GDL リファレンスガイド

GRAPHISOFT®

GRAPHISOFT®

特約店および製品在庫情報に関しては、GRAPHISOFT 社ウェブサイト *(http://www.graphisoft.co.jp)* をご覧ください。

GDL リファレンスガイド

商標

はじめに

このマニュアルはグラフィソフト社が独自に開発した GDL （Geometric Description Language - 幾何記述言語）に関する全容を説明しています。このマニュアルは、グラフィソフト社のソフトウェアに付属している組み立てツールやオブジェクトライブラリの機能を拡張するために用意されています。このマニュアルでは、GDL の詳細（構文定義、コマンド、変数等）を説明しています。

目次

概要 12
スタート 12
スクリプト 12
ライブラリ部品の構造 12
分析、分解、そして簡略化 13
最終調整 14
初級 15
中級 16
上級 19
上級 21
3D 生成 21
3D 空間 21
座標変換 22
GDL インタプリタ 22
GDL スクリプト解析 22
GDL 構文 25
ステートメント 25
行 25
ラベル 25
文字 25
文字列 26
識別子 26
変数 26
パラメータ 27
単純タイプ 27
派生タイプ 27
座標変換 29
2D 変換 29
3D 変換 30
変換スタックの操作 33
3D 形状 35
基本形状 35
BLOCK 35
BRICK 35
CYLIND 36
SPHERE 36
ELLIPS 37
CONE 38
PRISM 39
PRISM_ 40
CPRISM_ 43
CPRISM_{2} 44
BPRISM_ 44
FPRISM_ 47
HPRISM_ 49
SPRISM_ 50
SPRISM_{2} 51
SLAB 54
SLAB_ 54
CSLAB_ 55
CWALL_ 55
BWALL_ 59
XWALL_ 62
XWALL_{2} 65
BEAM 68
CROOF_ 68
CROOF_{2} 72
MESH 72
ARMC 75
ARME 76
ELBOW 78
3D での平面形状 79
HOTSPOT 79
LIN_ 79
RECT 79
POLY 80
POLY_ 80
PLANE 80
PLANE_ 81
CIRCLE 81
ARC 82

ポリラインから生成される形状 - - - 82
EXTRUDE . . . 84
PYRAMID . . . 88
REVOLVE . . . 90
RULED. . . 94
RULED{2} . . . 95
SWEEP. . . 98
TUBE . . . 102
TUBEA. . . 107
COONS. . . 109
MASS_ . . . 113
ビジュアル化のための要素 - - - 116
LIGHT . . . 116
PICTURE . . . 121
3D テキスト要素 - - - 122
TEXT . . . 122
RICHTEXT . . . 123
プリミティブ要素 - - - 123
VERT . . . 124
TEVE . . . 124
VECT . . . 124
EDGE . . . 126
PGON . . . 126
PIPG . . . 127
COOR . . . 127
BODY . . . 130
BASE . . . 132
3D での切り取り - - - 132
CUTPLANE . . . 132
CUTPOLY . . . 135
CUTPOLYA . . . 139
CUTSHAPE . . . 141
CUTFORM. . . 142
ソリッド図形コマンド - - - 143
GROUP. . . 149
ENDGROUP . . . 149
ADDGROUP . . . 149
SUBGROUP. . . 149
ISECTGROUP . . . 149
ISECTLINES . . . 149
PLACEGROUP . . . 150
KILLGROUP. . . 150
SWEEPGROUP. . . 152
バイナリ 3D - - - 152
2D 形状 ___ 155
図面要素 - - - 155
HOTSPOT2 . . . 155
LINE2 . . . 156
RECT2 . . . 156
POLY2 . . . 157
POLY2_ . . . 158
POLY2_A. . . 159
POLY2_B. . . 159
POLY2_B{2} . . . 159
POLY2_B{3} . . . 160
POLY2_B{4} . . . 160
POLY2_B{5} . . . 160
ARC2 . . . 162
CIRCLE2 . . . 162
SPLINE2 . . . 163
SPLINE2A . . . 165
PICTURE2. . . 166
PICTURE2{2} . . . 166
テキスト要素 - - - 167
TEXT2 . . . 167
RICHTEXT2 . . . 167
バイナリ 2D - - - 168
FRAGMENT2 . . . 168
FRAGMENT2 . . . 168
2D での 3D 投影 - - - 168
PROJECT2. . . 168
PROJECT2{2}. . . 169
PROJECT2{3}. . . 172
リスト内の図面 - - - 173
DRAWING2 . . . 173
DRAWING3 . . . 173
DRAWING3{2}. . . 173
DRAWING3{3}. . . 173
グラフィカル編集 ___ 175

ホットスポットベースの編集コマンド ---------- 175
HOTSPOT 175
HOTLINE2 180
HOTARC2 180
HOTLINE 180
HOTARC 180
ステータスコード 181
ステータスコードの構文 ---------- 181
追加ステータスコード ---------- 183
ポリラインの前の部分：現在の位置および接線が定義されます。 . 183
絶対終点による線分 183
相対終点による線分 184
長さと方向による線分 184
長さによる接線線分 185
始点の設定 185
ポリラインを閉じる 186
接線の設定 186
中心点の設定 187
終点までの正接円弧 187
半径と角度による正接円弧 188
中心点と最終半径上の点による円弧 188
中心点と角度による円弧 189
中心点と半径による完全な円 189
属性 195
指示文 ---------- 195
3D および 2D スクリプトの指示文 195
[LET] 195
RADIUS 196
RESOL 197
TOLER 198
PEN 198
LINE_PROPERTY 199
[SET] STYLE 199
SET STYLE 0 199
3D スクリプトでのみ使用される指示文 199
MODEL 199
[SET] MATERIAL 200
SECT_FILL 201
SHADOW 202
2D スクリプトでのみ使用される指示文 203
DRAWINDEX 203
[SET] FILL 203
[SET] LINE_TYPE 204
インライン属性定義 ---------- 204
材質 204
DEFINE MATERIAL 204
DEFINE MATERIAL BASED_ON 207
DEFINE TEXTURE 208
塗りつぶし 210
DEFINE FILL 210
DEFINE FILLA 213
DEFINE SYMBOL_FILL 215
DEFINE SOLID_FILL 216
DEFINE EMPTY_FILL 216
DEFINE LINEAR_GRADIENT_FILL 217
DEFINE RADIAL_GRADIENT_FILL 217
DEFINE TRANSLUCENT_FILL 217
DEFINE IMAGE_FILL 217
線種 218
DEFINE LINE_TYPE 218
DEFINE SYMBOL_LINE 218
スタイル 219
DEFINE STYLE 219
DEFINE STYLE {2} 220
PARAGRAPH 220
テキストブロック 221
追加データ 222
外部ファイルの依存性 223
非ジオメトリックスクリプト 225
特性スクリプト ---------- 225
DATABASE_SET 225
DESCRIPTOR 226
REF DESCRIPTOR 226
COMPONENT 226
REF COMPONENT 227
BINARYPROP 227
SURFACE3D ( ) 227
VOLUME3D ( ) 227

POSITION 227
DRAWING 228
パラメータスクリプト - - - - - 228
VALUES 229
PARAMETERS 230
LOCK 231
HIDEPARAMETER 231
ユーザーインターフェイススクリプト - - - - - 231
UI_DIALOG 231
UI_PAGE 231
UI_CURRENT_PAGE 232
UI_BUTTON 232
UI_SEPARATOR 233
UI_GROUPBOX 233
UI_PICT 234
UI_STYLE 234
UI_OUTFIELD 235
UI_INFIELD 235
UI_INFIELD {2} 235
UI_INFIELD {3} 236
UI_RADIOBUTTON 242
UI_TOOLTIP 243
式と関数 245
数式 - - - - - 245
DIM 245
VARDIM1(expr) 246
VARDIM2(expr) 246
オペレータ - - - - - 248
算術オペレータ 248
関係オペレータ 248
ブールオペレータ 248
関数 - - - - - 249
算術関数 249
ABS 249
CEIL 249
INT 249
FRA 249
ROUND_INT 249
SGN 249
SQR 249
三角関数 250
ACS 250
ASN 250
ATN 250
COS 250
SIN 250
TAN 250
PI 250
指数関数 250
EXP 250
LGT 250
LOG 251
ブール関数 251
NOT 251
統計関数 251
MIN 251
MAX 251
RND 251
ビット関数 251
BITTEST 251
BITSET 251
特殊関数 252
文字列関数 252
STR 252
STR 252
STR{2} 252
SPLIT 255
STW 257
STRLEN 257
STRSTR 257
STRSUB 258
制御ステートメント 259
フロー制御ステートメント - - - - - 259
FOR 259
NEXT 259
DO 260
IF 262
GOTO 263
GOSUB 263

RETURN. 263
END / EXIT 264
パラメータバッファの操作 - - - - - 264
マクロオブジェクト - - - - - 268
出力ステートメント - - - - - 270
ファイル操作 - - - - - 270
OPEN 270
INPUT. 271
VARTYPE. 271
OUTPUT. 271
CLOSE 271
USING DETERMINISTIC ADD-ONS - - - - - 271
INITADDONSCOPE 272
PREPAREFUNCTION 272
CALLFUNCTION 272
CLOSEADDONSCOPE. 272
その他 273
グローバル変数 - - - - - 273
一般環境情報. 273
フロア情報. 274
フライスルー情報. 274
一般要素のパラメータ. 275
オブジェクト、ランプ、ドア、窓のパラメータ. 275
オブジェクト、ランプのパラメータ. 276
オブジェクト、ランプ、ドア、窓のパラメータ、カーテンウォール付属品 - リストとラベル用のみ 276
オブジェクト、ランプ、カーテンウォール付属品 - リストとラベル用のみ. 276
窓、ドア、壁終端のパラメータ. 277
窓、ドアのパラメータ - リストとラベル用のみ 278
ランプパラメータ - リストとラベル用のみ 278
ラベルのパラメータ. 279
壁のパラメータ - 建具のみに使用可能 280
壁パラメータ - リストとラベル用のみ 281
柱パラメータ - リストとラベル用のみ 283
梁パラメータ - リストとラベル用のみ 285
スラブパラメータ - リストとラベル用のみ 286
屋根パラメータ - リストとラベル用のみ 288
塗りつぶしパラメータ - リストとラベル用のみ. 290
メッシュパラメータ - リストとラベル用のみ. 290
カーテンウォールパラメータ - リストとラベル用のみ. 291
カーテンウォールフレームパラメータ - リストとラベル用のみ. . . 292
カーテンウォールパネルのパラメータ - リストとラベル用のみ. . . 293
カーテンウォール接続部パラメータ - リストとラベル用のみ 293
カーテンウォール付属品パラメータ - リストとラベル用のみ 293
自由なユーザーグローバル変数 294
古いグローバル変数 295
要求 - - - - - 297
REQ 297
REQUEST 298
APPLICATION_QUERY. 308
LIBRARYGLOBAL. 309
建具 - - - - - 310
概要 310
建具ライブラリ部品の作成 311
直線壁の矩形の建具 311
3D の調整. 313
2D の調整. 321
平面図から作成される GDL - - - - - 324
キーワード - - - - - 325
共通のキーワード 325
予約キーワード 326
3D 専用 327
2D 専用 329
2D および 3D 用 330
非ジオメトリックスクリプト 330
特性スクリプト 330
特性スクリプト 331
インターフェイススクリプト 331
現在の GDL キーワードのアルファベット順リスト 332
A. 332
B. 332
C. 333
D. 337
E. 339
F. 339

G....340
H....340
I....340
K....341
L....341
M....341
N....342
O....342
P....343
R....345
S....347
T....349
U....349
V....351
W....351
X....353
パラメータ命名規則....354
GDL Data I/O アドオン----354
データベースの説明....354
データベースを開く....354
データベースから値を読み取る....357
データベースに値を書き込む....358
データベースを閉じる....358
GDL DateTime アドオン----359
チャネルを開く....359
情報を読み取る....361
チャネルを閉じる....361
GDL File Manager I/O アドオン----361
フォルダを指定する....361
ファイル / フォルダ名を取得する....362
フォルダのスキャンを終了する....362
GDL Text I/O アドオン----364
ファイルを開く....364
値を読み取る....366
値を書き込む....367
ファイルを閉じる....368
Property GDL アドオン----368
OPEN....369
CLOSE....369
INPUT....370
OUTPUT....372
GDL XML 拡張機能----372
XML ドキュメントを開く....374
XML ドキュメントを読み取る....374
XML ドキュメントを修正する....379
ポリゴン操作拡張機能----382
チャネルを開く....382
ポリゴンコンテナの管理....382
ポリゴンの管理....383
ポリゴン操作の設定....384
ポリゴン操作....384
結果として生成されたポリゴンを取得する....386
チャネルを閉じる....386
索引 387

GDL リファレンスガイド

はじめに

*このマニュアルはグラフィソフト社が独自に開発した GDL （Geometric Description Language - 幾何記述言語）に関する全容を説明しています。*このマニュアルは、GRAPHISOFT 社のソフトウェアに付属している組み立てツールやオブジェクトライブラリの機能を拡張するために用意されています。*このマニュアルでは、GDL の詳細（構文定義、コマンド、変数等）を説明しています。*

概要

GDL は、BASIC に似たパラメトリックな**プログラミング言語**です。GDL は、ドア、窓、家具、構造要素、階段のような 3D ソリッドオブジェクトおよびこれらを平面図で表す 2D シンボルを記述します。これらのオブジェクトを総称して**ライブラリ部品**と呼んでいます。

スタート

デザインに対する必要、プログラミングの経験、および画法幾何学の知識によって、GDL の学習をどこから開始するかが決まります。

段階を踏んで少しずつ自分の技量を上げながら、GDL を学習するようにします。ある程度理解できたら、次の作業に進みます。次に示す経験レベル別の推奨学習方法を参考にしてください。

BASIC などのプログラミング言語をよく知っているユーザーは、既存のスクリプトを見れば GDL の組み立て方に慣れることができます。ArchiCAD に付属するライブラリ部品を開き、2D と 3D の GDL スクリプトをよく考察すると、さらに多くのことを学ぶことができます。また、平面図の要素を GDL 形式で保存して、そのスクリプトを確認することもできます。

BASIC のようなプログラミング言語の知識がないユーザーでも、積み木のようなブロックで遊んだことがあるのなら、GDL を使いながら覚えることができます。まず最も簡単なコマンドを使用し、ライブラリ部品の 3D ウィンドウでその結果を確認しながら学習を進めてゆきます。

ライブラリ部品の環境設定についての詳細は、『ArchCAD ヘルプ』の「バーチャルビルディング」章の「パラメトリックオブジェクト」を参照してください。

GRAPHISOFT 社では、GDL プログラミングおよびオブジェクトライブラリの開発に関する本を何冊か発行しています。『Object Making With ArchiCAD』は、初心者に最適です。David Nicholson Cole 著の『GDL Cookbook』は、入門者および高度な GDL プログラマ用として最も人気のある教材です。『GDL Technical Standards』には、プロのライブラリ開発者向けのグラフィソフト公式規格が含まれています。この本は、グラフィソフトのウェブサイトから無料でダウンロードすることができます。

スクリプト

ライブラリ部品の構造

GDL によって記述された全てのライブラリ部品は、**スクリプト**を持っています。スクリプトは、3D 形状と 2D シンボルを組み立てるための実際の GDL コマンドをリストにしたものです。ライブラリ部品は、ArchiCAD における数量計算のための記述も保持することができます。

マスタースクリプトコマンドは、ライブラリ部品の他のスクリプトよりも先に実行されます。

2D スクリプトは、パラメトリックな 2D 図面の記述に使用します。ライブラリ部品の**バイナリ 2D データ**（2D シンボルウィンドウの内容）は、FRAGMENT2 コマンドを使用して参照することができます。2D スクリプトの記述がない場合は、バイナリ 2D データが平面図上のライブラリ部品の表示に使用されます。

3D スクリプトは、パラメトリックな 3D モデルの記述に使用します。**バイナリ 3D データ**（インポートまたはエクスポートの操作中に生成されたもの）は、BINARY コマンドで参照することができます。

特性スクリプトは、要素リスト、構成要素リスト、ゾーンリストに使用する構成要素と記述項目を含んでいます。ライブラリ部品の**構成要素**と**記述項目**のセクションに記述されている**バイナリ特性データ**を参照するには、BINARYPROP コマンドを使用します。特性スクリプトおよびマスタスクリプトの記述がない場合は、リスト作成中にバイナリ特性データが使用されます。

ユーザーインターフェイススクリプトを使って、入力ページを定義することができます。これは、標準パラメータリストに代わってパラメータ値の編集に使用することがでます。

パラメータスクリプトでは、ライブラリ部品のパラメータとして有効値のセットを定義することができます。

パラメータセクションに設定されているパラメータセットは、平面図にライブラリ部品を配置するときにライブラリ部品の設定のデフォルトとして使用されます。

プレビュー画像は、現在のライブラリ部品の検索時に、ライブラリ部品の設定ダイアログボックスに表示されます。プレビュー画像は、3D スクリプトおよび 2D スクリプトから、PICTURE および PICTURE2 コマンドによって参照することができます。

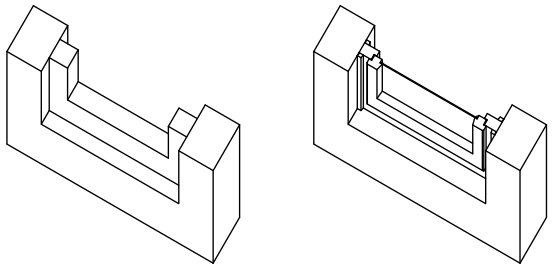
ライブラリ部品に関するテキスト情報は、**コメント**セクションに格納されます。

ArchiCAD は、ビジュアル化機能、構文やエラーのチェック機能を持っており、GDL スクリプトを書くための快適な環境を用意しています。

分析、分解、そして簡略化

どんなに複雑なものでも、たいていのオブジェクトは単純な図形形状の建物ブロックに分解することができます。常に、オブジェクトの単純な分析から始め、このオブジェクトを構成する全ての図形ユニットを定義します。このようにすれば、この建物ブロックを GDL スクリプト言語に変換することができます。分析が適切になされていれば、これらを組み上げたときには理想に近いものになります。

より良い分析を行うためには、空間的な知覚と、少なくとも幾何学の基礎知識を持っている必要があります。

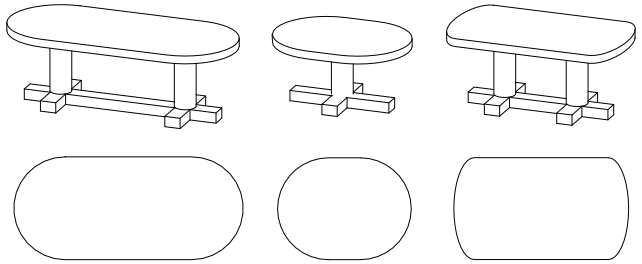


窓のさまざまな表現方法

初期段階で学習意欲を失うことがないよう、あらかじめ寸法の決まったオブジェクトを使用して、できるだけ簡潔なわかりやすい状態で作業を進めてください。モデリングの習熟度に応じて、最終目的に近い高度なオブジェクトにレベルアップしていきます。

複雑であることが理想的であるとは限りません。建築プロジェクトの性質によって、理想的なライブラリ部品は基本的なものであったり洗練されたものであったりします。上図左側の窓は、設計図のビジュアル化という目的に適しています。右側の窓は、細部までリアルに表現されているので、プロジェクトの終わりの方の組み立て図段階で使用することができます。

最終調整



目的に応じて、カスタムパラメトリックオブジェクトの精度さを変えることができます。社内のみで使用するカスタムオブジェクトは、営利を目的としたオブジェクトや一般使用のオブジェクトより現実的でなくてもよいことになります。

シンボルが平面図上ではあまり重要でないか、パラメトリックな（可変的な）変更を 2D で表示する必要がない場合は、パラメトリック 2D スクリプトは省くことができます。

パラメトリックな変更が 2D 表示に関係していても、パラメトリック 2D スクリプトが必ずしも必要になるわけではありません。3D スクリプトウィンドウでパラメトリックな修正を行うまたは修正したオブジェクトの 3D 上面図を新しいシンボルとして使用して、修正したオブジェクトを新規の名称で保存することもできます。デフォルト値をパラメトリックに変更すると、オリジナルから似たようなオブジェクトを複数作成することができます。

最も複雑で高度なライブラリ部品は、パラメトリックな 3D 記述と対応するパラメトリックな 2D スクリプトからなります。設定の変更は、オブジェクトの 3D 画像だけでなく、平面図における 2D 表示にも影響します。

初級

これらのコマンドは簡単に理解して使用することができます。プログラミングの知識がなくても、これらのコマンドだけで十分役立つオブジェクトを作成できます。

単純な形状

形状は、複雑なライブラリ部品を構成する基本的な図形単位です。これらは、GDL の組み立てブロックです。GDL スクリプトにコマンドを書くことで、3D 空間に図形を配置します。

形状コマンドは、形状タイプを定義するキーワード、および形状の寸法を定義するいくつかの数値または英字のパラメータから構成されます。

値の数は、形状ごとに異なります。

最初は、パラメータを省いて、固定値だけで作業することをお勧めします。

次のような形状コマンドが初心者向きです。

3D GDL コマンド：

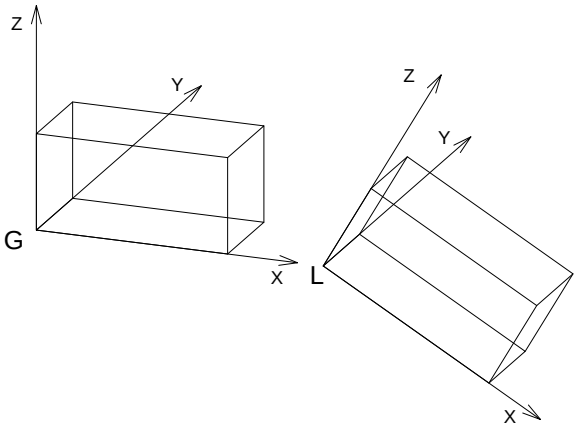
```
BLOCK CYLIND SPHERE PRISM
```

2D GDL コマンド：

```
LINE2 RECT2 POLY2 CIRCLE2 ARC2
```

座標変換

座標変換というのは、組み立てブロックを配置する前に、手を特定の場所に移動することと似ています。座標変換は、次の形状の位置、方向、スケールを準備します。




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

ライブラリ部品の 3D ウィンドウでは、存在するオブジェクトの座標軸の現在の位置（L = ローカル）と原点（G = グローバル）の位置を表示することができます。

最も簡単な座標変換は、次のとおりです。

3D GDL コマンド：

```
ADDX ADDY ADDZ
ROTX ROTY ROTZ
```

2D GDL コマンド：

```
ADD2 ROT2
```

ADD で始まるコマンドは次の形状を移動し、ROT コマンドはその形状のいずれかの軸を中心として回転します。

中級

これらのコマンドはやや複雑になります。プログラミングの知識が必要なためではなく、より複雑な形状やより抽象的な座標変換を記述するからです。

3D GDL コマンド：

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

2D GDL コマンド：

```
HOTSPOT2 POLY2_ TEXT2 FRAGMENT2
```

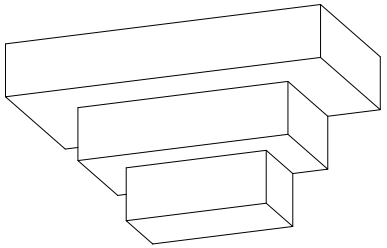
これらのコマンドでは、前述のコマンドよりも多くの値を設定する必要があります。辺や表面の可視性を制御するステータス値を必要とするものもあります。

座標変換

3D GDL コマンド：

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

2D GDL コマンド：



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

MUL で始まる座標変換は、円を楕円にまたは球を楕円形に変えるなど、以降の形状のスケールを変更します。負数を使用すると、ミラーに使用できます。2 行目のコマンド（ADD、MUL、ROT）は、空間の 3 つの寸法の全てに同時に影響します。

上級

これらのコマンドはさらに複雑になります。これは、図形自体の形状が複雑になるか、GDL をプログラミング言語として使用するからです。

3D GDL コマンド：

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

これらのコマンドを使用して、基準となるポリゴンで空間のポリゴンをトレースすることにより、滑らかな曲面を作り出すことができます。パラメータリストで材質への関連付けが必要となる形状もあります。

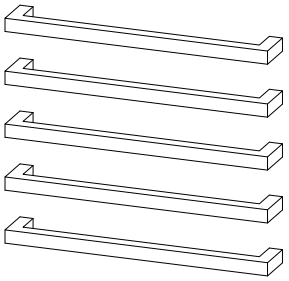
切断面、ポリゴン、形状を使用して、単純な形状から任意の複雑な形状を生成することができます。これに該当するコマンドは、CUTPLANE、CUTPOLY、CUTPOLYA、CUTSHAPE、CUTEND です。

2D GDL コマンド：

```
PICTURE2     POLY2_A
SPLINE2      SPLINE2_A
```

フロー制御と条件ステートメント

FOR NEXT
DO WHILE ENDWHILE
REPEAT UNTIL
IF THEN ELSE ENDIF GOTO GOSUB
RETURN END EXIT
これらは、コンピュータのプログラムの経験者にとっては一般的なコマンドです。しかし、非常に基本的なコマンドですので、プログラムを書いた経験のあるなしに関わらず、十分に理解できることでしょう。
少ないスクリプトで多くの形状を配置したり、計算結果によってその後の処理を決める場合に、スクリプトの部分的な繰り返しを行うことができます。



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

PARAMETERS

この段階まで進めば、固定値を変数名に置き換えることができます。これを実行すると、オブジェクトの柔軟性が向上します。これらの変数は、プロジェクトの作業中にライブラリ部品の設定ダイアログボックスから設定することができます。

マクロコール

標準の GDL 形状以外のものも利用可能です。既存のライブラリ部品は、そのまま GDL 形状として使用することができます。これを配置するには、標準の形状コマンドと同様に、部品の名前をコールして、必要なパラメータを渡します。

上級

前述の機能やコマンドの概略をよく理解できるようになったら、残りのコマンドは、必要に応じて使用することができるでしょう。

注記： コンピュータのメモリ容量に従って、GDL スクリプトのファイルの長さ、マクロコールの深さ、および変換の数に制限が生じます。

上記の GDL コマンドに関する追加情報は、このマニュアルのいろいろなところで説明しています。使用可能なコマンドとそのパラメータ構造の概要については、ご使用のソフトウェアの HTML 形式のヘルプファイルも参照してください。

3D 生成

3D モデリングは、浮動小数点演算をベースにしています。これは、モデルの図形サイズに制限がないことを意味しています。サイズに関わりなく、非常に細かい部分についても精度が保たれます。

最終的に画面上で見る 3D モデルは、**図形プリミティブ**によって構成されたものです。図形プリミティブは、メモリにバイナリフォーマットで保存され、3D エンジンが平面図で作図されたものに従って 3D モデルを生成します。建築平面図の要素からバイナリ 3D データへの変形を 3D 変換と呼んでいます。

プリミティブとは、次のとおりです。

・構造物の構成要素の全ての**頂点**

・頂点をつなげる全ての**辺**

・辺によって囲まれた全ての表面**ポリゴン**

これらプリミティブのグループは**ボディ（本体）**としてまとめて保持されます。ボディによって 3D モデルが作られます。スムージング、シャドウ投射、光沢がある材質や透過材質など、3D ビジュアル化の全ての機能は、このデータ構造をベースにしています。

3D 空間

3D モデルは、主座標系の X、Y、Z の軸によって測定された 3 次元空間で作成されます。主座標系の原点を**グローバル原点**と呼びます。

平面図ビューでは、グローバル原点は、特定の書類を開かずにプログラムを起動したときに表示されるワークシートの左下隅に表示されます。さらにグローバル原点は、平面図書類で参照される全てのフロアの GL レベルも定義します。

オブジェクトを配置するとき、平面図におけるその位置は主座標系の X 軸と Y 軸の方向に沿って定義されます。Z 軸に沿った位置については、[オブジェクトの設定] ダイアログボックスで設定することができます。また、3D 空間に配置された後

で直接調整することもできます。この位置が、オブジェクトの**ローカル座標系**の基点およびデフォルト位置となります。スクリプトで記述される形状は、このローカル座標系を基準として配置されます。

座標変換

全ての GDL 形状は、ローカル座標系の現在位置にリンクされています。例えば、BLOCK は原点にリンクされています。ブロックの長さ、幅、高さは、常に 3 軸の正方向に設定されます。したがって、BLOCK コマンドは軸沿いの 3 つの寸法パラメータを定義するだけで設定されます。

移動して回転したブロックはどのようにしたら生成できるのでしょうか。これは、BLOCK コマンドのパラメータ構造では記述できません。なぜなら、移動して回転するためのパラメータが存在しないからです。

このような場合は、BLOCK コマンドを実行する前に座標系を正しい位置に移動します。座標変換コマンドを使用すれば、位置や軸を中心とした回転を前もって定義できます。この変換はすでに生成されている形状にはまったく影響しません。このコマンドのあとに生成される形状にのみ影響します。

GDL インタプリタ

GDL スクリプトが実行されると、GDL インタプリタエンジンは、ライブラリ部品の位置、サイズ、回転角、ユーザー定義のパラメータ、およびミラー状態を確認します。その後、ローカル座標系を適切な位置に移動させ、ライブラリ部品のスクリプトから GDL コマンドを受け取る準備を整えます。インタプリタは、基本形状のコマンドを受け取るたびに、その形状を表現する図形プリミティブを生成します。

インタプリタが作業を終えると、メモリに完全なバイナリ 3D モデルが格納され、3D 投影、フライスルーレンダリング、時刻日影の計算を行えるようになります。

ArchiCAD には、GDL 用のプリコンパイラとインタプリタが含まれています。GDL スクリプトの解析は、プレコンパイルされたコードに対して行われます。これは解析のスピードアップを図るためです。GDL スクリプトが変更されると、新規のコードが作られます。

ほかのファイル形式（例えば、DXF、Zoom、Alias Wavefront）から変換されたデータ構造は、ライブラリ部品のバイナリ 3D セクションに格納されます。このセクションは、GDL スクリプトの BINARY ステートメントによって参照されます。

GDL スクリプト解析

ユーザーは、平面図上に配置されているライブラリ部品の解析順序を操作することはできません。GDL スクリプトの解析順序は、内部データ構造が基準になります。また、元に戻す操作、再実行操作、および修正操作の影響を受ける場合もあります。この規則の唯一の例外は、名前の先頭に「**MASTER_GDL**」または「**MASTEREND_GDL**」が付く現在のブライブラリの特殊 GDL スクリプトです。

名前の先頭に「MASTER_GDL」が付くスクリプトは、3D 変換を実行する前、断面 / 立面を作成する前、リストプロセスを開始する前、および現在のライブラリのロード後に実行されます。

名前の先頭に「MASTEREND_GDL」が付いているスクリプトは、3D 変換シーケンス後、断面 / 立面の作成後、リストプロセス終了時、および現在のライブラリの変更時（[ライブラリをロード]、[プロジェクトを開く]、[新規プロジェクト]、[終了]）に実行されます。

これらのスクリプトは、ライブラリ部品の編集時には実行されません。ライブラリにこのようなスクリプトが 1 つ以上含まれていると、全て不定の順序で実行されます。

MASTER_GDL スクリプトと MASTEREND_GDL スクリプトには、属性の定義、GDL ユーザーグローバル変数の初期設定、3D コマンド（3D モデルのみで有効）、値リストの定義（*229 ページ「VALUES」を参照*）、および GDL 機能拡張固有コマンドを含めることができます。これらのスクリプトで定義されている属性は、現在の属性セットに結合されます（同じ名前を持つ属性は置き換えられません。GDL から派生し、プログラムで編集されていない属性は必ず置き換えられます）。

GDL 構文

この章では、GDL 構文の基本要素であるステートメント、ラベル、識別子、変数、パラメータについて説明しています。表記規則についても細かく説明しています。

GDL 構文の規則

GDL では、引用符間の文字列以外、大文字小文字は区別されません。GDL スクリプトの論理的な終わりは、END または EXIT ステートメントまたはファイルの物理的な終わりによって示されます。

ステートメント

GDL プログラムはステートメントで構成されます。ステートメントは、キーワード（GDL 形状、座標変換、またはプログラム制御フローで定義される）、マクロ名、または等記号（=）と式が続く変数名で始めることができます。

行

ステートメントは、行区切り記号（改行コード）で複数行に分けることができます。

ステートメントの最後にコンマ（,）を置くと、次の行に続いていることを指示します。コロン（:）は、1 行内でステートメントを分割するのに使用します。感嘆符（!）の後にはコメントを書くことができます。GDL スクリプトには空白行を入れることができます。これは、何の影響も与えません。オペランドとオペレータの間には、複数のスペースまたはタブを使用することができます。ステートメントのキーワードとマクロコールの後には、必ずスペースまたはタブを入力してください。

ラベル

全ての行は、後続するステートメントの参照として使用するラベルで開始することができます。ラベルは整数か定数の文字列で、引用符で挟み、後ろにコロン（:）を付けます。文字列のラベルは大文字小文字を区別します。同じラベルが存在することはできません。プログラムの実行は、GOTO または GOSUB ステートメントによって指定されたラベルにジャンプして続けることができます。

文字

GDL テキストは、英語の小文字、大文字、数字、そして次の文字で構成されます。

<space> _（下線）~ ！：，；．+ ミ * / ^ = < > <= >= # ( ) [ ]{ } ¥ @ & |（縦線）″ ’ ` ´ “ ” ’ ‘ <end_of_line>

文字列

引用符（″、’、`、´）に挟まれた任意の文字列、または引用符のない任意のユニコード文字列。ただし、マクロコール、属性名、ファイル名などの与えられた値を伴う識別子としてスクリプトに現れる文字列は除きます。引用符のない文字列は全て大文字に変換されるので、引用符を使用することをお勧めします。文字列は、255 文字を超えることはできません。
ArchiCAD ユーザーインターフェイスは、GDL エンジンとは異なり、ユニコードに完全には対応していないため、ライブラリ部品エディタに入力できるのは現在のシステムコードページの文字だけです。
文字「¥」は特別な意味を持ちます。その意味は、後続の文字によって異なります。
¥¥　　「¥」文字
¥n 改行
¥t タブ文字
¥*new line* 改行を伴わずに次の行に継続する文字列
¥others 不正な指定のため警告を発生
例：
「これは文字列です」
「洗面台 1’-6″*1’-2」
「他の区切り文字は使用しないでください」

識別子

識別子は、特殊な文字列です。
・- 255 文字を超えないようにします。
・- 英字、「_」、または「~」で開始します。
・- 文字、数字、「_」、または「~」が使えます。
・- 大文字と小文字は区別されません。
識別子は、GDL のキーワード、グローバル変数、ローカル変数、または文字列（名前）としても使えます。キーワードとグローバル変数名は、GDL を使用しているプログラムで定義されます。ほかの識別子は、全て変数名として使えます。

変数

GDL プログラムは、（識別子で定義される）数字と文字列の変数、数字、および文字列を扱うことができます。
変数には、ローカル変数とグローバル変数の 2 種類があります。
キーワード、グローバル変数、属性名、マクロ名、およびファイル名ではない全ての識別子は、ローカル変数とみなされます。初期化されない（定義されない）場合、値は 0（整数）になります。ローカル変数は、マクロコール時にはスタックに置かれます。マクロコールから戻るときには、インタプリタによって値が復元されます。

グローバル変数には予約名があります（使用可能なグローバル変数のリストについては、*273 ページ「その他」*を参照してください）。グローバル変数は、マクロコール時でもスタックに置かれないため、モデリング用の特殊な値の格納と、マクロからの戻りコードのシミュレートが可能です。ユーザーグローバル変数は、どのスクリプトにも設定できますが、有効になるのはそれ以降のスクリプト内だけです。目的のスクリプトが最初に解析されるようにするには、ユーザーグローバル変数を MASTER_GDL ライブラリ部品に設定します。残りのグローバル変数をスクリプトに使用して、プログラムと連携することができます。

「=」コマンドを使用して、数値または文字列値をローカル変数とグローバル変数に代入することができます。

パラメータ

ライブラリ部品のパラメータリストに入っている識別子をパラメータと呼びます。パラメータ識別子は、32 文字を超えてはなりません。スクリプトでは、ローカル変数と同様の規則がパラメータにも適用されます。

テキストだけの GDL ファイルのパラメータは、文字 A から Z で識別されます。

単純タイプ

変数、パラメータ、および式には、数値または文字列の単純な 2 つの種類があります。

*数式*は、演算においては、定数、数値変数またはパラメータ、数値を返す関数、およびこれらの任意の組み合わせです。数式は整数または実数です。整数式は、結果が整数になる演算において、整数定数、変数またはパラメータ、整数を返す関数、およびこれらの任意の組み合わせです。実数式は、結果が実数になる演算において、実数定数、変数またはパラメータ、実数値を返す関数、およびこれら（または整数式）の任意の組み合わせです。整数または実数の数式は、コンパイルプロセスの間に定義され、結合に使用される定数、変数、パラメータおよび演算によって異なります。実数と整数の式は、同じように、数式が必要な場合にいつでも使用できます。ただし、その組み合わせによって精度の問題が生じるような場合には、コンパイラ警告が表示されます（関係オペレータ「=」か「<>」、またはブールオペレータ AND、OR、EXOR を使用した、実数、または実数と整数の比較。実数ラベル式付きの IF または GOTO ステートメント）。

*文字列式*は、結果が文字列になる演算において、定数文字列、文字列変数またはパラメータ、文字列を返す関数、およびこれらの任意の組み合わせです。

派生タイプ

変数とパラメータは、配列にすることもできます。パラメータは、単純タイプの値リストにすることもできます。

*配列*は、インデックスで直接アクセスできる数値または文字列、あるいはこの両方の 1 次元または 2 次元のテーブルです。

*値リスト*は、有効な数値または文字列値のセットです。値リストは、ライブラリ部品の値リストスクリプトまたは MASTER_GDL スクリプト内のパラメータに代入することができ、ポップアップメニュー形式でパラメータリストに表示されます。

[aaa]

角括弧内の要素は、省略可能です（太字の場合は、表記のとおりに入力する必要があります）。

{n}

コマンドのバージョン番号

...

