    statementn+m]
ENDIF
```

```
END [v1, v2, ..., vn]
```

```
GROUP "name"
    [statement1 ... statementn]
ENDGROUP
```

```
IF condition THEN
    [statement1
    statement2
    ...
```

```
    statementn]
[ELSE
    statementn+1
    statementn+2
    ...
    statementn+m]
ENDIF

PARAGRAPH name alignment, firstline_indent,
        left_indent, right_indent, line_spacing [,
        tab_position1, ...]
    [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

WHILE condition DO
    [statement1
    statement2
    ...
    statementn]
ENDWHILE
```

```
EXIT [v1, v2, ..., vn]

EXP (x)

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

EXTRUDEDSHELL topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4,
        defaultMat,
        n, offset, thickness, flipped, trimmingBody,
        x_tb, y_tb, x_te, y_te, topz, tangle,
        x_bb, y_bb, x_be, y_be, bottomz, bangle,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThicakenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n

EXTRUDEDSHELL{2} topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4,
        defaultMat,
        n, status, offset, thickness, flipped, trimmingBody,
        x_tb, y_tb, x_te, y_te, topz, tangle,
        x_bb, y_bb, x_be, y_be, bottomz, bangle,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThicakenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n

EXTRUDEDSHELL{3} topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4,
        defaultMat,
```

```
        n, status, offset, thickness, flipped, trimmingBody,
        x_tb, y_tb, x_te, y_te, topz, tangle,
        x_bb, y_bb, x_be, y_be, bottomz, bangle,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThicakenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n
```

F

```
FILE_DEPENDENCE "name1" [, "name2", ...]

[SET] FILL name_string

[SET] FILL index

IND (FILL, name_string)

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 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

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 SOLID_FILL name [[,] FILLTYPES_MASK fill_types]

DEFINE EMPTY_FILL name [[,] FILLTYPES_MASK fill_types]

DEFINE LINEAR_GRADIENT_FILL name [[,] FILLTYPES_MASK fill_types]]

DEFINE RADIAL_GRADIENT_FILL name [[,] FILLTYPES_MASK fill_types]]

DEFINE TRANSLUCENT_FILL name [[,] FILLTYPES_MASK fill_types]
        pat1, pat2, pat3, pat4, pat5, pat6, pat7, pat8,
        percentage

DEFINE IMAGE_FILL name image_name [[,] FILLTYPES_MASK fill_types]
        part1, part2, part3, part4, part5, part6, part7, part8,
        image_vert_size, image_hor_size, image_mask, image_rotangle

VALUES "fill_parameter_name" [[,] FILLTYPES_MASK fill_types], value_definition1
   [, value_definition2, ...]

FOR variable_name = initial_value TO end_value [ STEP step_value ]NEXT variable_name

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

FRA (x)

FRAGMENT2 fragment_index, use_current_attributes_flag

FRAGMENT2 ALL, use_current_attributes_flag

G

GET (n)

IF condition THEN label
IF condition GOTO label
IF condition GOSUB label

GOSUB label

IF condition THEN label
IF condition GOTO label
IF condition GOSUB label

GOTO label

GROUP "name"
    [statement1 ... statementn]
ENDGROUP

H

HASKEY (dictionary.key)

HIDEPARAMETER "name1" [, "name2", ..., "namen"]

HIDEPARAMETER ALL ["name1" [, "name2", ..., "namen"]]

HOTARC r, alpha, beta, unID

HOTARC2 x, y, r, startangle, endangle, unID

HOTLINE x1, y1, z1, x2, y2, z2, unID

HOTLINE2 x1, y1, x2, y2, unID

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

```
HOTSPOT2 x, y [, unID [, paramReference [, flags [, displayParam [,
 "customDescription"]]]]]
```

```
HPRISM_ top_mat, bottom_mat, side_mat,
        hill_mat,
        n, thickness, angle, hill_height, status,
        x1, y1, s1,
        ...
        xn, yn, sn
```

I

```
IF condition THEN label
IF condition GOTO label
IF condition GOSUB label
```

```
IF condition THEN label
IF condition GOTO label
IF condition GOSUB label
```

```
IF condition THEN label
IF condition GOTO label
IF condition GOSUB label
```

```
IF condition THEN statement [ELSE statement]
```

```
IF condition THEN
    [statement1
    statement2
    ...
    statementn]
[ELSE
    statementn+1
    statementn+2
```

```
    ...
    statementn+m]
ENDIF
IND (MATERIAL, name_string)
IND (BUILDING_MATERIAL, name_string)
IND (FILL, name_string)
IND (LINE_TYPE, name_string)
IND (STYLE, name_string)
IND (TEXTURE, name_string)
IND (PROFILE_ATTR, name_string, index)
INITADDONSCOPE (extension, parameter_string1, parameter_string2)
INPUT (channel, recordID, fieldID, variable1 [, variable2,...])
INT (x)
ISECTGROUP (g_expr1, g_expr2)
ISECTGROUP{2} (g_expr1, g_expr2, edgeColor, materialId, materialColor [, operationStatus])
ISECTGROUP{3} (g_expr1, g_expr2, edgeColor, materialId, materialColor [, operationStatus])
ISECTLINES (g_expr1, g_expr2)
```