前の要素の繰り返しを示します。

variable（変数）

GDL の変数名を示します。

prompt（プロンプト）

任意の文字列（引用符を含めてはなりません）

BOLD_STRING（ボールド文字列）

UPPERCASE_STRING（大文字文字列）

special characters （特殊文字）

表記とおりに入力をしなければならないことを示します。

other_lowercase_string_in_parameter_list （パラメータリスト内のその他の小文字文字列）

任意の GDL 式を示します。

座標変換

この章では、GDL で使用可能な変換のタイプ（座標系の移動、スケールの変更、回転）と変換の解釈および扱い方について説明しています。

変換とは

GDL においては、全ての図形要素はローカル座標系と強く結びついています。GDL では、右手法則の座標系が使用されます。例えば、ブロックの 1 つの角は原点に位置し、側面はそれぞれ X-Y、X-Z、Y-Z 平面にあります。

希望の位置に図形要素を配置するためには、2 つの手順が必要です。まず座標系をその位置に移動します。次に要素を作成します。座標系のある軸に沿ったまたは軸を中心とした移動、回転、ストレッチを総称して変換と呼びます。

変換はスタックに置かれます。解釈は、常にスタックの最後から逆順に行われます。スクリプトはこの変換スタックを継承します。そのため、新規の要素を追加することはできますが、ローカル座標系で定義した要素以外は削除できません。現在のスクリプトで定義した変換のみ削除できます。スクリプトから戻るとき、ローカル座標系で定義された変換はスタックから除去されます。

2D 変換

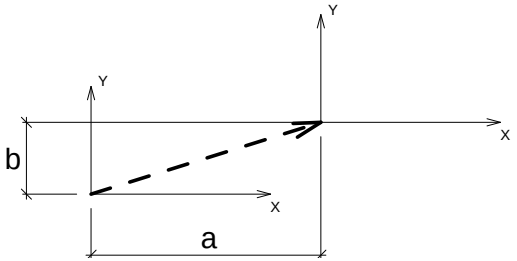
この変換は、ADD、MUL、ROTZ の 3D 変換の 2D 変換版です。

ADD2

ADD2 x, y

例：

ADD2 a, b



MUL2

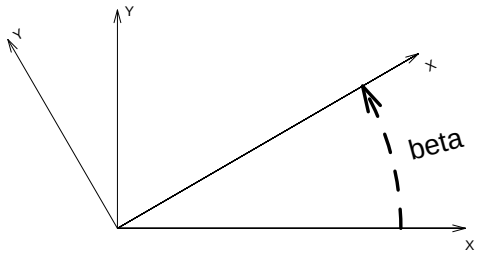
MUL2 x, y

ROT2

ROT2 alpha

例：

ROT2 beta



3D 変換

ADDX

ADDX dx

ADDY

ADDY dy

ADDZ

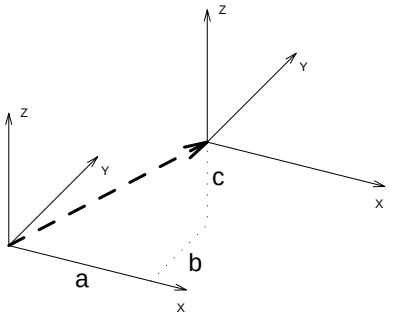
ADDZ dz

それぞれ、dx、dy、dz によって与えられた軸方向に、ローカル座標系を移動します。

ADD

ADD dx, dy, dz

シーケンス ADDX dx: ADDY dy: ADDZ dz を置換します。
スタックの項目は 1 つだけなので、DEL 1 で削除することができます。
例：
ADD a,b,c



MULX

MULX mx

MULY

MULY my

MULZ

MULZ mz

与えられた軸方向にローカル座標系をスケール変更します。mx、my、mz に負数を指定すると、スケール変更と同時にミラーが行われます。

MUL

MUL mx, my, mz

シーケンス MULX mx: MULY my: MULZ mz を置換します。スタックの項目は 1 つだけなので、DEL 1 で削除することができます。

ROTX

ROTX alphax

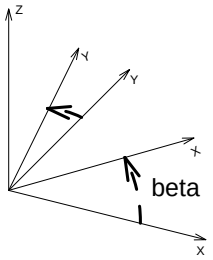
ROTY

ROTY alphay

ROTZ

ROTZ alphaz

与えられた軸を中心として、alphax、alphay、alphaz で指定された角度だけ、反時計回りにローカル座標系を回転します。
例：



ROTZ beta

ROT

ROT x, y, z, alpha

ローカル座標系を、ベクトル（x, y, z）で定義された軸を中心として、alpha で指定された角度だけ反時計回りに回転します。
スタックの項目は 1 つだけなので、DEL 1 で削除することができます。

XFORM

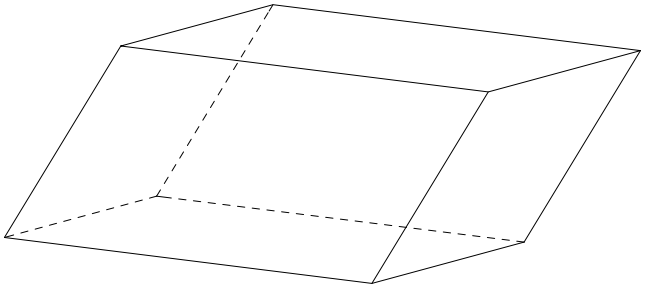
XFORM a11, a12, a13, a14,
a21, a22, a23, a24,
a31, a32, a33, a34

完全な変換マトリクスを定義します。これは、主に自動 GDL コード生成で使用します。スタックの項目は 1 つだけです。

```
x' = a11 * x + a12 * y + a13 * z + a14
y' = a21 * x + a22 * y + a23 * z + a24
z' = a31 * x + a32 * y + a33 * z + a34
```

例：

```
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
```



変換スタックの操作

DEL n

DEL n [, begin_with]

変換スタックから n 個のエントリを削除します。
パラメータ begin-with が指定されていない場合は、変換スタックの前方向に n 個のエントリを削除します。ローカル座標系は、1 つ前の位置に戻ります。
begin_with 変換が指定されていると、begin_with で表されたエントリから順方向に n 個のエントリを削除します。番号は 1 から始まります。パラメータ begin_with が指定されかつ n が負数の場合は、逆方向に削除します。
現在のスクリプトで発行された変換が、引数 n で与えられた数より少ない場合は、発行された変換だけが削除されます。

DEL TOP

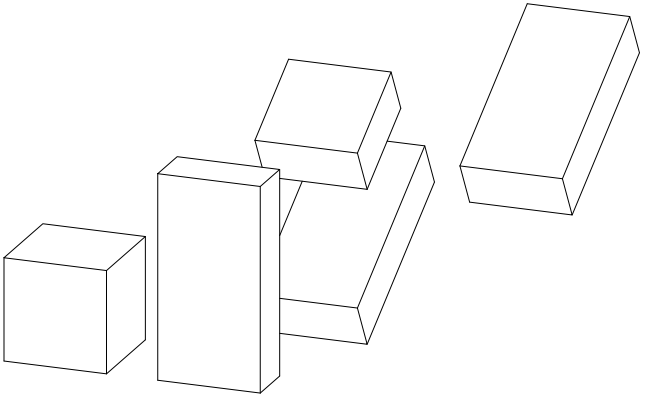
DEl TOP

現在のスクリプト内の現在の変換を全て削除します。

NTR

NTR ()

実際に変換した数を返します。
例：



```
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          !ADDX 2 変換
BLOCK 1, 0.5, 1
DEL 1, NTR() -2 を削除    !ADDZ 1.5 変換
BLOCK 1, 0.5, 2
DEL -2, 3           を削除！ ROTX -60 および ADDY 2.5 変換
BLOCK 1, 0.5, 2 を削除
```

3D 形状

この章では、GDL で使用可能な全ての 3D 形状作成コマンドを、最も基本的なものやポリラインから複雑な形状を生成するものまで解説しています。ビジュアル化のための要素（光源、画像）、そして 3D で表示されるテキストの定義についても解説しています。さらに、節点、ベクトル、辺、ボディからなる内部 3D データ構造のプリミティブ、バイナリデータの解釈、そして切断面の使い方の指針も細かく解説しています。

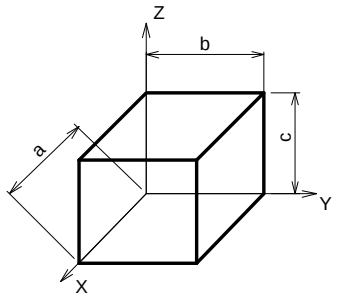
基本形状

BLOCK

BLOCK a, b, c

BRICK

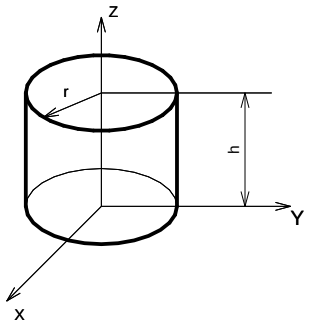
BRICK a, b, c



ブロックの最初の角はローカル座標の原点に位置し、長さ a、b、c の辺が、それぞれ X 軸、Y 軸、Z 軸に沿って延びています。
ゼロの値は、欠落したブロック（矩形または直線）を生成します。
パラメータの制限：
a, b, c >= 0

CYLIND

CYLIND h, r

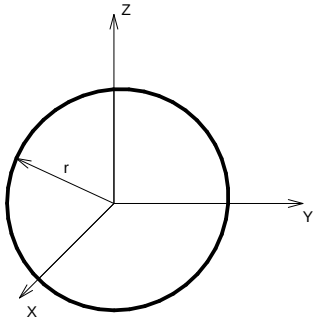


Z 軸と同軸で、高さが h、半径が r の直円柱。
h = 0 の場合、X-Y 平面上に円が生成されます。
r = 0 の場合、Z 軸沿いに直線が生成されます。

SPHERE

SPHERE r

中心が原点で、半径が r の球。



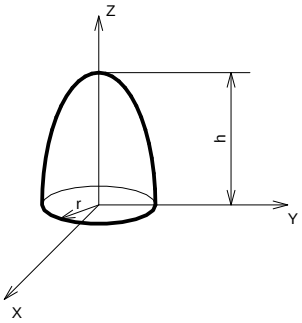
ELLIPS

ELLIPS h, r

半楕円球体。X-Y 平面の横断面は、原点を中心とする半径が r の円です。z 軸に沿った半軸の長さは h です。

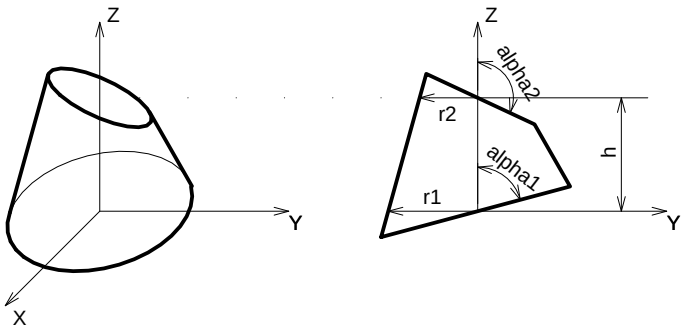
例：

```
ELLIPS r, r                    ! 半球
```



CONE

CONE h, r1, r2, alpha1, alpha2



alpha1 と alpha2 が両端の表面の z 軸に対する傾斜角、r1 と r2 が両端の円の半径、h が z 軸沿いの高さである円錐台。
h = 0 の場合、alpha1 と alpha2 の値は意味を持たず、X-Y 平面に環が作成されます。
alpha1 と alpha2 は、度（°）で指定します。

パラメータの制限：
0 < alpha1 < 180° および 0 < alpha2 < 180°
例：

```
CONE h, r, 0, 90, 90              ! 通常の円錐
```

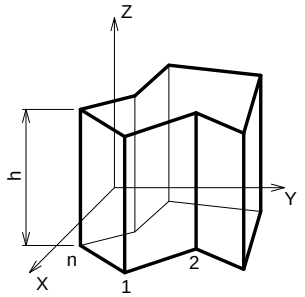
PRISM

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

基準ポリゴンが X-Y 平面上の角柱（*80 ページ「POLY」* および *80 ページ「POLY_」* のパラメータを参照）。Z 軸に沿った高さは、abs(h)。h は負の値をとることもできます。負の場合、2 番目の基準ポリゴンは、X-Y 平面より下になります。

パラメータの制限：

n >= 3



PRISM_

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

CPRISM_ ステートメントに似ていますが、水平の辺と側面のいずれかを省略することができます。

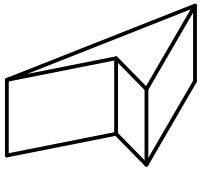
パラメータの制限：

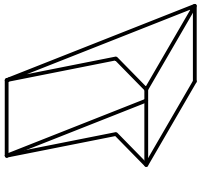
n >= 3

si: ポリゴンの辺と側面の可視性を制御できるようにするステータスコード。特殊な制約を使用して、穴を定義したり、ポリラインに線分や円弧を作成することもできます。

詳細は、181 ページ「ステータスコード」を参照してください。

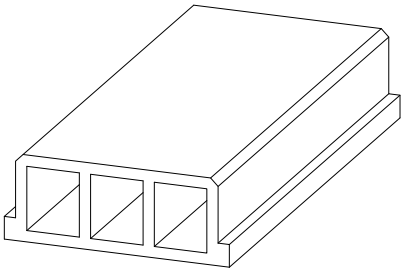
例：



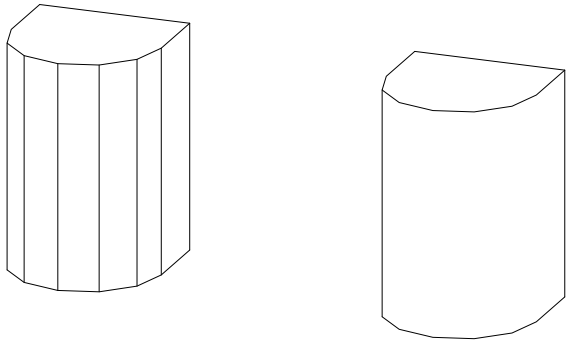


```
PRISM_ 4,1,
0,0,15,
1,1,15,
2,0,15,
1,3,15
```

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



```
ROTX 90
PRISM_ 26, 1.2,
        0.3, 0, 15,
        0.3, 0.06, 15,
        0.27, 0.06, 15,
        0.27, 0.21, 15,
        0.25, 0.23, 15,
        -0.25, 0.23, 15,
        -0.27, 0.21, 15,
        -0.27, 0.06, 15,
        -0.3, 0.06, 15,
        -0.3, 0, 15,
        0.3, 0, -1,                    ! 輪郭の終わり
        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,                ! 最初の穴の終わり
        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,                ! 2 番目の穴の終わり
        -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                  ! 3 番目の穴の終わり
```



```
    j7 = 0                      j7 = 1
R=1
H=3
PRISM_          9,              H,
    -R,         R,              15,
    COS(180)*R, SIN(180)*R,     15,
    COS(210)*R, SIN(210)*R,     15,
    COS(240)*R, SIN(240)*R,     15,
    COS(270)*R, SIN(270)*R,     15,
    COS(300)*R, SIN(300)*R,     15,
    COS(330)*R, SIN(330)*R,     15,
    COS(360)*R, SIN(360)*R,     15,
    R,          R,              15
ADDX 5
PRISM_          9,              H,
    -R,         R,              15,
    COS(180)*R, SIN(180)*R,     64+15,
    COS(210)*R, SIN(210)*R,     64+15,
    COS(240)*R, SIN(240)*R,     64+15,
    COS(270)*R, SIN(270)*R,     64+15,
    COS(300)*R, SIN(300)*R,     64+15,
    COS(330)*R, SIN(330)*R,     64+15,
    COS(360)*R, SIN(360)*R,     64+15,
    R,          R,              15
```

CPRISM_

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

PRISM_ ステートメントの拡張版。最初の 3 つのパラメータは上面、下面、および側面の材質の名前またはインデックスとして使用されます。それ以外のパラメータは前述の PRISM_ ステートメントと同じです。

パラメータの制限：

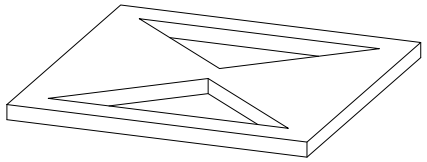
n >= 3

「204 ページ「材質」」も参照してください。

si: ポリゴンの辺と側面の可視性を制御できるようにするステータスコード。特殊な制約を使用して、穴を定義したり、ポリラインに線分や円弧を作成することもできます。

詳細は、181 ページ「ステータスコード」を参照してください。

例：



```
CPRISM_ "Iron", 0, T_,          !"Iron" は定義済み材質
!0 は一般材質

                                !T_ はグローバル変数
                                ! (材質のインデックス)
    13, 0.2,
    0, 0, 15,
    2, 0, 15,
    2, 2, 15,
    0, 2, 15,
    0, 0, -1,                    ! 輪郭の終わり
    0.2, 0.2, 15,
    1.8, 0.2, 15,
    1.0, 0.9, 15,
    0.2, 0.2, -1,                ! 最初の穴の終わり
    0.2, 1.8, 15,
    1.8, 1.8, 15,
    1.0, 1.1, 15,
    0.2, 1.8, -1                   ! 2 番目の穴の終わり
```

CPRISM_{2}

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

CPRISM_{2} は CPRISM_ コマンドの拡張版です。角柱のそれぞれの側面に異なる角度と材質を定義できます。
上部平面の定義は、CROOF_ ステートメントの場合と同じように定義します。
alphai: 角柱の辺 i に属する面と、基部に対して垂直の平面との間の角度
mati: 側面の材質を制御できるようにする材質への関連付け

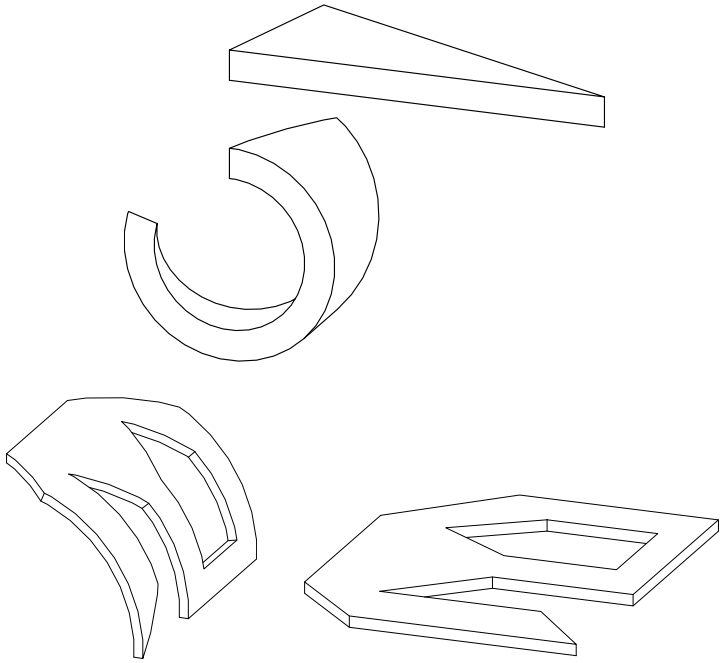
BPRISM_

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

直線の CPRISM_ 要素と同じデータ構造に基づいた、滑らかな曲面の角柱。半径のパラメータだけが追加されています。
対応する CPRISM_ の派生形。X-Y 平面をこの平面に正接する円柱の上に折り曲げることで作り出されます。X 軸沿いの辺は円形状の円弧に変形されますが、Y 軸沿いの辺は水平なままです。Z 軸沿いの辺は半径方向になります。
詳細は、59 ページ「BWALL_」を参照してください。
si: ポリゴンの辺と側面の可視性を制御できるようにするステータスコード。特殊な制約を使用して、穴を定義したり、ポリラインに線分や円弧を作成することもできます。

詳細は、181 ページ「ステータスコード」を参照してください。
例（対応する CPRISM_ との比較）:

```
BPRISM_ "Glass", "Glass", "Glass",
        3, 0.4, 1,                          ! 半径 = 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

PRISM_ ステートメントと似ていますが、hill_material、angle、および hill_height のパラメータが追加されています。丘部分は、正多角柱の上に追加されます。

hill_material: hill_material: 丘の側面の材質

angle: 丘の側面の辺の取り付け角。Restriction: 0 <= angle < 90。angle = 0 の場合、直交ビューから見た丘の側面の辺は、現在の解像度の四半円になります（コマンド *196 ページ「RADIUS」*、*197 ページ「RESOL」* および *198 ページ「TOLER」* を参照）。

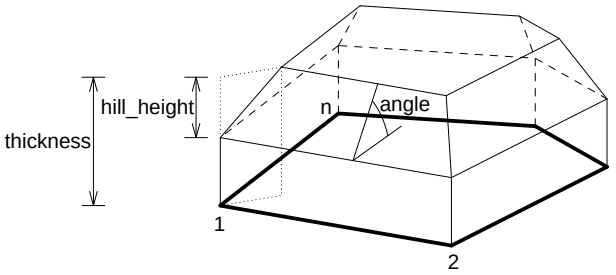
hill_height: 丘の高さ。パラメータ thickness は FPRISM_ の全高を表します。

パラメータの制限：

n >= 3, hill_height < thickness

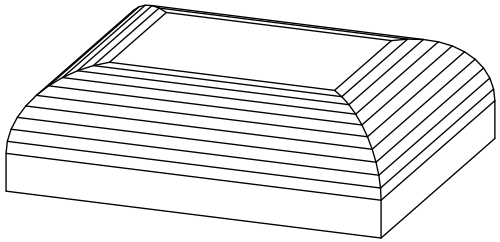
si: ポリゴンの辺と側面の可視性を制御できるようにするステータスコード。特殊な制約を使用して、穴を定義したり、ポリラインに線分や円弧を作成することもできます。

詳細は、181 ページ「ステータスコード」を参照してください。

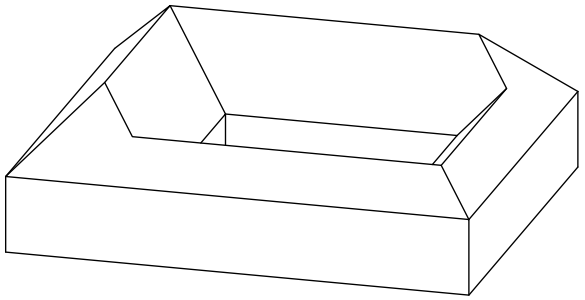


例：

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



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



HPRISM_

```
HPRISM_ top_mat, bottom_mat, side_mat,
     hill_mat,
     n, thickness, angle, hill_height, status,
     x1, y1, s1,
     ...,
     xn, yn, sn
```

FPRISM_ に似ていますが、丘の辺の可視性を制御するパラメータが追加されています。

status: 丘の辺の可視性を制御します。

0: 丘の辺は全て可視 (FPRISM_)

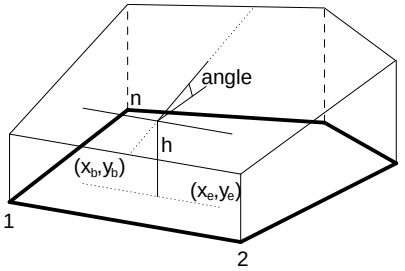
1: 丘の辺は不可視

SPRISM_

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

CPRISM_ ステートメントの拡張版。X-Y 平面と平行でない上部ポリゴンを設定できます。上部平面の定義は、CROOF_ ステートメントの場合と同じように定義します。多角柱の高さは、基準線で定義されます。上部と下部のポリゴンは交差しないように制限されます。

追加パラメータ：



xb, yb, xe, ye: 基準線（ベクトル）の開始部分と終了部分の座標、

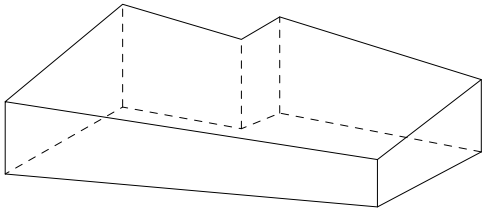
angle: 与えられた方向の基準線を中心とした上部ポリゴンの度数単位の回転角（反時計回り）。

si: ポリゴンの辺と側面の可視性を制御できるようにするステータスコード。特殊な制約を使用して、穴を定義したり、ポリラインに線分や円弧を作成することもできます。

詳細は、181 ページ「ステータスコード」を参照してください。

注記： 上部ポリゴンの各節点の計算された Z 座標は、正の値または 0 でなければなりません。

例：



```
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}

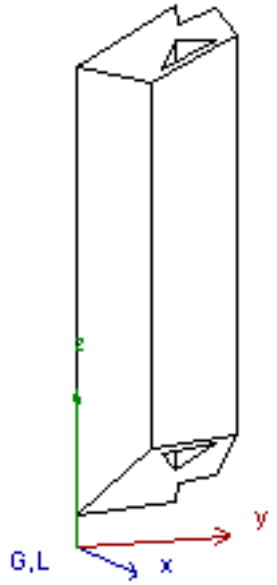
SPRISM_{2} top_material, bottom_material, side_material,
n,
xtb, ytb, xte, yte, topz, tangle,
xbb, ybb, xbe, ybe, bottomz, bangle,
x1, y1, s1, mat1,
...
xn, yn, sn, matn

SPRISM_ ステートメントの拡張版。X-Y 平面と平行でない上部ポリゴンおよび下部ポリゴンを設定できます。平面の定義は、CROOF_ ステートメントの場合と同じように定義します。多角柱の上部と下部は、基準線で定義します。上部と下部のポリゴンは交差しないように制限されます。

追加パラメータ：

xtb, ytb, xte, yte：上部ポリゴンの基準線（ベクトル）の開始部分と終了部分の座標、topz：上部ポリゴンの基準線の「z」レベル、
tangle：与えられた方向の基準線を中心とした上部ポリゴンの度数単位の回転角（反時計回り）、
xbb, ybb, xbe, ybe：下部ポリゴンの基準線（ベクトル）の開始部分と終了部分の座標、
bottomz：上部ポリゴンの基準線の「z」レベル、
bangle：与えられた方向の基準線を中心とした下部ポリゴンの度数単位の回転角（反時計回り）、
si：ポリゴンの辺と側面の可視性を制御できるようにするステータスコード。特殊な制約を使用して、穴を定義したり、ポ

リラインに線分や円弧を作成することもできます。
詳細は、181 ページ「ステータスコード」を参照してください。
mati: 側面の材質を制御できるようにする材質への関連付け
例：



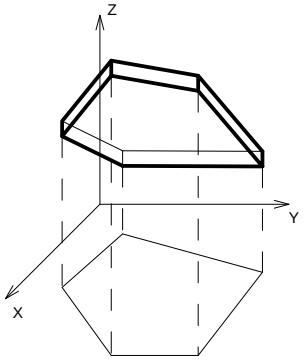

```
SPRISM_{2} 'Grass', 'Earth', 'Earth',
        11,
        0,   0,   11,   0,   30, -30.0,
        0,   0,    0,  11,    2,  30.0,

        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')
```

SLAB

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

傾斜した多角柱。横の面は、X-Y 平面に対して常に垂直です。底面は、X-Y 平面に平行な軸を中心として回転された平らなポリゴンです。h は負の値をとることもできます。負の場合、2 番目の基準ポリゴンは、与えられた基準ポリゴンの下になります。
点が実際に平面上にあるかどうかは確認されません。頂点が平面上にないと、シェーディングやレンダリングで正常な結果が出ません。
パラメータの制限：
n >= 3



SLAB_

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

SLAB ステートメントに似ていますが、水平の辺と側面のいずれかを省略することができます。このステートメントは、PRISM_ ステートメントにも似ています。
si: ポリゴンの辺と側面の可視性を制御できるようにするステータスコード。特殊な制約を使用して、穴を定義したり、ポリラインに線分や円弧を作成することもできます。

詳細は、181 ページ「ステータスコード」を参照してください。

CSLAB_

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

SLAB_ ステートメントの拡張版。最初の 3 つのパラメータは、上面、下面、および側面の材質の名前またはインデックスとして使用されます。それ以外のパラメータは、前述の SLAB_ ステートメントの場合と同様です。

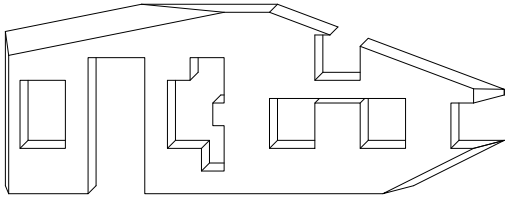
si: ポリゴンの辺と側面の可視性を制御できるようにするステータスコード。特殊な制約を使用して、穴を定義したり、ポリラインに線分や円弧を作成することもできます。

詳細は、181 ページ「ステータスコード」を参照してください。

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:
左、右および側面の表面の材質の名前またはインデックス （壁の左右の側面は、X 軸沿いになります)。



壁の基準線は、常に X 軸と一致するように変換されます。壁の側面は、X-Z 平面上になります。

height: 壁の基準面を基準とした壁高さ

x1ÅAx2ÅAx3ÅAx4: 次に示すような x-y 平面上にある壁の投影終了点です。壁が自立している場合、

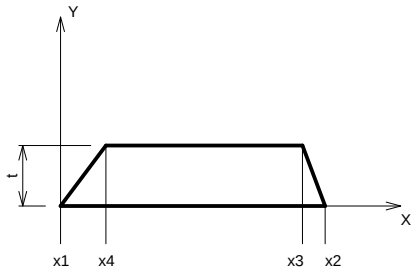
x1 = x4 = 0ÅAx2 = x3 = 壁の長さとなります。

t: 壁厚さ

t < 0: 壁のボディが X 軸より右の場合

t > 0: 壁のボディが X 軸より左の場合

t = 0: 壁をポリゴンによって表し、穴のまわりにフレームを生成する場合



mask1、mask2、mask3、mask4: 辺および側面ポリゴンの可視性を制御します。

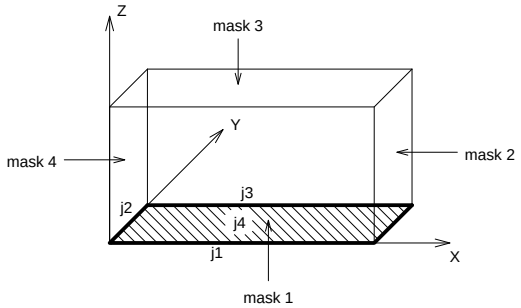
maski = j1 + 2*j2 + 4*j3 + 8*j4

ここで、j1、j2、j3、j4 は、0 または 1 をとります。

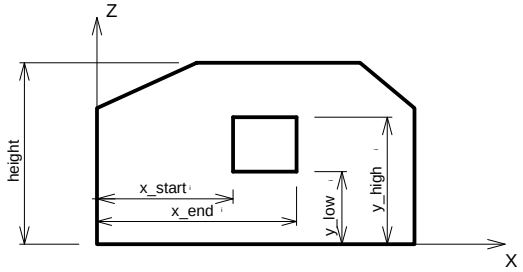
j1、j2、j3、j4 の値は、頂点および側面を表示する（1）か省略する（0）かを表します。

n: 壁の開口の数。

x_starti, y_lowi, x_endi, y_highi: 次に示すように、開口部の座標です。


Z
mask 3
Y
mask 4
mask 2
j3
j2
j4
j1
X
mask 1

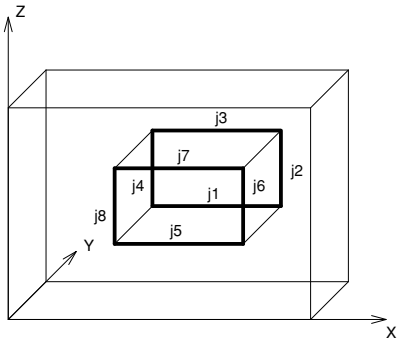
frame_showni values: 1: 穴の辺を表示する場合 ; 0: 穴の辺を表示しない場合。
負の値では、開口部の辺の可視性を個別に制御します。



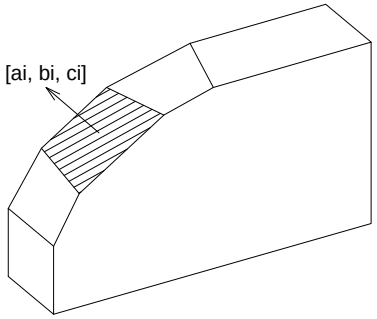
frame_showni = -(1*j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8), ここで、j1, j2... j8 は、0 または 1 をとります。次の図に示すように、j1 から j4 までの値で壁表面の左側にある穴の辺の可視性を、j5 から j8 の値で右側にある辺の可視性をそれぞれ制御します。

壁の表面に対して垂直な辺は、その両端から可視の辺が引かれている場合には、可視となります。

m: 切断面の数



ai, bi, ci, di: 切断面を定義する方程式の係数 [ai*x + bi*y + ci*z = di]。切断面の正方向の部分（つまり、ai*x + bi*y + ci*z > di）が切り取られ、削除されます。



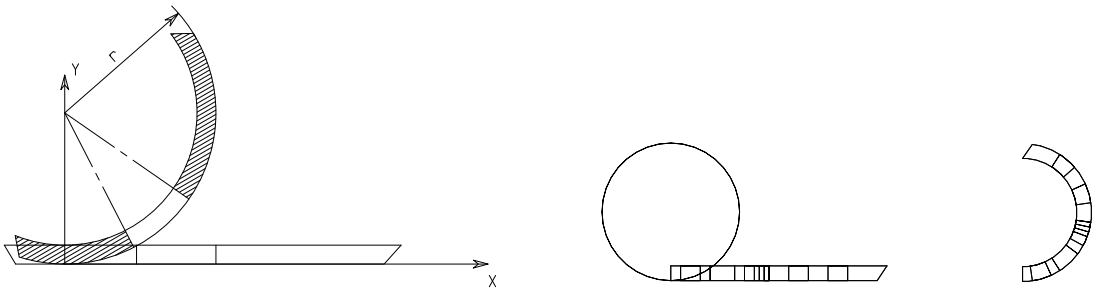
BWALL_

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

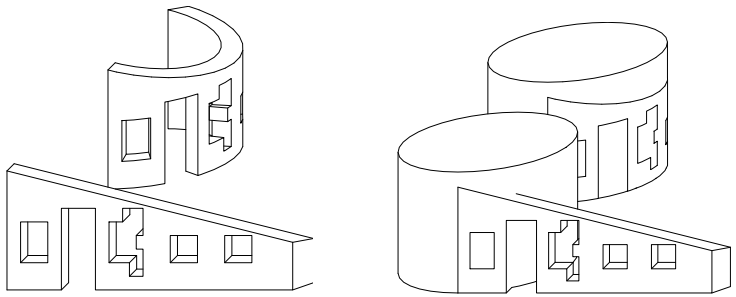
直線壁である CWALL_ 要素と同じデータ構造に基づいた、滑らかな曲面壁。半径のパラメータだけが追加されています。

対応する CWALL_ の派生形。X-Z 平面をこの平面に正接する円柱の上に折り曲げることで作り出されます。X 軸沿いの辺は円形状の円弧に変形されますが、Y 軸沿いの辺は半径方向になり、垂直な辺は垂直なままです。曲率は、現在の解像度で設定された辺数によって近似されます（コマンド *196 ページ「RADIUS」*、*197 ページ「RESOL」* および *198 ページ「TOLER」* を参照）。

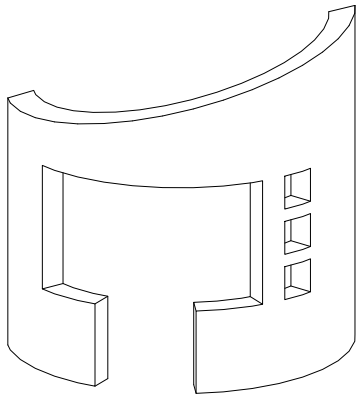
詳細は、55 ページ「CWALL_」を参照してください。


Y
r
X

*例： BWALL_ と対応する CWALL_*



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



XWALL_

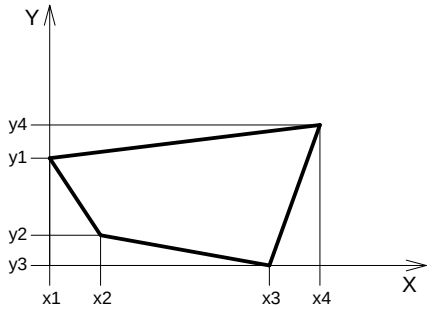
```
XWALL_ left_material, right_material, vertical_material,horizontal_material,
      height, x1, x2, x3, x4,
      y1, y2, y3, y4,
      t, radius,
      log_height, log_offset,
      mask1, mask2, mask3, mask4,
      n,
      x_start1, y_low1, x_end1, y_high1,
      frame_shown1,
      ...
      x_startn,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

BWALL_ 要素と同じデータ構造に基づいた壁の定義の拡張版。

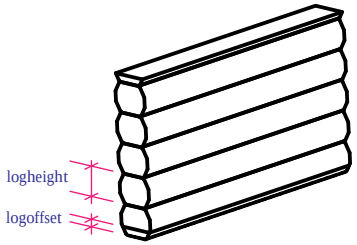
追加パラメータ：

vertical_material、horizontal_material: 垂直 / 水平側面の材質の名前またはインデックス

y1, y2, y3, y4: 次に示すような x-y 平面上にある壁の投影終了点



log_height、log_offset: 壁をログで構成できるようにする追加パラメータ。直線壁の場合のみ有効です。



status: ログで構成した壁の挙動を制御します。

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

j1: 水平な辺に右側面材質を適用

j2: 水平な辺に左側面材質を適用

j3: ハーフログから開始

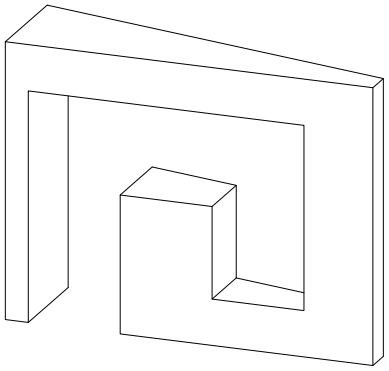
j6: 壁の辺に合わせてテクスチャを調整

j7: 曲がった側面の二重半径

j8: 右側面を角ログに

j9: 左側面を角ログに

例：



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

XWALL_{2}

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

XWALL_ 要素と同じデータ構造に基づいた壁定義の拡張版。

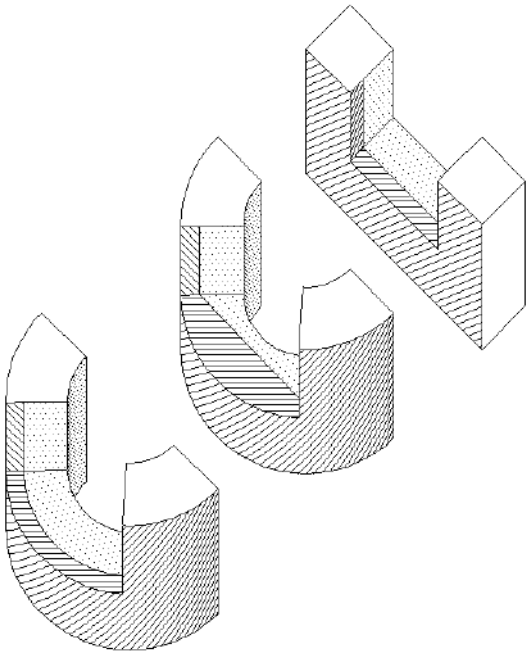
追加パラメータ：

silldepthi: 開口部の下枠 / 敷居の論理奥行き。frame_showni パラメータの j9 ビットが設定されている場合、壁の側面の材質は穴のポリゴンをラップします。silldepthi は、材質と材質との間の区切り線を定義します。

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

材質のラップを制御する追加の値は 2 つあります。値 j1、j2 ... j8 の意味は、CWALL_ コマンドと XWALL_ コマンドと同じです。値 j9 は、穴のポリゴンの材質を制御します。j9 が 1 の場合、穴は壁の側面材質を継承します。値 j10 は、折れ曲がった壁の場合、穴の上側のポリゴンと下側のポリゴンの穴材質間の区切り線の形式を制御します。値 j10 が 1 の場合区切り線は直線で、その他の場合は曲線です。

例：



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

```
          0, 0, 1, 1,
          0, 0,
          15, 15, 15, 15,
          1,
          1.6, 0.9, 4.6, 2.1, 0.3, -(255 + 256),
          0,
          0
BODY -1

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

BEAM

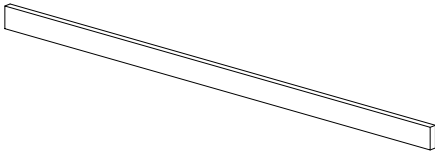
```
BEAM left_material, right_material, vertical_material, top_material, bottom_material,
     height, x1, x2, x3, x4,
     y1, y2, y3, y4, t,
     mask1, mask2, mask3, mask4
```

梁の定義 パラメータは XWALL_ 要素のものと似ています。

追加パラメータ：

top_material, bottom_material: 上面および下面の材質

例：



```
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
```

破風の切り落とし角度を設定できる勾配屋根。

top_material, bottom_material, side_material: 上面、下面および側面の材質の名前またはインデックス

n: 屋根ポリゴンに含まれる節点の数

xb, yb, xe, ye: 基準線（ベクトル）

height: 基準線（下面）での屋根の高さ

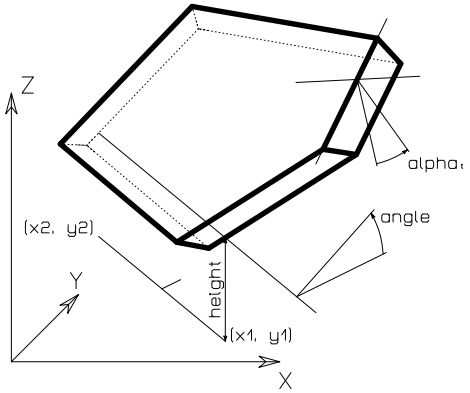
angle: 与えられた方向の基準線を中心とした屋根平面の度数単位の回転角（反時計回り）

thickness: 屋根平面に対して垂直に測定された屋根の厚さ

xi, yi: 屋根の下部ポリゴンの節点の座標

alphai: 屋根の辺 i に属している面と屋根平面に垂直な平面との角度。-90 < alphai < 90。屋根ポリゴンの各辺の正規の方向に対して反時計回りの回転角が正の角度になります。
屋根ポリゴンの辺が正規の向きになるのは、上面図で、輪郭を反時計回り方向、穴を時計回り方向にした場合です。

si: ポリゴンの辺と側面の可視性を制御できるようにするステータスコード。特殊な制約を使用して、穴を定義したり、ポリラインに線分や円弧を作成することもできます。

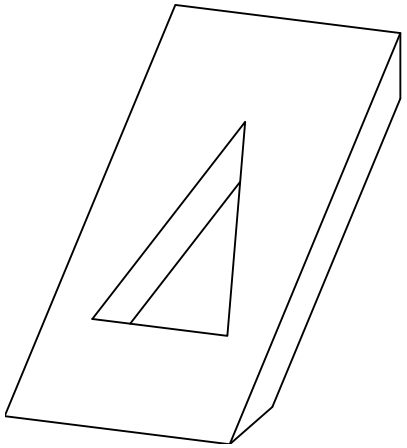


詳細は、181 ページ「ステータスコード」を参照してください。

パラメータの制限：

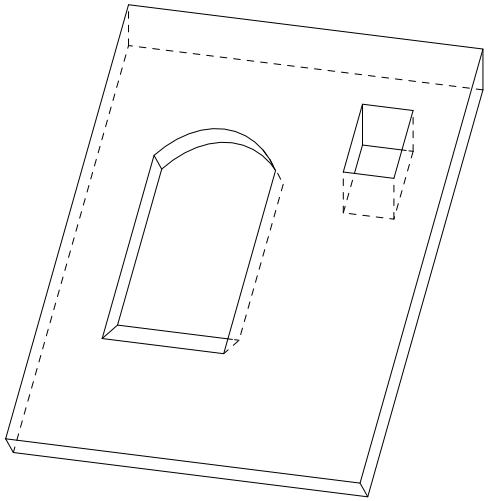
n >= 3

例：



```
CROOF_ 1, 1, 1,                 ！材質
          9,
          0, 0,
          1, 0,                   ！基準線 (xb,yb)(xe,ye)
          0.0,                    ！高さ
          -30,                    ！角度
          2.5,                    ！ 厚さ
          0, 0, -60, 15,
          10, 0, 0, 15,
          10, 20, -30, 15,
          0, 20, 0, 15,
          0, 0, 0, -1,
          2, 5, 0, 15,
          8, 5, 0, 15,
          5, 15, 0, 15,
          2, 5, 0, -1
L=0.25
R=(0.6^2+L^2)/(2*L)
A=ASN(0.6/R)
CROOF_ "Roof Tile", "Pine", "Pine",
          16, 2, 0, 0,
          0, 0, 45, -0.2*SQR(2),
          0, 0, 0, 15,
          3.5, 0, 0, 15,
```

```
3.5,  3,  -45,  15,
0,  3,  0,  15,
0,  0,  0,  -1,
0.65,  1,  -45,  15,
1.85,  1,  0,  15,
1.85,  2.4-L,  0,  13,
1.25,  2.4-R,  0,  900,
0,  2*A,  0,  4015,
0.65,  1,  0,  -1,
2.5,  2,  45,  15,
3,  2,  0,  15,
3,  2.5,  -45,  15,
2.5,  2.5,  0,  15,
2.5,   2,  0,  -1
```



CROOF_{2}

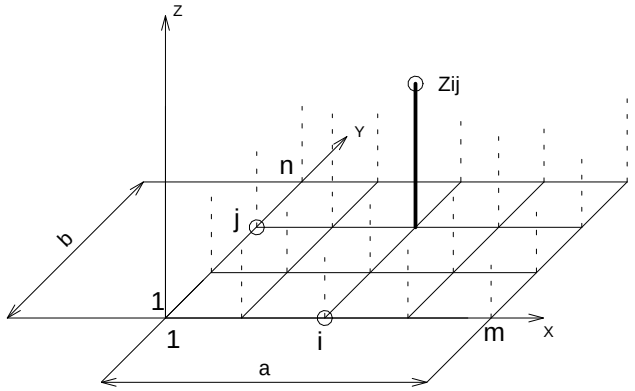
CROOF_{2} top_material, bottom_material, side_material,
n, xb, yb, xe, ye, height, angle thickness,
x1, y1, alpha1, s1, mat1
...
xn, yn, alphan, sn, matn

CROOF_{2} は CROOF_ ステートメントの拡張版です。側面に別々の材質を定義できます。
mati: 側面の材質を制御できるようにする材質への関連付け

MESH

MESH a, b, m, n, mask,
z11, z12, ... z1m,
z21, z22, ... z2m,
...
zn1, zn2, ... znm

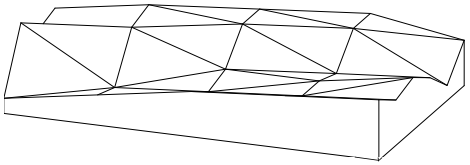
等間隔である矩形を基準にした単純で滑らかなメッシュ。基準矩形の辺は a と b で、m と n はそれぞれ X 軸と Y 軸に沿った点の数で、zij は節点の高さです。
マスキング:



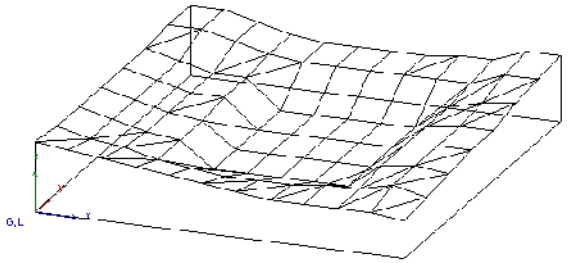
mask = j1 + 4*j3 + 16*j5 + 32*j6 + 64*j7
j1, j3, j5, j6, j7 は 0 または 1 をとります。
j1 (1): 基準面を表現

j3 (4): 側面を表現
j5 (16): 基準面および側面の辺は可視
j6 (32): 上面の辺は可視
j7 (64): 上面の辺は可視、表面はスムージングされない
パラメータの制限：
m >= 2, n >= 2