K

```
KILLGROUP g_expr
```

L

```
[LET] varnam = n
LGT (x)
LIBRARYGLOBAL (object_name, parameter, value)
```

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

LINE2 x1, y1, x2, y2

LINE_PROPERTY expr

[SET] LINE_TYPE name_string

[SET] LINE_TYPE index

IND (LINE_TYPE, name_string)

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

LOCK "name1" [, "name2", ..., "namen"]

LOCK ALL ["name1" [, "name2", ..., "namen"]]

LOG (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

MASS{2} 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

[SET] MATERIAL name_or_index

IND (MATERIAL, name_string)

MAX (x1, x2, ..., xn)

MESH a, b, m, n, mask,
z11, z12, ..., z1m,
z21, z22, ..., z2m,
...
zn1, zn2, ..., znm

MIN (x1, x2, ..., xn)

MODEL WIRE

MODEL SURFACE

MODEL SOLID

MUL mx, my, mz

MUL2 x, y

MULX mx

MULY my

MULZ mz

N

NEWPARAMETER "name", "type" [, dim1 [, dim2]]

FOR variable_name = initial_value TO end_value [STEP step_value]NEXT variable_name

NOT (x)

```
NSP

NTR ()

NURBSBODY shadowStatus, smoothnessMin, smoothnessMax

NURBSCURVE2D degree, nControlPoints,
        knot_1, knot_2, ..., knot_m,
        cPoint_1_x, cPoint_1_y, weight_1,
        cPoint_2_x, cPoint_2_y, weight_2,
        ...,
        cPoint_n_x, cPoint_n_y, weight_n

NURBSCURVE3D degree, nControlPoints,
        knot_1, knot_2, ..., knot_m,
        cPoint_1_x, cPoint_1_y, cPoint_1_z, weight_1,
        cPoint_2_x, cPoint_2_y, cPoint_2_z, weight_2,
        ...,
        cPoint_n_x, cPoint_n_y, cPoint_n_z, weight_n

NURBSEDGE vert1, vert2, curve, curveDomainBeg, curveDomainEnd, status, tolerance

NURBSFACE n, surface, tolerance,
        trim1, trim2, ..., trimn

NURBSFACE{2} n, surface, tolerance,
        wrap_method, wrap_flags,
        x1, y1, z1,
        x2, y2, z2,
        x3, y3, z3,
        x4, y4, z4,
         trim1, trim2, ..., trimn

NURBSLUMP n, face1, face2, ..., facen

NURBSSURFACE degree_u, degree_v, nu, nv,
       knoten_u_1, knoten_u_2, ..., knoten_u_mu,
       knoten_v_1, knoten_v_2, ..., knoten_v_mv,
```

```
        cPoint_1_1_x, cPoint_1_1_y, cPoint_1_1_z, weight_1_1,
        cPoint_1_2_x, cPoint_1_2_y, cPoint_1_2_z, weight_1_2,
        ...,
        cPoint_1_nv_x, cPoint_1_nv_y, cPoint_1_nv_z, weight_1_nv,
        cPoint_2_1_x, cPoint_2_1_y, cPoint_2_1_z, weight_2_1,
        ...,
        cPoint_nu_nv_x, cPoint_nu_nv_y, cPoint_nu_nv_z, weight_nu_nv
NURBSTRIM edge, curve, curveDomainBeg, curveDomainEnd, tolerance
NURBSTRIMSINGULAR vertex, curve, curveDomainBeg, curveDomainEnd, tolerance
NURBSVERT x, y, z, hard, tolerance
```

O

```
OPEN (filter, filename, parameter_string)
OUTPUT channel, recordID, fieldID, expression1 [, expression2, ...]
```