例：



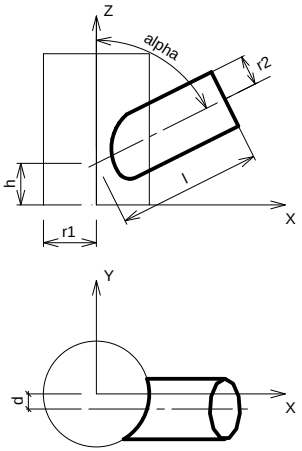
```
MESH 50,  30,  5,  6,  1+4+16+32+64,
          2,  4,  6,  7,  8,
          10,  3,  4,  5,  6,
          7,  9,  5,  5,  7,
          8,  10,  9,  4,  5,
          6,  7,  9,  8,  2,
          4,  5,  6,  8,  6
```




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

ARMC

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

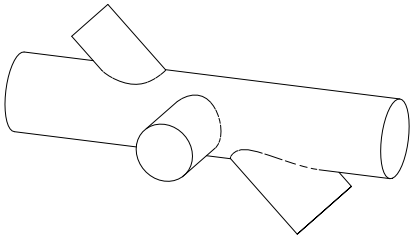


ほかの管から突き出ている管。パラメータについては図を参考にしてください（貫通する曲線についても計算され、描かれます)。alpha は度（°）で指定します。

パラメータの制限：

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

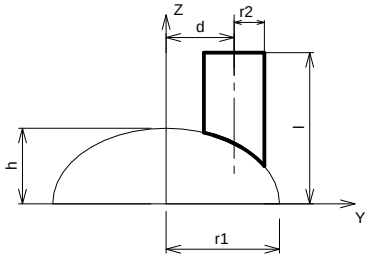
例：



```
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



Y-Z 平面の楕円から突き出ている管。パラメータについては図を参考にしてください（貫通する曲線についても計算され、描かれます)。

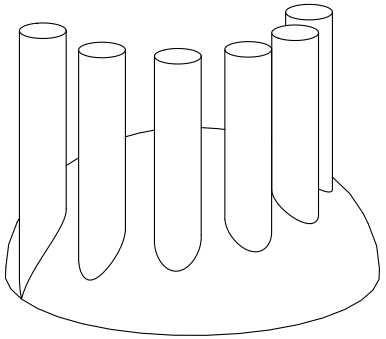
パラメータの制限：

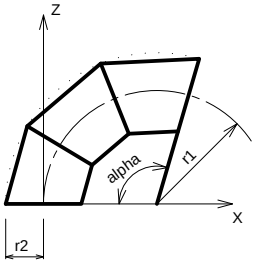
r1 >= r2+d

l >= h*√(1-(r2-d)2/r12)

例：

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





ELBOW

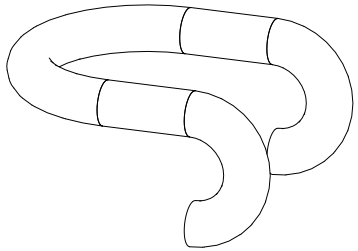
ELBOW r1, alpha, r2

X-Z 平面の線分化されたエルボ。円弧の半径は r1 で、角度は alpha、管線分の半径は r2。alpha は度（°）で指定します。

パラメータの制限：

r1 > r2

例：



```
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
```

3D での平面形状

このセクションで説明する図形要素は、3D スクリプトで使用して、3 次元空間の点、線、円弧、円、平面ポリゴンを定義することができます。

HOTSPOT

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

点（x, y, z）の 3D ホットスポット。

unID は、3D スクリプトにおけるホットスポットの一意の識別子です。これは可変数のホットスポットが存在する場合に便利です。

paramReference: グラフィカルホットスポットベースのパラメータ編集手法を使って編集可能なパラメータ。

displayParam: paramRefrence パラメータ編集中に情報パレットに表示するためのパラメータ。配列のメンバも使えます。

ホットスポットの使用方法については、175 ページ「グラフィカル編集」を参照してください。

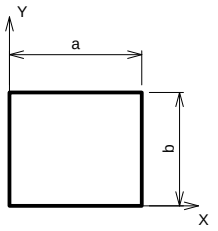
LIN_

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

点 P1（x1,y1,z1）と点 P2（x2,y2,z2）の間の直線線分。

RECT

RECT a, b



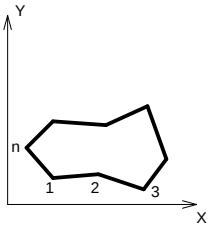
辺 a と b を持つ X-Y 平面上の矩形。

パラメータの制限：

a, b >= 0

POLY

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

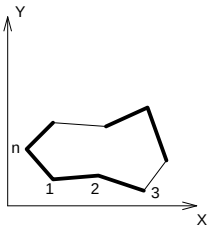


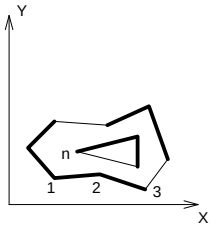
X-Y 平面上の n 個の辺を持つポリゴン。節点 i の座標は（xi, yi, 0)。
パラメータの制限：
n >= 3

POLY_

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

標準の POLY ステートメントとほとんど同じですが、辺のいずれかを省略することができます。si = 0 の場合、頂点（xi, yi）から延びる辺は省略されます。si = 1 の場合、辺は表示されます。
si = -1 は、直接開口（穴）を定義するのに使用します。
追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。
詳細は、183 ページ「追加ステータスコード」を参照してください。





パラメータの制限：
n >= 3

PLANE

PLANEn, x1, y1, z1, ... xn, yn, zn

任意の平面上の n 個の辺を持つポリゴン。節点 i の座標は（xi, yi, zi)。正しいシェーディングやレンダリングの結果を得るためには、ポリゴンは必ず平面でなければなりませんが、インタプリタはこれを確認しません。

パラメータの制限：

n >= 3

PLANE_

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

正規の PLANE ステートメントとほとんど同じですが、POLY_ ステートメントの場合と同様、辺のいずれかを省略することができます。

追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。

「183 ページ「追加ステータスコード」」を参照してください。

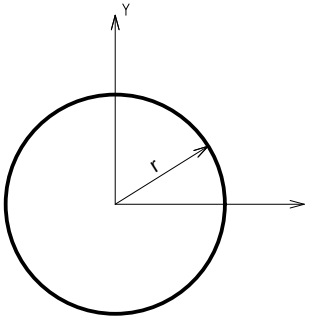
パラメータの制限：

n >= 3

CIRCLE

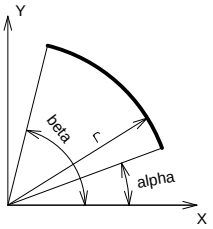
CIRCLE r

X-Y 平面上の、原点を中心とし、半径が r である円。



ARC

ARC r, alpha, beta



X-Y 平面上の、原点を中心とし、alpha で始まり beta で終わる角度を持ち、半径が r ある円弧（ワイヤフレームモード）または扇形（その他のモード）。
alpha と beta は度（° ）で指定します。

ポリラインから生成される形状

これらの要素では、ポリラインと内部規則を使用して複雑な 3D 形状を作成することができます。与えられたポリラインを回転、投影、変換することができます。その結果作成されたボディは、PRISM_ や CYLIND などのこれまでに説明した要素と同様に一般化されます。

1 本のポリラインから作成される形状：

EXTRUDE
PYRAMID
REVOLVE

2 本のポリラインから作成される形状：

RULED
SWEEP[SWEEP]
TUBE
TUBEA[TUBEA]

最初のポリラインは、常に X-Y 平面上にあります。各点は 2 つの座標値で決まります。3 番目の値はステータスを表わします（以下を参照）。2 本目のポリライン（RULED と SWEEP 用）は空間曲線になります。各頂点は 3 つの座標値で決まります。

4 本のポリラインで作成される形状：

COONS

任意の数のポリラインから作成される形状：

MASS[MASS]

ポリラインの一般的な制限

・隣接する頂点は異なる位置になければなりません（RULED を除く）。
・ポリラインは自己交差してはいけません（プログラムによる確認は行なわれないので、隠線除去やレンダリングが不適切になります）。
・ポリラインは開いていても閉じていても構いません。閉じている場合には、最初の節点をステートメントの最後で繰り返す必要があります。

マスキング

マスク値は 3D 形状の表面または辺、またはこの両方を表示または非表示するのに使用します。マスク値は要素ごとに異なります。詳細については、各要素を説明する節または章を参照してください。

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

j1、j2、j3、j4、j5、j6、j7 は 0 か 1 をとります。

j1、j2、j3、j4 の値は、表面がある（1）か省略する（0）かを表します。

j5、j6、j7 の値は、辺を表示する（1）か非表示にする（0）かを表します。

j1: 基準面

j2: 上面

j3: 側面

j4: 反対の側面

j5: 基準面の辺

j6: 上面の辺

j7: 横断面 / 表面の辺は可視、表面はスムージングされない

全ての面と辺を使用可能にするには、マスク値を 127 に設定します。

ステータス：

ステータス値は、ポリラインの与えられた点において、回転パスからのはっきりとした輪郭を表示するかどうかを決定します。

0: 節点から始まる縦の円弧 / 側面の辺を全て表示

1: 節点から始まる縦の円弧 / 側面の辺を輪郭の表示にのみ使用

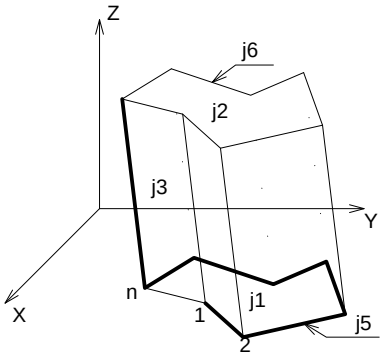
-1: EXTRUDE の場合のみ： 閉じたポリゴンまたは穴の終わりをマークし、次の節点が別の穴の最初の節点となることを示す

追加ステータスコードでは、特殊な制約を使用して、ポリライン内に線分や円弧を作成することができます。

詳細は、183 ページ「追加ステータスコード」を参照してください。
滑らかな 3D 形状を作成するには、全てのステータス値を 1 に設定します。status = 0 に設定すると、隆起線が作成されます。
その他の値は将来の拡張のために予約されています。

EXTRUDE

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



X-Y 平面のポリラインを底面として使用した一般的な多角柱。底面と底面の間の変位ベクトルは（dx, dy, dz）です。
これは、PRISM と SLAB ステートメントを汎用化したものです。基準ポリラインは閉じている必要はありません。側面の辺は、X-Y 平面に対して常に垂直ではありません。基準ポリラインには、PRISM_ と同様に穴を含めることができます。また、輪郭辺の可視性を制御することができます。
n: ポリラインの節点の数

mask: 下面、上面、および（開いているポリラインの場合）側面のポリゴンの有無を制御します。

si: 側面の辺のステータスまたは、ポリゴンや穴の終端をマークします。追加ステータスコード値を使用して、ポリラインに円弧および線分を定義することもできます。

パラメータの制限：

n > 2
マスキング：

mask = j1 + 2*j2 + 4*j3 + 16*j5 + 32*j6 + 65*j7 +128*j8
ここで、j1, j2, j3, j5, j6, j7, j8 は 0 または 1 をとります。
j1 (1): 基準面を表現

j2（2）：上面を表現

j3 （4）：（閉じている）側面を表現

j5（16）：基準面の辺は可視

j6（32）：上面の辺は可視

j7（64）：横断面の辺は可視、表面は明確

j8（128）：横断面の辺はシャープで、表面のスムージングは OpenGL とレンダリングで停止

ステータス値：

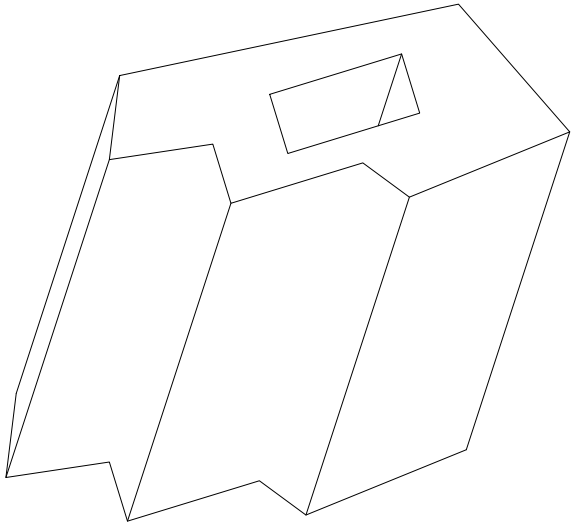
0：節点から延びる側面の辺は可視

1：節点から延びる側面の辺を輪郭の表示に使用

-1：閉じたポリゴンまたは穴の終わりをマークし、次の節点が別の穴の最初の頂点となることを示す

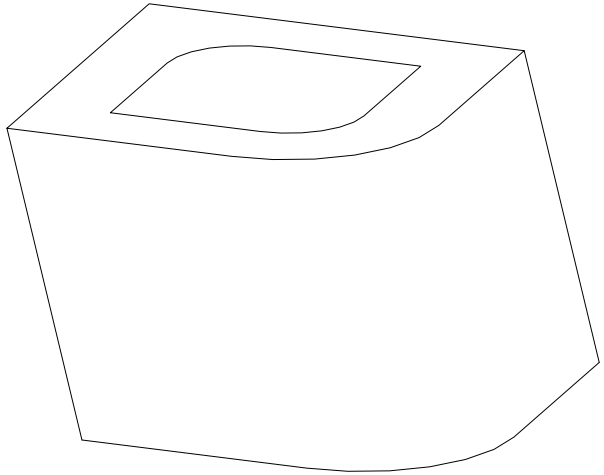
追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。
詳細は、183 ページ「追加ステータスコード」を参照してください。

例：



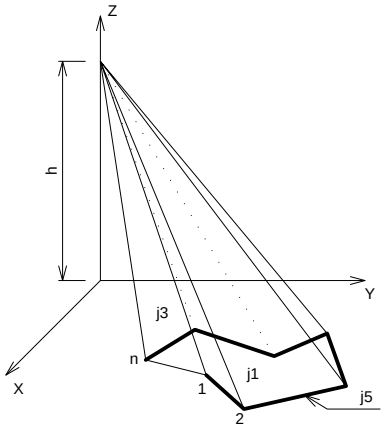
```
EXTRUDE 14, 1, 1, 4, 1+2+4+16+32,
          0, 0, 0,
          1, -3, 0,
          2, -2, 1,
          3, -4, 0,
          4, -2, 1,
          5, -3, 0,
          6, 0, 0,
          3, 4, 0,
          0, 0, -1,
          2, 0, 0,
          3, 2, 0,
          4, 0, 0,
          3, -2, 0,
          2, 0, -1
A=5: B=5
R=2: S=1
C=R-S
D=A-R
E=B-R
EXTRUDE 28, -1, 0, 4, 1+2+4+16+32,
          0, 0, 0,
          D+R*SIN(0),  R-R*COS(0), 1,
          D+R*SIN(15), R-R*COS(15), 1,
          D+R*SIN(30), R-R*COS(30), 1,
          D+R*SIN(45), R-R*COS(45), 1,
          D+R*SIN(60), R-R*COS(60), 1,
          D+R*SIN(75), R-R*COS(75), 1,
          D+R*SIN(90), R-R*COS(90), 1,
          A, B, 0,
          0, B, 0,
          0, 0, -1,
          C, C, 0,
          D+S*SIN(0), R-S*COS(0), 1,
          D+S*SIN(15), R-S*COS(15), 1,
          D+S*SIN(30), R-S*COS(30), 1,
          D+S*SIN(45), R-S*COS(45), 1,
          D+S*SIN(60), R-S*COS(60), 1,
          D+S*SIN(75), R-S*COS(75), 1,
          D+S*SIN(90), R-S*COS(90), 1,
          A-C,B-C,0,
          R-S*COS(90), E+S*SIN(90), 1,
          R-S*COS(75), E+S*SIN(75), 1,
```

```
R-S*COS(60), E+S*SIN(60), 1,
R-S*COS(45), E+S*SIN(45), 1,
R-S*COS(30), E+S*SIN(30), 1,
R-S*COS(15), E+S*SIN(15), 1,
R-S*COS(0), E+S*SIN(0), 1,
C, C, -1
```



PYRAMID

PYRAMID n, h, mask, x1, y1, s1, ... xn, yn, sn



X-Y 平面上のポリラインを基準とする角錐。角錐の先端は、(0,0,h) に配置されます。

n: ポリラインの節点の数

mask: 下面、上面、および（開いているポリラインの場合）側面のポリゴンの有無を制御します。

si: 側面の辺のステータス

パラメータの制限：

h > 0 および n > 2

マスキング ：

mask = j1 + 4*j3 + 16*j5

j1、j3、j5 は 0 または 1 をとります。

j1 (1): 基準面を表現

j3 (4): （閉じている）側面を表現

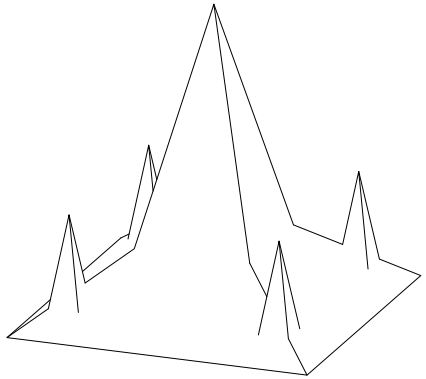
j5 (16): 基準面の辺は可視

ステータス値：

0: 節点から延びる側面の辺は全て可視

1: 節点から延びる側面の辺を輪郭の表示に使用

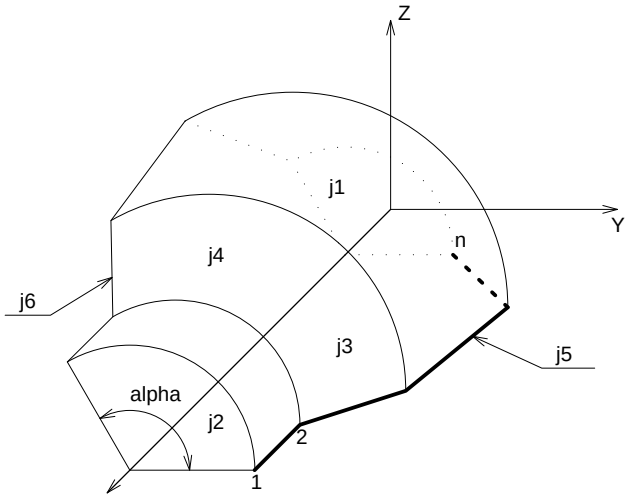
追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。
詳細は、183 ページ「追加ステータスコード」を参照してください。
例：



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

REVOLVE

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



X-Y 平面に定義されたポリラインを X 軸を中心として回転して生成される表面。

n: ポリラインの節点の数

alpha: 度数単位の掃引角度

mask: 下面、上面、および（alpha < 360 の場合の）側面のポリゴンの存在を制御します。

si: 縦の円弧のステータス

パラメータの制限：

n >= 2
yi >= 0.0
yi と yi + 1 （つまり、隣接する 2 つの節点の y 値）は同時に 0 にはできません。

マスキング ：

status = j1 + 2*j2 + *j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8
j1, j3, j5, j6, j7 は 0 または 1 をとります。

j1 (1): 基準面を表現

j2 (2): 端の表面を表現

j5 (16): 基準面の辺（座標 x2, y2, z2）は可視

j6 (32): 最後の辺（座標 xm-1, ym-1, zm-1）は可視

j7 (64): 横断面の辺は可視、表面は明確

j8 (128): 横断面の辺はシャープで、表面のスムージングは OpenGL とレンダリングで停止

ステータス値：

0: 節点から延びる側面の円弧は全て可視

1: 節点から延びる側面の円弧は輪郭の表示に使用される

2: ArchiCAD または Z バッファレンダリングエンジンを使用し、スムージングを設定する時は、この点に属する横の辺は区切りを定義します。この方法は、追加節点の定義と同等です。計算は、コンパイラによって行われます。ほかのレンダリング方式を使用すると、0 を使用した場合と同じ結果になります。

追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。
詳細は、183 ページ「追加ステータスコード」を参照してください。

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

REVOLVE の上位バージョン。開始角度と面材料が制御可能です。

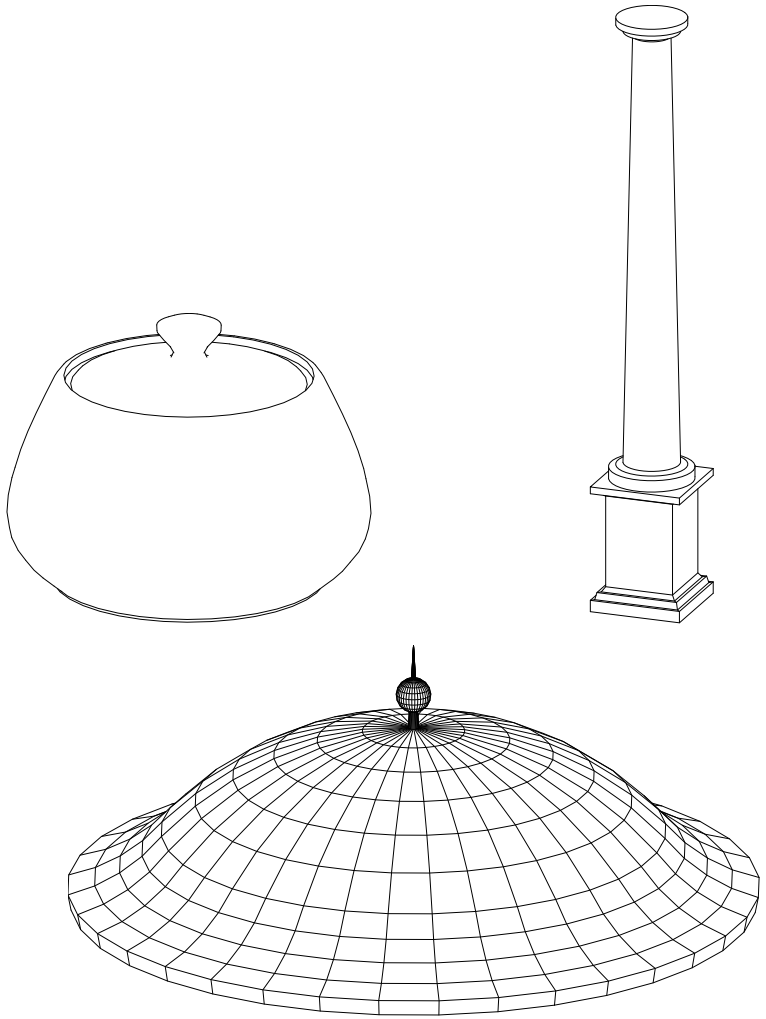
alphaOffset: 掃引開始角度

alpha: 度単位の掃引角度距離（負の値も可）

sideMat: 閉じている面の材質

mati: i 番目のエッジから生成されたフェースの材質

例：



```
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
```

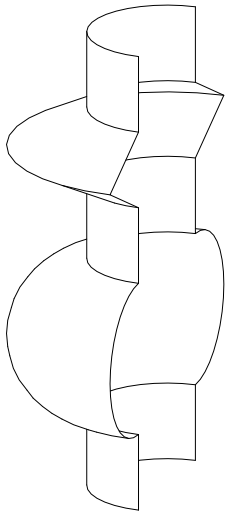
ステータスコード 2 と：

```
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,
```

結果が異なる場合の回避策 ステータスコード 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,
```

```
3+COS(75), 1+SIN(75), 1,          2, 1, 2,
3, 2, 1,                          1, 1, 0
3+COS(105), 1+SIN(105), 1,
3+COS(120), 1+SIN(120), 1,
3+COS(135), 1+SIN(135), 1,
3+COS(150), 1+SIN(150), 1,
3+COS(165), 1+SIN(165), 1,
2, 1, 0,
1.9999, 1, 0,
1, 1, 0
```

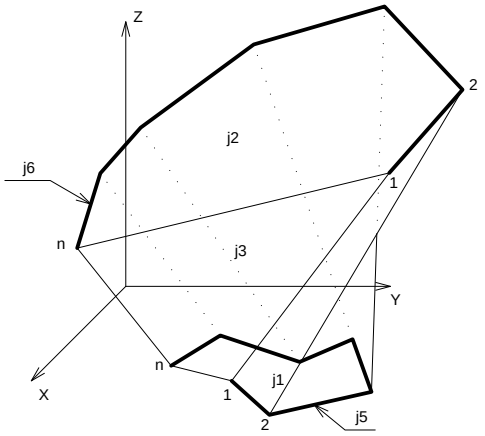


RULED

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

RULED{2}

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



RULED は、節点の数が同じ平面曲線と空間曲線に基づいた表面です。直線線分は、この 2 つのポリラインの対応する節点を接続します。

このコマンドは、隣接する節点が重なることを許可する唯一の GDL 要素です。

Ver. 2 の RULED {2} では、上部ポリゴンと下部ポリゴンの両方が定義された点の方向（右回りか、左回りか）を確認し、必要に応じて向きを反転します （オリジナルの RULED コマンドで考慮されるのは、ベースポリゴンだけであり、エラーが発生する可能性があります)。

n: 各曲線のポリラインの節点の数

mask: 下面、上面、および側面のポリゴンの存在と、元となったポリライン上の辺の可視性を制御します。側面のポリゴンは、曲線が閉じていない場合は、曲線の最初と最後の節点を接続します。

ui, vi: 平面曲線の節点の座標

si: 側面の辺のステータス

xi, yi, zi: 空間曲線の節点の座標

パラメータの制限：

n > 1

マスキング：

mask = j1 + 2*j2 + 4*j3 + 16*j5 + 32*j6 + 64*j7　ここで、j1, j2, j3, j5, j6, j7 は 0 または 1 をとります。

j1 (1): 基準面を表現

j2 (2): 上面を表現（上面が平面でない場合は無効）

j3 (4): 側面を表現（平面四角形あるいは 2 つの三角形）

j5 (16): 平面曲線の辺は可視

j6 (32): 空間曲線の辺は可視

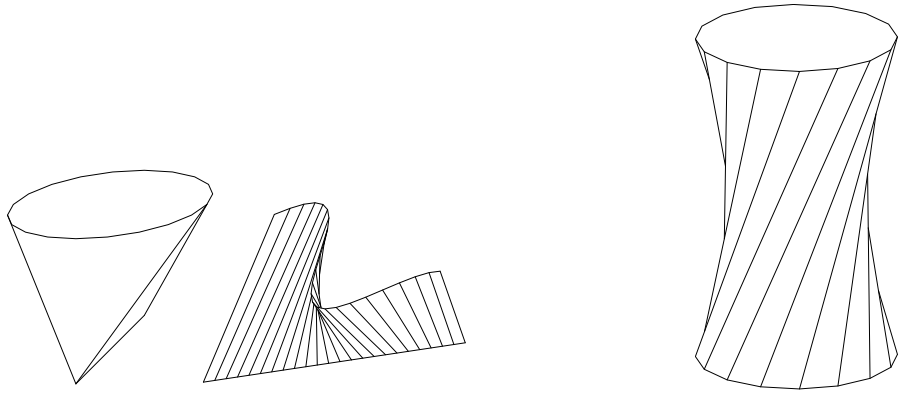
j7 (64): 表面上の辺は可視。表面はスムージングされない

ステータス値：

0: 節点から延びる側面の辺は全て可視

1: 節点から延びる側面の辺を輪郭の表示に使用

例：



```
R=3
RULED  16, 1+2+4+16+32,
       COS(22.5)*R, SIN(22.5)*R, 0,
       COS(45)*R, SIN(45)*R, 0,
       COS(67.5)*R, SIN(67.5)*R, 0,
       COS(90)*R, SIN(90)*R, 0,
       COS(112.5)*R, SIN(112.5)*R, 0,
       COS(135)*R, SIN(135)*R, 0,
       COS(157.5)*R, SIN(157.5)*R, 0,
       COS(180)*R, SIN(180)*R, 0,
       COS(202.5)*R, SIN(202.5)*R, 0,
       COS(225)*R, SIN(225)*R, 0,
       COS(247.5)*R, SIN(247.5)*R, 0,
       COS(270)*R, SIN(270)*R, 0,
       COS(292.5)*R, SIN(292.5)*R, 0,
       COS(315)*R, SIN(315)*R, 0,
       COS(337.5)*R, SIN(337.5)*R, 0,
       COS(360)*R, SIN(360)*R, 0,
       COS(112.5)*R, SIN(112.5)*R, 1,
       COS(135)*R, SIN(135)*R, 1,
       COS(157.5)*R, SIN(157.5)*R, 1,
       COS(180)*R, SIN(180)*R, 1,
       COS(202.5)*R, SIN(202.5)*R, 1,
       COS(225)*R, SIN(225)*R, 1,
       COS(247.5)*R, SIN(247.5)*R, 1,
       COS(270)*R, SIN(270)*R, 1,
```

```
COS(292.5)*R, SIN(292.5)*R, 1,
COS(315)*R, SIN(315)*R, 1,
COS(337.5)*R, SIN(337.5)*R, 1,
COS(360)*R, SIN(360)*R, 1,
COS(22.5)*R, SIN(22.5)*R, 1,
COS(45)*R, SIN(45)*R, 1,
COS(67.5)*R, SIN(67.5)*R, 1,
COS(90)*R, SIN(90)*R, 1
```

SWEEP

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

ポリラインの空間曲線パスに沿ったポリラインの掃引によって生成される表面。

ポリラインの平面は、パスの曲線に従います。空間曲線は X-Y 平面から始まる必要があります。そうでない場合、X-Y 平面から始まるように Z 軸方向に移動されます。

点（xi, yi, zi）の横断面は、点（xi-1, yi-1, zi-1）と（xi,yi, zi）の間の空間曲線線分に垂直です。

SWEEP は、急須の注ぎ口などの複雑な形状を作るときに使用できます。

n: ポリラインの節点の数

m: パスの節点の数

alpha: ポリラインの平面上での、パスのある節点から次の節点への増分回転

scale: パスのある節点から次の節点へのポリラインの増分スケール係数

mask: 下面および上面のポリゴンの表面と辺の可視性を制御

ui, vi: 基準ポリラインの節点の座標

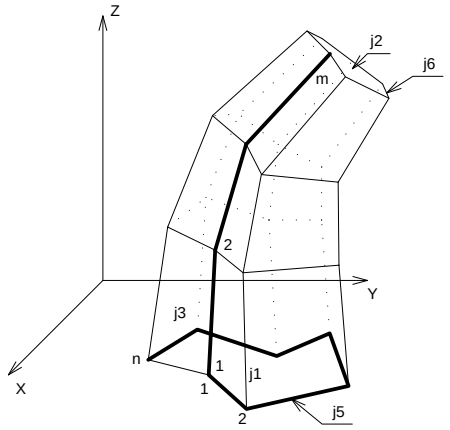
si: 側面の辺のステータス

xi, yi, zi: パス曲線の節点の座標

パラメータの制限：

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

マスキング：



mask = j1 + 2*j2 + 4*j3 + 16*j5 + 32*j6 + 64*j7　ここで、j1, j2, j3, j5, j6, j7 は 0 または 1 をとります。

j1（1）: 基準面を表現

j2（2）: 上面を表現

j3（4）: 側面を表現

j5（16）: 基準面の辺は可視

j6（32）: 上面の辺は可視

j7（64）: 横断面の辺は可視、表面は明確

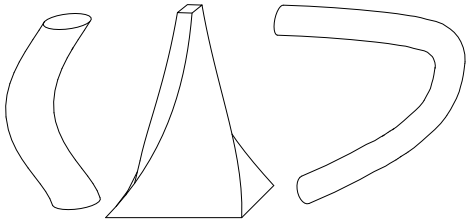
ステータス値：

0: 節点から延びる側面の辺は全て可視

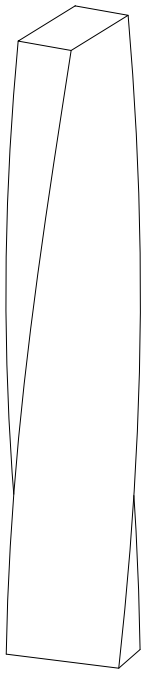
1: 節点から延びる側面の辺を輪郭の表示に使用

追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。
詳細は、183 ページ「追加ステータスコード」を参照してください。

例：



```
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

生成する横断面を歪めることなく、空間曲線パスに沿ってポリラインを掃引することによって生成される表面。内部の接続表面は、一時的な U-V-W 座標系の U-W 平面内で回転することができます。

V 軸： 対応する点における生成元の曲線の正接方向

W 軸： V 軸に垂直で、ローカル座標系の Z 軸を基準とした上方向

U 軸： V 軸および W 軸に垂直で、これらの軸と右手直交座標系を形成します。

V 軸が垂直の場合、W 軸の方向は正しく定義されません。この場合、水平方向を決めるためにパス上の前の節点での W 軸が使用されます。

パス線分で切り取られたチューブの横断面のポリゴンは、常に基準ポリゴンと等しくなります（u1, w1、... un, wn)。接合箇所の横断面ポリゴンは、接合箇所線分の 2 等分面になります。基準ポリゴンは、閉じていなければなりません。

n: ポリラインの節点の数

m: パスの節点の数

ui, wi: 基準ポリラインの節点の座標

si: 側面の辺のステータス

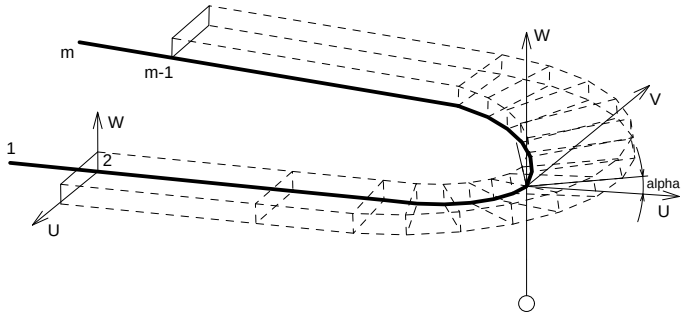
xi, yi, zi: パス曲線の節点の座標

注記： パスは、生成された横断面の数より 2 つ多い点で構成されます。最初の点と最後の点は、TUBE に含まれる最初と最後の表面の空間内の位置を決定します。これらの点は表面の法線を決定するだけで、パスの実際の節点ではありません。これらの点によって示される方向に TUBE がつながっている場合、表面の方向は、この 2 つの端点に最も近い節点で生成される表面の方向と一致します。

anglei: 横断面の回転角

マスキング：

mask = j1 + 2*j2 + 16*j5 + 32*j6 + 64128*j7　ここで、j1, j2, j3, j5, j6, j7 は 0 または 1 をとります。



j1（1）: 基準面を表現

j2（2）: 端の表面を表現

j5（16）: 基準面の辺（座標 x2, y2, z2）は可視

j6（32）: 最後の辺（座標 xm-1, ym-1, zm-1）は可視

j7（64）: 横断面の辺は可視、表面は明確

j8（128）: 横断面の辺はシャープで、表面のスムージングは OpenGL とレンダリングで停止

パラメータの制限：

n > 2 および m > 3

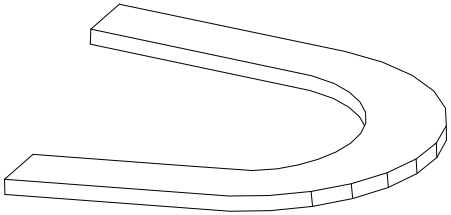
ステータス値：

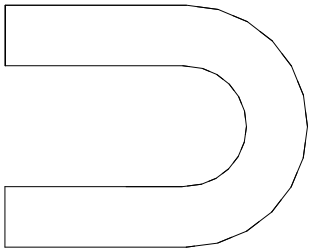
0: 節点から延びる側面の辺は全て可視

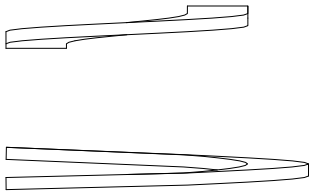
1: 節点から延びる側面の辺を輪郭の表示に使用

追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。*『ArchiCAD ヘルプ』の 「バーチャルビルディング」章の「パラメトリックオブジェクト」を参照してください。*

例：

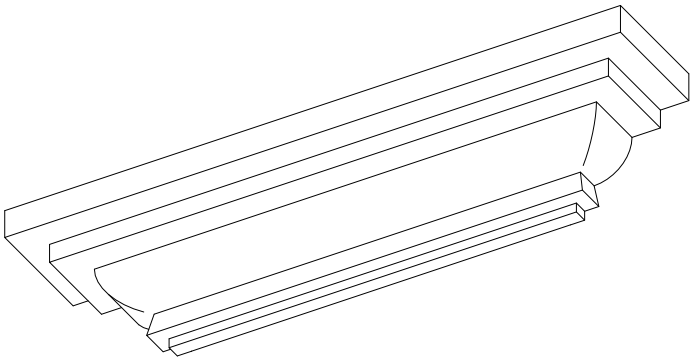




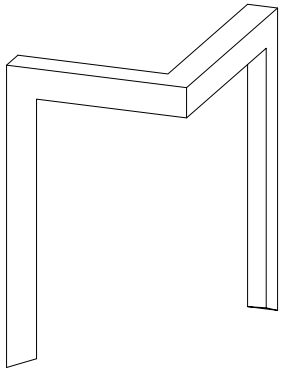


```
TUBE 4, 18, 16+32,
          2.0, 0.0, 0,
          0.0, 0.0, 0,
          0.0, 0.4, 0,
          2.0, 0.4, 0,
          -1, 0, 0, 0,
          0, 0, 0, 0,
          4, 0, 0.1, 0,
          6, 0, 0.15, 0,
   6+4*SIN(15), 4 - 4*COS(15), 0.2, 0,
   6+4*SIN(30), 4 - 4*COS(30), 0.25, 0,
   6+4*SIN(45), 4 - 4*COS(45), 0.3, 0,
   6+4*SIN(60), 4 - 4*COS(60), 0.35, 0,
   6+4*SIN(75), 4 - 4*COS(75), 0.4, 0,
   10, 4, 0.45, 0,
   6+4*SIN(105), 4 - 4*COS(105), 0.5, 0,
   6+4*SIN(120), 4 - 4*COS(120), 0.55, 0,
   6+4*SIN(135), 4 - 4*COS(135), 0.6, 0,
   6+4*SIN(150), 4 - 4*COS(150), 0.65, 0,
   6+4*SIN(165), 4 - 4*cos(165), 0.7, 0,
   6, 8, 0.75, 0,
   0, 8, 1, 0,
   -1, 8, 1, 0
TUBE  14, 6, 1+2+16+32,
      0, 0,0,
      0.03, 0,0,
      0.03, 0.02, 0,
      0.06, 0.02, 0,
      0.05, 0.0699, 0,
      0.05, 0.07, 1,
      0.05, 0.15, 901,
      1, 0, 801,
```

```
        0.08,  90,  2000,
        0.19,  0.15,  0,
        0.19,  0.19,  0,
        0.25,  0.19,  0,
        0.25,  0.25,  0,
        0,  0.25,  0,
        0,  1,  0,  0,
        0,  0.0001,  0,  0,
        0,  0,  0,  0,
        -0.8,  0,  0,  0,
        -0.8,  0.0001,  0,  0,
        -0.8,  1,  0,  0
```

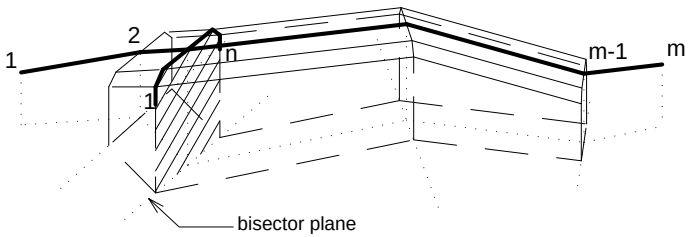


```
TUBE  3,  7,  16+32,
        0,  0,  0,
        -0.5,  0,  0,
        0,  0.5,  0,
          0.2,  0,  -0.2,  0,
        0,  0,  0,  0,
        0,  0,  5,  0,
        3,  0,  5,  0,
        3,  4,  5,  0,
        3,  4,  0,  0,
        3,  3.8,  -0.2,  0
```

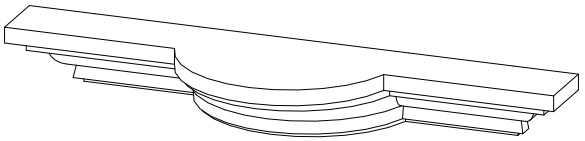


TUBEA

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

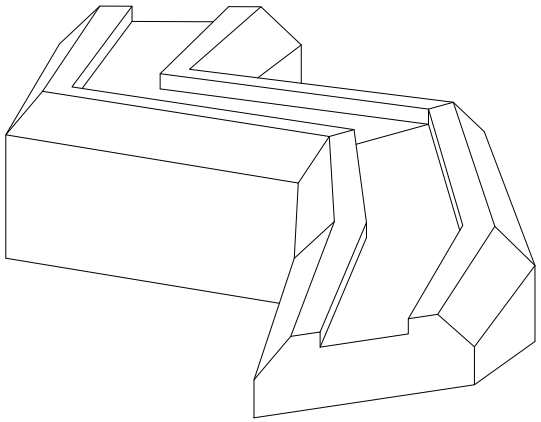


TUBEA は、TUBE ステートメントとは別のアルゴリズムで、空間曲線パスに沿ってポリラインを掃引することで生成される表面です。
パス曲線の各接合箇所で生成される横断面ポリゴンは、基準ポリゴン（u1, w1、... un, wn）と等しく、接合箇所線分のローカル座標系の X-Y 平面への投影の 2 等分平面に位置付けられます。基準面が開いている可能性がある： この場合、横断面ポリゴンは、REVOLVE 表面の場合と同じように、ローカル X-Y 平面に届くように生成されます。
パス線分で切り取られたチューブの横断面は、基準ポリゴンと異なる場合があります。
追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。
詳細は、183 ページ「追加ステータスコード」を参照してください。
例：



```
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
```

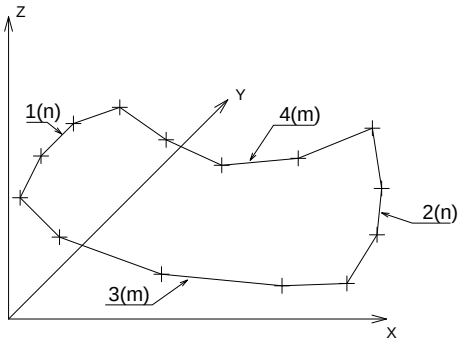


COONS

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

COONS は 4 本の境界曲線から生成されます。

マスキング：



mask = 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7　ここで、j3, j5, j6, j7 は 0 または 1 をとります。

j3 (4): 最初の境界 (x1, y1, z1) の辺は可視

j4 (8): 2 番目の境界 (x2, y2, z2) の辺は可視

j5 (16): 3 番目の境界 (x3, y3, z3) の辺は可視

j6 (32): 4 番目の境界 (x4, y4, z4) の辺は可視

j7 (64): 表面上の辺は可視、表面はスムージングされない

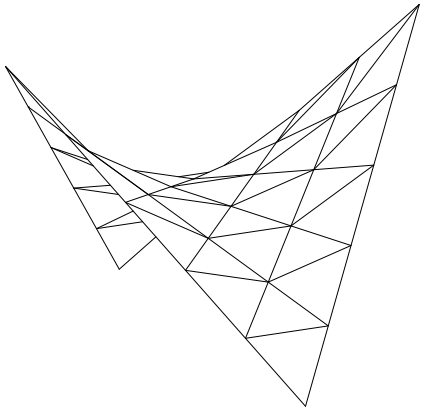
パラメータの制限：

n, m > 1

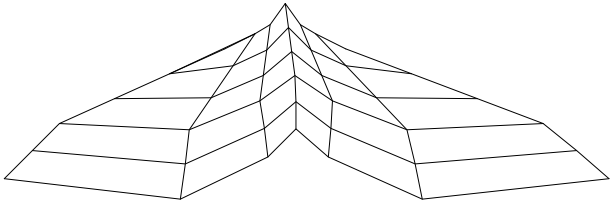
例：