P

```
PARAGRAPH name alignment, firstline_indent,
        left_indent, right_indent, line_spacing [,
        tab_position1, ...]
    [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

PARAMETERS name1 = expression1 [,
        name2 = expression2, ...,
        namen = expressionn]

CALL macro_name_string [,]
    PARAMETERS [ALL][name1=value1, ..., namen=valuen][[,]
    RETURNED_PARAMETERS r1, r2, ...]

 CALL macro_name_string [,]PARAMETERS
    value1 or DEFAULT [, ..., valuen or DEFAULT]


 PARVALUE_DESCRIPTION (parname [, ind1 [, ind2]])

PEN n

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

PGON{2} n, vect, status, wrap, edge_or_wrap1, ..., edge_or_wrapn

PGON{3} n, vect, status, wrap_method, wrap_flags, edge_or_wrap1, ..., edge_or_wrapn

PI

PICTURE expression, a, b, mask

PICTURE2 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

POINTCLOUD "data_file_name"

POLY n, x1, y1, ..., xn, yn

POLY2 n, frame_fill, x1, y1, ..., xn, yn

POLY2_ n, frame_fill, 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

POLY2_B{3} n, frame_fill,
        fill_pen, fill_background_pen,
        fillOrigoX, fillOrigoY,
        mxx, mxy, myx, myy, x1, y1, s1, ..., xn, yn, sn

POLY2_B{4} n, frame_fill,
        fill_pen, fill_background_pen,
        fillOrigoX, fillOrigoY,
        mxx, mxy, myx, myy,
        gradientInnerRadius,
        x1, y1, s1, ..., xn, yn, sn

POLY2_B{5} n, frame_fill, fillcategory, distortion_flags,
        fill_pen, fill_background_pen,
        fillOrigoX, fillOrigoY,
```

```
        mxx, mxy, myx, myy,
        gradientInnerRadius,
        x1, y1, s1, ..., xn, yn, sn

POLY2_B{6} n, frame_fill, fillcategory, distortion_flags,
                    fill_pen, fill_background_pen,
                    fillOrigoX, fillOrigoY,
                    mxx, mxy, myx, myy,
                    gradientInnerRadius,
                    x1, y1, s1, pen1, linetype1, ..., xn, yn, sn, penn, linetypen

POLYROOF defaultMat, k, m, n,
        offset, thickness, applyContourInsidePivot,
        z_1, ..., z_k,
        pivotX_1, pivotY_1, pivotMask_1,
        roofAngle_11, gableOverhang_11, topMat_11, bottomMat_11,
        ...
        roofAngle_1k, gableOverhang_1k, topMat_1k, bottomMat_1k,
        ...
        pivotX_m, pivotY_m, pivotMask_m,
        roofAngle_m1, gableOverhang_m1, topMat_m1, bottomMat_m1,
        ...
        roofAngle_mk, gableOverhang_mk, topMat_mk, bottomMat_mk,
        contourX_1, contourY_1, contourMask_1, edgeTrim_1, edgeAngle_1, edgeMat_1,
        ...
        contourX_n, contourY_n, contourMask_n, edgeTrim_n, edgeAngle_n, edgeMat_n

POLYROOF{2} defaultMat, k, m, n,
        offset, thickness, totalThickness, applyContourInsidePivot,
        z_1, ..., z_k,
        pivotX_1, pivotY_1, pivotMask_1,
        roofAngle_11, gableOverhang_11, topMat_11, bottomMat_11,
        ...
        roofAngle_1k, gableOverhang_1k, topMat_1k, bottomMat_1k,
```

```
        ...
        pivotX_m, pivotY_m, pivotMask_m,
        roofAngle_m1, gableOverhang_m1, topMat_m1, bottomMat_m1,
        ...
        roofAngle_mk, gableOverhang_mk, topMat_mk, bottomMat_mk,
        contourX_1, contourY_1, contourMask_1, edgeTrim_1, edgeAngle_1, edgeMat_1,
        ...
        contourX_n, contourY_n, contourMask_n, edgeTrim_n, edgeAngle_n, edgeMat_n
POLYROOF{3} defaultMat, mask, k, m, n,
          offset, thickness, totalThickness, applyContourInsidePivot,
          z_1, ..., z_k,
          pivotX_1, pivotY_1, pivotMask_1,
          roofAngle_11, gableOverhang_11, topMat_11, bottomMat_11,
          ...
          roofAngle_1k, gableOverhang_1k, topMat_1k, bottomMat_1k,
          ...
          pivotX_m, pivotY_m, pivotMask_m,
          roofAngle_m1, gableOverhang_m1, topMat_m1, bottomMat_m1,
          ...
          roofAngle_mk, gableOverhang_mk, topMat_mk, bottomMat_mk,
          contourX_1, contourY_1, contourMask_1, edgeTrim_1, edgeAngle_1, edgeMat_1,
          ...
          contourX_n, contourY_n, contourMask_n, edgeTrim_n, edgeAngle_n, edgeMat_n
POLYROOF{4} defaultMat, mask, k, m, n,
          offset, thickness, totalThickness, applyContourInsidePivot,
          z_1, ..., z_k,
          pivotX_1, pivotY_1, pivotMask_1,
          roofAngle_11, gableOverhang_11, topMat_11, bottomMat_11,
          ...
          roofAngle_1k, gableOverhang_1k, topMat_1k, bottomMat_1k,
          ...
          pivotX_m, pivotY_m, pivotMask_m,
```

```
          roofAngle_m1, gableOverhang_m1, topMat_m1, bottomMat_m1,
          ...
          roofAngle_mk, gableOverhang_mk, topMat_mk, bottomMat_mk,
          contourX_1, contourY_1, contourMask_1, edgeTrim_1, edgeAngle_1, edgeMat_1,
          ...
          contourX_n, contourY_n, contourMask_n, edgeTrim_n, edgeAngle_n, edgeMat_n
POLY_ n, x1, y1, s1, ..., xn, yn, sn

POSITION position_keyword

PREPAREFUNCTION channel, function_name, expression1 [, expression2, ...]

PRINT expression [, expression, ...]

PRISM n, h, x1, y1, ..., xn, yn

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

VALUES "profile_parameter_name" [[,] PROFILETYPES_MASK profile_types], value_definition1
            [, value_definition2, ...]

IND (PROFILE_ATTR, name_string, index)

PROJECT2 projection_code, angle, method

PROJECT2{2} projection_code, angle, method [, backgroundColor,
        fillOrigoX, fillOrigoY, filldirection]

PROJECT2{3} projection_code, angle, method, parts [, backgroundColor,
        fillOrigoX, fillOrigoY, filldirection][[,]
        PARAMETERS name1=value1, ..., namen=valuen]

PROJECT2{4} projection_code, angle,
       useTransparency, statusParts,
       numCutplanes,
       cutplaneHeight1, ..., cutplaneHeightn,

       method1, parts1,
       cutFillIndex1,
```