```
COONS  6, 6, 4+8+16+32+64,
       ! 1 番目の境界 , n=6
       0, 0, 5,
       1, 0, 4,
       2, 0, 3,
       3, 0, 2,
```

```
        4,  0,  1,
        5,  0,  0,
        ! 2 番目の境界 , n=6
        0,  5,  0,
        1,  5,  1,
        2,  5,  2,
        3,  5,  3,
        4,  5,  4,
        5,  5,  5,
        ! 3 番目の境界 , m=6
        0,  0,  5,
        0,  1,  4,
        0,  2,  3,
        0,  3,  2,
        0,  4,  1,
        0,  5,  0,
        ! 4 番目の境界 , m=6
        5,  0,  0,
        5,  1,  1,
        5,  2,  2,
        5,  3,  3,
        5,  4,  4,
        5,  5,  5
ROTZ  -90
ROTY  90
```



```
ROTZ
ROTY
COONS 7, 6, 4+8+16+32+64,
      ! 1 番目の境界 , 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 番目の境界 , 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 番目の境界 , m=6
      1, 2, 0,
      2, 4, -0.5,
      3, 6, -1,
      4, 8, -1.5,
      5, 9, -1.8,
      6, 10, -2,
      ! 4 番目の境界 , m=6
      1, -2, 0,
      2, -4, -0.5,
      3, -6, -1,
      4, -8, -1.5,
      5, -9, -1.8,
      6, -10, -2
```



MASS_

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

ArchiCAD のメッシュツールで生成される形状と同等のもの。

top_material, bottom_material, side_material: 上面、下面および側面の材質の名前またはインデックス

n: マスポリゴンに含まれる節点の数

m: 隆起線上の節点数

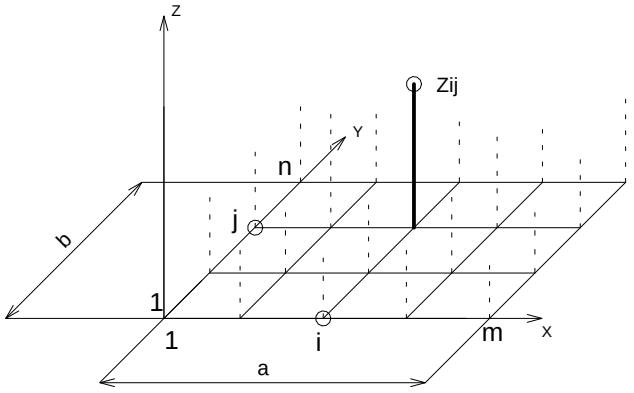
h: へりの高さ（負数にできます）

xi, yi, zi: 節点の座標

si: PRISM_ ステートメントと同様。追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。

詳細は、183 ページ「追加ステータスコード」を参照してください。

マスキング：



mask = j1 + 4*j3 + 16*j5 + 32*j6 + 64*j7　ここで、j1、j3、j5、j6、j7 は 0 または 1 をとります。

j1 (1): 基準面を表現

j3 (4): 側面を表現

j5 (16): 基準面および側面の辺は可視

j6 (32): 上面の辺は可視

j7 (64): 上面の辺は可視、表面はスムージングされない

j8 (128): 全ての隆起線はシャープで、表面はスムージングされる

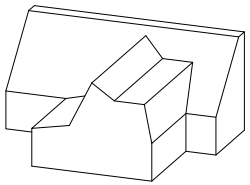
パラメータの制限：

n >=3, m >= 0

例：

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

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

```
19,  17,  5,  0,
19,  0,   5,  -1
```



ビジュアル化のための要素

LIGHT

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

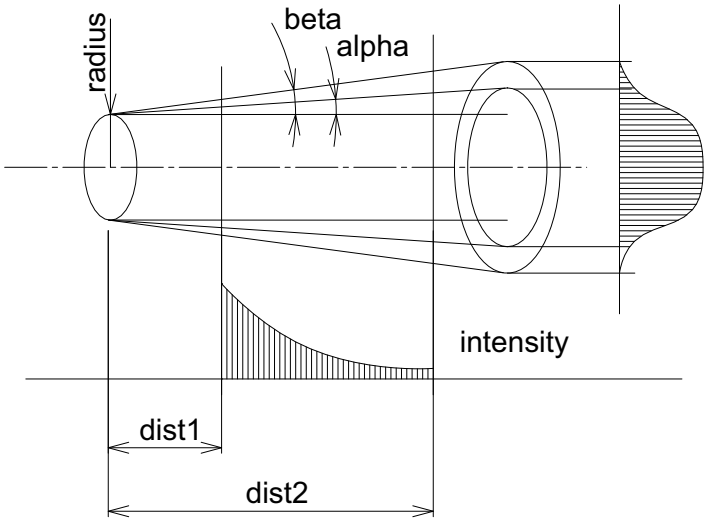
光源は、ローカル座標系の原点から X 軸方向にカラー（赤、緑、青）の付いた光を放射します。光は、点または円形の光源から X 軸に平行に投射されます。光は、alpha 角度の円錐台内では最大輝度を保ち、beta 角度の円錐台でゼロになります。減衰は、パラメータ angle_falloff で制御します （ゼロを指定すると光の境界が鮮明になり、高い値にすると減衰が滑らかになります)。 光の効果は、軸に沿って、distance1 と distance2 の値によって制限されます。パラメータ distance_falloff は、距離によって変化する輝度の減衰を制御します （ゼロを指定すると一定の輝度となり、大きな値を指定すると減衰が激しくなります)。

GDL の変換は、光の開始点と方向のみに影響します。

パラメータ shadow は、光源のシャドウ投射を制御します。

0: シャドウ投射なし

1: シャドウ投射あり



パラメータの制限：

alpha <= beta <= 80°

次のようなパラメータの組合せには、特別な意味があります。

radius = 0, alpha = 0, beta = 0

これは点光源で、全ての方向に光を照射しますが、シャドウ投射は行いません。shadow と angle_falloff のパラメータは無視され、shadow = 0、angle_falloff = 0 と解釈されます。

radius > 0, alpha = 0, beta = 0

指向性のある光

光の定義では、キーワード

ADDITIONAL_DATA の後に省略可能な追加のデータ定義を含めることができます。追加データには、名前（namei）と値（valuei）があり、これはいずれかのタイプの式にすることができます。また、配列にすることもできます。パラメータ名がサブ文字列「_file」で終わっている場合、その値はファイル名とみなされてアーカイブプロジェクトに含められます。

ArchiCAD または ArchiCAD のアドオンで、さまざまな意味の追加データを定義して使用することができます。

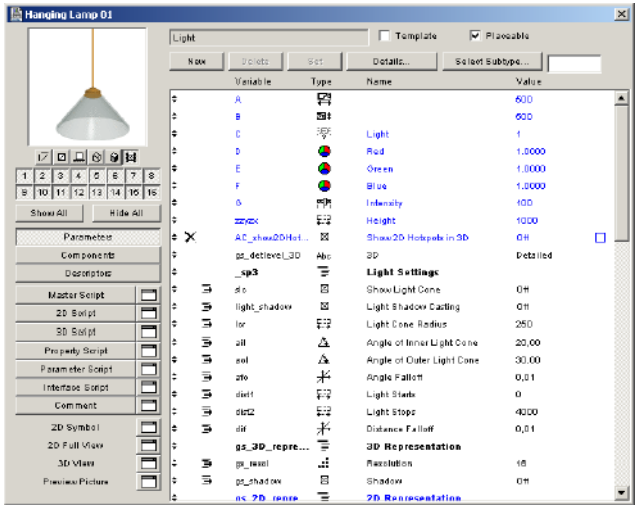
LightWorks アドオンパラメータの意味については、
http://www.graphisoft.com/support/developer/documentation/LibraryDevDoc/11 を参照してください。

例：

```
LIGHT 1.0,0.2,0.3,      ! RGB
      1,                  ! シャドウ
      1.0,                ! 半径
```

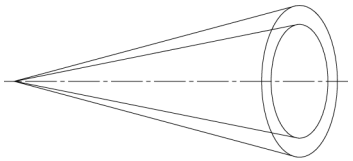
```
45.0,60.0,          ! angle1, angle2
0.3,                 ! angle_falloff
1.0,10.0,           ! distance1, distance2
0.2                    ! distance_falloff
```

ArchiCAD の光に関するライブラリ部品ダイアログボックス

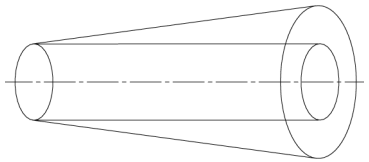


対応する GDL スクリプトの一部：

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



r = 0, alpha > 0, beta > 0



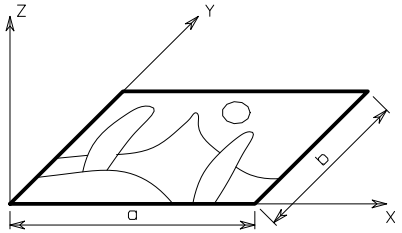
r > 0, alpha = 0, beta > 0

r > 0, alpha = 0, beta = 0
さまざまな alpha、beta パラメータによる光のタイプ

PICTURE

PICTURE expression, a, b, mask

レンダリング用の画像要素



文字列タイプの expression は、ライブラリ部品に格納されている画像のファイル名、数式、またはインデックスを意味します。0 のインデックスは特殊な値です。この値は、ライブラリ部品のプレビュー画像を参照します。ほかの画像は、画像を GDL オブジェクトとして含むプロジェクトまたは選択した要素を保存するときに、ライブラリ部品に格納できるだけです。
インデックス付きの画像参照は、属性が現在の属性セットに結合されているときには、MASTER_GDL スクリプトでは使用できません。画像は、ほかのいずれかの 3D 投影方法で RECT として扱われる矩形にはめ込まれます。

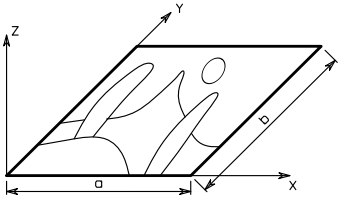
mask = alpha + distortion

alpha: アルファチャンネルの制御

0: アルファチャンネルは使わず、画像は矩形

1: アルファチャンネルを使い、画像の一部は透過も可

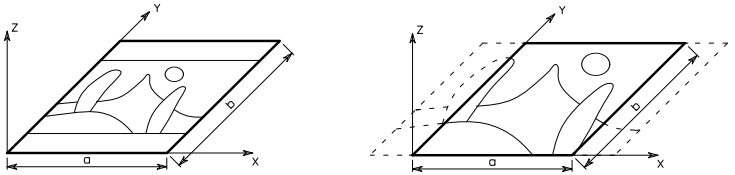
distortion: 歪みの制御



0: 画像を与えられた矩形に収める

2: 画像の自然な縦横比を保ちつつ、矩形の中央に収める

4: 画像の自然な縦横比を保ちつつ、矩形の中央に画像を配置



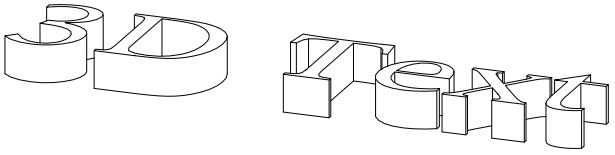
3D テキスト要素

TEXT

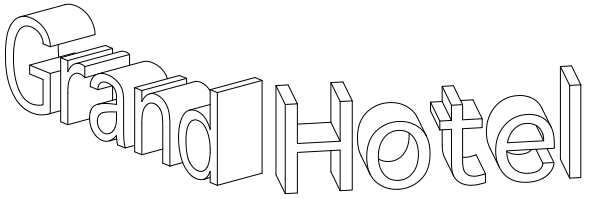
TEXT d, 0, expression

文字列タイプまたは数値タイプの式の値の現在のスタイルでの 3D 表現。
199 ページ「[SET] STYLE」および 219 ページ「DEFINE STYLE」を参照してください。
d: 文字の厚さ（メートル単位）

GDL の現在のバージョンでは、2 番目のパラメータは常にゼロです。
例：



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




```
name = "Grand"
ROTX 90
ROTY -30
TEXT 0.003, 0, name
```

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

注記： 2D GDL スクリプトとの互換性を保つため、DEFINE STYLE ステートメントでは、文字の高さは常にミリメートルで解釈されます。

RICHTEXT

RICHTEXT x, y,
height, 0, textblock_name

あらかじめ定義された TEXTBLOCK の 3D 表現。詳細は、*221 ページ「テキストブロック」*を参照してください。

x, y: リッチテキストの位置の X-Y 座標。

height: メートル単位の文字の厚さ

textblock_name: 前もって定義された TEXTBLOC の名前

GDL の現在のバージョンでは、4 番目のパラメータは常にゼロです。

プリミティブ要素

3D データ構造のプリミティブは、VERT、VECT、EDGE、PGON、BODY です。ボディは複数の表面と、それらの接続によって表現されます。3D 切断を実行するための情報は、この接続情報から導きます。

インデックスは 1 から始まり、BASE ステートメントまたは新規のボディ（暗黙的な BASE ステートメント）は 1 にリセットされます。各辺に、隣接したポリゴンのインデックス（最大 2）が格納されます。辺の方向は、1 番目と 2 番目の 2 つの頂点によって定義されます。

ポリゴンは、辺のインデックスを含む、方向を持った辺のリストです。これらの数値は、負の接頭語を持つことができます。この場合、与えられた辺は逆方向に使用されます。ポリゴンには穴を含めることができます。辺のリストで、ゼロのインデックスは新規の穴を意味します。穴は、ほかの穴を含むことはできません。1 つの辺は、0 から 2 つのポリゴンに属することができます。閉じたボディの場合、2 つのポリゴンの辺リストにおいて辺の接頭語が異なっていれば、ポリゴンの方向は適切です。

ポリゴンの法線ベクトルは、別々に保存されます。閉じたボディの場合、法線ベクトルはボディの内側から外側を指します。外側から見た場合は、辺リストの方向は反時計回り（数学的な正の方向）です。穴の方向は、親であるポリゴンの方向の逆です。開いているボディの法線ベクトルは、ボディの同じ側を指し示さなければなりません。

ボディの内側と外側を決定するためには、ボディは閉じている必要があります。閉じたボディは、それぞれの辺に 2 つの正確に隣接したポリゴンがあるものと定義されます。

開いているボディに対しては、切り取り、隠線処理、レンダリングのアルゴリズムの能率は悪くなります。正規パラメータを持つ各複合 3 次元要素は、内部の 3D データ構造においては閉じられたボディになっています。

輪郭線の検索は、辺のステータスビットと辺と、隣接するポリゴンを基にしています。これは複合曲面要素の場合は自動的に設定されますが、プリミティブ要素の場合、これらのビットを正しく設定するのはユーザーです。

簡略化された定義（PGON で vect = 0 または status < 0）の場合、他から参照されるプリミティブは、その前に定義しておく必要があります。この場合、推奨する順序は次のとおりです。

VERT (TEVE)
EDGE
(VECT)
PGON (PIPG)
COOR
BODY[BODY]

隣接するポリゴンの辺による検索は、BODY ステートメントが実行されている間に行なわれます。

VERT、EDGE、VECT、PGON の番号付けは、最後の（明示的または暗黙的な）BASE ステートメントが基準になります。

ステータス値は、プリミティブについての特殊な情報を保存するために使用されます。ステータスの各ビットがそれぞれ独立した意味を持っていますが、いくつかの例外もあります。

与えられた値は、お互いに加算することができます。次に示すビットの組み合わせ以外は、内部使用のために予約されています。これらのステータスのデフォルト値はゼロです。

VERT

VERT x, y, z

3 つの座標によって定義された、X-Y-Z 空間内の節点。

TEVE

TEVE x, y, z, u, v

テクスチャの座標定義を含む、VERT ステートメントの拡張版。自動テクスチャラッピングの代わりにユーザー定義のテクスチャ座標が必要な場合は、VERT ステートメントの代わりに使用することができます（*127 ページ「COOR」*のステートメントを参照）。

x, y, z: 節点の座標

u, v: 節点のテクスチャ座標

現在のボディの各頂点の（u, v）座標を指定しなければなりません。それぞれの頂点にはテクスチャ座標を 1 つだけ指定できます。ボディの定義に VERT ステートメントと TEVE ステートメントが混在していると、(u, v) 座標が無効になります。

注記： (u, v) テクスチャ座標が有効になるのはレンダリング時だけです。ベクトル塗りつぶしマッピングには使用できません。

VECT

VECT x, y, z:

3 つの座標によるポリゴンの法線ベクトルの定義。簡略化された定義（PGON で vect = 0）の場合、これらのステートメントは省略することができます。

EDGE

EDGE vert1, vert2, pgon1, pgon2, status

辺の定義。

vert1、vert2: 終端のインデックス。vert1 と vert2 のインデックスは、異なっている必要があり、前もって定義されている VERT を参照します。

pgon1、pgon2: 隣接するポリゴンのインデックス。ゼロと負の値は、次のような特別な意味があります。

0: 最も外側か、単独の辺

<0: 有効な隣接するポリゴンを検索

ステータスビット:

1: 非表示の辺

2: 曲面の辺

将来の使用のために予約されているステータスビット:

4: 曲面の最初の辺（2 と共にのみ使用）

8: 曲面の最後の辺（2 と共にのみ使用）

16: 円弧線分の辺

32: 円弧の最初の線分（16 と共にのみ使用）

62: 円弧の最後の線分（16 と共にのみ使用）

PGON

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

ポリゴンの定義。

n: 辺リスト内の辺の数

vect: 法線ベクトルのインデックス。前もって定義された VECT を参照する必要があります。

注記: vect = 0 の場合、解析中に法線ベクトルが計算されます。

インデックス edge1、edge2、... edgen は、定義済みの EDGE を参照する必要があります。ゼロの値は、穴の定義の開始または終了を表します。

負のインデックスは、格納されている法線ベクトルまたは辺の方向をポリゴン内で逆に変更します （格納されているベクトルまたは辺そのものは変更されません。ほかのポリゴンは、正のインデックスを持つオリジナルの方向を使用して、これを参照することができます)。

ステータスビット：
1: 非表示のポリゴン

2: 曲面のポリゴン

16: 凹形ポリゴン

32: 穴の開いたポリゴン

64: 凸状の穴 (32 と共にのみ使用)

将来の使用のために予約されているステータスビット：
4: 曲面の最初のポリゴン (2 と共にのみ使用)

8: 曲面の最後のポリゴン (2 と共にのみ使用)

ステータス値が負の場合、インタプリタエンジンはポリゴンのステータスを計算します (凹形のポリゴンや穴の開いたポリゴンのように)。
n = 0 は、特定の目的のために使用します。

PIPG

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

画像ポリゴンの定義。最初の 4 つのパラメータは、PICTURE 要素と同じで、残りのパラメータは PGON 要素と同じです。

COOR

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

塗りつぶしおよびテクスチャのマッピングのための BODY のローカル座標系。
wrap: ラッピングモード + 投影タイプ

ラッピングモード：
1: 平面状

2: ボックス状

3: 円柱状

4: 球体状

5: 円柱状塗りつぶしマッピングと同じ、ただし上面と下面のレンダリングでは円状マッピング

投影タイプ:

256: 塗りつぶしは常にローカル座標系の原点から開始

1024: 二次テクスチャ投影（推奨）

2048: 平均距離に基づく線形テクスチャ投影

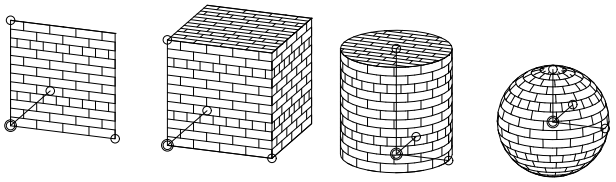
2096: 通常の三角形分割に基づく線形テクスチャ投影

注記: 最後の 3 つの値が有効になるのは、カスタムテクスチャ座標定義の場合だけです（*124 ページ「TEVE」を参照してください*）。

vert1: ローカル座標系の原点を表す VERT のインデックス

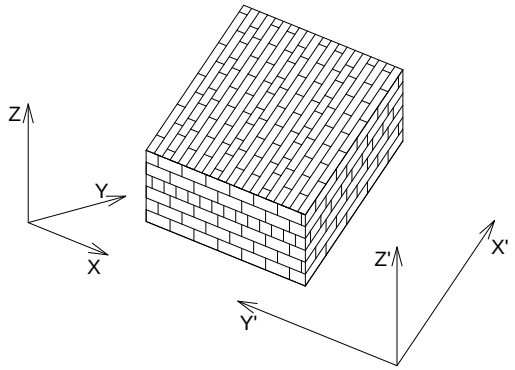
vert2、vert3、vert4: 3 つの座標軸を定義する VERT のインデックス

ローカル座標系の定義のみに使う場合は、VERT のインデックスの前に負の符号（-）を付けてください。



カスタムテクスチャ軸の例：

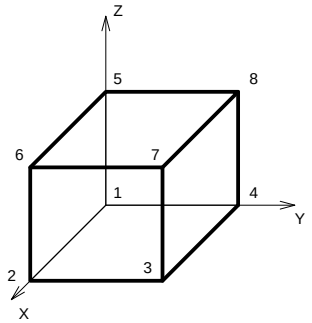
```
CSLAB_ "Face brick", "Face brick", "Face brick",
          4, 0.5,
          0, 0, 0, 15,
          1, 0, 0, 15,
          1, 1, 1, 15,
          0, 1, 1, 15
BASE
VERT      1, 0, 0                    !#1
VERT      1, 1, 1                    !#2
VERT      0, 0, 0                    !#3
VERT      1, 0, 1                    !#4
COOR      2, -1, -2,-3, -4
BODY      1
```



BODY

BODY ステータス

前記のプリミティブによって定義されたボディを作成します。
ステータスビット：
1：閉じたボディ
2：曲面（1 つ以上）を含むボディ
4：表面モデル。ボディを切り取った場合、切断面には面は存在しない
32：前に選択されたアルゴリズムとは無関係に、ボディは常にシャドウ投射を行う
62：ボディはシャドウ投射を行わない
32 と 64 のどちらも設定されていないと、事前に選択された自動シャドウ投射が実行されます。
202 ページ「SHADOW」を参照してください。
ステータス値が負の場合、インタプリタエンジンはボディのステータスを計算します。
例：



1：詳細説明

```
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: （ポリゴンまたはベクトルへの直接参照はなく、計算されます）

```
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

低レベルの図形要素（VERT、TEVE、VECT、EDGE、PGON、PIPG）ステートメントのカウンターをリセットします。各複合要素定義の後に暗黙的に発行されます。

3D での切り取り

CUTPLANE

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

または

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

切断面を作成し、閉じた形状の切り取られた部分を削除します。CUTPLANE は、異なる数のパラメータを持つことができます。

CUTPLANE のパラメータの数とその意味は、次のとおりです。

0: X-Y 平面

1: 切断面は X 軸を横切り、angle は切断面と X-Y 平面の角度です。

2: 切断面は Z 軸と平行で、与えられた値で X 軸と Y 軸と交差します。

3: 切断面は与えられた値で、X 軸、Y 軸、Z 軸と交差します。

4: 最初の 3 つのパラメータは上記と同じで、その他に次のような side 値があります。

side = 0: 切断面より上の部分を削除（デフォルト）；
side = 1: 切断面より下の部分を削除； x-y、x-z、y-z の場合は、軸の負の方向にある部分を削除

side パラメータを指定しないと、切断面より上の部分が削除されます。最初の 3 つのパラメータが X-Y、X-Z、または Y-Z 平面を定義する場合（例えば、1.0, 1.0, 0.0 は X-Y 平面を定義）、3 番目の軸の正の方向にある部分が削除されます。

CUTPLANE と CUTEND の間にはステートメントをいくつでも追加できます。また、マクロに CUTPLANE を含めることもできます。

CUTPLANE のパラメータは、現在の座標系を参照します。

CUTPLANE と CUTEND の間の変換はまさにこの切断面上では無効ですが、連続する CUTPLANE は変換されます。そのため、必要数の変換を使って CUTPLANE を設定し、次にこれらの変換を削除してから、切り取る形状を定義することをお勧めします。

CUTPLANE-CUTEND の対は、ネストさせることができ、ループ内でも使用できます。最後に CUTEND を指定しないと、有効になっている CUTPLANE は、スクリプトの最後まで全ての形状で有効になります。

マクロ内の CUTPLANE は、CUTEND がなくてもマクロ内の形状だけに有効です。

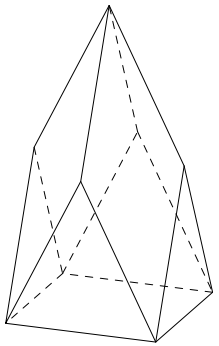
CUTPLANE と CUTEND の間でマクロがコールされた場合、マクロ内の形状が切り取られます。

現在の材質、ペン、および塗りつぶしの設定は、断面で有効です。

注記： CUTPLANE を CUTEND で終了しないと、最悪の場合、全ての形状が完全に削除されます。そのため、CUTEND が欠落していると必ず警告メッセージが出ます。

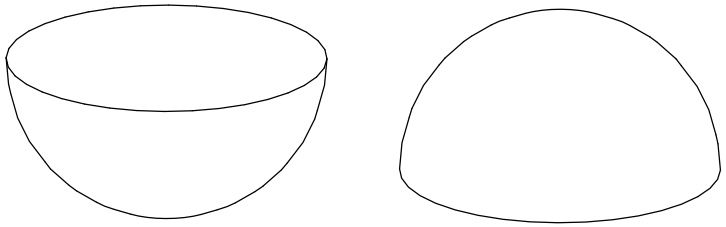
CUTPLANE の位置を指定するためだけに行った変換を削除しないと、実際に形状を移動した場合、CUTPLANE が間違った位置に指定されているように表示されます。

例：

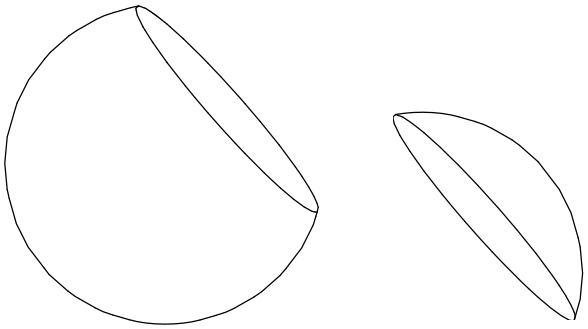


```
CUTPLANE 2, 2, 4
CUTPLANE -2, 2, 4
CUTPLANE -2, -2, 4
```

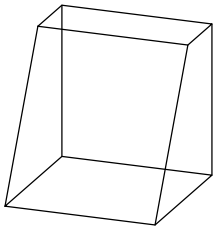
```
CUTPLANE 2, -2, 4
     ADD -1, -1, 0
     BRICK 2, 2, 4
     DEL 1
CUTEND
CUTEND
CUTEND
CUTEND
```

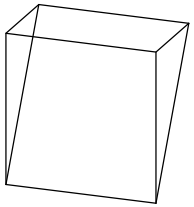


```
CUTPLANE                    CUTPLANE 1, 1, 0, 1
   SPHERE 2                   SPHERE 2
CUTEND                        CUTEND
```



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





```
CUTPLANE 60                CUTPLANE -120
   BRICK 2, 2, 2              BRICK 2, 2, 2
CUTEND                        CUTEND
```

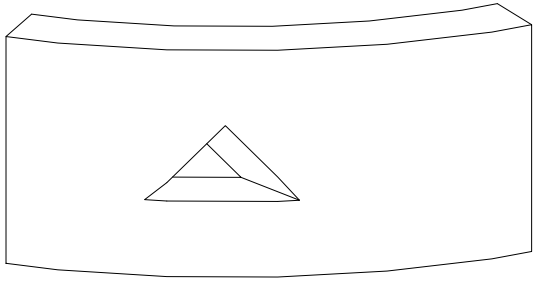
CUTPOLY

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

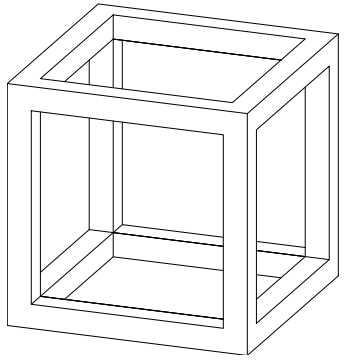
CUTPLANE コマンドと同じように、CUTPOLY のパラメータも、現在の座標系を参照します。ポリゴンは、自己交差することはできません。切り取り方向は Z 軸ですが、省略可能な（x, y, z）ベクトルを指定することもできます。

このパラメータは、無限の「管」を定義します。ポリゴンは管の横断面で、切り取り方向は管の方向です。管の内側のあらゆる要素が削除されます。

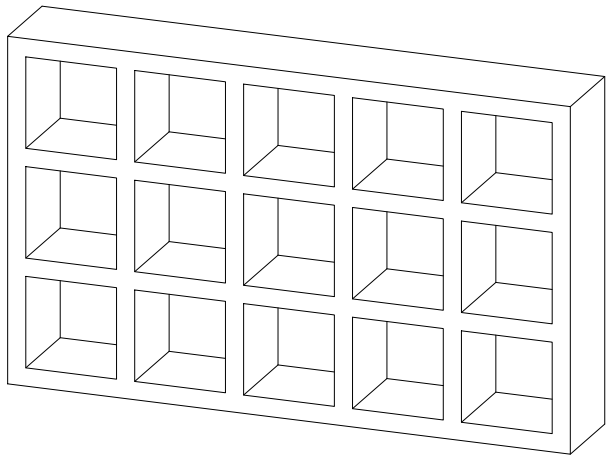
例：



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



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



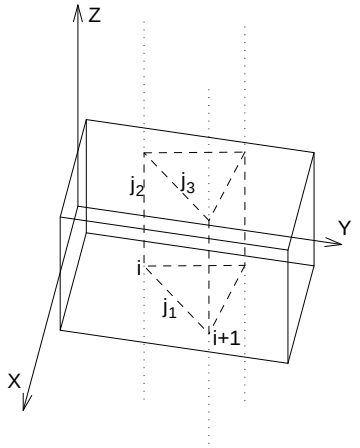
```
ROTX 90
FOR I=1 TO 3
    FOR J=1 TO 5
        CUTPOLY 4,
        0, 0, 1, 0,
        1, 1, 0, 1
        ADDX 1.2
    NEXT J
    DEL 5
    ADDY 1.2
NEXT I
DEL NTR()-1
ADD -0.2, -0.2, 0
BRICK 6.2, 3.8, 1
FOR K=1 TO 15
    CUTEND
NEXT K
DEL TOP
```

CUTPOLYA

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

CUTEND

CUTPOLY 定義と似ていますが、生成されたポリゴンの辺の可視性を制御することができます。切り取り形式は、ポリゴンの横断面が定義されている半無限チューブです。切り取り形式の最後がボディの中に垂れ下がっていると、対応する領域が切り抜かれます。



status: 生成された切り取りポリゴンの扱いを制御します。

1: 生成されたポリゴンと辺にボディの属性を使用

2: 生成された切り取りポリゴンは正規ポリゴンとして扱う

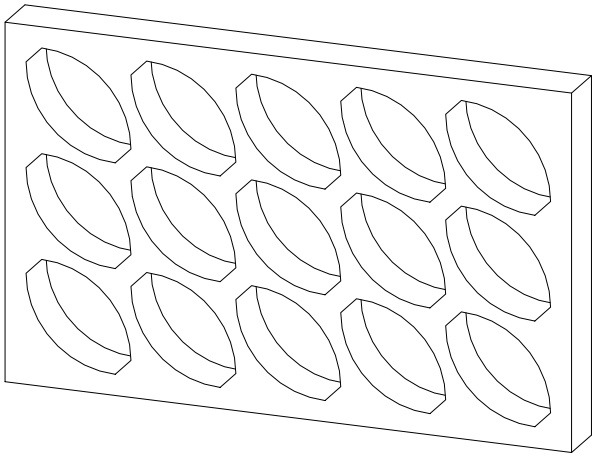
d: ローカルの原点と半無限チューブの距離

d = 0 の場合、無限チューブで切り取られます。

maski: PRISM_ ステートメントと同様。

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

例：

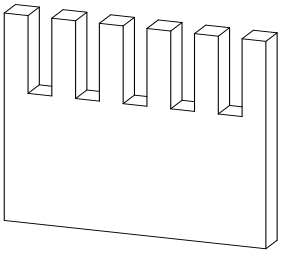


```
ROTX 90
FOR I=1 TO 3
    FOR J=1 TO 5
        CUTPOLYA 6, 1, 0,
                   1, 0.15, 5,
                   0.15, 0.15, 900,
                   0, 90, 4007,
                   0, 0.85, 5,
                   0.85, 0.85, 900,
                   0, 90, 4007
        ADDX 1
    NEXT J
    DEL 5
    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
status: 生成された切り取りポリゴンの扱いを制御します。指定がなければ（互換性を確保するため)、デフォルト値は 3 です。
status= j1 + 2*j2
j1: 生成されたポリゴンと辺にボディの属性を使用
j2: 生成された切り取りポリゴンは、通常のポリゴンとして取り扱われる



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

CUTFORM

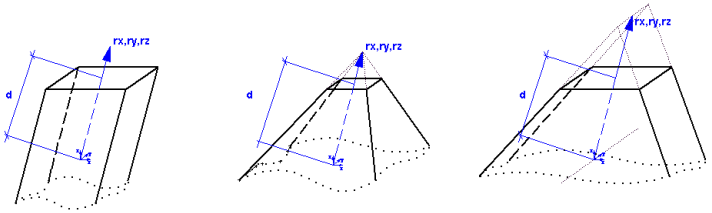
CUTFORM n, method, status,
rx, ry, rz, d,
x1, y1, mask1,
...
xn, yn, maskn

CUTPOLYA 定義と似ていますが、切り取りボディの形式と範囲を制御することができます。
method: 切り取りボディの形式を制御します。

1: 角柱形状

2: 角錐形状

3: ウエッジ形状の切り取りボディ。ウエッジ上部エッジの方向は Y 軸に平行で、その位置は rx, ry, rz （ry は無視）。



status: 切り取りボディの範囲と生成された切り取りポリゴンおよび新規の辺の取り扱いを制御します。

status = j1 + 2*j2 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8
j1: 生成されたポリゴンと辺にボディの属性を使用

j2: 生成された切り取りポリゴンは正規ポリゴンとして扱う

j4, j5: 切り取りの制限を定義します：
j4 = 0 および j5 = 0: 有限切り取り、
j4 = 0 および j5 = 1: 半無限切り取り、
j4 = 1 および j5 = 1: 無限切り取り
j6: 切り取りボディを使用して、ブール差ではなくブール積を生成 （CUTFORM コマンドでのみ使用可）。

j7 ： 切り取りボディの下部で生成される辺は表示されません。

j8 ： 切り取りボディの上部で生成される辺は表示されません。

rx, ry, rz: 切り取り形式が角柱形状の場合は切り取り方向を、切り取り形式が角錐状の場合は角錐の上端を定義します。

d: 切り取りの終わりまでの rx,ry,rz に沿った距離を定義します。切り取りが無限の場合、このパラメータは無効です。切り取りが有限の場合、切り取りボディの始点はローカル座標系になり、ボディは rx,ry,rz で定義された方向に沿った距離 d で終わります。

切り取りが半無限の場合、切り取りボディは rx, ry, rz で定義された方向に沿った d の距離から始まり、半無限切り取りの方向は rx, ry, rz で定義された方向とは逆になります。

mask: 切り取りボディの辺の可視性を定義します。

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

j1: ポリゴンは、切り取られるボディに入り込むときに可視の辺を作成

j2: 切り取り形式の縦方向の辺は可視

j3: ポリゴンは切り取られるボディから出るときに可視の辺を作成

j4: 切り取り形式の下辺は可視

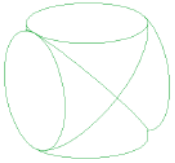
j5: 切り取り形式の上辺は可視

j7: 縦方向の辺の視点に依存する可視性を制御

ソリッド図形コマンド

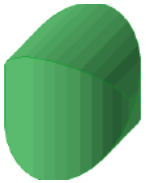
GDL はグループによって表されたソリッド間で特定の 3D 操作を実行することができます。これらの操作は、次のいずれかになります。

・**ADDGROUP**: 2 つのソリッドのブール和の形成



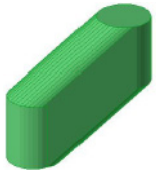
・**SUBGROUP**: 2 つのソリッドのブール差の形成

・ISECTGROUP: 2 つのソリッドのブール積の形成

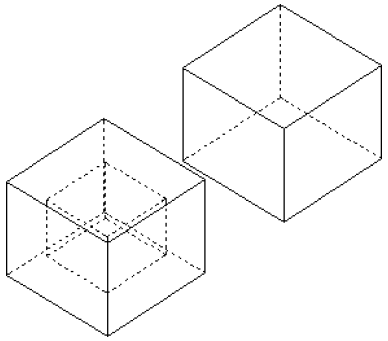


・ISECTLINES: 2 つのソリッドの交差線の計算

・SWEEPGROUP: ベクトルに沿ったソリッドの掃引



GDL のソリッドは、モデルでは別々のボディに見える、1 つ以上の立体からなります。立体には、必ず 1 つの外側殻があり、空洞を含むこともできます (空洞は、立体内の負の内側殻として記述することができます)。次の図のソリッドは、2 つの立体からなり、その 1 つには空洞があります。



BLOCK、SPHERE などの GDL ボディは、グループでは外側殻として表されます。次の組み立てを使用して、ソリッド内に複数の殻を置くことができます（BODY -1 ステートメントに注意）。

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

上のソリッドには 2 つの立体があり、そのそれぞれが 1 つの殻からなります。空洞は、プリミティブを使用して定義できます。また、ブール差の結果として発生させることもできます（例えば、大きな直方体の中央から小さな直方体を減算する）。

「123 ページ「プリミティブ要素」」も参照してください。

グループ操作はソリッドオブジェクトを使用した作業を目的としてはいますが、サーフェス、ワイヤフレーム、ハイブリッドモデルにも適用することができます （ハイブリッドモデルは、基本的には、隣接する面を持たずに辺を含むことのできる表面です）。このようなモデルでの操作の結果は次のようになります。

和（ベース » ツール）

	ソリッドベース	**サーフェスベース**	**ワイヤフレームベース**	**ハイブリッドベース**
ソリッドツール	ソリッドの結果	サーフェスの結果（結合）	ワイヤフレームの結果（結合）	ハイブリッドの結果（結合）
サーフェスツール	サーフェスの結果（結合）	サーフェスの結果（結合）	ハイブリッドの結果（結合）	ハイブリッドの結果（結合）
ワイヤフレームツール	ワイヤフレームの結果（結合）	ハイブリッドの結果（結合）	ワイヤフレームの結果（結合）	ハイブリッドの結果（結合）
ハイブリッドツール	ハイブリッドの結果（結合）	ハイブリッドの結果（結合）	ハイブリッドの結果（結合）	ハイブリッドの結果（結合）

差（ベース ¥ ツール）

	ソリッドベース	サーフェスベース	ワイヤフレームベース	ハイブリッドベース
ソリッドツール	ソリッドの結果	サーフェスの結果	ワイヤフレームの結果	ハイブリッドの結果
サーフェスツール	ソリッドベース（影響なし）	サーフェスベース（影響なし）	ワイヤフレームベース（影響なし）	ハイブリッドベース（影響なし）
ワイヤフレームツール	ソリッドベース（影響なし）	サーフェスベース（影響なし）	ワイヤフレームベース（影響なし）	ハイブリッドベース（影響なし）
ハイブリッドツール	ソリッドベース（影響なし）	サーフェスベース（影響なし）	ワイヤフレームベース（影響なし）	ハイブリッドベース（影響なし）

積（ベース « ツール）

	ソリッドベース	サーフェスベース	ワイヤフレームベース	ハイブリッドベース
ソリッドツール	ソリッドの結果	サーフェスの結果	ワイヤフレームの結果	ハイブリッドの結果
サーフェスツール	サーフェスの結果	空白の結果	空白の結果	空白の結果
ワイヤフレームツール	ワイヤフレームの結果	空白の結果	空白の結果	空白の結果
ハイブリッドツール	ハイブリッドの結果	空白の結果	空白の結果	空白の結果

交差線（ベース « ツール）

	ソリッドベース	サーフェスベース	ワイヤフレームベース	ハイブリッドベース
ソリッドツール	ワイヤフレームの結果	ワイヤフレームの結果	空白の結果	ワイヤフレームの結果
サーフェスツール	ワイヤフレームの結果	空白の結果	空白の結果	空白の結果
ワイヤフレームツール	空白の結果	空白の結果	空白の結果	空白の結果
ハイブリッドツール	ワイヤフレームの結果	空白の結果	空白の結果	空白の結果

掃引

ソリッド	サーフェス	ワイヤフレーム	ハイブリッド
有効な結果	サーフェスベース（影響なし）	ワイヤフレームベース（影響なし）	ハイブリッドベース（影響なし）

サーフェスは、MODEL SURFACE コマンドで明示的に作成することも、モデルから隣接しない面ポリゴンを省くことで暗黙的に作成することもできます。ワイヤフレームは、MODEL WIRE ステートメントを使用する方法と面ポリゴンなしでオブジェクトを定義する方法のどちらかで作成します。ハイブリッドモデルは、モデルから隣接する面ポリゴンを省くという間接的な方法でのみ作成できます。

多くの場合、必要なモデルはソリッドです。GDL ボディはグループ定義では殻として表されるので、高速で信頼性の高い操作を実現するには、生成される殻の幾何学的正確さが重要な問題です。不適切なオブジェクトを取り扱うと GDL エンジンに

負荷がかかり、意図した操作が完了するのに余計な時間がかかることになります。GDL のグループ操作を効率的に使用するために考慮すべき主な規則を要約すると、次のようになります。*物理的に存在している空間オブジェクトに合わせたモデル*。これは実際には、次のような指針となります。

・自己交差オブジェクトを避ける

・自己接触オブジェクトを避ける（小さな間隔を適用する）

・オブジェクトではサイズがゼロの部分を避ける（小さな厚みを適用する）

前述のように、殻（ボディによって定義される）の場合はこれらの規則に従うべきですが、ソリッド（グループによって定義される）の場合は当てはまりません （前述の GROUP 組み立て法のスクリプトによって作成されたソリッドは、適切にモデリングされます。これは、このソリッドを構成する殻は互いに接触しますが、殻自体は幾何学的に正しいからです)。