cutFillFgPen1, cutFillBgPen1,
cutFillOrigoX1, cutFillOrigoY1, cutFillDirection1,
cutLinePen1, cutLineType1,
projectedFillIndex1,
projectedFillFgPen1, projectedFillBgPen1,
projectedFillOrigoX1, projectedFillOrigoY1,
projectedFillDirection1,
projectedLinePen1, projectedLineType1,
...
method(numCutplanes+1)), parts(numCutplanes+1),
cutFillIndex(numCutplanes+1),
cutFillFgPen(numCutplanes+1), cutFillBgPen(numCutplanes+1),
cutFillOrigoX(numCutplanes+1), cutFillOrigoY(numCutplanes+1),
cutFillDirection(numCutplanes+1),
cutLinePen(numCutplanes+1), cutLineType(numCutplanes+1),
projectedFillIndex(numCutplanes+1),
projectedFillFgPen(numCutplanes+1), projectedFillBgPen(numCutplanes+1),
projectedFillOrigoX(numCutplanes+1), projectedFillOrigoY(numCutplanes+1),
projectedFillDirection(numCutplanes+1),
projectedLinePen(numCutplanes+1), projectedLineType(numCutplanes+1)

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

REF COMPONENT code [, keycode [, numeric_expression]]

REF DESCRIPTOR code [, keycode]

```
REMOVEKEY (dictionary.key)

REPEAT [statement1
    statement2
    ...
    statementn]
UNTIL condition

REQ (parameter_string)

REQUEST (question_name, name | index, variable1 [, variable2, ...])

REQUEST (extension_name, parameter_string, variable1, variable2, ...)

RESOL n

RETURN

CALL macro_name_string [,]
    PARAMETERS [ALL][name1=value1, ..., namen=valuen][[,]
    RETURNED_PARAMETERS r1, r2, ...]

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

REVOLVEDSHELL topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4,
        defaultMat,
        n, offset, thickness, flipped, trimmingBody, alphaOffset, alpha,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n

REVOLVEDSHELLANGULAR topMat, bottomMat,
        sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
        n, offset, thickness, flipped, trimmingBody, alphaOffset, alpha,
        segmentationType, nOfSegmente,
```

```
        preThickenTran_11, preThickenTran_12, preThickenTran_13,
        preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23,
        preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33,
        preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n
REVOLVEDSHELLANGULAR{2} topMat, bottomMat,
        sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
        n, status, offset, thickness, flipped, trimmingBody, alphaOffset, alpha,
        segmentationType, nOfSegments,
        preThickenTran_11, preThickenTran_12, preThickenTran_13,
        preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23,
        preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33,
        preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n
REVOLVEDSHELLANGULAR{3} topMat, bottomMat,
        sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
        n, status, offset, thickness, flipped, trimmingBody, alphaOffset, alpha,
        segmentationType, nOfSegments,
        preThickenTran_11, preThickenTran_12, preThickenTran_13,
        preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23,
        preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33,
        preThickenTran_34,
```

```
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n
REVOLVEDSHELL{2} topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4,
        defaultMat,
        n, status, offset, thickness, flipped, trimmingBody, alphaOffset, alpha,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n
REVOLVEDSHELL{3} topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4,
        defaultMat,
        n, status, offset, thickness, flipped, trimmingBody, alphaOffset, alpha,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        x_1, y_1, s_1,
        ...
        x_n, y_n, s_n
REVOLVE{2} n, alphaOffset, alpha, mask, sideMat,
        x1, y1, s1, mat1, ..., xn, yn, sn, matn
REVOLVE{3} n, alphaOffset, alpha, betaOffset, beta, mask, sideMat,
        x1, y1, s1, mat1, ..., xn, yn, sn, matn
REVOLVE{4} n, alphaOffset, alpha, betaOffset, beta, mask, sideMat,
        x1, y1, s1, mat1, ..., xn, yn, sn, matn
REVOLVE{5}n, alphaOffset, alpha, betaOffset, beta, mask, sideMat,
        x1, y1, s1, mat1, ..., xn, yn, sn, matn
RICHTEXT x, y,
```

```
        height, 0, textblock_name
RICHTEXT2 x, y, textblock_name
RND (x)
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
RULEDSEGMENTED n, mask,
        x11, y11, z11, s1,..., x1n, y1n, z1n, sn,
        x21, y21, z21, ..., x2n, y2n, z2n
RULEDSEGMENTED{2} top_material, bottom_material,
        n, mask, textureMode,
        x11, y11, z11, s1, mat1..., x1n, y1n, z1n, sn, matn,
        x21, y21, z21, ..., x2n, y2n, z2n
RULEDSHELL topMat, bottomMat,
        sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
        n, m, g,
        offset, thickness, flipped, trimmingBody,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        firstpolyX_1, firstpolyY_1, firstpolyS_1,
        ...
```

```
        firstpolyX_n, firstpolyY_n, firstpolyS_n,
        secondpolyX_1, secondpolyY_1, secondpolyS_1,
        ...
        secondpolyX_m, secondpolyY_m, secondpolyS_m,
        profile2Tran_11, profile2Tran_12, profile2Tran_13, profile2Tran_14
        profile2Tran_21, profile2Tran_22, profile2Tran_23, profile2Tran_24
        profile2Tran_31, profile2Tran_32, profile2Tran_33, profile2Tran 34
        generatrixFirstIndex_1, generatrixSecondIndex_1,
        ...
        generatrixFirstIndex_g, generatrixSecondIndex_g

RULEDSHELL{2} topMat, bottomMat,
        sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
        n, m, g, status,
        offset, thickness, flipped, trimmingBody,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        firstpolyX_1, firstpolyY_1, firstpolyS_1,
        ...
        firstpolyX_n, firstpolyY_n, firstpolyS_n,
        secondpolyX_1, secondpolyY_1, secondpolyS_1,
        ...
        secondpolyX_m, secondpolyY_m, secondpolyS_m,
        profile2Tran_11, profile2Tran_12, profile2Tran_13, profile2Tran_14
        profile2Tran_21, profile2Tran_22, profile2Tran_23, profile2Tran_24
        profile2Tran_31, profile2Tran_32, profile2Tran_33, profile2Tran 34
        generatrixFirstIndex_1, generatrixSecondIndex_1,
        ...
        generatrixFirstIndex_g, generatrixSecondIndex_g

RULEDSHELL{3} topMat, bottomMat,
        sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
        n, m, g, status,
```

```
        offset, thickness, flipped, trimmingBody,
        preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
        preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
        preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
        firstpolyX_1, firstpolyY_1, firstpolyS_1,
        ...
        firstpolyX_n, firstpolyY_n, firstpolyS_n,
        secondpolyX_1, secondpolyY_1, secondpolyS_1,
        ...
        secondpolyX_m, secondpolyY_m, secondpolyS_m,
        profile2Tran_11, profile2Tran_12, profile2Tran_13, profile2Tran_14
        profile2Tran_21, profile2Tran_22, profile2Tran_23, profile2Tran_24
        profile2Tran_31, profile2Tran_32, profile2Tran_33, profile2Tran 34
        generatrixFirstIndex_1, generatrixSecondIndex_1,
        ...
        generatrixFirstIndex_g, generatrixSecondIndex_g

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

S

```
SECT_ATTRS fill, fill_background_pen,
        fill_pen, contour_pen [, line_type]

SECT_ATTRS{2} contour_pen [, line_type]

SECT_FILL fill, fill_background_pen,
        fill_pen, contour_pen

[SET] STYLE name_string

[SET] STYLE index

[SET] MATERIAL name_or_index
```

```
[SET] BUILDING_MATERIAL name_or_index
     [, cut_fill_pen [, cut_fill_bkgd_pen, [iOverrideFlag]]]

[SET] FILL name_string

[SET] FILL index

[SET] LINE_TYPE name_string

[SET] LINE_TYPE index

SETMIGRATIONGUID guid

SGN (x)

SHADOW casting[, catching]

SIN (x)