GROUP

GROUP "name"

グループ定義の始まり。次の ENDGROUP ステートメントまでの全てのボディは「name」グループの一部になります。グループは、実際に生成（配置）されませんが、グループ操作で使用するまたは PLACEGROUP コマンドで明示的に配置することができます。グループ定義は入れ子にできませんが、グループ定義を含むマクロコールとほかのグループを使用する PLACEGROUP コマンドは含められます。

グループ名は、現在のスクリプト内では一意でなければなりません。グループ定義外の変換、断面はグループの部品には影響を及ぼしません。また、グループ定義内で使用された変換、断面は、定義外のボディには影響しません。グループ定義は、属性 DEFINE および SET（ペン、材質、塗りつぶし）に対しては透過です。定義の前に定義または設定された属性と定義の中で定義または設定された属性は、全て有効です。

ENDGROUP

ENDGROUP

グループ定義の終わり。

ADDGROUP

ADDGROUP (g_expr1, g_expr2)

SUBGROUP

SUBGROUP (g_expr1, g_expr2)

ISECTGROUP

ISECTGROUP (g_expr1, g_expr2)

ISECTLINES

ISECTLINES (g_expr1, g_expr2)

和、差、積、交差線のグループ操作。戻り値は新規のグループで、これは PLACEGROUP コマンドで配置したり、変数に格納したり、ほかのグループ操作でパラメータとして使用することができます。グループ操作は、既に定義されているグループ同士の間で、またはほかのグループ操作の結果のグループ同士間で実行することができます。e_expr1、g_expr2 は、グループタイプの式です。グループタイプの式は、グループ名（文字列式）とグループタイプ変数のどちらかか、結果がグループとなる操作でのこれらの任意の組み合わせです。

PLACEGROUP

PLACEGROUP g_expr

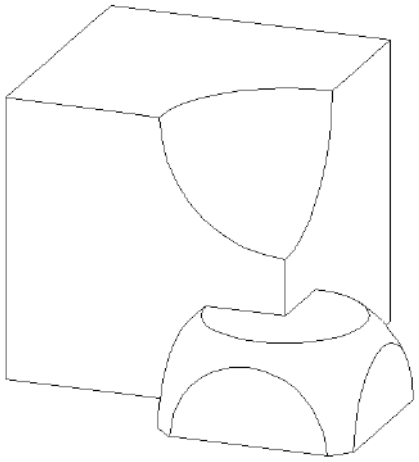
グループを配置すると、ボディが実際に生成されます。断面と変換は有効で、グループ式は評価され、結果のボディは 3D データ構造に格納されます。

KILLGROUP

KILLGROUP g_expr

指定されたグループのボディをメモリからクリアします。KILLGROUP 操作後は、グループは空白になります。クリアは、スクリプトの最後またはマクロコールから戻るときに、自動的に実行されます。パフォーマンスの観点から、このコマンドは、グループが必要なくなったときに使用してください。

例：



```
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[ENDGROUP]
! "ボックス"から"球面"を減算
result_1=SUBGROUP("box","sphere")
! "半球"と"レンガ"を交差
result_2=ISECTGROUP("semisphere","brick")
! 生成された以下を追加 result_3=ADDGROUP(result_1,result_2)
PLACEGROUP result_3
KILLGROUP "box"
KILLGROUP "sphere"
KILLGROUP "semisphere"
KILLGROUP "brick"
```

SWEEPGROUP

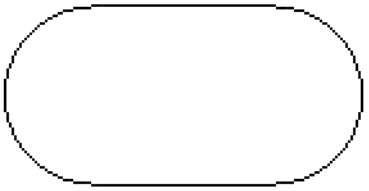
SWEEPGROUP (g_expr, x, y, z)

グループパラメータを与えられた方向に掃引することによって作成されたグループを返します。このコマンドは、ソリッドモデルに対してのみ有効です。

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

SWEEPGROUP と SWEEPGROUP (2) の違いとしては、前者では、現在の座標系に対して、実際の変換マトリックスが掃引操作の方向ベクトルに再度適用される点です （後者の SWEEPGROUP では、グローバルの座標系に対して、現在の変換が方向ベクトルに 2 度適用されます)。

例 :



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

バイナリ 3D

BINARY

BINARY mode [, section]

インラインバイナリオブジェクトを GDL マクロに含めるための特別なコマンド。頂点、ベクトル、辺、ポリゴン、ボディ、材質のセットがライブラリ部品ファイルの特定の部分から読み込まれます。このとき現在の変換に従って変形が行なわれ、3D モデルに結合されます。バイナリセクションのデータは、ユーザーが直接編集することはできません。

mode: ペンカラーと材質属性の定義の使用方法を定義します。

0: 現在の PEN および MATERIAL の設定は有効

1: 現在の PEN および MATERIAL の設定は無効。ライブラリ部品は、保存されているカラーと材質の定義で表示されます。表面の外観は一定です。

2: 保存されている PEN および MATERIAL の設定が使用され、定義されていない材質は、現在の設定に置き換えられる

3: 保存されている PEN および MATERIAL の設定が使用され、定義されていない材質は、保存されているデフォルト属性に置き換えられる

section: バイナリ部品のインデックス（1 から 16 まで）

セクションインデックスに 0 を指定すると、既存の全てのバイナリ部品を同時に参照できます。
インデックス値が 1 のセクションだけが GDL 内部から保存できます。セクション引き数を指定しない BINARY コマンドを使って参照することもできます。ほかのセクションインデックスは、サードパーティのツールで使用されます。
ArchiCAD と異なるデータ構造を持つファイル（例えば DXF や ZOOM）を開くと、その 3D 記述はバイナリフォーマットに変換されます。
メニューから *[名前を付けて保存...]* コマンドでメインのライブラリ部品編集ウィンドウを開いて、ライブラリ部品をバイナリフォーマットで保存することができます。*[名前を付けて保存...] 任意の順番*でダイアログボックスの *[バイナリフォーマットで保存]* チェックボックスがオンの場合、現在のライブラリ部品の GDL テキストはバイナリ記述に置き換えられます。

ヒント： 3D 切断後の 3D モデルをバイナリフォーマットで保存すると、切り取られたモデルが保存されます。この方法で、切り取り形状を作成できます

ライブラリ部品の 3D モデルをすでに生成している場合のみ、ライブラリ部品をバイナリフォーマットで保存することができます。
ライブラリ部品の GDL 記述をバイナリ記述で置き換えると、3D 変換にかかる時間を大幅に短縮することができます。一方、バイナリ 3D 記述はパラメトリックでなく、演算的な GDL プログラムよりディスク空間を必要とします。

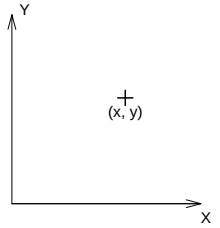
2D 形状

この章では、2D 空間で、線や円弧のような単純な形からポリゴンやスプラインのような複雑な形状を生成するのに使用するコマンドおよび 2D 空間でのテキスト要素の定義を紹介します。また、2D 空間でバイナリデータがどのように取り扱われるか、3D スクリプトによって作成された形状が 2D ビューでどのように投影されるか、そしてオブジェクトの 3D および 2D での外観間で一貫性を保つ方法についても説明しています。これから説明するコマンドでは、グラフィック要素を計算用に作成された要素リストに置くことができます。

図面要素

HOTSPOT2

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



unID は、2D スクリプトにおけるホットスポットの一意の識別子です。これは可変数のホットスポットが存在する場合に便利です。

paramReference: グラフィカルホットスポットベースのパラメータ編集手法を使って編集可能なパラメータ。

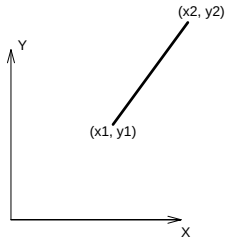
displayParam: paramRefrence パラメータ編集中に情報パレットに表示するためのパラメータ。配列のメンバも使えます。

HOTSPOT2 の使用については、175 ページ「グラフィカル編集」を参照してください。

LINE2

LINE2 x1, y1, x2, y2

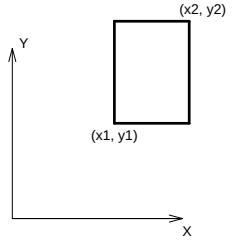
2 点間の線。



RECT2

RECT2 x1, y1, x2, y2

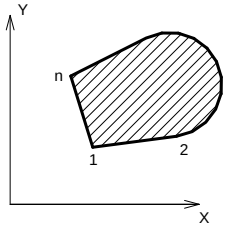
2 つの節点による矩形定義。



POLY2

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

n 個の辺を持つ開いたまたは閉じたポリゴン。



パラメータの制限：

n >= 2

frame_fill = j1 + 2*j2 + 4*j3

ここで、j1、j2、j3 は、0 または 1 をとります。

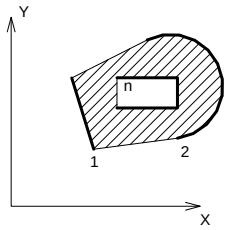
j1 (1): 輪郭線のみ

j2 (2): 塗りつぶしのみ

j3 (4): 開いたポリゴンを閉じる

POLY2_

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



標準の POLY2 ステートメントとほとんど同じですが、任意の辺を省略することができます。si = 0 の場合、頂点（xi,yi）から延びる辺は省略されます。si = 1 の場合、頂点が表示され、si = -1 は直接開口（穴）を定義するのに使用します。追加ステータスコード値を使用して、ポリラインに円弧および線分を定義することもできます。

パラメータの制限：

n >= 2

frame_fill = j1 + 2*j2 + 4*j3 + 8*j4 + 32*j6 + 64*j7

ここで、j1、j2、j3 は、0 または 1 をとります。

j1（1）：輪郭線のみ

j2（2）：塗りつぶしのみ

j3（4）：開いたポリゴンを閉じる

j4（8）：ローカルの塗りつぶし向き

j6：切断塗りつぶし（デフォルトは作図塗りつぶし）

j7：表面塗りつぶし（j6 = 0 のの場合のみ、デフォルトは作図塗りつぶし）

ステータス値：

s = j1 + 16*j5 + 32*j6

ここで、j1、j5、j6 は、0 または 1 をとります。

j1（1）：次の線分を表示

j5（16）：次の線分は内側の線（0 の場合は、一般的な線）

j6（32）：次の線分は輪郭線（j5 が設定されていない場合のみ有効）

-1：輪郭線の終端

POLY2_ 線のデフォルトの線特性は 0 （一般的な線）で、LINE_PROPERTY ステートメントは POLY2_ 辺では無効です。
追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。
詳細は、183 ページ「追加ステータスコード」を参照してください。

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

POLY2_ コマンドの詳細版には、塗りつぶしペンと塗りつぶし背景ペンの追加パラメータがあります。その他のパラメータは、全て POLY2_ ステートメントで説明しているものとほぼ同じです。追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。
詳細は、183 ページ「追加ステータスコード」を参照してください。

POLY2_B{2}

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

POLY2_B コマンドの上級版で、塗りつぶしペン、背景ペン、原点、および方向を定義することができます。
frame_fill = j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 +64*j7
j1、j2、j3、j4、j5、j6、j7 は 0 か 1 をとります。
j1 (1): 輪郭線のみ
j2 (2): 塗りつぶしのみ
j3 (4): 開いたポリゴンを閉じる
j4 (8): ローカルの塗りつぶし向き
j5 (16): グローバルの塗りつぶし原点（j4 が設定されている場合のみ有効）
j6 (32): 切り取りカテゴリの塗りつぶし（j7 に固有、なしに設定されている場合は作図カテゴリ）
j7 (64): 表面カテゴリの塗りつぶし（j6 に固有、なしに設定されている場合は下書きカテゴリ）
追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。
詳細は、183 ページ「追加ステータスコード」を参照してください。

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

POLY2_ コマンドの上級版で、塗りつぶしの向きをマトリクスを使用して定義することができます。
frame_fill = j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 +128*j8
j1、j2、j3、j4、j5、j6、j7、j8 は 0 か 1 をとります。
j1-j7: 前の POLY2_ コマンドと同様
j8 (128): スロープの塗りつぶしを使用
mxx, mxy, myx, myy: j8 が設定されている場合に、このマトリクスで塗りつぶしの向きを定義します。
追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。
詳細は、183 ページ「追加ステータスコード」を参照してください。

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

POLY2_ B{3} の上級版で、円形グラデーションの内部半径を設定することができます。
gradientInnerRadius: ポリゴンに対して円形グラデーションを選択した場合のグラデーションの内部半径。

POLY2_B{5}

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

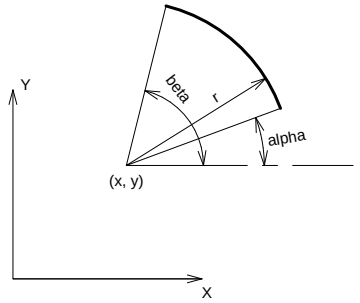
POLY2_ B{4} の上級版で、塗りつぶしの歪みを高度な方法で制御することができます。
frame_fill = j1 + 2*j2 + 4*j3
ここで、j1、j2、j3 は、0 または 1 をとります。

j1 (1): 輪郭線のみ
j2 (2): 塗りつぶしのみ
j3 (4): 開いたポリゴンを閉じる
塗りつぶしカテゴリ : 0 - 作図 1 - 切断 2- 表面
distortion_flags = j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 64*j7
ここで、各 ji フラグは 0 または 1 をとります。distortion_flags では、0 から 127 までの値が有効です。この範囲以外の値を使用しないでください。
j1 (1): 塗りつぶし原点の X 座標はグローバル原点の X 座標であり、j4 が設定されている場合のみ有効です。
fillOrigo は、(mxx, mxy) ベクトル線上の投影線原点 (0,0) です。
j2 (2): 塗りつぶし原点の Y 座標はグローバル原点の Y 座標であり、j4 が設定されている場合のみ有効です。
fillOrigo は、(myx, myy) ベクトル線上の投影線原点 (0,0) です。
j3 (4): innerRadius パラメータを使用して円形の歪みを作成
j4 (8): ローカルの方向を使用、歪み配列 (mij パラメータ) を使用
j5 (16): (シンボル塗りつぶしにのみ有効) 定義した X ベクトルの長さ (mxx, mxy) に、パターンの X サイズをリセット
j6 (32): (シンボル塗りつぶしにのみ有効) 定義した Y ベクトルの長さ (myx, myy) に、パターンの Y サイズをリセット
j7 (64): (シンボル塗りつぶしにのみ有効) シンボル塗りつぶしパターンの比率を維持、j5 および j6 の 1 つが設定されている場合にのみ有効
innerRadius: 円形塗りつぶしの歪みの半径、元の円の原点は (0, -innerRadius) 位置の Y 塗りつぶし軸上に配置されます。

ARC2

ARC2 x, y, r, alpha, beta

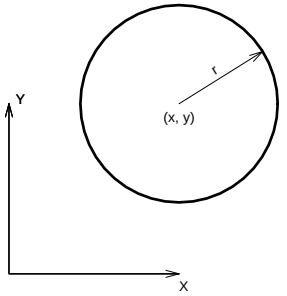
中心が（x, y）で、alpha で始まり beta で終わる角度を持つ、半径が r の円弧。
alpha と beta は度（° ）で指定します。



CIRCLE2

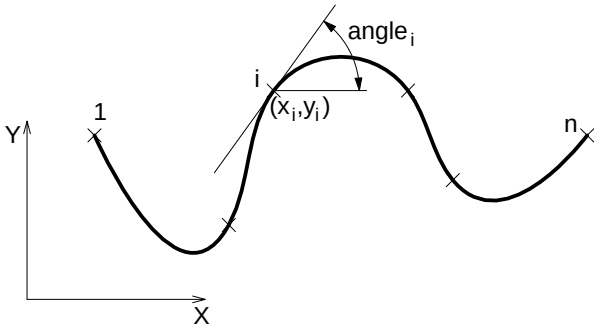
CIRCLE2 x, y, r

中心が（x, y）で、半径が r の円。

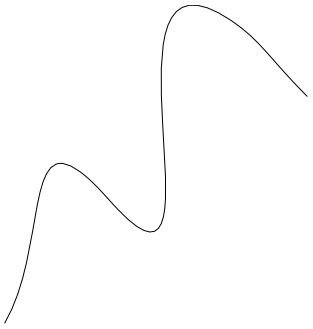


SPLINE2

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

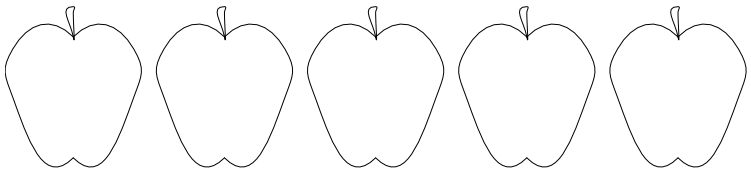


制限：
n >= 2
制御点が n 個のスプライン。制御点（xi,yi）でのスプラインの接点は、角度 i、つまり X 軸との度数単位の角度によって定義されます。
ステータス値：
0: デフォルト
1: 閉じたスプライン。スプラインの最初と最後の節点が接続された閉じたスプラインになります。
2: 自動的にスムージングされたスプライン。スプラインの生成時には、最初と最後の節点との間の節点の角度パラメータ値は使用されません。内蔵されている自動スムージングアルゴリズムが使用されます。



例：

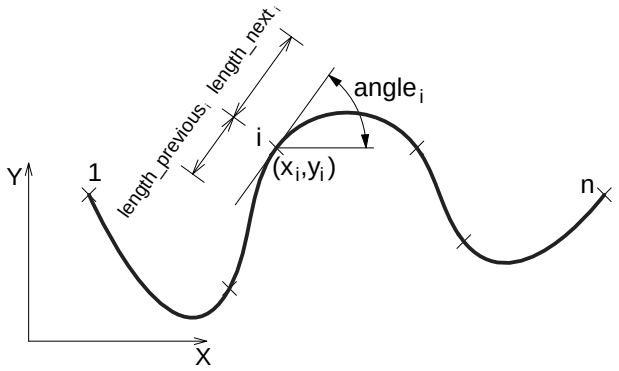
```
SPLINE2 5, 2,
          0, 0, 60,
          1, 2, 30,
          1.5, 1.5, -30,
          3, 4, 45,
          4, 3, -45
```



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

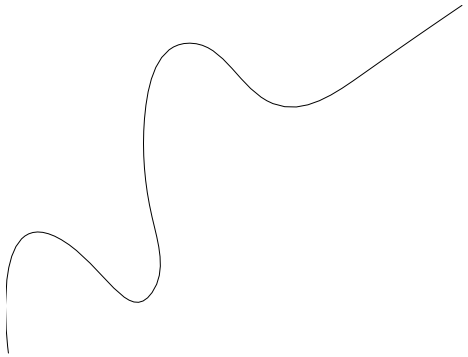
SPLINE2A

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



SPLINE2 ステートメントの拡張版（ベジェスプライン）。複雑なため、主に自動 2D スクリプト生成で使用されます。
詳細は、『ArchiCAD ヘルプ』の「ドキュメンテーション」章の「線／スプラインの作図」を参照してください。
ステータスコード:
0: デフォルト
1: 閉じたスプライン。スプラインの最初と最後の節点が接続されて閉じたスプラインになります。
2: 自動的にスムージングされたスプライン。スプラインの生成時には、最初と最後の節点にはさまれた節点の角度、length_ previous$_i$ と length_next$_i$ のパラメータ値は使用されません。内蔵されている自動スムージングアルゴリズムが使用されます。
xi, yi: 制御点の座標
length_previousi、length_nexti: 前と次の制御点の接線の長さ
anglei: 接線の方位角

例：



```
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

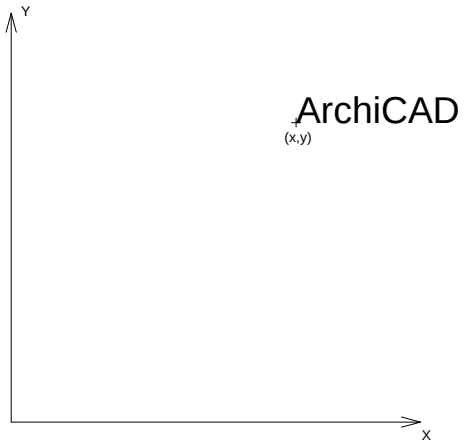
3D の PICTURE コマンドと同じように 2D で使用できます。3D とは異なり、マスク値は 2D 画像には影響しません。
文字列タイプの expression は画像ファイル名、数値の expression はライブラリ部品に格納されている画像のインデックスを意味します。0 のインデックスは特殊な値です。この値は、ライブラリ部品のプレビュー画像を参照します。PICTURE2{2} の場合、mask = 1 は完全に白のピクセルが透明であることを意味します。ほかの画像は、画像を GDL オブジェクトとして含むプロジェクトまたは選択した要素を保存するときに、ライブラリ部品に格納できるだけです。

テキスト要素

TEXT2

TEXT2 x, y, expression

計算された数式または文字列式の値を、x, y 座標位置に設定スタイルで書き込みます。
コマンド 199 ページ「[SET] STYLE」および 219 ページ「DEFINE STYLE」も参照してください。



RICHTEXT2

RICHTEXT2 x, y, textblock_name

前もって定義された TEXTBLOCK を配置します。
詳細は、221 ページ「テキストブロック」を参照してください。
x, y: X-Y リッチテキストの位置の座標

textblock_name: 前もって定義された TEXTBLOC の名前

バイナリ 2D

FRAGMENT2

FRAGMENT2 fragment_index,
use_current_attributes_flag

与えられたインデックスを持つ断片を、現在の変換と共に 2D 全体表示に挿入します。
use_current_attributes_flag: 現在の属性を使用するかどうかを定義します。

0: 定義されたカラー、線種、および塗りつぶし種類で断片を表示
1: 断片のカラー、線種、および塗りつぶし種類の代わりにスクリプトの現在の設定を使用

FRAGMENT2

FRAGMENT2 ALL,use_current_attributes_flag

全ての断片を現在の変換と共に 2D 全体表示に挿入します。
use_current_attributes_flag: 現在の属性を使用するかどうかを定義します。

0: 定義されたカラー、線種、および塗りつぶし種類で断片を表示
1: 断片のカラー、線種、および塗りつぶし種類の代わりにスクリプトの現在の設定を使用

2D での 3D 投影

PROJECT2

PROJECT2 projection_code, angle, method

PROJECT2{2}

PROJECT2{2} projection_code, angle,method [,backgroundColor, fillOrigoX, fillOrigoY, filldirection]

同じライブラリ部品に 3D スクリプトの投影を作成し、生成した線を 2D 全体表示に追加します。第 2 バージョンの PROJECT2{2} 、これより前の SET FILL コマンドと共に、2D スクリプトの結果である図面の塗りつぶし背景、原点、および方向をユーザーが制御できるようにします。

projection_code: 投影のタイプ

3: 上面図
4: 側面図
6: 斜投影
7: 等角投影
8: 不等角投影
9: 二等角投影
-3: 下面図
-6: 見上げ斜投影
-7: 見上げ等角投影
-8: 見上げ不等角投影
-9: 見上げ二等角投影

angle: [3D 投影の設定] ダイアログボックスの方位角設定

method: 選択された画像処理方式

1: ワイヤフレーム
2: 陰線処理（分解法）
3: シェーディング
16: 加算変更子。ベクトルハッチングを描画（隠線処理およびシェーディングモードでのみ有効）
32: 加算変更子。3D からの属性の代わりに現在の属性を使用（シェーディングモードでのみ有効）
64: 加算変更子。ローカル塗りつぶしの向き（シェーディングモードでのみ有効）
128: 加算変更子： 線は全て内側の線（32 と共にのみ有効）。デフォルトは generic。
256: 加算変更子： 線は全て輪郭線（128 が設定されておらず、32 と共にのみ有効）。デフォルトは generic。
512: 加算変更子： 塗りつぶしは全て断面（32 と共にのみ有効）。デフォルトは作図塗りつぶし。
1024: 加算変更子： 塗りつぶしは全て表面（512 が設定されておらず、32 と共にのみ有効）。デフォルトは作図塗りつぶし。

BackgroundColor: 塗りつぶし背景カラー

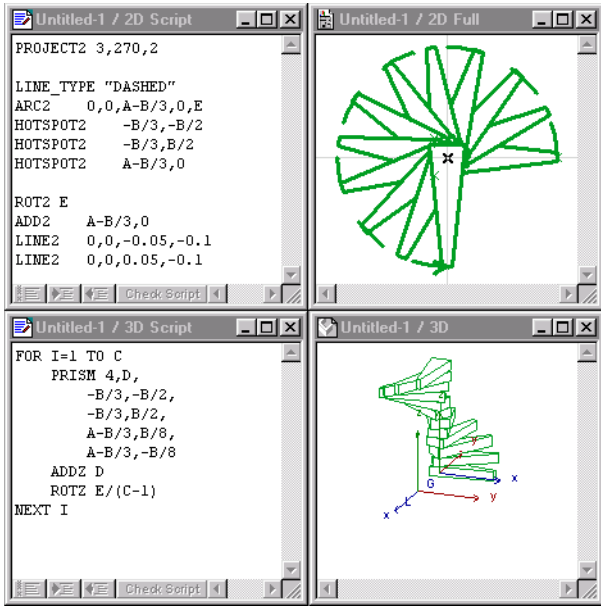
fillOrigoX: 塗りつぶし原点の X 座標

fillOrigoY: 塗りつぶし原点の Y 座標

filldirection: 塗りつぶしの方位角

注記： SET FILL は、PROJECT2{2} に対して有効です。

例：



互換性に関する注記： メソッドビット 32 をセットせずにメソッドビット 3 をセット（シェーディング）して PROJECT2 を使用した場合、ステータスビット 2 をセットせずに CUTPOLYA を使用してモデルを切り取る（切り取りポリゴンを生成）と、異なる属性を持つ切り取りポリゴンが生成されます。切り取りポリゴンは、3D スクリプトの SECT_FILL コマンドで定義された属性で生成されます。

PROJECT2{3}

PROJECT2{3} projection_code, angle, method , parts[, backgroundColor, fillOrigoX, fillOrigoY, filldirection][[,]
PARAMETERS name1=value1 , ... namen=valuen]

同じライブラリ部品に 3D スクリプトの投影を作成し、生成した線を 2D 全体表示に追加します。第 2 バージョンの PROJECT2{2} 、これより前の SET FILL コマンドと共に、2D スクリプトの結果である図面の塗りつぶし背景、原点、および方向をユーザーが制御できるようにします。第 3 バージョンの PROJECT2{3} は、投影モデルのどの部分が必要か定義し、線種を含めた切り取り部分とビュー部分の属性を個別に制御できる能力を追加します。現時点のパラメータをコマンドに設定して投影を生成することもできます。

method: 選択された画像処理方式

1: ワイヤフレーム
2: 陰線処理（分解法）
3: シェーディング
16: 加算変更子。ベクトルハッチングを描画（陰線処理およびシェーディングモードでのみ有効）
32: 加算変更子。3D からの属性の代わりに現在の属性を使用（シェーディングモードでのみ有効）
64: 加算変更子： ローカルの塗りつぶしの向き（シェーディングモードでのみ有効）。
128: 加算変更子： 線は全て内側の線（32 と共にのみ有効）。デフォルトは generic。
256: 加算変更子： 線は全て輪郭線（128 が設定されておらず、32 と共にのみ有効）。デフォルトは generic。
512: 加算変更子： 塗りつぶしは全て断面（32 と共にのみ有効）。デフォルトは作図塗りつぶし。
1024: 加算変更子： 塗りつぶしは全て表面（512 が設定されておらず、32 と共にのみ有効）。デフォルトは作図塗りつぶし。
2048: 加算変更子： 変更子 16、32、64、128、256、512、1024 および塗りつぶしの属性パラメータは投影のビュー部分でのみ有効。デフォルトでは全ての部分で有効。
4096: 加算変更子： 変更子 16、32、64、128、256、512、1024 および塗りつぶしの属性パラメータは投影の切り取り部分でのみ有効。デフォルトでは全ての部分で有効。
8192: 加算変更子： 切断塗りつぶしは傾斜。

既知の制限： 切り取り部分の線は個別に扱うことはできません。全ての線をまとめて内側または輪郭に設定することしかできません。

parts: 生成する部分を定義 15 は全ての部分を意味します。

part = j1 + 2*j2 + 4*j3 + 8*j4
ここで j1, j2, j3, j4 は 0 または 1 をとります。j1、j2、j3、j4 は、投影したモデルの該当部分を表示する（1）か省略する（0）かを表します。

j1: 切り取りポリゴン（SECT_FILL で定義したデフォルトの塗りつぶし属性）（シェーディングモードでのみ有効）
j2: 切り取りポリゴンの辺 j3: ポリゴンの表示
j4: ポリゴンの辺の表示

リスト内の図面

これらのコマンドは、ArchiCAD で要素リストが作成されたときのみ有効です。
特殊な特性タイプのライブラリ部品が、平面図内のライブラリ部品（オブジェクト、ドア、窓、または光源）に関連付けられているときに、以下のコマンドを 2D スクリプトに記述すると、そのライブラリ部品の 2D 部分と 3D 部分が参照されます。これは仮想参照と呼ばれるもので、リストプロセスの実行時に、現在リストされている要素の 2D または 3D スクリプトを使用して解決されます。

DRAWING2

DRAWING2 [expression]

expression の値に基づいて、このコマンドが記述されている特性オブジェクトに関連付けられたライブラリ部品の図面（expression = 0、デフォルト）または要素のラベル（expresion = 1）を作成します。

DRAWING3

DRAWING3 projection_code, angle, method

DRAWING3{2}

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

PROJECT2 と同様に、このコマンドを含む特性ライブラリ部品に関連付けられたライブラリ部品の 3D スクリプトの投影を作成します。全てのパラメータは、PROJECT2 および PROJECT2{2} のパラメータとほとんど同じです。
DRAWING3{2} の新規手法フラグ：
3: シェーディング
32: 3D からの属性の代わりに現在の属性を使用
64: ローカルの塗りつぶし向き

DRAWING3{3}

DRAWING3{3} projection_code, angle, method , parts[, backgroundColor, fillOrigoX, fillOrigoY,
filldirection][[,]
PARAMETERS name1=value1 , ... namen=valuen]

PROJECT2 と同様に、このコマンドを含む特性ライブラリ部品に関連付けられたライブラリ部品の 3D スクリプトの投影を作成します。全てのパラメータは、PROJECT2、PROJECT2{2} および PROJECT2{3} のパラメータとほとんど同じです。
DRAWING3 ¥{3¥} の新規手法フラグ：
2048: 加算変更子： 変更子 16、32、64、128、256、512、1024 および塗りつぶしの属性パラメータは投影のビュー部分でのみ有効。
投影 デフォルトでは全ての部分で有効。

4096: 加算変更子 : 変更子 16、32、64、128、256、512、1024 および塗りつぶしの属性パラメータは投影の切り取り部分でのみ有効。
投影 デフォルトでは全ての部分で有効。
8192: 加算変更子 : 切断塗りつぶしは傾斜。

グラフィカル編集

長さと角度タイプの GDL パラメータのホットスポットを基準とした対話形式のグラフィカル編集です。

ホットスポットベースの編集コマンド

HOTSPOT

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

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

unID: ユニーク ID。ライブラリ部品で定義されているホットスポットの中で一意でなければなりません。

paramReference: グラフィカルホットスポットベースのパラメータ編集手法を使って編集可能なパラメータ。

displayParam: paramRefrence パラメータ編集中に情報パレットに表示するためのパラメータ。配列のメンバも使えます。

有効な引数の例は、次のとおりです。

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

flags: ホットスポットのタイプ + ホットスポットの属性

type:

1: 長さタイプの編集、基準ホットスポット
2: 長さタイプの編集、移動ホットスポット
3: 長さタイプの編集、参照ホットスポット（常に非表示）
4: 角度タイプの編集、基準ホットスポット
5: 角度タイプの編集、移動ホットスポット
6: 角度タイプの編集、角度の中心（常に非表示）
7: 角度タイプの編集、参照ホットスポット（常に非表示）

属性は、次の値または 0 の組み合わせが可能です。

128: ホットスポットを非表示（タイプ 1、2、4、5 で有効）
256: 編集可能な基準ホットスポット（タイプ 1、4 で有効）
512: 2D で角度を反転（タイプ 6）

長さタイプのパラメータを編集するには、3 つのホットスポットをタイプ 1、2 および 3 で定義する必要があります。編集線の負の方向は、参照ホットスポットから基準ホットスポットへのベクトルで指定されます。移動ホットスポットは、この線に沿って、基準ホットスポットから測定された対応するパラメータの値によって決まる距離に配置する必要があります。

角度タイプのパラメータを編集するには、4 つのホットスポットをタイプ 4、5、6 および 7 で定義する必要があります。角度の平面は、中心ホットスポットから参照ホットスポットまでのベクトルに直交します。角度の測定における正の方向は、参照ホットスポットから平面を見た場合は反時計回りです。2D では平面は既に与えられているので、参照ホットスポットは

無視され、角度の測定の正の方向は、デフォルトでは反時計回りです。これは、中心ホットスポット（タイプ 6）の 512 属性フラグを設定することによって変更することができます。一貫性を持たせるために、中心ホットスポットから移動ホットスポットおよび基準ホットスポットへのベクトルは、中心ホットスポットから参照ホットスポットへのベクトルと垂直でなければなりません。移動ホットスポットは、中心ホットスポットを中心とした基準ホットスポットから測定された対応するパラメータによって決められる角度に配置する必要があります。

同一のパラメータの編集のためにホットスポットのセットが複数定義されている場合、ホットスポットは、ホットスポットコマンドの実行順にグループ化されます。基準ホットスポットに対して編集可能な属性が設定されている場合、基準ホットスポットをドラッグすることでパラメータも編集することができます。基準ホットスポットはオブジェクトの座標フレームに固定されることになるので（つまり、基準ホットスポットの位置はこれにアタッチされているパラメータと無関係でなければなりません）、オブジェクト全体が基準ポイントに沿ってドラッグされたり、基準ポイントを中心として回転されます（パラメータの値は変化しても、移動ホットスポットの位置は変化しません）。

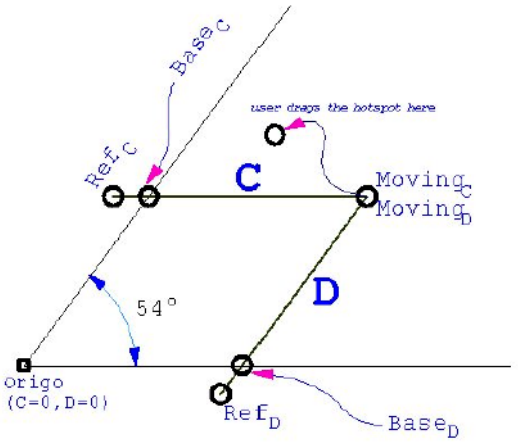
2 つまたは 3 つの長さタイプのホットスポットのセットを組み合わせて、1 回のドラッグで 2 つまたは 3 つのパラメータを編集できます。2 つを組み合わせた場合、ホットスポットの動きは線の制約を受けなくなりますが、長さ編集ホットスポットセットそれぞれの 2 つの線によって決まる平面の制約を受けます。3D では、長さ編集ホットスポットのセットを 3 つ組み合わせることで、ホットスポットを空間内の任意の場所に配置できます。2 本の線は、互いに平行であってはならず、3 本の線は同一平面上にあってはなりません。組み合わされたパラメータ編集操作は、選択した点の位置に対応パラメータが異なる 2 つまたは 3 つの編集可能ホットスポット（移動ホットスポットまたは編集可能な基準ホットスポット）がある場合に開始されます。パラメータが組み合わせ編集用に設計されている場合、基準ホットスポットと参照ホットスポットはオブジェクトの座標フレームに固定されませんが、残りのパラメータの値が変化するにつれて移動しなければなりません。

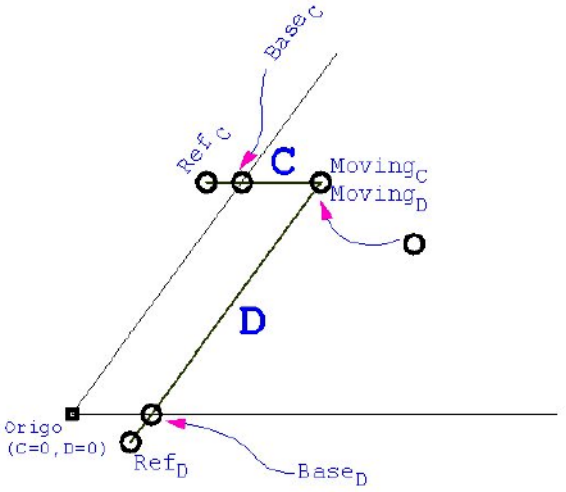
次の図と例 2 を参照してください。

例 1、2D での角度編集：

```
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
```

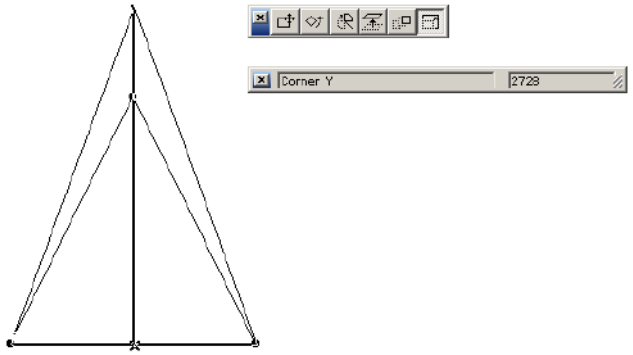
例 2、パラメータが 2 つの組み合わされた長さタイプ編集：





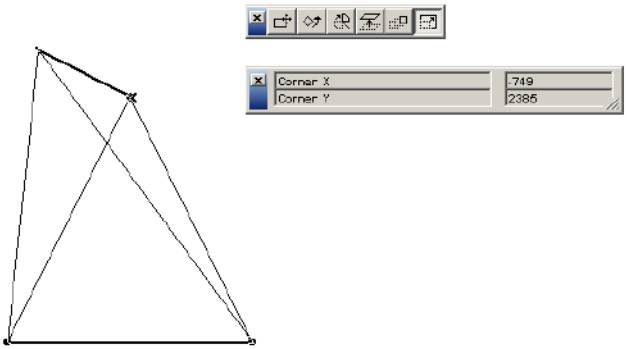

```
RECT2     0, 0, A, B
RECT2     0, 0, sideX, sideY
HOTSPOT2 sideX, 0, 1, sideY, 1
HOTSPOT2 sideX, -0.1, 2, sideY, 3
HOTSPOT2 sideX, sideY, 3, sideY, 2
HOTSPOT2 0, sideY, 4, sideX, 1
HOTSPOT2 -0.1, sideY, 5, sideX, 3
HOTSPOT2 sideX, sideY, 6, sideX, 2
```

例 3、パラメータが 1 つの単純な長さタイプ編集：




```
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
```

例 4： パラメータが 2 つの組み合わされた長さタイプ編集：




```
2D SCRIPT:
HOTSPOT2 -1, 0, 1
HOTSPOT2 1, 0, 2
HOTSPOT2 corner_x, 0, 3, corner_y, 1+128
HOTSPOT2 corner_x, -1, 4, corner_y, 3
HOTSPOT2 corner_x, corner_y, 5, corner_y, 2
HOTSPOT2 0, corner_y, 3, corner_x, 1+128
HOTSPOT2 -1, corner_y, 4, corner_x, 3
HOTSPOT2 corner_x, corner_y, 5, corner_x, 2
LINE2 -1, 0, 1, 0
LINE2 -1, 0, corner_x, corner_y
LINE2  1, 0, corner_x, corner_y
3D SCRIPT:
HOTSPOT -1, 0, 0, 1
HOTSPOT -1, 0, 0.5, 2
HOTSPOT  1, 0, 0, 3
HOTSPOT  1, 0, 0.5, 4
HOTSPOT corner_x, 0, 0, 5, corner_y, 1+128
HOTSPOT corner_x, -1, 0, 6, corner_y, 3
HOTSPOT corner_x, corner_y, 0, 7, corner_y, 2
HOTSPOT 0, corner_y, 0, 8, corner_x, 1+128
HOTSPOT -1, corner_y, 0, 9, corner_x, 3
HOTSPOT corner_x, corner_y, 0, 10, corner_x, 2
HOTSPOT corner_x, 0, 0.5, 11, corner_y, 1+128
HOTSPOT corner_x, -1, 0.5, 12, corner_y, 3
HOTSPOT corner_x, corner_y, 0.5, 13, corner_y, 2
HOTSPOT 0, corner_y, 0.5, 14, corner_x, 1+128
HOTSPOT -1, corner_y, 0.5, 15, corner_x, 3
```