SLAB n, h, x1, y1, z1, ..., xn, yn, zn

SLAB_ n, h, x1, y1, z1, s1, ..., xn, yn, zn, sn

MODEL SOLID

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

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
```

```
SPRISM_{2} top_material, bottom_material, side_material,
        n,
        xtb, ytb, xte, yte, topz, tangle,
        xbb, ybb, xbe, ybe, bottomz, bangle,
        x1, y1, s1, mat1,
        ...
        xn, yn, sn, matn
SPRISM_{3} top_material, bottom_material, side_material, mask,
        n,
        xtb, ytb, xte, yte, topz, tangle,
        xbb, ybb, xbe, ybe, bottomz, bangle,
        x1, y1, s1, mat1,
        ...
        xn, yn, sn, matn
SPRISM_{4} top_material, bottom_material, side_material, mask,
        n,
        xtb, ytb, xte, yte, topz, tangle,
        xbb, ybb, xbe, ybe, bottomz, bangle,
        x1, y1, s1, mat1,
        ...
        xn, yn, sn, matn
SQR (x)
FOR variable_name = initial_value TO end_value [ STEP step_value ]NEXT variable_name
STORED_PAR_VALUE ("oldparname", outputvalue)
STR (numeric_expression, length, fractions)
STR (format_string, numeric_expression)
STRLEN (string_expression)
STRSTR (string_expression1, string_expression2[, case_insensitivity])
```

```
STRSUB (string_expression, start_position, characters_number)
STRTOLOWER (string_expression)
STRTOUPPER (string_expression)
STR{2} (format_string, numeric_expression [, extra_accuracy_string])
STW (string_expression)
[SET] STYLE name_string
[SET] STYLE index
IND (STYLE, name_string)
SUBGROUP (g_expr1, g_expr2)
SUBGROUP{2} (g_expr1, g_expr2, edgeColor, materialId, materialColor [, operationStatus])
SUBGROUP{3} (g_expr1, g_expr2, edgeColor, materialId, materialColor [, operationStatus])
MODEL SURFACE
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)
SWEEPGROUP{2} (g_expr, x, y, z)
SWEEPGROUP{3} (g_expr, x, y, z, edgeColor, materialId, materialColor, method)
SWEEPGROUP{4} (g_expr, x, y, z, edgeColor, materialId, materialColor, method, status)
SWEEPGROUP{5} (g_expr, x, y, z, edgeColor, materialId, materialColor, method, status)
```


T


```
TAN (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', ...]

TEXTBLOCK_ name width, anchor, angle, width_factor, charspace_factor, fixed_height, n,
        'expr_1' [, 'expr_2', ..., 'expr_n']

IND (TEXTURE, name_string)

IF condition THEN label
IF condition GOTO label
IF condition GOSUB label

IF condition THEN statement [ELSE statement]

IF condition THEN
    [statement1
    statement2
    ...
    statementn]
[ELSE
    statementn+1
    statementn+2
    ...
    statementn+m]
ENDIF

FOR variable_name = initial_value TO end_value [ STEP step_value ]NEXT variable_name

TOLER d

DEL TOP

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
TUBE{2} top_material, bottom_material, cut_material,
        n, m, mask,
        u1, w1, s1, mat1,
        ...
        un, wn, sn, matn,
        x1, y1, z1, angle1,
        ...
        xm, ym, zm, anglem
```

U

```
UI_BUTTON type, text, x, y [, width, height, id [, url]]
UI_BUTTON type, text, x, y, width, height [, id [, url]] [ UI_TOOLTIP tooltiptext ]
UI_COLORPICKER "redParamName", "greenParamName", "blueParamName", x0, y0 [, width [,
 height]]
UI_COLORPICKER{2} redParamName, greenParamName, blueParamName, x0, y0 [, width [, height]]
UI_CURRENT_PAGE index
UI_CUSTOM_POPUP_INFIELD "name", x, y, width, height,
```

```
        storeHiddenId, treeDepth,
        groupingMethod, selectedValDescription,
        value1, value2, valuesArray1, .... valuen, valuesArrayn

UI_CUSTOM_POPUP_INFIELD "name", x, y, width, height , extra parameters ...
        [ UI_TOOLTIP tooltiptext ]
UI_CUSTOM_POPUP_INFIELD{2} name, x, y, width, height,
        storeHiddenId, treeDepth,
        groupingMethod, selectedValDescription,
        value1, value2, valuesArray1, .... valuen, valuesArrayn

UI_CUSTOM_POPUP_INFIELD{2} name, x, y, width, height , extra parameters ...
        [ UI_TOOLTIP tooltiptext ]
    UI_CUSTOM_POPUP_LISTITEM itemID, fieldID, "name", childFlag, image, paramDesc,
    storeHiddenId, treeDepth,
    groupingMethod, selectedValDescription,
    value1, value2, valuesArray1, .... valuen, valuesArrayn

UI_CUSTOM_POPUP_LISTITEM itemID, fieldID, "name", childFlag , image , paramDesc,
        extra parameters ...
        [ UI_TOOLTIP tooltiptext ]
    UI_CUSTOM_POPUP_LISTITEM{2} itemID, fieldID, name, childFlag, image, paramDesc,
    storeHiddenId, treeDepth,
    groupingMethod, selectedValDescription,
    value1, value2, valuesArray1, .... valuen, valuesArrayn

UI_CUSTOM_POPUP_LISTITEM{2} itemID, fieldID, name, childFlag , image , paramDesc,
        extra parameters ...
        [ UI_TOOLTIP tooltiptext ]
```