```
HOTSPOT corner_x, corner_y, 0.5, 16, corner_x, 2
PRISM_  4, 0.5,
        -1, 0, 15,
        1, 0, 15,
        corner_x, corner_y, 15,
        -1, 0, -1
```

HOTLINE2

HOTLINE2 x1, y1, x2, y2

2 点間のステータス線

HOTARC2

HOTARC2 x, y, r, startangle, endangle

中心が（x, y）で、alpha で始まり beta で終わる角度を持つ、半径が r のステータス円弧。

HOTLINE

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

点 P1（x1,y1,z1）と点 P2（x2,y2,z2）の間の直線線分。

HOTARC

HOTARC r, alpha, beta, unID

x-y 平面上の、中心が原点で、alpha で始まり beta で終わる角度を持つ、半径が r のステータス円弧。
alpha と beta は度（°）で指定します。

ステータスコード

この章で説明するステータスコードでは、特別な制約を使用して平面ポリライン内に線分および円弧を作成することができます。

節点にステータスコードを持つ平面ポリラインは、多くの GDL 要素の基準です。

POLY_, PLANE_, PRISM_, CPRISM_, BPRISM_, FPRISM_, HPRISM_, SPRISM_, SLAB_, CSLAB_, CROOF_, EXTRUDE, PYRAMID, REVOLVE, SWEEP, TUBE, TUBEA

ステータスコードでは、次のことができます。

- 平面ポリラインの辺の可視性を制御する
- ポリライン内の穴を定義する
- 側面の辺および表面の可視性を制御する
- ポリライン内に線分および円弧を作成する

ステータスコードの構文

si の番号は、バイナリ整数（0 ～ 127）または -1 です。

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

j1、j2、j3、j4、j5、j6、j7 は 0 か 1 をとります。

j1、j2、j3、j4 は、頂点および側面を表示する（1）か、省略する（0）か、を表します。

j1: 水平方向の下の辺

j2: 垂直方向の辺

j3: 水平方向の上の辺

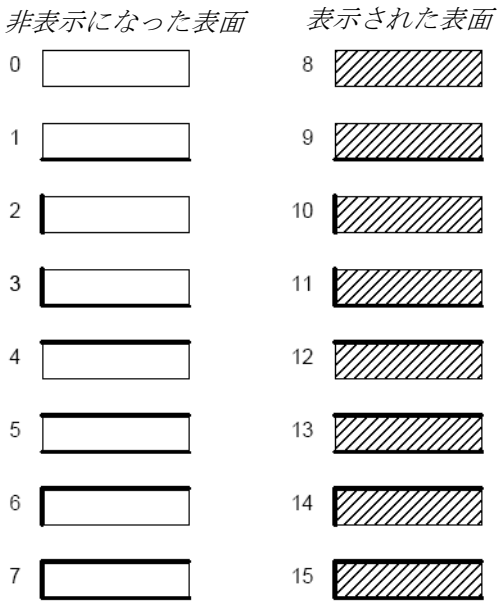
j4: 側面

j5: 陰線処理をした水平方向の辺（PRISM_ 形状の場合のみ）

j6: 陰線処理をした垂直方向の辺（PRISM_ 形状の場合のみ）

j7: j2=1 の場合のみ有効な特殊なステータス値で、現在の垂直方向の辺の表示に応じて視点を制御
j2=0: 水平方向の辺を常に非表示
j2=1 および j7=1: ビューの現在の方向から輪郭を表示する場合のみ、垂直方向の辺を表示
j2=1 および j7=0: 垂直方向の辺を常に表示

有効なステータス値は次のとおりです（太線は可視の辺を意味します)。



a_code：追加ステータスコード（省略可)。ポリライン内に線分および円弧を作成できるようにします。

`si = -1` は、角柱に直接穴を定義するのに使用します。これは輪郭の終わりと、その輪郭内の穴の始まりを示します。また各穴の輪郭の終了と、別の穴の開始を示すのにも使われます。その値の前の座標は、輪郭または穴の最初の点の座標と一致していなければなりません。マスク値として -1 を使った場合には、パラメータリストの最後のマスク値は、最後の穴の終わりを示すために必ず -1 でなければなりません。

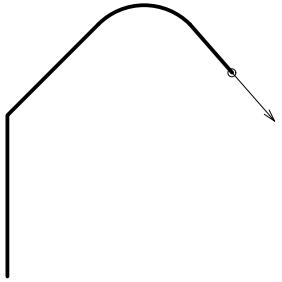
正しいシェーディングまたはレンダリング結果を得るためには、ポリゴンにおいて穴同士は分離していること、そして内部交差がないことが必要です。

追加ステータスコード

以下の追加ステータスコードでは、特殊な制約を使用して、ポリライン内に線分や円弧を作成することができます。これらのコードは、次の線分または円弧を参照します。オリジナルのステータスコードは、指定されている場所でのみ有効です(追加コードの後には「+s」が含まれます)。

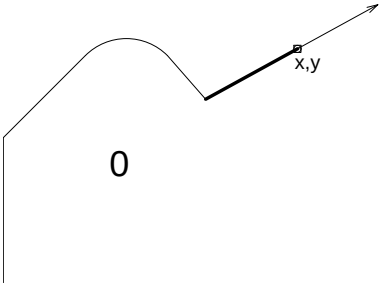
注記： 円弧の解像度は、「属性」の章で説明している指示文によって制御されます。POLY2_ ステートメントの場合、解像度が 8 より大きいと、真の円弧が生成され、それ以下の値にすると、生成される円弧は全て線分化されます。

ポリラインの前の部分： 現在の位置および接線が定義されます。



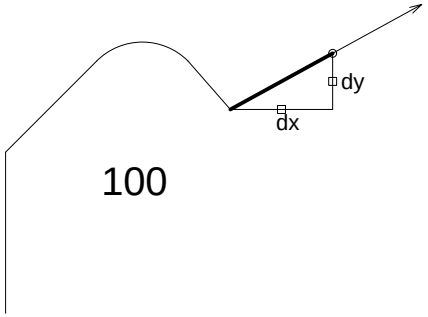
絶対終点による線分

x, y, s
ここで、0 < s < 100



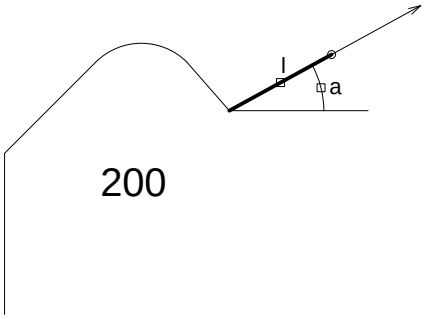
相対終点による線分

dx, dy, 100+s,
ここで、0 < s < 100



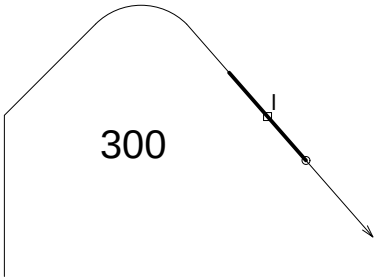
長さと方向による線分

l, a, 200+s,
ここで、0 < s < 100



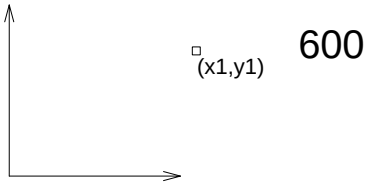
長さによる接線線分

l, 0, 300+s,
ここで、0 < s < 100



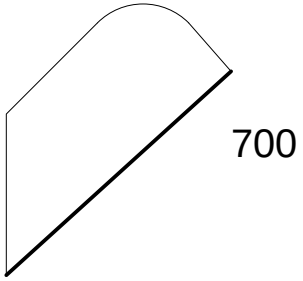
始点の設定

x1, y1, 600,



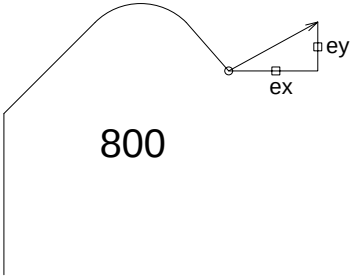
ポリラインを閉じる

```
0, 0, 700,
```



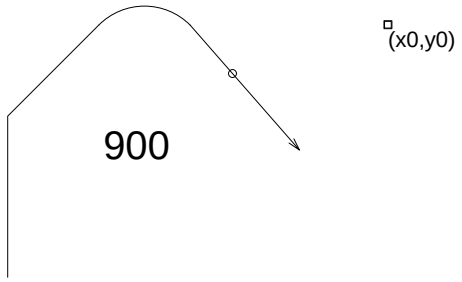
接線の設定

```
ex, ey, 800,
```



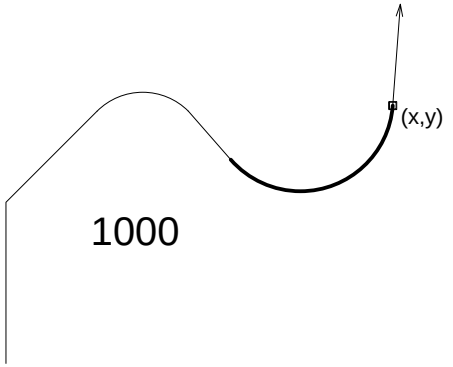
中心点の設定

x0, y0, 900,



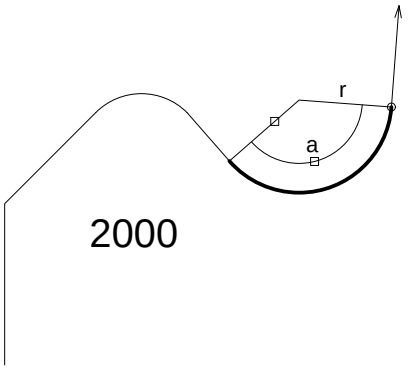
終点までの正接円弧

x, y, 1000+s,
ここで、0 < s < 100



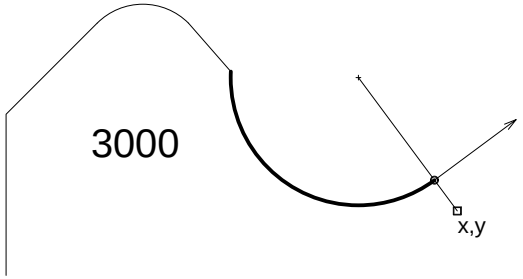
半径と角度による正接円弧

r, a, 2000+s,
ここで、0 < s < 100



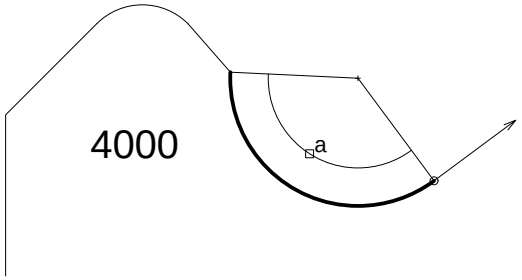
中心点と最終半径上の点による円弧

x, y, 3000+s,
ここで、0 < s < 100



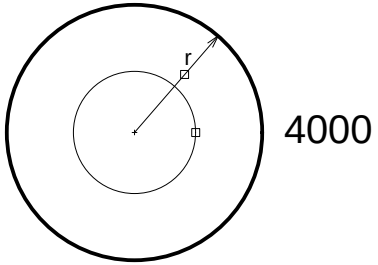
中心点と角度による円弧

0，a，4000+s,
ここで、0 < s < 100



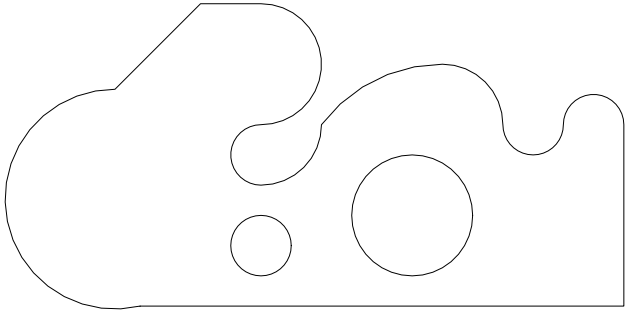
中心点と半径による完全な円

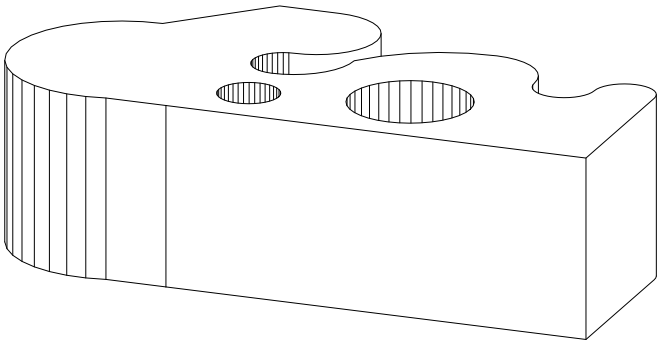
r，360，4000+s,
ここで、0 < s < 100



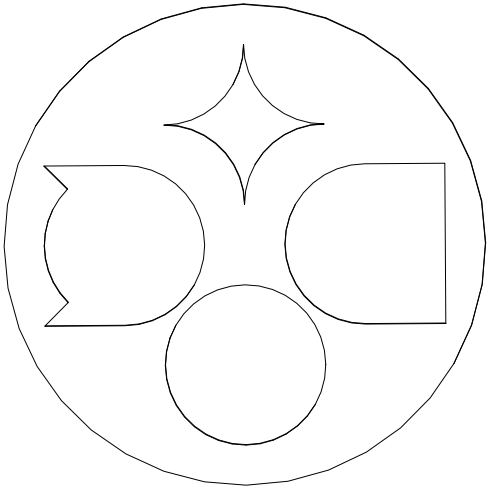
この場合、ステータス s が円全体を参照します。
全ての角度値は度（°）単位です。0 で示された省略された座標 0 （コード 300、700、4000）はどのような値でも構いません。

例：



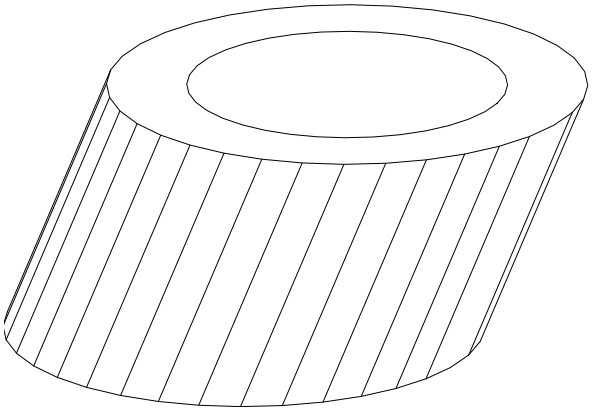


```
EXTRUDE 21, 0, 0, 3, 1+2+4+16+32,
 0, 0, 0,
 7, 0, 0,
 7, 3, 1,
 6, 3, 1000,        ! 終点への正接円弧
 5, 3, 1001,        ! 終点への正接円弧
 1, 90, 2000,       ! 半径と角度による正接円弧
 2, 3, 1001,        ! 終点への正接円弧
 1, 3, 900,         ! 終点を設定
 1, 2, 3000,        ! 始点、中心点、最後の半径上の点を使った円弧
 1, 2.5, 900,       ! 中心点を設定
 0, -180, 4001,     ! 始点、中心点、角度を使った円弧
 1, 5, 1000,        ! 終点への正弦円弧
-1, 0, 100,         !(dx, dy)
 2, 225, 200 によるセグメント,      !(len, angle)
-1, 0 によるセグメント, 800,        ! 正接を設定
-1, 0, 1000,        ! 終点への正接円弧
 0, 0, -1,          ! 輪郭の終端
 1, 1, 900,         ! 中心点を設定
0.5, 360, 4000,     ! 中心点と半径による円
3.5, 1.5, 900,      ! 中心点を設定
1, 360,  4001       ! 中心点、半径による円
```

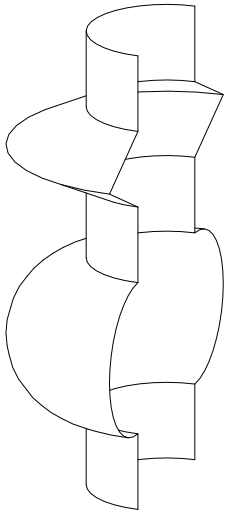


```
EXTRUDE 2+5+10+10+2, 0, 0, 3, 1+2+4+16+32,
      0, 0, 900,
      3, 360, 4001,
      2.5, -1, 0,
      2.5, 1, 0,
      1.5, 1, 1,
      1.5, -1, 1001,
      2.5, -1, -1,
      0, 2.5, 600,
      0, -1, 800,
      1, 1.5, 1001,
      -1, 0, 800,
      0, 0.5, 1001,
      0, 1, 800,
      -1, 1.5, 1001,
      1, 0, 800,
      0, 2.5, 1001,
      0, 2.5, 700,
      -1.5, 0, 900,
      -2.5, 0, 600,
      -2.5, 1, 3000,
      -2.5, 1, 0,
      -1.5, 1, 0,
      -1.5, -1, 1001,
```

```
-2.5, -1, 0,
SQR(2)-1, 45, 200,
-2.5, 0, 3000,
-2.5, 0, 700,
0, -1.5, 900,
1, 360, 4000
```



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



```
ROTY -90
REVOLVE 9, 180, 16+32,
 7, 1, 0,
 6, 1, 0,
 5.5, 2, 0,
 5, 1, 0,
 4, 1, 0,
 3, 1, 900,     ！中心点を設定
 0, 180, 4001, ！始点、中心点、角度を使った円弧
 2, 1, 0,
 1, 1, 0
```

属性

この章の最初の部分では、GDL ステートメントの解釈に影響する指示文を解説しています。指示文では、円柱要素で使われる滑らかさ、3D 表示での表現モード、または以降の形状の属性（カラー、材質、テキストスタイルなど）の割り当てを定義できます。次にインライン属性定義について解説しています。この機能では、プロジェクトの現在の属性セットにはないカスタマイズした材質、テクスチャ、塗りつぶしパターン、線種、テキストスタイルをオブジェクトに割り当てることができます。

指示文

指示文は、それに続く GDL ステートメントの解釈に影響し、その影響は次の指示文まで、またはスクリプトが終了するまで続きます。コールされたスクリプトは、現在の設定を継承します。変更内容はローカルに対して影響を与えます。スクリプトから戻るときには、マクロコールの前の設定に戻ります。

3D および 2D スクリプトの指示文

[LET]

[**LET**] varnam = n

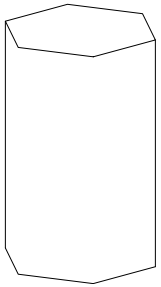
値の割り当て。LET 指示文は省略できます。変数は、n の評価された値を格納します。

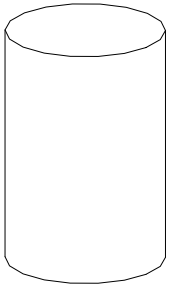
RADIUS

RADIUS radius_min, radius_max

ポリライン内の円柱要素と円弧の滑らかさを設定します。
半径がｒの円が表現されます。
・r <radius_min の場合は、六角形
・r ≥ radius_max の場合は、辺が 36 のポリゴン
・radius_min < r <radius_max の場合は、辺が (6+30*(r-radius_min)/(radius_max-radius_min)) のポリゴン
円弧についても同様に変換されます。
RADIUS ステートメントの後では、前に使用された RESOL と TOLER ステートメントの効果はなくなります。
パラメータの制限：
r_min ≤ r_max
例：

```
RADIUS 1.1, 1.15        RADIUS 0.9, 1.15
CYLIND 3.0, 1.0         CYLIND 3.0, 1.0
```



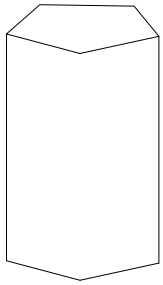


RESOL

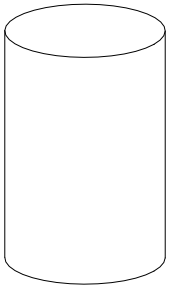
RESOL n

ポリライン内の円柱要素と円弧の滑らかさを設定します。円は、辺の数が n の正規ポリゴンに変換されます。
円弧についても同様に変換されます。
RESOL ステートメントの後では、前に使用された RADIUS と TOLER ステートメントの効果はなくなります。
パラメータの制限：
n >= 3
デフォルト：
RESOL 36
例：

RESOL 5
CYLIND 3.0, 1.0



RESOL 36
CYLIND 3.0, 1.0

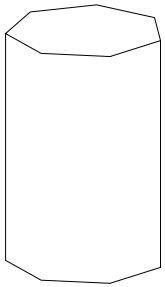


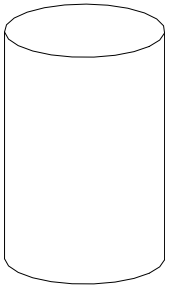
TOLER

TOLER d

ポリライン内の円柱要素と円弧の滑らかさを設定します。円弧の近似誤差（つまり、論理円弧と生成された弦の最大相違）は、d よりも小さくなります。
TOLER ステートメントの後では、前の RADIUS と RESOL ステートメントの効果はなくなります。
例：

```
TOLER 0.1                    TOLER 0.01
CYLIND 3.0, 1.0              CYLIND 3.0, 1.0
```





注記：指示文 RADIUS、RESOL、および TOLER は、曲線を使用して、円柱状の 3D 要素（CIRCLE、ARC、CYLIND、SPHERE、ELLIPS、CONE、ARMC、ARME、ELBOW、REVOLVE）および 2D ポリライン内の円弧の滑らかさを設定します。
「183 ページ「追加ステータスコード」」を参照してください。

PEN

PEN n

カラーを設定します。
パラメータの制限：
0 < n <= 255
デフォルト：
PEN 1
スクリプトに PEN ステートメントがない場合。
(ライブラリ部品では、ライブラリ部品設定からデフォルト値が読み取られます。スクリプトが存在していないインデックスを参照した場合、PEN 1 がデフォルト設定となります)

LINE_PROPERTY

LINE_PROPERTY expr

2D スクリプトでこの後に生成される、線の全ての特性を次の LINE_PROPERTY ステートメントまで定義します（RECT2、LINE2、ARC2、CIRCLE2、SPLINE2、SPLINE2A、POLY2、FRAGMENT2 コマンド）。デフォルトの値は generic です。
ステータスの値には以下のものがあります。
0: 全ての線が一般的な線

1: 全ての線が内側の線

2: 全ての線が輪郭線

[SET] STYLE

[SET] STYLE name_string

[SET] STYLE index

この後に生成される全てのテキストは、次の SET STYLE ステートメントまでこの文字スタイルを使用します。
index は、内部データ構造のスタイルスタックを参照する定数です（負のインデックスは、GDL スクリプトで前に定義された材質のデータ構造のインデックスを意味します）。このスタックは GDL 解析中に修正されますが、プログラム内からも修正することができます。スタイル名の代わりにインデックスを使用することは、IND 関数を先に使用している場合のみ推奨します。
デフォルト：

SET STYLE 0

スクリプトに SET STYLE ステートメントがない場合（アプリケーションフォント、サイズ = 5 mm、アンカ = 1、標準スタイル）。

3D スクリプトでのみ使用される指示文

MODEL

MODEL WIRE

MODEL SURFACE

MODEL SOLID

現在のスクリプトでの表現モードを設定します。
MODEL WIRE: ワイヤフレームのみで、表面も体積もありません。オブジェクトは透過です。

MODEL SURFACE、MODEL SOLID: 断面の表面は境界表面との関係に基づいて生成されるので、どちらの方法も同じ 3D 内部データ構造を生成します。オブジェクトは不透過です。

ボディの一部を切り取ったときのみ、以下のような違いが出ます。

MODEL SURFACE: ボディの内側が表示される

MODEL SOLID: 新規の表面が表示される

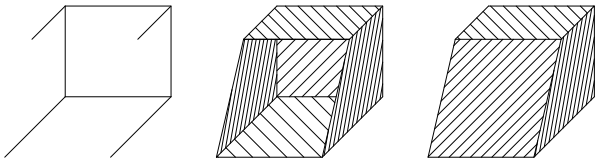
デフォルト:

MODEL SOLID:

3 つのモデリング方法を説明するため、3 つのブロックを例に挙げます。

```
MODEL WIRE
BLOCK 3,2,1
ADDY  4
MODEL SURFACE
BLOCK 3,2,1
ADDY  4
MODEL SOLID
BLOCK 3,2,1
```

平面を使用して切り取った結果:



[SET] MATERIAL

[SET] MATERIAL name_string

[SET] MATERIAL index

この後に生成される全ての表面は、次の MATERIAL ステートメントまでこの材質を表します。ただし、

BPRISM_, CPRISM_, FPRISM_, HPRISM_

SPRISM_, CSLAB_, CWALL_, BWALL_, XWALL_,

CROOF_, MASS

上記のボディ内の表面は、この規則に従いません。

index は、内部データ構造のマテリアルスタックを参照する定数です（負のインデックスは、GDL スクリプトで前に定義された材質のデータ構造のインデックスを意味します)。このスタックは GDL 解析中に修正されますが、プログラム内からも

修正することができます。材質名の代わりにインデックスを使用することは、IND 関数を先に使用している場合のみ推奨します。

index 0 には特別な意味があります。表面に現在のペンカラーが適用され、マットな表示になります。

デフォルト：

MATERIAL 0
スクリプトに MATERIAL ステートメントがない場合。
(ライブラリ部品では、ライブラリ部品設定からデフォルト値が読み取られます。スクリプトが存在していないインデックスを参照した場合、MATERIAL 0 がデフォルト設定となります)

SECT_FILL

SECT_FILL fill, fill_background_pen,
fill_pen, contour_pen

または

SECT_ATTRS fill, fill_background_pen,
fill_pen, contour_pen [, line_type]

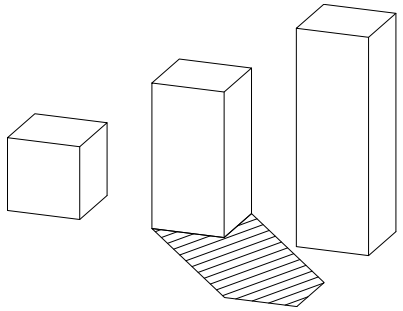
断面 / 立面ウィンドウおよび PROJECT2{3} コマンドで、3D 要素の切り取り部分に対して使用する属性を定義（互換性を保つために、PROJECT2 以前のバージョンでは無効)。

fill: 塗りつぶし名またはインデックス番号
fill_background_pen: 塗りつぶし背景ペンカラー番号
fill_pen: 塗りつぶしペンカラー番号
contour_pen: 塗りつぶし輪郭ペンカラー番号
line_type: ポリゴンの辺の線種

SHADOW

SHADOW keyword_1[, keyword_2]

レンダリングおよびベクトルシャドウ投射における要素のシャドウ投射を制御します。
keyword_1:ON、AUTO あるいは OFF
keyword_2: ON あるいは OFF
ON: この後の全ての要素は、全ての状況においてシャドウを投射します。
OFF: この後の全ての要素は、どのような状況においてもシャドウを投射しません。
AUTO: シャドウ投射は自動的に決まります。
・非表示の部品に対して SHADOW OFF を設定すると、メモリと処理時間の節約になります。
・SHADOW ON を設定すると、細かい部分もシャドウ投射されます。
省略可能な 2 番目のキーワードは、表面上のシャドウの外観を制御します。
・SHADOW keyword_1 の後の keyword_2 が OFF の場合、この後の表面でのベクトルシャドウ投射は使用不可になります。
・SHADOW keyword_1 の後の keyword_2 が ON の場合、背面のベクトルシャドウ投射はオンになります。
デフォルト:
SHADOW AUTO



```
SHADOW OFF
BRICK 1, 1, 1
ADDX 2
SHADOW ON
BRICK 1, 1, 2
ADDX 2
SHADOW OFF
BRICK 1, 1, 3
```

2D スクリプトでのみ使用される指示文

DRAWINDEX

DRAWINDEX number

2D スクリプト要素の描画順序を定義します。描画インデックスが小さい要素から描画されます。

パラメータの制限：

0 < number <= 50

(現在のバージョンの GDL では、10、20、30、40 および 50 の DRAWINDEX の値だけが有効です。その他の値はこれらの値に丸められます)

DRAWINDEX 指示文がない場合、デフォルトの描画順序は次のとおりです。

1 図形

2 塗りつぶし

3 線

4 テキスト要素

[SET] FILL

[SET] FILL name_string

[SET] FILL index

この後に生成される全ての 2D ポリゴンは、次の SET FILL ステートメントまでこの塗りつぶしを表します。

index は、内部データ構造の塗りつぶしスタックを参照する定数です。このスタックは GDL 解析中に修正されますが、プログラム内からも修正することができます。塗りつぶしの名前の代わりにインデックスを使用することは、IND 関数を先に使用しているときのみ推奨します。

デフォルト：

SET FILL 0

つまり、スクリプトに SET FILL ステートメントがない場合は、空の塗りつぶしとなります。

[SET] LINE_TYPE

[SET] LINE_TYPE name_string

[SET] LINE_TYPE index

この後に生成される全ての 2D 線は、次の SET LINE_TYPE ステートメントまで（線、円弧、ポリラインで）この線種を表します。
index は、内部データ構造の線種スタックを参照する定数です。このスタックは GDL 解析中に修正されますが、プログラム内からも修正することができます。線種名の代わりにインデックスを使用することは、IND 関数を先に使用しているときのみ推奨します。
デフォルト：
SET LINE_TYPE 1
つまり、スクリプトに SET LINE_TYPE ステートメントがない場合、実線になります。

インライン属性定義

属性は、材質、塗りつぶし、線種のダイアログボックスで作成できます。これらの平面図属性は、どの GDL スクリプトからでも参照できます。
属性は、GDL スクリプトでも定義できます。以下の 2 つの状況が想定されます。

- MASTER_GDL スクリプト内での属性定義。MASTER_GDL スクリプトは、このスクリプトを含むライブラリがメモリにロードされるときに解釈されます。MASTER_GDL 属性は、平面図の属性と結合されます。同じ名前を持つ属性は置き換えられません。MASTER_GDL がロードされたら、その中で定義されている属性をスクリプトから参照できます。
- ライブラリ部品内での属性定義。この方法で定義した材質とテクスチャは、スクリプトと、そこからコールされるスクリプトで使用できます。2D スクリプトで定義され、使用される塗りつぶし種類と線種は、MASTER_GDL スクリプトで定義されているかのように動作します。

スクリプトウィンドウの［**GDL スクリプトのチェック**］コマンドを使うと、材質、塗りつぶし、線種、またはスタイルのパラメータが正しいかどうかを検証できます。
ライブラリ部品の 3D 解釈で使用される材質、塗りつぶし、線種、スタイルが目的のものと異なり、エラーメッセージも表示されない場合には、いずれかのパラメータ値が間違っている可能性があります。［**GDL スクリプトのチェック**］コマンドを使えば、詳細なメッセージが表示されるので、パラメータを調べることができます。

材質

DEFINE MATERIAL

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

注記： このコマンドには、追加のデータ定義を含めることができます。
詳細は、222 ページ「追加データ」を参照してください。

全ての GDL スクリプトには、材質名を参照する前に、その材質の定義を含めることができます。この材質は、これが定義されたスクリプトとそこからコールされるスクリプトの 3D 要素に対してのみ使用できます。

name: 材質の名前。

type: 材質タイプ。材質を定義するパラメータの実際の番号 (n) は、タイプによって異なります。パラメータの意味と制限については、「例」のコメントで説明しています。

0: 一般定義、n = 161:

1: 単純定義、n = 9 (その他のパラメータは定数または与えられた値で計算されます)

2-7: 定義済みの材質タイプ、n=3
3 つの値は表面カラーの RGB 構成要素です。その他のパラメータは定数か、カラーから計算されます。
2: マット
3: 金属
4: プラスチック
5: ガラス
6: 放射
7: 定数

10: 塗りつぶしパラメータを持つ一般定義、n = 17

11: 塗りつぶしパラメータを持つ単純定義、n = 10

12-17: 塗りつぶしパラメータを持つ、定義済みの材質タイプ、n = 4

20: 塗りつぶし、塗りつぶしのカラーインデックス、テクスチャパラメータのインデックスを持つ一般定義、n = 19

21: 塗りつぶし、塗りつぶしのカラーインデックス、テクスチャパラメータのインデックスを持つ単純定義、n = 12

22-27: 塗りつぶし、塗りつぶしのカラーインデックス、テクスチャパラメータのインデックスを持つ定義済の材質タイプ、n = 6

20-27 タイプの特別な意味：
ペン番号がゼロの場合、ベクトルハッチングはアクティブなペンで生成されます。
テクスチャインデックスの値をゼロにすると、ベクトルハッチングまたはテクスチャなしで材質を定義できます。

例：

```
DEFINE MATERIAL "water" 0,
    0.5284, 0.5989, 0.6167,
!   表面 RGB [0.0..1.0]
    1.0, 0.5, 0.5, 0.9,
!   環境、拡散、鏡面、透過
!   係数 [0.0..1.0]
2.0,
```

```
!    光沢 [0.0..100.0]
     1,
!    透過減衰量 [0.0..4.0]
     0.5284, 0.5989, 0.6167,
!    鏡面 RGB [0.0..1.0]
     0, 0, 0,
!    放射 RGB [0.0..1.0]
     0.0
!    放射減衰量 [0.0..65.5]
DEFINE MATERIAL "asphalt" 1,
     0.1995, 0.2023, 0.2418,
!    表面 RGB [0.0..1.0]
     1.0, 1.0, 0.0, 0.0,
!    環境、拡散、鏡面、透過
!    係数 [0.0..1.0]
     0,
!    光沢 [0..100]
     0
!    透過減衰量 [0..4]
DEFINE MATERIAL "matte red" 2,
     1.0,  0.0,  0.0
!    表面 RGB [0.0..1.0]
DEFINE MATERIAL "Red Brick" 10,
     0.878294, 0.398199, 0.109468,
     0.58, 0.85, 0.0, 0.0,
     0,
     0.0,
     0.878401, 0.513481, 0.412253,
     0.0, 0.0, 0.0,
     0,
IND(FILL, "common brick")
!    塗りつぶしインデックス
DEFINE MATERIAL "Yellow Brick+*" 20,
     1, 1, 0,
!    表面 RGB [0.0 .. 1.0]
     0.58,  0.85,  0,  0,
!    環境、拡散、鏡面、透過
!    係数 [0.0 .. 1.0]
     0,
!    光沢 [0.0 .. 100.0]
     0,
!    透過減衰量 [0.0 .. 4.0]
```

```
    0.878401,  0.513481,  0.412253,
!   鏡面 RGB [0.0 .. 1.0]
    0, 0, 0,
!   放射 RGB [0.0 .. 1.0]
    0,
!   放射減衰量 [0.0 .. 65.5]
    IND(FILL, "common brick"), 61,
    IND(TEXTURE, "Brick")
!   塗りつぶしインデックス、カラーインデックス、テクスチャインデックス、
```

DEFINE MATERIAL BASED_ON

DEFINE MATERIAL name [,] **BASED_ON** orig_name [,] **PARAMETERS** name1 = expr1 [, ...][[,] **ADDITIONAL_DATA** name1 = expr1 [, ...]]

既存の材質に基づいた材質の定義。オリジナルの材質の指定されたパラメータが新規の値で上書きされ、その他のパラメータはそのままになります。実際のパラメータがないコマンドを使用すると、オリジナルとまったく同じ材質になります。ただし、材質の名前は変更されます。材質のパラメータ値は、"REQUEST{2} ("Material_info", name_or_index, param_name, value_or_values)" 関数を使って取得できます。

orig_name: オリジナルの材質の名前（既存の GDL で定義済み、または平面図の材質の名前）

namei: 新規の値で上書きされる材質のパラメータ名。材質の定義のパラメータに対応する名前。

gs_mat_surface_r, gs_mat_surface_g, gs_mat_surface_b（表面 RGB [0.0..1.0]）
gs_mat_ambient（環境 [0.0..1.0]）
gs_mat_diffuse（拡散 [0.0..1.0]）
gs_mat_specular（鏡面 [0.0..1.0]）
gs_mat_transparent（透過 [0.0..1.0]）
gs_mat_shining（光沢 [0.0..100.0]）
gs_mat_transp_att（透過減衰量 [0.0..4.0]）
gs_mat_specular_r, gs_mat_specular_g, gs_mat_specular_b（鏡面カラー RGB[0.0..1.0]）
gs_mat_emission_r, gs_mat_emission_g, gs_mat_emission_b（放射カラー RGB[0.0..1.0]）
gs_mat_emission_att（放射減衰量 [0.0..65.5]）
gs_mat_fill_ind（塗りつぶしインデックス）
gs_mat_fillcolor_ind（塗りつぶしカラーインデックス）
gs_mat_texture_ind（テクスチャインデックス）

expri: 材質の指定されたパラメータを上書きする新規の値。値の範囲は、材質の定義のときと同じです。

例：

```
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

全てのGDLスクリプトには、テクスチャ名を参照する前に、そのテクスチャの定義を含めることができます。このテクスチャは、これが定義されたスクリプトとそこからコールされるスクリプトでのみ使用できます。

name: テクスチャの名前

expression: テクスチャに関連付けられた画像。文字列式は画像ファイル名、数式はライブラリ部品に格納されている画像のインデックスを意味します。0のインデックスは特殊な値です。この値は、ライブラリ部品のプレビュー画像を参照します。

x: テクスチャの論理幅

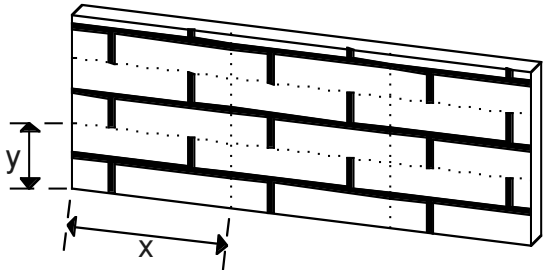
y: テクスチャの論理高さ

mask: j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8 + 256*j9　ここで、j1、j2、j3、j4、j5、j6、j7、j8、j9は0または1をとります。

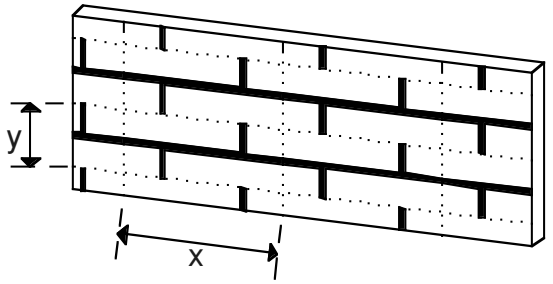
Alpha channel controls (j1... j6):
j1: アルファチャネルによってテクスチャの透過が変化
j2: バンプマッピングつまり表面法線のかく乱。
バンプマッピングでは、アルファチャネルを使用して、表面法線の振幅を定義します。
j3: アルファチャネルがテクスチャの拡散カラーを変化させる
j4: アルファチャネルがテクスチャの鏡面カラーを変化させる
j5: アルファチャネルがテクスチャの環境カラーを変化させる
j6: アルファチャネルがテクスチャの表面カラーを変化させる

接続の制御 (j7... j9):

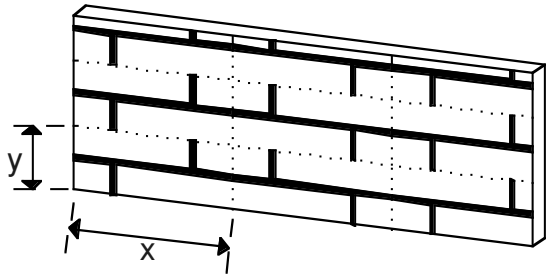
値がゼロの場合、標準モードが選択されます。



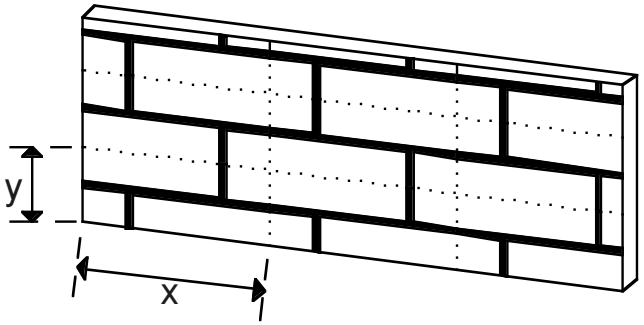
j7: テクスチャはランダムにシフトされます。



j8: ‘x’方向にミラー



j9: ‘y’方向にミラー



angle: 回転の角度

例:

```
DEFINE TEXTURE "Brick" "Brick.PICT", 1.35, 0.3, 256+128, 35.0
```

塗りつぶし

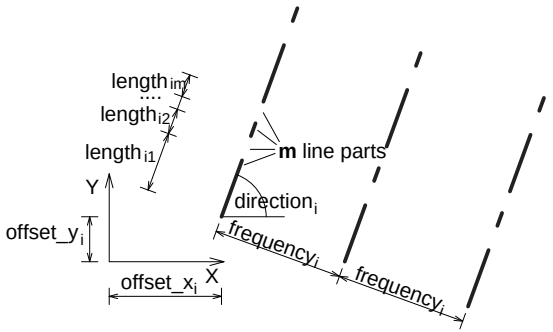
DEFINE FILL

```
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
```

注記: このコマンドには、追加のデータ定義を含めることができます。

詳細は、222 ページ「追加データ」を参照してください。

全ての GDL スクリプトには、塗りつぶし名を参照する前に、その塗りつぶしの定義を含めることができます。このようにして定義された塗りつぶしは、これが定義されたスクリプトとそこからコールされるスクリプトでのみ使用できます。



name: 塗りつぶしの名前

fill_types = j1 + 2 * j2 + 4 * j3

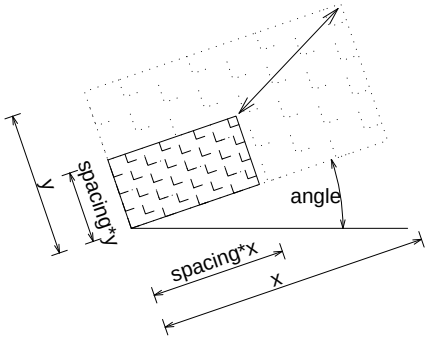
j1: 切断塗りつぶし

j2: 表面塗りつぶし

j3: 作図塗りつぶし

j ビットがセットされている場合、指定されたタイプに対応して、定義された塗りつぶしを ArchiCAD で使用できます。デフォルトは全ての塗りつぶし (0) です。

pattern definition: pattern1、pattern2、pattern3、pattern4、pattern5、pattern6、pattern7、pattern8: 0 と 255 の間のバイナリ値を示す 8 つの番号 塗りつぶしのビットマップパターンを定義します。



spacing: ハッチング間隔 - 塗りつぶし全体のグローバルスケール係数を定義します。全ての値はこの数値によって x 方向と y 方向で乗算されます。

angle: グローバル回転角（度単位）

n: ハッチングラインの数

frequencyi: ラインの出現頻度（2 つの線の間隔は spacing*frequencyi）

diri: ラインの方向角度（度単位）
offset_xi, offset_yi: 原点からのラインのオフセット

mi: ラインの本数

lengthij: ラインの長さ（実際の長さは spacing * lengthij)。ラインは線分とスペースが交互に組み合わされたものです。ラインの最初の部分が線分で、長さがゼロのときには点になります。

ビットマップパターンは、pattern1 ～ pattern8 パラメータでのみ定義され、[オプション] メニューの [表示オプション] の [ポリゴンの塗りつぶし] が [ビットマップパターン] に設定されている場合のみ使用されます。これを定義するには、塗りつぶしの最小単位を選択してから、8 x 8 の位置がある矩形グリッドを使用して、ドットおよび空のスペースとして表現します。8 パターンのパラメータは、グリッドの線の中でバイナリ値の 10 進表現になります（1 つのドットは 1、空のスペースは 0)。

ベクトルハッチングは、塗りつぶし定義の 2 番目の部分によって定義されます。与えられた頻度（frequencyi）で反復される破線の集まりとして定義されます。破線の集まりの各ラインは、その方向（directioni)、原点からのオフセット (offset_xi, offset_yi)、および与えられた長さ（lengthij）の線分とスペースの反復を含む破線定義によって記述されます。

注記： DEFINE FILL コマンドでは、単純な塗りつぶししか定義できません。シンボル塗りつぶしは定義できません。

例：

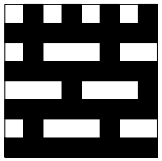
```
DEFINE FILL  "brick" 85, 255, 136, 255,
         34, 255, 136, 255,
         0.08333, 0.0, 4,
         1.0, 0.0, 0.0, 0.0, 0,
         3.0, 90.0, 0.0, 0.0, 2,
         1.0, 1.0,
         3.0, 90.0, 1.5, 1.0, 4,
         1.0, 3.0, 1.0, 1.0,
         1.5, 90.0, 0.75, 3.0, 2,
         1.0, 5.0
```

ビットマップパターン：

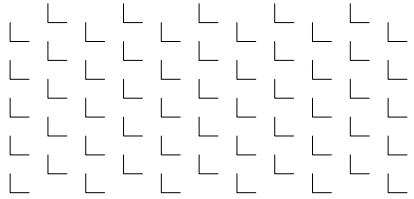
```
パターン： バイナリ値：
pattern1 = 85     01010101     ・・・・
pattern2 = 255    11111111    ・・・・・・・・
pattern3 = 136    10001000    ・   ・
pattern4 = 255    11111111    ・・・・・・・・
pattern5 = 34     00100010      ・   ・
```

```
pattern6 = 255    11111111    ........
pattern7 = 136     10001000    .   .
pattern8 = 255    11111111    ........
```

ビュー：



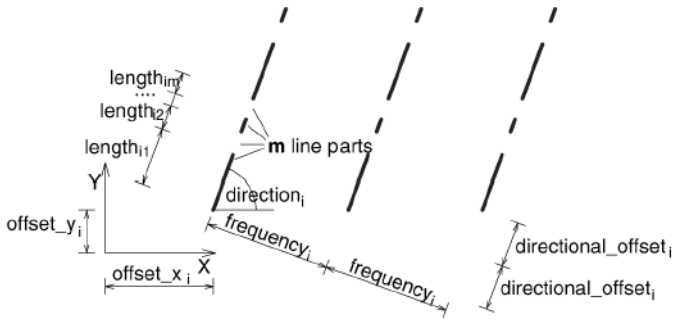
ベクトルハッチング：



DEFINE FILLA

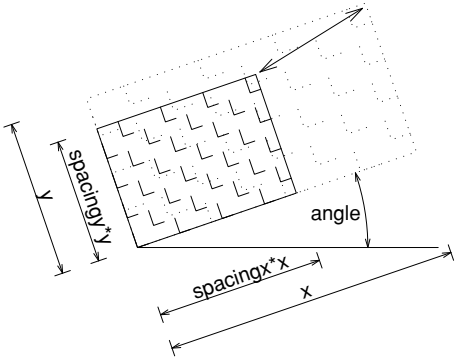
DEFINE FILLA name [,] [**FILLTYPES_MASK** fill_types,] pattern1, pattern2, pattern3, pattern4, pattern5, pattern6, pattern7, pattern8, spacing_x, spacing_y, angle, n, frequency1, directional_offset1, direction1, offset_x1, offset_y1, m1, length11, ... length1m, ... frequencyn, directional_offsetn, directionn, offset_xn, offset_yn, mn, lengthn1, ... lengthnm

注記： このコマンドには、追加のデータ定義を含めることができます。
詳細は、222 ページ「追加データ」を参照してください。



DEFINE FILL ステートメントの拡張版。

追加パラメータ：



spacing_x, spacing_y: x 方向および y 方向それぞれの間隔係数。この 2 つのパラメータは、塗りつぶし全体のグローバルスケール係数を定義します。x 方向の全ての値は spacing_x で乗算され、y 方向の全ての値は spacing_y によって乗算されます。

directional_offseti: ラインの方向に沿って計測される次の同じようなハッチングラインの開始部分のオフセット。それぞれのラインは、frequencyi によって定義される距離をとって、directional_offseti によって定義されるオフセットで描画されます。オフセットの実際の長さは、spacing x directional_offseti になります。

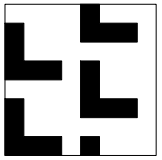
例：

```
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
```

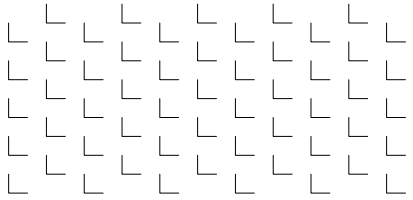
ビットマップパターン：

パターン：	バイナリ値：	
pat1 = 8	00001000	.
pat2 = 142	10001110	・ ・・・
pat3 = 128	10000000	・
pat4 = 232	11101000	・・・ ・
pat5 = 8	00001000	・
pat6 = 142	10001110	・ ・・・
pat7 = 128	10000000	・
pat8 = 232	11101000	・・・ ・

ビュー：



ベクトルハッチング：

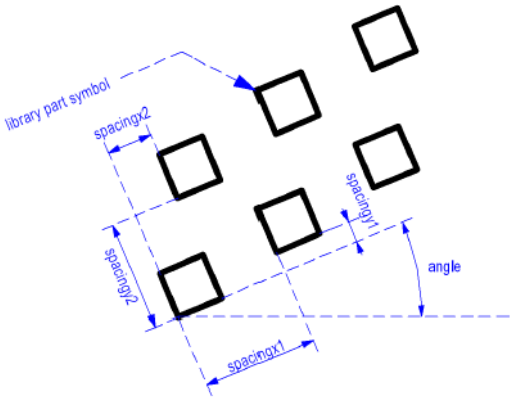


DEFINE SYMBOL_FILL

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]

注記： このコマンドには、追加のデータ定義を含めることができます。

詳細は、222 ページ「追加データ」を参照してください。



DEFINE FILL ステートメントの拡張版で、塗りつぶし定義にライブラリ部品の図面を含めることができます。macro_name の使い方とパラメータは、CALL コマンドと同じです。

パラメータ：

spacingx1, spacingx2: 垂直方向の間隔

spacingy1, spacingy2: 水平方向の間隔

scaling2: 水平方向のスケール

scaling2: 垂直方向のスケール

macro_name: ライブラリ部品の名前

DEFINE SOLID_FILL

DEFINE SOLID_FILL name [[,] FILLTYPES_MASK fill_types]]

無地塗りつぶしを定義します。

注記： このコマンドには、追加のデータ定義を含めることができます。

詳細は、222 ページ「追加データ」を参照してください。

DEFINE EMPTY_FILL

DEFINE EMPTY_FILL name [[,] FILLTYPES_MASK fill_types]

空の塗りつぶしを定義します。

注記： このコマンドには、追加のデータ定義を含めることができます。

詳細は、222 ページ「追加データ」を参照してください。

DEFINE LINEAR_GRADIENT_FILL

DEFINE LINEAR_GRADIENT_FILL name [[,] FILLTYPES_MASK fill_types]

線形グラデーションを定義します。

DEFINE RADIAL_GRADIENT_FILL

DEFINE RADIAL_GRADIENT_FILL name [[,] FILLTYPES_MASK fill_types]

円形グラデーションを定義します。

DEFINE TRANSLUCENT_FILL

```
DEFINE TRANSLUCENT_FILL name [[,] FILLTYPES_MASK fill_types]
      pat1, pat2, pat3, pat4, pat5, pat6, pat7, pat8,
      percentage
```

指定されたパーセント値での定義に従って、背景色と前景色を混合して表示する塗りつぶしを定義します。

percentage: 前景色の量；0 は背景色のみを表示し（空白塗りつぶしと同様)、100 は前景色のみを表示します（無地塗りつぶしと同様)。

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
```

画像パターンに基づいた塗りつぶしを定義します。

image_name: 現在のライブラリにロードされているパターン画像の名前。

image_vert_size, image_hor_size: パターンのモデルサイズ。

image_mask: タイリング指示文。

image_rotangle: 標準座標システムからのパターンの回転角度。

線種

DEFINE LINE_TYPE

DEFINE LINE_TYPE name spacing, n,
lengthl, ... lengthn

注記： このコマンドには、追加のデータ定義を含めることができます。

詳細は、222 ページ「追加データ」を参照してください。

全ての GDL スクリプトには、線種名を参照する前に、その線種の定義を含めることができます。このようにして定義された線種は、これが定義されたスクリプトとそこからコールされるスクリプトの 2D 要素でのみ使用できます。

name: 線種の名前

spacing: 間隔係数

n: ラインの本数

lengthij: ラインの長さ（実際の長さは spacing * lengthij）。ラインは線分とスペースが交互に組み合わされたものです。ラインの最初の部分が線分で、長さがゼロのときには点になります。

注記: DEFINE LINE_TYPE コマンドでは単純な線種、つまり線分とスペースのみで構成されたラインを定義できます。シンボル線は定義できません。

例：

```
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]

注記： このコマンドには、追加のデータ定義を含めることができます。

詳細は、222 ページ「追加データ」を参照してください。

DEFINE LINE ステートメントの拡張版で、ラインの定義にライブラリ部品図面を含めることができます。macro_name の使い方とパラメータは、CALL コマンドと同じです。

パラメータ：

dash: 両方のライン構成要素のスケール

gap: 各構成要素間の間隔

スタイル

DEFINE STYLE

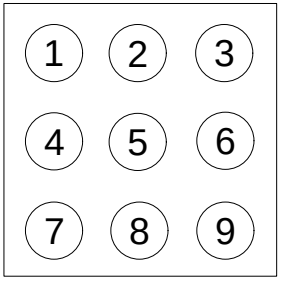
DEFINE STYLE name font_family, size, anchor, face_code

TEXT2 コマンドおよび TEXT コマンドの使用をお勧めします。
GDL スクリプトには、スタイル名を参照する前に、そのスタイル定義を含めることができます。このようにして定義されたスタイルは、これが定義されたスクリプトとそこからコールされるスクリプトでのみ使用できます。
name: スタイルの名前

font_family: 使用されているフォントファミリの名前（Garamond など）

size: 文字「l」の高さ（ペーパースペースでのミリメートル、またはモデルスペースでのメートル単位）

定義されたスタイルが TEXT2 コマンドおよび TEXT コマンドと共に使用されている場合、size はミリメートル単位の文字の高さになります。
RICHTEXT2 コマンドおよび RICHTEXT コマンドの文字列 PARAGRAPH と共に使用されている場合、size は、TEXTBLOCK 定義の fixed_height パラメータに応じてミリメートルまたはメートルになります。輪郭とシャドウの face_code およびアンカの値は無効です。
anchor: テキスト配置位置のコード



face_code: 次の値の組み合わせ：

0	normal
1	**bold**
2	*italic*
4	underline
8	outline
16	shadow

注記： 輪郭およびシャドウの値は、Macintosh プラットフォームおよび ArchiCAD バージョン 8.1 までででのみ有効です。

DEFINE STYLE {2}

DEFINE STYLE{2} name font_family, size, face_code

スタイル定義の新規のバージョンは、PARAGRAPH 定義と共に使用することをお勧めします。

name: スタイルの名前

font_family: 使用されているフォントファミリの名前（Garamond など）

size: 文字列の高さ（モデルスペースでの mm または m 単位）

face_code: 次の値の組み合わせ：

0 normal
1 **bold**
2 *italic*
4 underline
32 superscript
64 subscript
128 ~~strikethrouth~~

定義されたスタイルが TEXT2 コマンドと共に使用されている場合、size はミリメートル単位の文字の高さになります。上付き文字、下付き文字および取り消し線の face_code 値は無効です。RICHTEXT2 コマンドおよび RICHTEXT コマンドで文字列 PARAGRAPH と共に使用されている場合、size は、TEXTBLOCK 定義の fixed_height パラメータに応じてミリメートルまたはメートルになります。

PARAGRAPH

PARAGRAPH

```
PARAGRAPH name alignment, firstline_indent,
      left_indent, right_indent, line_spacing [,
      tab_size1, ...]
      [PEN index]
      [[SET] STYLE style1]
      [[SET] MATERIAL index]
      'string1'
      'string2'
      ...
      'string n'
      [PEN index]
      [[SET] STYLE style2]
      [[SET] MATERIAL index]
```

```
'string1'
'string2'
...
'string n'
...
```

ENDPARAGRAPH
GDL スクリプトには、スタイル名を参照する前に、そのスタイル定義を含めることができます。このようにして定義されたスタイルは、これが定義されたスクリプトとそこからコールされるスクリプトでのみ使用できます。段落は、属性が異なる（スタイル、ペンおよび材質（3D））それぞれ最大 256 文字までの任意の数の文字列として定義されます。段落定義内に属性が指定されていない場合、実際の（またはデフォルトの）属性が使用されます。文字列は、段落文字列に（特殊文字'¥n'を使って）含まれる新規のラインによって自動的に、1 行ずつの同一の段落に分割されます。段落定義は、TEXTBLOCK コマンドの名前で参照できます。長さタイプのパラメータは全て（firstline_indent、left_indent、right_indent、tab_position)、TEXTBLOCK 定義の fixed_height パラメータに応じてミリメートルまたはメートルになります。

name: 段落の名前

alignment: 段落文字列の整列。有効値：
1: 左揃え、2: 中央揃え、3: 右揃え、4: 両端揃え

firstline_indent: 最初の行のインデント（モデルスペースの mm または m 単位）

left_indent: 左インデント（モデルスペースの mm または m）

right_indent: 右インデント（モデルスペースの mm または m 単位）

line_spacing: 文字間隔係数。実際のスタイルで定義される行間のデフォルト距離（文字サイズ + 次の行までの距離）にこの数値が乗算されます。

tab_positioni: 連続したタブ位置（それぞれ段落の冒頭に対する）（モデルスペースの mm または m)。段落文字列のタブがこの位置にスナップされます。タブ位置が指定されていない場合、デフォルト値が使用されます（12.7 mm)。

テキストブロック

TEXTBLOCK

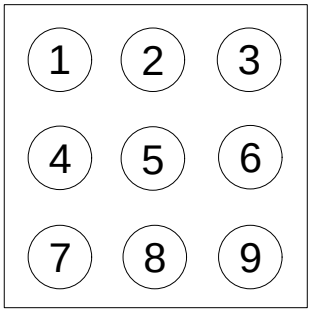
TEXTBLOCK name width, anchor, angle, width_factor, charspace_factor, fixed_height, 'string_expr1' [, 'string_expr2', ...]

テキストブロックの定義。GDL スクリプトには、スタイル名を参照する前に、そのスタイル定義を含めることができます。このようにして定義されたスタイルは、これが定義されたスクリプトとそこからコールされるスクリプトでのみ使用できます。テキストブロックは、*167 ページ「RICHTEXT2」*と *123 ページ「RICHTEXT」*を使用して配置できる、任意の数の文字列ま

たは段落として定義されます。”REQUEST ("TEXTBLOCK_INFO", textblock_name, width, height)”を使用すると、計算された TEXTBLOCK の幅および高さに関する情報を取得できます。

width: テキストブロックの幅（モデルスペースの mm または m)。0 の場合、自動的に計算されます。

anchor: テキスト配置位置のコード



angle: テキストブロックの回転角（度単位）

width_factor: 文字幅係数。実際のスタイルによって定義された文字幅に、この数値が乗算されます。

charspace_factor: 文字間隔係数。水平方向に並んだ 2 つの文字間の距離に、この数値が乗算されます。

fixed_height: 有効値

1: 配置された TEXTBLOCK はスケールに左右されず、指定された長さタイプのパラメータは全てミリメートルになります。

0: 配置された TEXTBLOCK は、スケールによって異なり、指定された長さタイプのパラメータは全てモデルスペースのメートルになります。

string_expri: 以前定義済みであれば段落の名前に、定義されていなければ単なる文字列になります（デフォルトの段落パラメータあり)。

追加データ

属性定義には、ADDITIONAL_DATA キーワードの後に省略可能な追加データ定義を含めることができます。追加データは、属性コマンドの既に定義されているパラメータの後に入力する必要があります。追加のデータには名前（namei）と値（valuei）があります。これは、いずれかのタイプの式ですが、配列にすることもできます。文字列パラメータ名が従属文字列「_file」で終わっている場合、その値はファイル名として扱われ、アーカイブプロジェクトに含められます。ArchiCAD または ArchiCAD のアドオンによって、追加データの別の意味が定義、使用されます。

LightWorks アドオンパラメータの意味については、
http://www.graphisoft.com/support/developer/documentation/LibraryDevDoc/13（英文）を参照してください。

追加データ定義は、以下のコマンドで使用可能です。

DEFINE MATERIAL

DEFINE MATERIAL parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE FILL

DEFINE FILL parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE FILLA

DEFINE FILLA parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE SYMBOL_FILL

DEFINE SYMBOL_FILL parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE LINE_TYPE

DEFINE LINE_TYPE parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE SYMBOL_LINE

DEFINE SYMBOL_LINE parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

外部ファイルの依存性

FILE_DEPENDENCE ″name1″ [, ″name2″, ...]

GDL スクリプトが依存している外部ファイルのリストを出すことができます。ファイル名は定数文字列である必要があります。
ここで指定した全てのファイルはアーカイブプロジェクトに含められます（CALL ステートメントで使用される定数マクロ名やさまざまな GDL コマンドで使用される定数画像名など)。このコマンドはこのレベルでのみ機能します。指定されたファイルがライブラリ部品の場合、そのコールされたマクロファイルは含まれません。
このコマンドは、外部ファイルが GDL スクリプトのカスタム位置で参照する場合に便利です。例えば、ファイル操作における、ADDITIONAL_DATA ファイルパラメータ、データファイル

非ジオメトリックスクリプト

GDL オブジェクトの外観を定義する 3D および 2D のスクリプトウィンドウ以外にも、GDL オブジェクトに補完情報を追加するためのスクリプトも用意されています。これらのスクリプトには、数量計算のための特性スクリプト、さまざまなパラメータの有効値のリストを含むパラメータスクリプト、パラメータ入力用のカスタムインターフェイスを作成するユーザーインターフェイススクリプトがあります。ここでは、これらのスクリプトタイプ全てで使用可能なコマンドの詳細を説明しています。

特性スクリプト

ライブラリ部品には特性スクリプト用に予約されている GDL ウィンドウがあります。このスクリプトでは、ライブラリ部品の特性をパラメータ従属にすることができます。また指示文を使用して、最終構成要素リストでの特性の位置を定義できます。いくつかのコマンドを使用することによって、スクリプトのローカル記述項目と構成要素（旧バージョンの ArchiCAD の［特性］ウィンドウで作成した）で定義することができます。外部データベースで使用されている記述項目と構成要素を参照することもできます。コード長は、32 文字以下でなければなりません。
特性スクリプトでは、形状を生成しない GDL コマンドを使用できます。

DATABASE_SET

DATABASE_SET set_name [descriptor_name, component_name, unit_name, key_name, criteria_name, list_set_name]

データベースセットの定義またはデータベースセットの選択。このコマンドを MASTER_GDL スクリプト内に配置すると、記述項目、構成要素、単位、キー、条件、一覧表のファイルを含むデータベースセットが定義されます。
このデータベースセットの名前は次に、REF COMPONENT と REF DESCRIPTOR が参照する実際のデータベースセットを選択して、指示文として set_name パラメータだけを持つ同じコマンドを使って特性スクリプトから参照できます。デフォルトのデータセット名は「デフォルトセット（"Default Set"）」で、ほかのセットが選択されなかった場合に使用されます。デフォルトのデータベースセットのファイル名は、DESCDATA、COMPDATA、COMPUNIT、LISTKEY、LISTCRIT、LISTSET です。ArchiCAD のローカライズ版では、これらの名前は全て翻訳されています。
スクリプトには任意の数の DATABASE_SET の選択を含めることができます。
set_name：データベースセット名
descriptor_name：記述項目データファイル名
component_name：構成要素データファイル名
unit_name：単位データファイル名
key_name：キーデータファイル名
criteria_name：条件ファイル名
list_set_name：一覧表のファイル名

DESCRIPTOR

DESCRIPTOR name [,code, keycode]

ローカル記述項目の定義。スクリプトには任意の数の DESCRIPTOR を含めることができます。

name: 複数行に拡張できます。改行は文字「¥n」で、タブは文字「¥t」で定義できます。行の終わりに「¥」を追加すると、改行せずに文字列を次の行につなげることができます。文字列内で文字「¥」を重ねると（¥¥)、制御の意味がなくなり、単に「¥」と解釈されます。
文字列の長さは、改行文字も含めて 255 文字以下でなければなりません。これを超えた文字は、コンパイラに切り捨てられます。長いテキストが必要な場合は、複数の DESCRIPTOR を使用します。

code: 文字列。記述項目のコードを定義します。

keycode: 文字列。外部データベースのキーを参照します。

このキーは記述項目に割り当てられます。

REF DESCRIPTOR

REF DESCRIPTOR code [, keycode]

コードとキーコード文字列による外部データベースの記述項目への参照。

COMPONENT

COMPONENT name, quantity, unit [, proportional_with, code, keycode, unitcode]

ローカル構成要素の定義。スクリプトには任意の数の COMPONENT を含めることができます。

name: 構成要素の名前（128 文字以下）

quantity: 数量を示す数式

unit: 単位の記述に使用される文字列

proportional_with: 1 〜 6 の範囲のコード リストする時は、上で定義されている構成要素の数量が、現在リストされている要素用に計算された値で自動的に乗算されます。
1: 項目
2: 長さ
3: 表面 A
4: 表面 B
5: 表面
6: 体積

code: 文字列。構成要素のコードを定義します。

keycode: 文字列。外部データベースのキーを参照します。キーは構成要素に割り当てられます。

unitcode: 文字列。構成要素数量の出力形式を制御する外部データベース内の単位を参照します。この文字列によって、ローカルに定義されている単位文字列が置換されます。

REF COMPONENT

REF COMPONENT code [, keycode [, numeric_expression]]

コードとキーコード文字列による外部データベースの構成要素への参照。構成要素データベース内で乗算する値は、ここで指定する省略可能な数式で上書きすることができます。

BINARYPROP

BINARYPROP

BINARYPROP は、構成要素および記述項目のセクション内のライブラリ部品で定義されているバイナリ特性データ（構成要素と記述項目）への参照です。
DATABESE_SET 指示文は、バイナリデータには有効ではありません。

SURFACE3D ( )

SURFACE3D ( )

SURFACE3D () 関数は、ライブラリ部品の 3D 形状の表面積を返します。
警告： 同じパラメータを使用して、複数の形状を同じ場所に配置すると、この関数は全ての形状の総面積を返します。

VOLUME3D ( )

VOLUME3D ( )

VOLUME3D () 関数は、ライブラリ部品の 3D 形状の体積を返します。
警告： 同じパラメータを使用して、複数の形状を同じ場所に配置すると、この関数は全ての形状の総体積を返します。

POSITION

POSITION position_keyword

構成要素リスト内でのみ有効です。
以下の記述項目と構成要素に対応付けられている要素のタイプだけを変更します。特性スクリプトにそのような指示文がない場合は、記述項目と構成要素が、デフォルトの要素タイプと共にリストされます。
以下のキーワードがあります。

WALLS
COLUMNS
BEAMS
DOORS
WINDOWS
OBJECTS
CEILS

PITCHED_ROOFS
LIGHTS
HATCHES
ROOMS
MESHES
指示文は次の指示文が与えられるまで、それ以降の DESCRIPTOR と COMPONENT でそのまま有効になります。スクリプトにはいくつでも指示文を記述できます。

例：

```
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 ()          ！ワードローブ表面
COMPONENT "glue", 1.5, "kg"
COMPONENT "handle",2*c,"nb"！ドアの c の数
COMPONENT "paint", 0.5 * s, "kg"
POSITION WALLS
REF COMPONENT "0002"
```

DRAWING

DRAWING

DRAWING: 同じライブラリ部品の 2D スクリプトに記述されている図面を参照します。図面を材質表に配置するときに使用します。

パラメータスクリプト

パラメータリストは、有効な数値または文字列値のセットです。パラメータリストは、ライブラリ部品のパラメータスクリプトまたは MASTER_GDL スクリプトに定義されているとおりに、パラメータに適用することができます。パラメータのタイプは単純タイプにする必要があります。タイプの互換性は、GDL コンパイラによって確認されます。

パラメータスクリプトは、値リストタイプのパラメータ値が変更されるたびに解釈され、スクリプトに定義されている有効値がポップアップメニューに表示されます。

VALUES

VALUES "fillparam_name" [[,] FILLTYPES_MASK fill_types,] [,]value_definition1
[, value_definition2, ...]

name: パラメータ名

fill_types = j1 + 2 * j2 + 4 * j3

j1: 切断塗りつぶし

j2: 表面塗りつぶし

j3: 作図塗りつぶし

塗りつぶし種類のパラメータにのみ使用できます。j ビットが設定されている場合、「fillparam_name」パラメータに対応する塗りつぶしポップアップには自動的に、指定したタイプの塗りつぶしだけが入ります。デフォルトは全ての塗りつぶし (0) です。

value_definitioni: 値の定義は、次のように行います。

expressioni: 数式または文字列式、または

CUSTOM: キーワード。どのカスタム値でも入力できることを意味します。または、

RANGE left_delimiter [$expression_1$], [$expression_2$]
right_delimiter [**STEP** step_start_value,
step_value]

range definition, with optional step
left_delimiter: [, meaning '>=', or (, meaning '>'
expression1: lower limit expression
expression2: upper limit expression
right_delimiter:], meaning '<=', or), meaning '< '
step_start_value: starting value
step_value: step value

例：

```
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
! ファイルから全ての文字列の値を読み取り、
! 値リスト DIM sarray[] でその値を使用する例

! パラメータデータファイル名
 = "ProjectNotes.txt"
ch1 = OPEN ("text", filename, "MODE=RO, LIBRARY")
i = 1
j = 1
sarray[1] = "" を含むライブラリ内のファイル
! 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 の全ての文字列を収集
! ファイル
VALUES "RefNote" sarray から読み取った文字列を持つパラメータポップアップ
```

PARAMETERS

```
PARAMETERS name1 = expression1 [,
      name2 = expression2, ...,
      namen = expressionn]
```

namei: パラメータ名

expressioni: パラメータの新規値

このコマンドを使用して、ライブラリ部品のパラメータ値をパラメータスクリプトによって修正できます。
修正は次の解釈についてのみ有効になります。マクロ内のコマンドは、コールする側のパラメータを参照します。パラメータが値リストである場合、選択される値は既存値、カスタム値、値リストの最初の値のいずれかになります。
また、グローバル文字列変数 GLOB_MODPAR_NAME は最後にユーザーが修正したパラメータ名を含みます。

LOCK

LOCK name1 [, name2, ..., namen]

設定ダイアログボックスで名前が付けられているパラメータをロックします。ロックされたパラメータは、ダイアログボックスではグレー表示され、ユーザーはその値を修正できません。

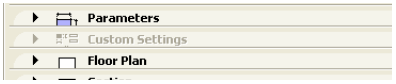
HIDEPARAMETER

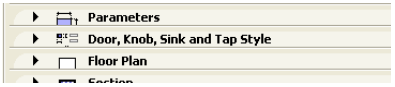
HIDEPARAMETER name1 [, name2, ..., namen]

設定ダイアログボックスで、名前の付いたパラメータ（1 つ以上）とその子パラメータを非表示にします。パラメータスクリプトでこのコマンドを使用して非表示にしたパラメータは、自動的にパラメータリストから消えます。

ユーザーインターフェイススクリプト

次の GDL コマンドを使用して、設定ダイアログボックスでライブラリ部品の［カスタム設定］パネルのカスタムインターフェイスを定義できます。ライブラリ部品エディタの［デフォルトとして設定］ボタンをクリックすると、そのオブジェクト（ドア、窓などの）の設定ダイアログボックスでは、デフォルトでカスタムインターフェイスが使用されます。カスタムコントロールのパラメータは、オリジナルのパラメータリストでは自動的に非表示にされませんが、ライブラリ部品エディタで手動で非表示にできます。





座標系の原点は左上隅にあります。サイズと座標の値はピクセル単位で測定されます。

UI_DIALOG

UI_DIALOG title [, size_x, size_y]

ダイアログボックスのタイトルを定義します。現在では、使用できる領域のサイズは 444 × 266 ピクセルに固定されていて、size_x パラメータと size_y パラメータは使用されていません。

制限： インターフェイススクリプトには 1 つの UI_DIALOG コマンドしか含められません。

UI_PAGE

UI_PAGE page_number

ページ指示文。インターフェイス要素が配置されるページを定義します。ページ数は 1 から始まります。ページ間の移動は、次の 2 つの方法で定義することができます。1 つは、UI_NEXT と UI_PREV コマンドで作成した 2 つのボタンを使用する方法です。もう 1 つは、UI_CURRENT_PAGE コマンドを使用して動的なページハンドリングを作成する方法です。

インターフェイススクリプトに UI_PAGE コマンドがない場合、各要素はデフォルトで最初のページに配置されます。

警告： ページリストの連続性を断ち切ると、ボタンなしの新規ページが強制的に挿入され、そこから別のページに移動できなくなります。

UI_CURRENT_PAGE

UI_CURRENT_PAGE index

表示する現在のタブページの定義。

警告：存在しないページにジャンプすると、ボタンとコントロールのない新規ページが強制的に挿入され、そこから別のページに移動できなくなります。

index: 表示する UI_PAGE の有効なインデックス

UI_BUTTON

UI_BUTTON type, text, x, y, width, height [, id [, url]]

UI_PICT_BUTTON type, text, picture reference, x, y, width, height [, id [, url]]

現在のページでのボタン定義。ボタンは、ページ間の移動、ウェッブページを開く、パラメータスクリプトで定義したアクションの実行など、さまざまな用途で使用できます。ボタンはテキストまたは画像を設定できます。

type: 次に挙げるボタンのタイプ

UI_PREV: クリックすると、前のページが表示されます。

UI_NEXT: クリックすると、次のページが表示されます。

UI_FUNCTION: クリックすると、GLOB_UI_BUTTON_ID グローバル変数は式で指定したボタンに設定されます。

UI_LINK: クリックすると、式の URL はデフォルトのウェッブブラウザで開きます。

text: テキストタイプのボタンに表示されるテキスト。画像ボタンの場合、このパラメータは省略。

picture_reference: ライブラリ部品に格納される画像ファイル名、またはインデックス番号。インデックス 0 は、ライブラリ部品のプレビュー画像を参照します。

x, y: ボタンの位置

width, height: ボタンの幅と高さ（ピクセル単位）

id: 整数のユニーク ID

url: URL を含む文字列

UI_PREV および UI_NEXT ボタンは、前 / 次のページがない場合は無効です。これらのボタンを押すと、ライブラリ部品の gs_ui_current_page パラメータは、この名前のパラメータがあれば、表示するページのインデックスに設定されます。

例：

```
! 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.co.jp"
! parameter script
if GLOB_UI_BUTTON_ID = 3 then
parameters  gs_ui_current_page = 9, ...
ENDIF
```

UI_SEPARATOR

UI_SEPARATOR x1, y1, x2, y2

区切り線を生成します。x1 = x2 または y1 = y2 の場合、区切り線は単数の（縦または横の）区切り線となります。i

x1, y1: 左上の節点座標（ラインの始点座標）

x2, y2: 右下の節点座標（ラインの終点座標）

UI_GROUPBOX

UI_GROUPBOX text, x, y, width, height

グループボックスは矩形の区切りです。これは、論理的に関連付けられたパラメータを視覚的にグループ化するのに使用できます。

text: グループボックスのタイトル

x, y: 左上角の位置

width, height: 幅と高さ（ピクセル単位）

UI_PICT

UI_PICT picture reference, x, y [,width, height[, mask]]

ダイアログボックス内の画像要素。画像ファイルはロードしたライブラリのいずれかに配置しなければなりません。

picture_reference: ライブラリ部品に格納される画像ファイル名、またはインデックス番号。インデックス 0 は、ライブラリ部品のプレビュー画像を参照します。x, y: 画像の左上隅の位置を示します。

width, height: 省略可能な幅と高さ（ピクセル単位)。デフォルトでは画像のオリジナルの幅と高さの値が使用されます。

mask = alpha + distortion

詳細については“PICTURE”要素を参照してください。

UI_STYLE

UI_STYLE fontsize, face_code

このキーワードの後に生成される全ての UI_OUTFIELD と UI_INFIELD は、次の UI_STYLE ステートメントまでこのスタイルで表現されます。

fontsize: 次のフォントサイズ値のいずれか。

0: 小
1: 極小
2: 大

face_code: STYLE 定義と似ていますが、値を組み合わせて使用することはできません。

0: normal
1: **bold**
2: *italic*
4: underline
8: outline
16: shadow

UI_OUTFIELD

UI_OUTFIELD expression,x,y,width,height [, flags]]

静的なテキストを生成します。

expression: 数式または文字列式

x, y: テキストブロックの左上角の位置

width, height: 幅と高さ（ピクセル単位）

flags = j1 + 2*j2 + 4*j3

j1 (1) および j2 (2) - 水平整列
j1 = 0, j2 = 0 : 左端揃え（デフォルト）
j1 = 1, j2 = 0 : 右端揃え
j1 = 0, j2 = 1 : 中央揃え
j1 = 1, j2 = 1 : 使用しない
j3 (4): - グレーテキスト

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,
...,
expression_imagen, textn, value_definitionn]

パラメータの入力のための編集テキストまたはポップアップメニューを生成します。パラメータのタイプが値リスト、材質、塗りつぶし、線種、またはペンカラーである場合は、ポップアップが生成されます。

コマンドのオプションのパラメータが存在する場合は、値リストをサムネイルビューフィールドとして表示できます。異なるサムネールコントロールタイプが使用可能です。これは指定された画像と対応付けられたテキストを表示し、ポップアップメニューのように一度に 1 つの項目を選択できるようになっています。

バージョン 1 および 2 の infield では、サムネール項目と値リストの項目はインデックスで関連付けられています。

バージョン 3 の infield は、サムネール項目を関連するパラメータの値リスト項目にバインドする、値の関連付けを定義します。サムネール項目で定義された値がパラメータの値リストに存在しない場合は、その値はコントロールで表示されません。

インターフェイススクリプトは、パラメータが修正された後は新規の値で再構築されます。

name: UI_INFIELD の文字列式としてのパラメータ名。UI_INFIELD{2} の配列の場合は、省略可能なインデックス値を持つパラメータ名。

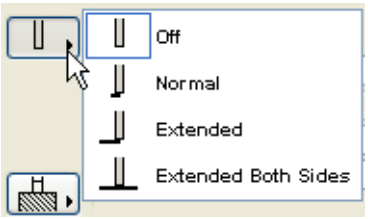
x, y: 編集テキスト、ポップアップ、またはコントロールの位置

width, height: 幅と高さ (ピクセル単位)

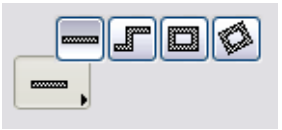
method: コントロールのタイプ

1: リストビューコントロール

2: ポップアップメニューコントロール

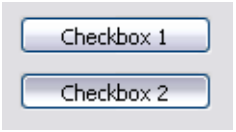


3: ポップアップアイコンラジオコントロール



4: プッシュアイコンラジオコントロール

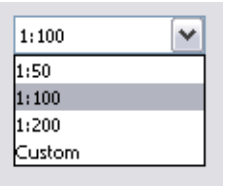
5: テキスト付きプッシュボタン



6：画像付きプッシュボタン

7: テキスト付きチェックボックス

8. テキスト付きポップアップリスト



picture_name: 連結画像のマトリクス、または空の文字列を含む共通画像ファイル名

images_number: マトリクス内の画像の数、ブールパラメータの場合は 0 または 2

rows_number: マトリクス内の行の数

cell_x, cell_y: 画像とテキストを含む、サムネイルビューフィールド内のセルの幅と高さ

image_x, image_y: セル内の画像の幅と高さ

expression_imagei: マトリクス内の i 番目の画像のインデックス、または個々のファイル名 共通の画像ファイル名が指定されている場合は、ここではインデックスを使用する必要があります。インデックスと個々のファイル名の組み合わせは使えません。

texti: セル番号が i のセル内のテキスト

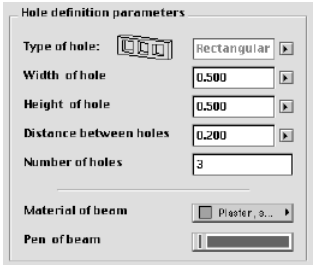
value_definitioni: セルの値を値リスト項目の値と一致させる値定義

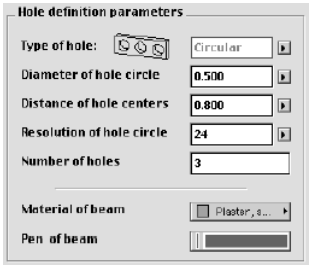
expression: 数式または文字列式、または

CUSTOM: キーワードで、どんなカスタム値でも入力できることを意味します。

例 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
```





例 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
```

UI_RADIOBUTTON

UI_RADIOBUTTON name, value, text, x, y, width, height
ラジオボタングループのラジオボタンを生成します。ラジオボタングループは、パラメータ名によって定義されます。同一グループの項目は、相互に排他的です。

name: 配列の場合は、省略可能なインデックス値を持つパラメータ名

value: このラジオボタンが設定された場合、パラメータはこの値に設定されます。

text: ラジオボタンの隣に表示されるテキスト

x, y: ボタンの位置

width, height: 幅と高さ（ピクセル単位）

例：

```
UI_RADIOBUTTON "ceilingPlan", 0, `Floor Plan`, 10, 140, 100, 20
UI_RADIOBUTTON "ceilingPlan", 1, `Ceiling Plan`, 10, 160, 100, 20
```

UI_TOOLTIP

UI_BUTTON type, text, 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_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]
ユーザーインターフェイスページ上のコントロールに対してツールチップを定義します。ツールチップはボタン、インフィールド、アウトフィールドおよび画像で使用できます。
tooltiptext: そのコントロールに対してツールチップとして表示するテキスト

式と関数

GDL 形状の全てのパラメータは、計算によって導き出すことができます。例えば、円柱の高さをその半径の 5 倍に定義したり、立方体を定義する前に、その半分のサイズ分だけ座標系を各方向に移動させて、最初の原点を立方体の左下角ではなく中心に持って来たりすることができます。これらの計算を定義するために、GDL には数式、演算子、関数といった多数の数学ツールが用意されています。

数式

GDL ステートメントに複合式を記述することができます。式は数値タイプまたは文字列タイプにすることができます。定数、変数、パラメータ、または関数コールを使用します。オペレータでこれらを組み合わせることもできます。一組の括弧 ( ) (優先順位 1) は、オペレータのデフォルト優先順位を無効にするときに使用します。

同じスクリプト内でも、*単純タイプ*の変数に数値および文字列値を指定したり、この変数をそれぞれ数値タイプまたは文字列タイプの式に使用することもできます。結果が文字列になる演算は、マクロコールでマクロ名として直接使用することはできません。材質、塗りつぶし、線種またはスタイルの定義で属性名としても使用することはできません。文字列値が与えられている変数は、そのように扱われ、文字列値が必須となる場合に使用できます。それ以降のスクリプト内で同じ変数に数値が指定されている場合も、再度文字列値が与えられるまで数式に使用できます。可能な場合は、コンパイル前のプロセスで、式のタイプが確認されます。

GDL は 1 次元および 2 次元の寸法配列をサポートしています。変数は、次元が指定される宣言ステートメントの後で*配列*になります。

DIM

```
DIM var1[dim_1], var2[dim_1][dim_2], var3[ ],
      var4[ ][ ], var5[dim_1][ ],
      var5[ ][dim_2]
```

キーワード DIM の後に、カンマで区切った変数名をいくつでも使用できます。var1、var2、... は配列名で、角括弧内の数字は配列の次元（数字定数）を表しています。変数式を次元として使用することはできません。次元の指定がない場合は、配列は動的（1 次元、2 次元、またはこの両方）に宣言されます。
ライブラリ部品のパラメータも配列にすることができます。実際の次元はライブラリ部品のダイアログボックスで指定します。パラメータ配列は、スクリプト内で宣言する必要はなく、デフォルトでは動的です。CALL ステートメントを使用してライブラリ部品を参照する時は、配列パラメータの実際の値は任意の次元を持つ配列とすることができます。
配列の要素はスクリプト内のどの場所でも参照できますが、変数の場合は参照できるのは宣言の後だけです。

var1[num_expr] あるいは var1
var2[num_expr1][num_expr2] あるいは var2[num_expr1]
あるいは var2

実際のインデックス値を指定せずに配列名を記述すると、状況によって受け入れられる配列全体（または一連の 2 元配列）が参照されます（CALL、PRINT、LET、PUT、REQUEST、INPUT、OUTPUT のステートメントの場合）。動的配列の場合、実際のインデックス値に制限はありません。解析中に、存在しない動的配列要素に値が与えられると、必要な量のメモリが割り当てられて、存在しない要素は全て 0 （数値）に設定されます。

警告： 状況によっては、予想外のメモリ不足エラーが起きる場合があります。インタプリタはエラー条件を検出できないので、各インデックスは、間違った大きな値であっても、有効であると判断されます。存在しない動的配列要素は 0 （数値）です。

固定の次元を持つ配列の場合、その固定次元上での実際のインデックスの妥当性が確認されます。固定長の配列変数は、割り当て内の動的配列値を受け入れられません。ただし、配列値全体が与えられた動的配列は、これらの値を受け入れます。これは、戻りパラメータとして配列参照全体が使用できるいくつかのステートメントでも同じです。（REQUEST、INPUT、SPLIT）
配列要素は、どの数式または文字列式を使用して、文字列値または数値を与えることができます。
1 で始まるインデックスと数式をインデックスとして使用できます。
配列要素は、異なる単純タイプ（数字、文字列、グループ）にすることができます。配列全体のタイプ（’メイン’タイプ）は、その最初の要素（[1] または [1][1]）のタイプです。パラメータ配列およびグローバル変数配列は、混合タイプにはできません。

VARDIM1(expr)

VARDIM1(expr)

VARDIM2(expr)

VARDIM2(expr)

これらの関数は、パラメータとして指定された（配列）式に対して実際の大きさの値を返します。これらは、動的配列または配列パラメータの実際の要素全てを適切に扱いたい場合には使用する必要があります 動的配列のどの要素もまだ設定されていない場合は、戻り値は 0 です。1 次元の寸法配列の場合、VARDIM2 は 0 を返します。

数式の例：

```
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
```

文字列式の例：

```
″Constant string″
name + STR (″%m″, i) + ″.″ + ext
string_param <> ″Mode 1″
```

配列値を使用する式の例：

```
DIM tab [5], tab2 [3][4] ! declaration
tab [1] + tab [2]
tab2 [2][3] + A
PRINT tab

DIM f1 [5], v1[], v2[][]
v1[3] = 3 !v1[1] = 0, v1[2] = 0, array of 3
                ! elements
v2[2][3] = 23  ! all other elements(2 X 3) = 0
PRINT v1, v2

DIM f1 [5], v1[], v2[][]
FOR i = 1 TO VARDIM1(f1)
    f1 [i] = i
NEXT I
v1 = f1
v2 [1] = f1
PRINT v1, v2
```

オペレータ

以下のオペレータは、優先順位の高い順に並んでいます。式の評価は、優先順位の最も高いオペレータから始められ、かつ左から右に行われます。

算術オペレータ

^ （あるいは **) 累乗 優先順位 2
***** 乗算 優先順位 3
/ 除算 優先順位 3
MOD （または %） 剰余 優先順位
X MOD Y = X - Y * INT (X/Y)
\+ 加算 優先順位 4
\- 減算 優先順位 4

注記： + （加算）も文字列の式に適用できます。結果として、文字列が連結されます。
'/' （除算）の結果は常に実数です。その他の演算の結果はオペランドのタイプによって異なります。全てのオペランドが整数の場合、結果は整数となり、整数以外の場合は実数となります。

関係オペレータ

= 等しい 優先順位 5
< より小さい 優先順位 5
> より大きい 優先順位 5
<= より小さいか等しい 優先順位 5
>= より大きいか等しい 優先順位 5
<> （あるいは **#**） 等しくない 優先順位 5

注記： これらの演算は、任意の 2 つの文字列の式間でも使用できます（文字列の比較は大文字と小文字を区別します）。結果は 1 整数または 0 整数です。結果は 1 整数または 0 整数です。「=」（等しい）、「<=」（より小さいか等しい）、「>=」（より大きいか等しい）、「<>」（あるいは #）（等しくない）の各オペレータを実数オペランドと共に使用すると、演算結果の精度に問題が生じるおそれがあるため、お勧めできません。

ブールオペレータ

AND （あるいは **&**）論理積 優先順位 6

OR （あるいは |） 論理和 優先順位 7

EXOR （あるいは **@**）排他的論理和 優先順位 8

注記： ブールオペレータは、整数と共に機能します。つまり 0 は「偽（false)」を意味し、それ以外の数値は「真(true)」を意味します。論理式での値もやはり実数です。つまり、1 は「真」で、0 は「偽」になります。ブールオペレータを実数オペランドと共に使用すると、演算結果の精度に問題が生じるおそれがあるため、お勧めできません。

関数

算術関数

ABS

ABS（x）x の絶対値（x が整数の場合は整数、それ以外の場合は実数）を返します。

CEIL

CEIL（x）x よりは大きい最小の整数値（常に整数）を返します。（例，CEIL(1.23) = 2; CEIL (-1.9) = -1).