```
UI_DIALOG title [, size_x, size_y]

UI_GROUPBOX text, x, y, width, height

UI_INFIELD "name", x, y, width, height [,
        method, picture_name,
        images_number,
        rows_number, cell_x, cell_y,
        image_x, image_y,
        expression_image1, text1,
        ...
        expression_imagen, textn]

UI_INFIELD "name", x, y, width, height [, extra parameters ... ]
        [ UI_TOOLTIP tooltiptext ]

UI_INFIELD{2} name, x, y, width, height [,
        method, picture_name,
        images_number,
        rows_number, cell_x, cell_y,
        image_x, image_y,
        expression_image1, text1,
        ...
        expression_imagen, textn]

UI_INFIELD{2} name, x, y, width, height [, extra parameters ... ]
        [ UI_TOOLTIP tooltiptext ]

UI_INFIELD{3} name, x, y, width, height [,
        method, picture_name,
        images_number,
        rows_number, cell_x, cell_y,
        image_x, image_y,
        expression_image1, text1, value_definition1,
        ...
        [picIdxArray, textArray, valuesArray,
```

```
        ...]
        expression_imagen, textn, value_definitionn]

UI_INFIELD{3} name, x, y, width, height [, extra parameters ... ]
        [ UI_TOOLTIP tooltiptext ]

UI_INFIELD{4} "name", x, y, width, height [,
        method, picture_name,
        images_number,
        rows_number, cell_x, cell_y,
        image_x, image_y,
        expression_image1, text1, value_definition1,
        ...
        [picIdxArray, textArray, valuesArray,
        ...]
        expression_imagen, textn, value_definitionn]

UI_INFIELD{4} "name", x, y, width, height [, extra parameters ... ]
        [ UI_TOOLTIP tooltiptext ]

UI_LISTFIELD fieldID, x, y, width, height [, iconFlag [, description_header [,
 value_header]]]

UI_LISTFIELD fieldID, x, y, width, height [, iconFlag [, description_header [,
 value_header]]]
        [ UI_TOOLTIP tooltiptext ]

UI_LISTITEM itemID, fieldID, "name" [, childFlag [, image [, paramDesc]]]

UI_LISTITEM itemID, fieldID, "name" [, childFlag [, image [, paramDesc]]]
        [ UI_TOOLTIP tooltiptext ]

UI_LISTITEM{2} itemID, fieldID, name [, childFlag [, image [, paramDesc]]]

UI_LISTITEM{2} itemID, fieldID, name [, childFlag [, image [, paramDesc]]]
        [ UI_TOOLTIP tooltiptext ]

UI_OUTFIELD expression, x, y [, width, height [, flags]]
```

```
UI_OUTFIELD expression, x, y, width, height [, flags] [ UI_TOOLTIP tooltiptext ]

UI_PAGE page_number [, parent_id, page_title [, image]]

UI_PICT picture_reference, x, y [, width, height [, mask]]

UI_PICT expression, x, y [,width, height [, mask]] [ UI_TOOLTIP tooltiptext ]

UI_PICT_BUTTON type, text, picture_reference,
        x, y, width, height [, id [, url]]

UI_PICT_BUTTON type, text, picture_reference,
        x, y, width, height [, id [, url]] [ UI_TOOLTIP tooltiptext ]

UI_PICT_PUSHCHECKBUTTON name, text, picture_reference,
         frameFlag, x, y, width, height [UI_TOOLTIP tooltip]

UI_PICT_PUSHCHECKBUTTON{2} "name", text, picture_reference,
         frameFlag, x, y, width, height [UI_TOOLTIP tooltip]

UI_PICT_RADIOBUTTON name, value, text,
         picture_reference, x, y, width, height [UI_TOOLTIP tooltip]

UI_PICT_RADIOBUTTON{2} "name", value, text,
         picture_reference, x, y, width, height [UI_TOOLTIP tooltip]

UI_RADIOBUTTON "name", value, text, x, y, width, height

UI_RADIOBUTTON name, value, text, x, y, width, height [ UI_TOOLTIP tooltiptext ]

UI_RADIOBUTTON{2} name, value, text, x, y, width, height

UI_SEPARATOR x1, y1, x2, y2

UI_SLIDER "name", x0, y0, width, height [, nSegments [, sliderStyle]]

UI_SLIDER{2} name, x0, y0, width, height [, nSegments [, sliderStyle]]

UI_STYLE fontsize, face_code

UI_TEXTSTYLE_INFIELD name, faceCodeMask, x, y,
         buttonWidth, buttonHeight[, buttonOffsetX]
```

```
UI_TEXTSTYLE_INFIELD{2} "name", faceCodeMask, x, y,
         buttonWidth, buttonHeight [, buttonOffsetX]

UI_BUTTON type, text, x, y, width, height [, id [, url]] [ UI_TOOLTIP tooltiptext ]

UI_PICT_BUTTON type, text, picture_reference,
        x, y, width, height [, id [, url]] [ UI_TOOLTIP tooltiptext ]

UI_INFIELD "name", x, y, width, height [, extra parameters ... ]
        [ UI_TOOLTIP tooltiptext ]

UI_INFIELD{2} name, x, y, width, height [, extra parameters ... ]
        [ UI_TOOLTIP tooltiptext ]

UI_INFIELD{3} name, x, y, width, height [, extra parameters ... ]
        [ UI_TOOLTIP tooltiptext ]

UI_INFIELD{4} "name", x, y, width, height [, extra parameters ... ]
        [ UI_TOOLTIP tooltiptext ]

UI_CUSTOM_POPUP_INFIELD "name", x, y, width, height , extra parameters ...
        [ UI_TOOLTIP tooltiptext ]

UI_CUSTOM_POPUP_INFIELD{2} name, x, y, width, height , extra parameters ...
        [ UI_TOOLTIP tooltiptext ]

UI_RADIOBUTTON name, value, text, x, y, width, height [ UI_TOOLTIP tooltiptext ]

UI_OUTFIELD expression, x, y, width, height [, flags] [ UI_TOOLTIP tooltiptext ]

UI_PICT expression, x, y [,width, height [, mask]] [ UI_TOOLTIP tooltiptext ]

UI_LISTFIELD fieldID, x, y, width, height [, iconFlag [, description_header [,
 value_header]]]
        [ UI_TOOLTIP tooltiptext ]

UI_LISTITEM itemID, fieldID, "name" [, childFlag [, image [, paramDesc]]]
        [ UI_TOOLTIP tooltiptext ]

UI_LISTITEM{2} itemID, fieldID, name [, childFlag [, image [, paramDesc]]]
        [ UI_TOOLTIP tooltiptext ]
```

```
UI_CUSTOM_POPUP_LISTITEM itemID, fieldID, "name", childFlag , image , paramDesc,
        extra parameters ...
        [ UI_TOOLTIP tooltiptext ]

UI_CUSTOM_POPUP_LISTITEM{2} itemID, fieldID, name, childFlag , image , paramDesc,
        extra parameters ...
        [ UI_TOOLTIP tooltiptext ]

REPEAT [statement1
    statement2
    ...
    statementn]
UNTIL condition

USE (n)
```

V

```
VALUES "parameter_name" [,]value_definition1 [, value_definition2, ...]

VALUES "fill_parameter_name" [[,] FILLTYPES_MASK fill_types], value_definition1
   [, value_definition2, ...]

VALUES "profile_parameter_name" [[,] PROFILETYPES_MASK profile_types], value_definition1
            [, value_definition2, ...]

VALUES{2} "parameter_name" [,]num_expression1, description1,
   [, num_expression2, description2, ...]

VALUES{2} "parameter_name" [,]num_values_array1, descriptions_array1
   [, num_values_array2, descriptions_array2, ...]

VARDIM1 (expr)

VARDIM2 (expr)

VARTYPE (expression)

VECT x, y, z
```

```
VERT x, y, z
VERT x, y, z, hard
VOLUME3D ()
```

W

```
WALLARC2 x, y, r, alpha, beta
WALLBLOCK2 n, fill_control, fill_pen, fill_background_pen,
        fillOrigoX, fillOrigoY, fillAngle,
        x1, y1, s1,...
        xn, yn, sn
WALLBLOCK2{2} n, frame_fill, fillcategory, distortion_flags,
        fill_pen, fill_background_pen,
        fillOrigoX, fillOrigoY,
        mxx, mxy, myx, myy,
        innerRadius,
        x1, y1, s1,
        ...
        xn, yn, sn
WALLHOLE n, status,
        x1, y1, mask1,
        ...
        xn, yn, maskn
        [, x, y, z]
WALLHOLE2 n, fill_control, fill_pen, fill_background_pen,
        fillOrigoX, fillOrigoY, fillAngle,
        x1, y1, s1,
        ...
        xn, yn, sn
WALLHOLE2{2} n, frame_fill, fillcategory, distortion_flags,
```

```
        fill_pen, fill_background_pen,
        fillOrigoX, fillOrigoY,
        mxx, mxy, myx, myy,
        innerRadius,
        x1, y1, s1,
        ...
        xn, yn, sn

WALLLINE2 x1, y1, x2, y2

WALLNICHE n, method, status,
        rx, ry, rz, d,
        x1, y1, mask1, [mat1,]
        ...
        xn, yn, maskn[, matn]

DO [statement1
    statement2
    ...
    statementn]
WHILE condition

WHILE condition DO
    [statement1
    statement2
    ...
    statementn]
ENDWHILE

MODEL WIRE
```

X

```
XFORM newx_x, newy_x, newz_x, offset_x,
      newx_y, newy_y, newz_y, offset_y,
      newx_z, newy_z, newz_z, offset_z
```

```
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

XWALL_{3} 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
```