INT

INT（x）x の整数部（常に整数）を返します。（例，INT(1.23) = 1, INT(-1.23) = -2）.

FRA

FRA（x）x の小数部（x が整数の場合は整数 0、それ以外の場合は実数）を返します。（例，FRA(1.23) = 0.23, FRA(-1.23) = 0.77）.

ROUND_INT

ROUND_INT（x）x の丸めた整数部を返します。'i = ROUND_INT (x)'式は、スクリプト
IF x < 0.0 THEN i = INT (x - 0.5) ELSE i = INT (x + 0.5) と同等です。

SGN

SGN（x） x が正の時は整数 +1 を、x が負の時は整数 -1 を返します。それ以外の時は整数 0 を返します。

SQR

SQR（x） x の平方根（常に実数）を返します。

三角関数

これらの関数は、引数（COS、SIN、TAN）と戻り値（ACS、ASN、ATN）の単位として度数を使用します。

ACS

ACS（x） xのアークコサインを返します。（-1.0 <= x <= 1.0; 0° <= ACS(x) <= 180°）。

ASN

ASN（x）xのアークサインを返します。（-1.0 <= x <= 1.0; -90° <= ASN(x) <= 90°）。

ATN

ATN（x）xのアークタンジェントを返します。（-90° <= ATN(x) <= 90°）。

COS

COS（x）xのコサインを返します。

SIN

SIN（x）xのサインを返します。

TAN

TAN（x）xのタンジェントを返します。

PI

PI：円周率を返します。（p = 3.1415926...）.

注記： 全ての戻り値は*実数*です。

指数関数

EXP

EXP（x）e （e = 2.7182818）のx乗を返します。

LGT

LGT（x）xの常用対数を返します。

LOG

LOG（x）x の自然対数を返します。

注記： 全ての戻り値は*実数*です。

ブール関数

NOT

NOT（x）x が真（<>0）の時は偽（= 整数 0）、x が偽（=0）（論理否定）の時は真（= 整数 1）を返します。

注記： パラメータの値は*整数*でなければなりません。

統計関数

MIN

MIN（x1, x2, ... xn）: 引数の中で最も小さい値を返します。

MAX

MAX（x1, x2, ... xn）: 引数の中で最も大きい値を返します。

RND

RND（x）0.0 から x の間のランダムな値を返します（x > 0.0）。

ビット関数

BITTEST

BITTEST（x, b）

x の b ビット目がたっている場合は 1 を返します。それ以外の場合は 0 を返します。

BITSET

BITSET（x, b [, expr]）

expr は 0 またはそれ以外にすることができます。デフォルト値は 1 です。指定された式の値に従って x の b ビット目を 1 または 0 に設定し、その結果を返します。パラメータの値は*整数*でなければなりません。戻り値は*整数*です。

特殊関数

特殊関数（グローバル変数も）は、プログラムとの通信を目的としてスクリプト内で使用することができます。特殊関数は、プログラムの現在の状態とさまざまな環境設定を問い合わせるか、ライブラリ部品の現在の環境を参照します。要求コールも GDL 機能拡張との通信に使用することができます。

特殊関数には、要求関数と IND 関数という 2 つのタイプがあります。

REQ (parameter_string)

REQUEST (question_name, name | index, variable1 [, variable2,...])

IND (MATERIAL, name_string)

IND (FILL, name_string)

IND (LINE_TYPE, name_string)

IND (STYLE, name_string)

IND (TEXTURE, name_string)

これらの要求の戻り値は常に、正常に検出された値の数（整数）ですが、検出された値のタイプはそれぞれの要求で部分的に定義されます。

IND 関数は属性のインデックス（整数）値を返します。

文字列関数

STR

STR (numeric_expression, length, fractions)

STR

STR (format_string, numeric_expression)

STR{2}

STR{2}(format_string, numeric_expression [, exta_accuracy_string])

この関数の最初の形式は、数式の現在の値から文字列を作成します。文字列内の数の文字数の最小は length で、fractions は浮動小数点の後の桁数を表します。変換された値の文字数が length より多い場合、必要に応じて拡張されます。length よりも文字数が少ない場合は、左詰め（length > 0）または右詰め（length < 0）されます。

例：

```
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
```

2 番目のケースでは、format_string は変数と定数のどちらかとすることができます。format に何も指定しないと、小数点以下 3 位までのメートルとして解釈されます（整数の場合は 0 が表示されます）。format_string に追加の精度フラグを設定すると、STR{2} 関数は対応する追加精度文字列を 3 番目のパラメータに返します。

format_string の形式は以下のとおりです。

```
%[0 or more flags][field_width][.precision] conv_spec
```

flags（m、mm、cm、e、df、di、sqm、sqcm、sqf、sqi、dd、gr、rad、cum、l、cucm、cumm、cuf、cui、cuy、gal の場合）:

```
(none)   right justify (default)
-        left justify
+        explicit plus sign
(space)  in place of a + sign
'*' 0    extra accuracy Off (default)
'*' 1    extra accuracy .5
'*' 2    extra accuracy .25
'*' 3    extra accuracy .1
'*' 4    extra accuracy .01
'*' 5    extra accuracy .5
'*' 6     extra accuracy .25
```

flags（m、mm、cm、df、di、sqm、sqcm、sqf、sqi、dd、fr、rad、cum、l、cucm、cumm、cuf、cui、cuy、gal の場合）:

```
'#'      0 整数を非表示
```

flags（ffi、fdi、fi の場合）:

```
'0'      0 インチを表示
```

flags（m、mm、cm、e、df、di、sqm、sqcm、sqf、sqi、dd、gr、rad、cum、l、cucm、cumm、cuf、cui、cuy、gal の場合）:

「~」は 0 小数を非表示にします（「#」フラグが指定されていない場合にのみ有効）。

「^」は、小数区切り文字や桁集合化文字を変更しません（指定されていない場合、これらの文字は現在のシステムで設定されているとおりに置き換えられます）。

field_width: 符号なしの 10 進整数、生成する最小文字数

precision: 符号なしの 10 進整数、生成する小数部桁数

conv_spec（変換規則子）：
e：指数形式（メートル）
m：メートル
mm：ミリメートル
cm：センチメートル
ffi：フィートと分数インチ
fdi：フィートと小数インチ
df：小数フィート
fi：分数インチ
di：少数インチ
pt：ポイント

面積の場合：
sqm：平方メートル
sqcm：平方センチメートル
sqmm：平方ミリメートル
sqf：平方フィート
sqi：平方インチ

角度の場合：
dd：小数表示の角度
dms：度、分、秒
gr：grad
rad：ラジアン
surv：測量の単位

体積の場合：
cum：立方メートル
l：リットル
cucm：立方センチメートル
cumm：立方ミリメートル
cuf：立方フィート
cui：立方インチ
cuy：立方ヤード
gal：ガロン

例：

```
h = 23
nr = 0.345678
TEXT2 0, h, STR ("%m", nr) !0.346
TEXT2 0, h-1, STR ("% 10.2m", nr) !35
TEXT2 0, h-2, STR ("% 4cm", nr) !34.5678
TEXT2 0, h-3, STR ("% 12.4m", nr) !34.5678
TEXT2 0, h-4, STR ("% 6mm", nr) !345.678000
TEXT2 0, h-5, STR ("%+15e", nr)!+3.456780e-01
TEXT2 0, h-6, STR ("%ffi", nr) !1'-2"
TEXT2 0, h-7, STR ("%0.16ffi", nr) !1'-1 5/8"
TEXT2 0, h-8, STR ("% .3fdi", nr) ! 1'-1.609"
TEXT2 0, h-9, STR ("% -10.4df", nr) ! 1.1341'
TEXT2 0, h-10, STR ("%0.64fi", nr) !13 39/64"
TEXT2 0, h-11, STR ("%+12.4di", nr)!+13.6094"
TEXT2 0, h-12, STR ("%#.3sqm", nr) ! 346
TEXT2 0, h-13, STR ("%+sqcm", nr) !+3,456.78
TEXT2 0, h-14, STR ("% .2sqmm", nr) ! 345,678.00
TEXT2 0, h-15, STR ("%-12sqf", nr) !3.72
TEXT2 0, h-16, STR ("%10sqi", nr) ! 535.80
TEXT2 0, h-17, STR ("%.2pt", nr) !0.35
alpha = 88.657
TEXT2 0, h-18, STR ("%+10.3dd", alpha)!+88.657°
TEXT2 0, h-19, STR ("%.1dms", alpha)!88° 39'
TEXT2 0, h-20, STR ("%.2dms", alpha) !88° 39'25"
TEXT2 0, h-21, STR ("%10.4gr", alpha) ! 98.5078G
TEXT2 0, h-22, STR ("%rad", alpha) !1.55R
TEXT2 0, h-23, STR ("%.2surv", alpha) !N 1° 20'35" E
```

SPLIT

SPLIT (string, format, variable1 [, variable2, ..., variablen])

1 つ以上の数値部分または文字列部分内のフォーマットに従って、文字列パラメータを分割します。分割プロセスは、最初の一致しない部分が検出されたときに停止します。読み取れた値（*整数*）の数を返します。

string: 分割する文字列

format: 定数文字列 %s および %n -s の任意の組み合わせ。文字列内の部分は、定数文字列に適合している必要があります。%s はスペースまたはタブで区切られた文字列値、%n は数値を表します。

$variable_i$: 分割された文字列部分を格納する変数の名前

例：

```
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)

現在のスタイルで表示された文字列の（*実数*）幅をメートル単位で返します。メートル単位の幅は、STW (string_expression) / 1000 * GLOB_SCALE です。

例：


```
DEFINE STYLE "own" "Monaco", 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)

文字列の（*整数の*）長さ（文字数）を返します。

STRSTR

STRSTR (string_expression1, string_expression2)

2 番目の文字列が 1 番目の文字列に最初に現れる（*整数の*）位置を返します。1 番目の文字列に 2 番目の文字列が含まれていない場合、この関数は 0 を返します。

STRSUB

STRSUB (string_expression, start_position, characters_number)

start_position パラメータで与えられた位置から始まる characters_number 文字数の長さの文字列を返します。

例：

```
ss = ""
    n = REQUEST ("Linear_dimension", "",ss)
    unit = ""
IF STRSTR (ss, "m")  > 0 THEN unit = "m"
IF STRSTR (ss, "mm") > 0 THEN unit = "mm"
IF STRSTR (ss, "cm") > 0 THEN unit = "cm"
TEXT2 0, 0, STR (ss, a) + " " + unit !1.00 m
string = "Flowers.PICT"
len = STRLEN (string)
n = STRSTR (string, ".")
TEXT2 0, -1, STRSUB (string, 1, n - 1) !Flowers
TEXT2  0, -2, STRSUB (string, len - 4, 5) !.PICT
```

制御ステートメント

この章では、スクリプト内のループやサブルーチンを制御するのに使用可能な GDL コマンドについて解説し、繰り返し使用するパラメータ値を格納するために用意されたバッファ操作の概念を紹介します。また、オブジェクトをマクロコールとして使用する方法および計算された式を画面上に表示する方法も説明します。

フロー制御ステートメント

FOR

FOR variable_name = initial_value **TO** end_value [**STEP** step_value]

FOR ループの最初のステートメント。キーワード STEP と step_value を省略すると、ステップは 1 とみなされます。

ループ制御変数としてグローバル変数を使用することは許されません。

例：

```
FOR I=1 TO 10 STEP 2
    PRINT I
NEXT I
```

NEXT

NEXT variable_name

FOR ループの最後のステートメント。

ループ変数の値は、初期値 initial_value から終了値 end_value まで、ループ本体（FOR と NEXT ステートメントに挟まれたステートメント）の実行のたびに step_value の増加量（あるいは減少量）で変化します。ループ変数の値が end_value を超えると、プログラムは NEXT ステートメントから後のステートメントに実行を移します。

注記： ループの実行中に step_value を変更しても効果はありません。

次の 2 つのプログラムは、同じ結果になります。

```
! 1st
  A = B
1:IF C > 0 AND A > D OR C < 0 AND A < D THEN 2
  PRINT A
  A = A + C
  GOTO 1
2:
! 2nd
  FOR A = B TO D STEP C
     PRINT A
  NEXT A
```

上記は、step_value = 0 で無限ループとなる例を示しています。

FOR ステートメントの後には、NEXT ステートメントを 1 つだけ使うことができます。GOTO （または IF ... GOTO）ステートメントでループを抜け、その後また戻ることができますが、FOR ステートメントを無視してループに入ることはできません。

DO

DO

```
      [statment1
      statement2
      ...
      statementn]
```

WHILE 条件

キーワードとキーワードとの間のステートメントは、条件が真であれば実行されます。

条件が確認されるのはそれぞれのステートメントの実行後です。

WHILE 条件 DO

```
      [statement1
      statement2
      ...
      statementn]
```

ENDWHILE

キーワードとキーワードとの間のステートメントは、条件が真であれば実行されます。

条件が確認されるのはそれぞれのステートメントの実行前です。

REPEAT
[statement1
statement2
...
statementn]

UNTIL 条件
条件が真になるまで、キーワードとキーワードの間のステートメントが実行されます。
条件が確認されるのはそれぞれのステートメントの実行後です。
例：
以下の 4 つの GDL コマンドシーケンスは等価とみなされます。

```
! 1st
FOR i = 1 TO 5 STEP 1
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
NEXT i
! 2nd
i = 1
DO
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
    i = i + 1
WHILE i <= 5
! 3rd
i = 1
WHILE i <= 5 DO
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
    i = i + 1
ENDWHILE
! 4th
i = 1
REPEAT
    BRICK 0.5, 0.5, 0.1
    ADDZ 0.3
    i = i + 1
UNTIL i > 5
```

IF

IF 条件 **THEN** ラベル

IF 条件 **GOTO** ラベル

IF 条件 **GOSUB** ラベル

条件付きジャンプステートメント。条件式の値が 0 の場合、コマンドの影響はなく、その他の値の場合、実行はラベルに移って継続されます。

例：

```
IF  A THEN 28
IF  I > J GOTO 200+I*J
IF  I > 0 GOSUB 9000
IF  条件 THEN statement [ELSE statement]
```

または

```
IF  条件 THEN [statement1
    statement2
    ...
    statementn]
 [ELSE
    statementn+1
    statementn2
    ...
    statementn+m]
ENDIF
```

同じ行内のキーワード THEN または ELSE、またはこの両方の後にコマンドを 1 つだけ記述する場合は、ENDIF は不要です。同じ行内の THEN または ELSE の後に指定したコマンドは、ENDIF を意味します。

THEN の後に新規の行があると、条件の式が真（ゼロ以外）の場合に限って、キーワード ELSE または ENDIF が現れるまで、以降の全てのコマンドが実行されます。それ以外の場合、ELSE 以降のコマンドが実行されます。キーワード ELSE がない場合は、ENDIF 以降のコマンドが実行されます。

例：

```
IF a = b THEN height = 5 ELSE height = 7
IF needdoors THEN
    CALL   "door_macro" PARAMETERS
    ADDX   a
ENDIF
IF simple THEN
    HOTSPOT2 0, 0
    RECT2 a, 0, 0, b
ELSE PROJECT2 3, 270, 1
IF name = "Sphere" THEN
    ADDY b
    SPHERE 1
ELSE
    ROTX 90
    TEXT 0.002, 0, name
ENDIF
```

GOTO

GOTO ラベル

無条件ジャンプステートメント。プログラムは、ラベルの値（数値または文字列）によって示されるステートメントへの分岐を実行します。変数ラベルの式ではランタイムが決定したアドレスにジャンプするため、解釈が遅くなることがあります。

例：

```
GOTO K+2
```

GOSUB

GOSUB ラベル

エントリポイントがラベルであるサブルーチンの内部サブルーチンコール。ラベル値は、どんな数式または文字列式でも使用できます。変数ラベルの式ではランタイムが決定したアドレスにジャンプするため、解釈が遅くなることがあります。

RETURN

RETURN

内部サブルーチンから戻ります。

END / EXIT

END / EXIT [v1, v2, ..., vn]

現在の GDL スクリプトの終わり。プログラムは、終了するか、1 つ上のレベルに戻ります。GDL ファイル内で複数の END または EXIT を使用することができます。省略可能な値リストが指定されている場合は、現在のスクリプトがこれらの戻り値をコールする側に渡します。

268 ページ「CALL」の戻りパラメータの受け取りについての説明を参照してください。

BREAKPOINT

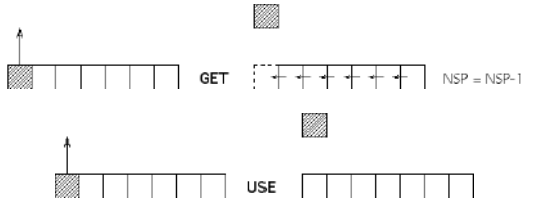
BREAKPOINT expression

このコマンドでは、GDL スクリプト内にブレークポイントを指定することができます。GDL デバッガは、パラメータの値（数式）が真（1）の場合はこのコマンドで停止し、デバッガの［ブレークポイントを有効に］オプションがオンにされます。通常の実行モードでは、GDL インタプリタは単にこのコマンドを無視します。

パラメータバッファの操作

パラメータバッファは、内蔵データ構造です。数式を使用して記述できる特定の規則の後で、いくつかの値（例えば、座標）が変化した場合に使用することができます。これは、例えば変数の現在の値を格納するのに便利です。

パラメータバッファは無限に長い配列です。PUT コマンドを使用して、ここに数値を格納できます。PUT コマンドは、与えられた値をバッファの最後に格納します。これらの値は、GET または USE コマンドの使用時に入力されたときの順序に従って、後で使うことができます。そのため、最初に格納された値が最初に使われます。GET (n) または USE (n) コマンドは、カンマで区切られた n 個の値に対応します。したがって、n 個の値が必要な任意の GDL パラメータリストで使うことができます。



PUT

PUT expression [, expression, ...]

与えられた値を、与えられた順番で内部パラメータバッファに格納します。

GET

GET (n)

内部パラメータバッファから次の n 個の値を取り出して使い、その後それを削除します。

USE

USE (n)

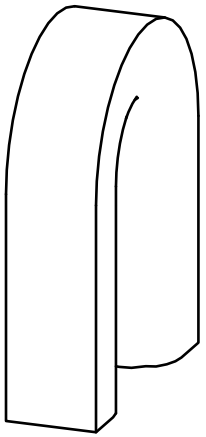
内部パラメータバッファから次の n 個の値を取り出して使いますが、その後それらを削除しません。以降の USE および GET 関数で、同一のパラメータシーケンスを使用できます。

NSP

NSP

内部バッファに格納されているパラメータの個数を返します。

パラメータバッファの使用例は、次のとおりです。



```
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
```

スクリプトの完全な記述は、次のとおりです。

```
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
```

マクロオブジェクト

必要な 3D オブジェクトは常に複合要素またはプリミティブ要素に分解することができますが、特定のアプリケーションでは、複合要素を定義する方がいい場合があります。このような個別に定義された要素をマクロと呼びます。

CALL

CALL macro_name_string [,parameter_list]

CALL macro_name_string [,] **PARAMETERS** [name1=value1 , ... namen=valuen][[,] **RETURNED_PARAMETERS** r1, r2, ...]

CALL macro_name_string [,] **PARAMETERS ALL** [name1=value1 , ... namen=valuen][[,][**RETURNED_PARAMETERS** r1, r2, ...]

CALL macro_name_string [,]**PARAMETERS** value1 *or* **DEFAULT** [, ... valuem *or* **DEFAULT**]

マクロ名は、31 文字以下にします。

マクロ名には、文字列定数、文字列変数、またはパラメータを使用できます。ただし、文字列演算をマクロ名としてマクロコールと共に使用することはできません。

警告： 文字列変数またはパラメータをマクロ名として使用すると、[ロードされているライブラリの全部品を含む] オプションをオンにしている場合に限り、コールされたマクロはアーカイブプロジェクトに組み込まれない場合があります。

識別子の定義と一致しない場合には、マクロ名を引用符（″、’、`、’、″、’）で囲む必要があります。つまり、マクロ名を、英字、または「_」または「~」の文字で開始し、英字、数字、および「_」と「~」の文字だけが含まれるようにします。さもなければ、CALL ステートメントで使う引用符を、最初と最後で同じものにしなければなりません。また、引用符は、マクロ名で使用するどの文字とも異なったものにしてください。

マクロ名自体も、キーワード CALL を指定せずにコマンドとして使えます。

macro_name [parameter_list]

macro_name **PARAMETERS** [name1=value1, ... namen=valuen]

macro_name **PARAMETERS ALL**

最初のタイプのマクロコールは、パラメータリストに 1 文字のパラメータ（A から Z）だけが含まれる場合には、いずれかのライブラリ部品だけでなく単純な GDL テキストファイルと共に使用できます。この形式のマクロコールは、旧バージョンとの互換性を保つために使用できますが、2 番目のタイプを使用することをお勧めします。パラメータリストの意味は、パラメータ A の値はリストの最初の値、パラメータ B の値は 2 番目の値というようになっています。値に対応する 1 文字のパラメータがライブラリ部品のマクロにない場合は、この値を無視して解釈が継続されますが、プログラムから警告されます。この方式では、文字列タイプの式は使用できません。

2 番目のタイプは、フル機能のライブラリ部品または普通の GDL テキストファイルに使用することができます。キーワード PARAMETERS の後で、コールされたマクロのパラメータ名を、それぞれについて記号「=」と値の両方を使用して、任意の順序でリストする必要があります。ここでは文字列タイプの式を使うことができますが、コールされたマクロの文字列タイプのパラメータには文字列値だけを与えてください。配列パラメータには全配列値を与える必要があります。パラメータリストのパラメータ名がコールされたマクロに含まれていない場合には、エラーメッセージが出されます。マクロコールにリス

トされていないコールされたマクロのパラメータには、マクロとしてコールされたライブラリ部品に定義されているオリジナルのデフォルト値が与えられます。

3 番目のタイプは、フル機能のライブラリ部品でのみ使用することができます。この場合、パラメータを個別に指定する必要はありません。コールする側の全てのパラメータはコールされたマクロに渡されます。コールする側で見つからないマクロのパラメータに対しては、デフォルト値が使用されます。パラメータ値を個別に指定した場合は、コールする側からの値またはコールされたマクロのパラメータをデフォルト値にします。この場合でも、マクロはパラメータを返すことができます。コールする側では、変数リストに続く RETURNED_PARAMETERS キーワードを使用して、戻り値を集めることができます。戻り値は、コールされたマクロで戻ってきた順番にこれらの変数に格納されます。マクロのコールとリターン部で変数の数とタイプを同じにしないでください。コールする側で指定した変数の方が多い場合は、0 整数に設定されます。タイプの互換性は確認しません。コールする側で指定した変数タイプは戻り値のタイプに設定されます。コールする側の変数値の 1 つが動的配列の場合は、全ての後続値はその中に格納されます。

264 ページ「END / EXIT」の戻りパラメータの構文を参照してください。

GDL マクロはコールする順序に従った、独自の環境を持っています。次のオプションの場合、

MODEL, RADIUS, RESOL, TOLER, PEN, LINE_TYPE, MATERIAL, FILL, STYLE, SHADOW

現在の値と現在の変換はマクロにおいて全て有効です。これらを使用したり修正することはできますが、修正はマクロ内でのみ有効です。コールしたマクロのレベルには影響しません。

マクロコールにパラメータを与えるということは、そのマクロのレベルでの暗黙の値の割り当てを意味します。

パラメータ A と B は一般的に、オブジェクトのサイズ変更に使います。

4 番目のマクロコールのタイプは、フル機能のライブラリ部品でのみ使用することもできます。この場合は、実際のパラメータ値は、コールされたライブラリ部品に存在する順番に個別に指定する必要があります。リストの最後から指定しているのでなければ、どの値も省略できません。パラメータの実際の値の代わりに DEFAULT キーワードを使用すると、実際の値がデフォルト値としてライブラリ部品に格納されます。値がない場合は、自動的にデフォルト値が使用されます（実際の値の数 *m* がパラメータの数より小さくてもかまいません）。この種類のマクロコールを解釈するときには、パラメータを名前で検索してそれに実際の値を割り当てる必要はありません。前より使い心地はよくないですが、より良いパフォーマンスを得ることができます。

例：

```
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
     ! パラメータのデフォルト値
```

"door-1" PARAMETERS を使用します。

名前が長いパラメータまたは文字列タイプのパラメータが必要ない場合は、テキストタイプの GDL で十分です。このタイプの GDL は、パラメータリストがないので、最初のタイプのマクロコールによってのみコールすることができます。これに対し、マクロパラメータ名を A から Z の文字だけに限定したくない場合、またはパラメータリストに文字列を含めたい場合は、マクロをライブラリ部品にして 2 番目のタイプの GDL 構文に従ってコールしてください。

出力ステートメント

PRINT

PRINT expression [, expression, ...]

全ての引数をダイアログボックスに書き込みます。引数は、カンマで区切られた、任意の順番の任意の数の文字列式または数式とすることができます。

例：

```
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
```

ファイル操作

以下のキーワードを使用して外部ファイルを開くと、GDL スクリプトとの値のやりとりによるファイルの読み書きおよび操作が可能になります。これには、特別なアドオン機能拡張を使用する必要があります。テキストファイルは、"TEXT GDL I/O" アドオンで処理できます。ほかのファイルの種類用のアドオンは、サードパーティによる開発が可能です。

「その他」の「GDL Text I/O アドオン」も参照してください。

OPEN

OPEN (filter, filename, parameter_string)

filter: 文字列。既存の機能拡張の名前。

filename: 文字列。ファイル名。

parameter_straing: 文字列。操作可能な機能拡張固有の区切り文字とオープン命令モードが含まれています。内容は機能拡張によって解釈されます。

指示されたとおりにファイルを開きます。戻り値は正の整数で、これで特定のファイルが識別されます。この値は、チャネル番号で、以降のインスタンスではファイルの参照番号になります。参照されたファイルをアーカイブプロジェクトに組み込むには、/FILE_DEPENDENCE "name1" [, "name2", ...]」コマンドをファイル名と組み合わせて使用します。

INPUT

INPUT (channel, recordID, fieldID, variable1 [, variable2,...])

recordID, fieldID: 文字列または数値タイプで、読み込みの開始位置を示します。内容は機能拡張によって解釈されます。

与えられたパラメータの数が、channel 値によって識別されたファイルの開始位置から読み込む値の数を定義します。パラメータリストには少なくとも 1 つの値が必要です。この関数は、読み込んだ値を順序どおりにパラメータに格納します。これらの値は、格納用に定義されたパラメータタイプとは関係なく、数値タイプまたは文字列タイプにすることができます。
戻り値は、正常に読み込まれた値の数です。ファイルの終わり文字に達すると、-1 が返されます。

VARTYPE

VARTYPE (expression)

式のタイプが数値の場合は 1 を、文字列の場合は 2 を返します。
INPUT コマンドを使って変数の値を読み取る際に役立ち、現在の値に従って変数のタイプを変更できます。
これらの変数のタイプは、コンパイル作業中は確認されません。

OUTPUT

OUTPUT channel, recordID, fieldID, expression1 [, expression2, ...]

recordID, fieldID: 文字列または数値タイプで、書き込みの開始位置を示します。内容は機能拡張によって解釈されます。

channel 値によって識別されたファイルに、与えられた位置から、定義済の式と同じ個数の値を出力します。少なくとも 1 個の式が必要です。出力値のタイプは、式のタイプと同じです。

CLOSE

CLOSE channel

channel 値で識別されたファイルを閉じます。

USING DETERMINISTIC ADD-ONS

以下にあげるキーワードを使って、決定的関数を提供する GDL アドオンをコールすることができます。つまり、特定の操作の結果が、指定されたパラメータによってのみ左右されます。これには、特別なアドオン機能拡張を使用する必要があります。例えば、ポリゴンの操作を PolyOperations アドオンによって実行することができます。ほかの操作用のアドオンは、サードパーティによる開発が可能です。
「その他」の「ポリゴン操作拡張機能」も参照してください。

INITADDONSCOPE

INITADDONSCOPE (extension, parameter_string1, parameter_string2)

extension: 文字列。既存の機能拡張の名前。

parameter_string1: 文字列。内容は拡張機能によって解釈されます。

parameter_string2: 文字列。内容は拡張機能によって解釈されます。

指示されたとおりにチャネルを開きます。戻り値は正の整数で、これで特定の接続が識別されます。この値は、チャネル番号で、以降のインスタンスでは接続の参照番号になります。

PREPAREFUNCTION

PREPAREFUNCTION channel, function_name, expression1 [, expression2, ...]

function_name: コールされる関数の文字列または数値 ID。内容は機能拡張によって解釈されます。

expression: 準備ステップのために渡されるパラメータ

後の関数をコールするための準備ステップとして、アドオンに何らかの値を設定します。

CALLFUNCTION

CALLFUNCTION (channel, function_name, parameter, variable1 [, variable2,...])

function_name: コールされる関数の文字列または数値 ID。内容は機能拡張によって解釈されます。

parameter: 入力パラメータ。内容は拡張機能によって解釈されます。

variablen: 出力パラメータ

アドオン内の、*channel* で指定された *function_name* という名前の関数がコールされます。パラメータリストには少なくとも 1 つの値が必要です。この関数は、戻り値を順序どおりにパラメータに格納します。戻り値は、正常に設定された値の数です。

CLOSEADDONSCOPE

CLOSEADDONSCOPE channel

channel 値で識別された接続を閉じます。

その他

GDL は、特殊なアドオンアプリケーションを介して、外部ファイル上でたくさんの操作を取り扱うこともできます。このためのコマンドをこの章で、例をあげて解説しています。

グローバル変数

グローバル変数を使用して、モデルの特殊な値を保存できます。これにより、GDL マクロの環境に関する図形情報にアクセスできます。例えば、壁に収まるように窓を定義するときに壁パラメータをコールすることができます。グローバル変数は、マクロコール中はスタックされません。

また、ドア、窓、ラベルおよび特性ライブラリ部品では、固定名の省略可能なパラメータを介して、ArchiCAD と通信することも可能です。これらのパラメータは、ライブラリ部品のパラメータリストに存在すれば、ArchiCAD によって設定されます。固定名のパラメータリストと Basic Library ドキュメントの詳細については、
http://www.graphisoft.com/support/developer/documentation/LibraryDevDoc/13（英文）を参照してください。

一般環境情報

GLOB_SCRIPT_TYPE T~ **現在のスクリプトのタイプ**

1- 特性スクリプト、2-2D スクリプト、3-3D スクリプト、4- ユーザーインターフェイススクリプト、5- パラメータスクリプト、6- マ

GLOB_CONTEXT **表示状況**

1- ライブラリ部品エディタ、2- 平面図、3-3D 表示、4- 断面 / 立面、5- 設定ダイアログ、6- リスト、7- 詳細図、8- レイアウト、22- 平面図からのフィードバックモード編集、23-3D 表示からのフィードバックモード編集、24- 断面 / 立面からのフィードバックモード編集、28- レイアウトからのフィードバックモード編集、43-3D 表示からオペレータとして生成、44- 立面 / 断面からオペレータとし

GLOB_SCALE A_ **図面スケール**

現在のウィンドウに基づく

GLOB_DRAWING_BGD_PEN **図面背景カラーのペン**

現在のウィンドウの背景カラーに最適な（印刷可能な）現在のパレット内のペン

GLOB_NORTH_DIR U~ **プロジェクトの北方向に投影**

［太陽光］ダイアログでの設定に従ったデフォルトプロジェクト座標系を標準とする

GLOB_WORLD_ORIGO_OFFSET_X

GLOB_WORLD_ORIGO_OFFSET_Y

ワールド原点を基準にしたプロジェクト原点の位置 294 ページ「グローバル変数 GLOB_WORLD_ORIGO_... の使用例：」を参照してくだ

GLOB_MODPAR_NAME **最後に修正されたパラメータ名**

設定ダイアログまたはライブラリ製品の編集可能なホットスポットを通して修正されたパラメータ パラメータスクリプトでのみ有効。

GLOB_UI_BUTTON_ID **UI ページで押されたボタンの ID**

または最後の動作が ID 付きのボタンを押されてなかった場合は 0。

GLOB_CUTPLANES_INFO ［4］ -

長さ値 4 の配列：1：切断面の高さ，2：切断面の最高レベル，3：切断面の最低レベル，4：ライブラリ部品のローカル座標システム［平面図の切断面］ダイアログ

GLOB_STRUCTURE_DISPLAY **構造表示の詳細**

躯体表示オプション設定に関する情報（整数）：0：全構造、1：芯のみ、2：仕上げなし

フロア情報

GLOB_HSTORY_ELEV	B_	配置フロアの立面
配置フロアは、オブジェクトが配置されているフロアです。		
GLOB_HSTORY_HEIGHT	Q_	配置フロアの高さ
配置フロアは、オブジェクトが配置されているフロアです。		
GLOB_CSTORY_ELEV	Q~	配置フロアの立面
配置フロアは、[平面図] ウィンドウに現在表示されているフロアです。		
GLOB_CSTORY_HEIGHT	R~	配置フロアの高さ
配置フロアは、[平面図] ウィンドウに現在表示されているフロアです。		
GLOB_CH_STORY_DIST	S~	配置フロアを基準にした現在のフロアの位置
配置フロアは、[平面図] ウィンドウに現在表示されているフロアです。		

フライスルー情報

GLOB_FRAME_NR	N_	アニメーションの現在のフレーム番号
アニメーションにのみ有効、静止画像の場合は -1		
GLOB_FIRST_FRAME	O_	フライスルーの最初のフレームインデックス
アニメーションにのみ有効、静止画像の場合は 0		
GLOB_LAST_FRAME	P_	フライスルーの最後のフレームインデックス
アニメーションにのみ有効、静止画像の場合は 0		
GLOB_EYEPOS_X	K~	現在のカメラ位置 (x)
アニメーションと静止画像のパース投影の場合のみ有効		
GLOB_EYEPOS_Y	L~	現在のカメラ位置 (y)
アニメーションと静止画像のパース投影の場合のみ有効		
GLOB_EYEPOS_Z	M~	現在のカメラ位置 (z)
アニメーションと静止画像のパース投影の場合のみ有効		
GLOB_TARGPOS_X	N~	現在の目標位置 (x)
アニメーションと静止画像のパース投影の場合のみ有効		
GLOB_TARGPOS_Y	O~	現在の目標位置 (y)
アニメーションと静止画像のパース投影の場合のみ有効		
GLOB_TARGPOS_Z	P~	現在の目標位置 (z)
アニメーションと静止画像のパース投影の場合のみ有効		
GLOB_SUN_AZIMUTH		太陽の方位
[太陽光] ダイアログボックスでの設定に従う		
GLOB_SUN_ALTITUDE		太陽の高度
[太陽光] ダイアログボックスでの設定に従う		

一般要素のパラメータ

GLOB_LAYER		要素のレイヤー
要素が割り当てられるレイヤーの名前		
GLOB_ID		要素のユーザー ID
設定ダイアログボックスで設定されている ID		
GLOB_INTGUID		要素の内部 GUID
プログラムによって生成される一意の内部 GUID （ユーザーによる制御は不可）		
GLOB_ELEVATION	J_	要素の基準高さ
配置フロアを基準とする（ドア、窓を除く。下端高さ、現在の設定に従う）		
GLOB_ELEM_TYPE		要素のタイプ。ラベルおよび特性オブジェクトの場合は、親要素のタイプも含む
0- なし（個々のラベル）、1- オブジェクト、2- ランプ、3- 窓、4- ドア、5- 壁、6- 柱、7- スラブ、8- 屋根、9- 塗りつぶし、10- メッシュ、11- ゾーン、12- 梁		

オブジェクト、ランプ、ドア、窓のパラメータ

SYMB_LINETYPE		ライブラリ部品の線種
2D シンボルのデフォルト線種として適用される		
SYMB_FILL		ライブラリ部品の塗りつぶし種類
断面 / 立面ウィンドウ内のライブラリ部品の断面に適用される		
SYMB_FILL_PEN		ライブラリ部品の塗りつぶしペン
断面 / 立面ウィンドウ内のライブラリ部品の断面に適用される		
SYMB_FBGD_PEN		ライブラリ部品の塗りつぶし背景ペン
断面 / 立面ウィンドウ内のライブラリ部品の断面に適用される		
SYMBL_SECT_PEN		断面のライブラリ部品のペン
断面 / 立面ウィンドウ内のライブラリ部品の断面の輪郭に適用される		
SYMB_VIEW_PEN	L_	ライブラリ部品のデフォルトペン
3D ウィンドウ内の全ての辺と断面 / 立面ウィンドウ内のビューの辺に適用される		
SYMB_MAT	M_	ライブラリ部品のデフォルト材質
SYMB_POS_X	X~	ライブラリ部品の位置 (x)
プロジェクト原点を基準とする（ドア、窓、壁終端を除く。含んでいる壁の始点を基準とする）		
SYMB_POS_Y	Y~	ライブラリ部品の位置 (y)
プロジェクト原点を基準とする（ドア、窓、壁終端を除く。含んでいる壁の始点を基準とする）注記：Y および Z 軸の方向については、310 ページ「建具」を参照してください。		
SYMB_POS_Z	Z~	ライブラリ部品の位置 (z)

プロジェクト原点を基準とする（ドア、窓、壁終端を除く。含んでいる壁の始点を基準とする）注記：Y および Z 軸の方向については、310 ページ「建具」を参照してください。

オブジェクト、ランプのパラメータ

SYMB_ROTANGLE W~ **ライブラリ部品の回転角**

現在の配置基準点を中心にして、設定ダイアログの数値による回転が行われる

SYMB_MIRRORED V~ **ライブラリ部品のミラー**

0- ミラーなし、1- ミラーあり（ミラーは現在の配置基準点を中心にして行われる）ローカル座標系の原点が台形壁ポリゴンの非矩形頂点であるときを除き、壁終端では常に 0 となります。

オブジェクト、ランプ、ドア、窓のパラメータ、カーテンウォール付属品 - リストとラベル用のみ

SYMB_A_SIZE **ライブラリ部品図面用長さ / 幅**

オブジェクト / ランプの長さ、窓 / ドアの幅（固定パラメータ）、付属品の幅

SYMB_B_SIZE **ライブラリ部品図面用幅 / 高さ**

オブジェクト / ランプの幅、窓 / ドアの高さ（固定パラメータ）、付属品の高さ

オブジェクト、ランプ、カーテンウォール付属品 - リストとラベル用のみ

SYMB_Z_SIZE **ライブラリ部品図面用高さ / 長さ**

付属品の長さ、または最後のユーザーパラメータが zzyzx 形式で指定されている場合は図面用高さ、それ以外の場合は 0

窓、ドア、壁終端のパラメータ

WIDO_REVEAL_ON		建具外壁側抱きオン
0-外壁側抱きオフ、1-外壁側抱きオン		
WIDO_SILL	K_	建具の下端奥行き
建具設定の［抱き］タブページで設定された円弧壁の抱き奥行き： 実開口の角からの半径方向		
WIDO_SILL_HEIGHT		建具の図面用下端高さ
WIDO_RSIDE_SILL_HEIGHT		建具の外壁側の下端高さ
WIDO_OPRSIDE_SILL_HEIGHT		建具の外壁と反対側の下端高さ
WIDO_RIGHT_JAMB	B~	建具の右側の抱き
［建具の設定］ダイアログボックスの［抱き］タブページに設定したとおり		
WIDO_LEFT_JAMB		建具の左側の抱き
［建具の設定］ダイアログボックスの［抱き］タブページに設定したとおり		
WIDO_THRES_DEPTH	C~	建具の下端奥行き
［建具の設定］ダイアログボックスの［抱き］タブページに設定したとおり		
WIDO_HEAD_DEPTH	D~	建具の上端奥行き
［建具の設定］ダイアログボックスの［抱き］タブページに設定したとおり		
WIDO_HEAD_HEIGHT		建具の図面用上額縁の高さ
WIDO_RSIDE_HEAD_HEIGHT		建具の外壁側の上額縁の高さ
WIDO_OPRSIDE_HEAD_HEIGHT		建具の外壁と反対側の上額縁の高さ
WIDO_REVEAL_SIDE	E~	抱きは開口と反対側
1-反対側、0-同じ側。要素を配置するとき、デフォルト値は窓の場合 0、ドアの場合 1		
WIDO_FRAME_THICKNESS	F~	建具の枠の厚さ
建具を反転する時は、建具はミラーしてからこの値で自動的に再配置される		
WIDO_POSITION	H~	建具のオフセット
開口部の軸または壁終端と壁の開始点の法線ベクトルとの角度または距離		
WIDO_ORIENTATION		建具開口部向き
左／右 - うまく機能するのは建具がローカル標準に従って作成されている場合のみ		
WIDO_MARKER_TXT		建具のマーカーテキスト
［建具の設定］ダイアログから開く［建具の寸法］サブダイアログで設定されているとおり		
WIDO_SUBFL_THICKNESS		サブフロアの厚さ（敷居補正）
［建具の設定］ダイアログボックスの［パラメータ］タブページ		
WIDO_PREFIX		建具の下端高さ頭文字
［建具の設定］ダイアログから開く［建具の寸法］サブダイアログで設定されているとおり		
WIDO_CUSTOM_MARKER		建具のカスタムマーカースイッチ
1- パラメータは 2D スクリプトで使用可能、自動寸法設定はなし		
WIDO_ORIG_DIST	R_	壁の曲率の中心からのローカル原点の距離
曲線壁の中心点からローカル原点までの距離。直線壁の場合は 0 曲線壁の終点が 0 となる壁終端では負数となります。		
WIDO_PWALL_INSET		手すり壁インセット

窓、ドアのパラメータ - リストとラベル用のみ

WIDO_RSIDE_WIDTH	建具の外壁側の開口部幅
WIDO_OPRSIDE_WIDTH	建具の外壁側と反対側の開口部幅
WIDO_RSIDE_HEIGHT	建具の外壁側の開口部高さ
WIDO_OPRSIDE_HEIGHT	建具の外壁側と反対側の開口部高さ
WIDO_RSIDE_SURF	建具の外壁側の開口部面積
WIDO_OPRSIDE_SURF	建具の外壁側と反対側の開口部面積
WIDO_N_RSIDE_WIDTH	建具の外壁側の開口部図面用幅
WIDO_N_OPRSIDE_WIDTH	建具の外壁側と反対側の開口部図面用幅
WIDO_N_RSIDE_HEIGHT	建具の外壁側の開口部図面用高さ
WIDO_N_OPRSIDE_HEIGHT	建具の外壁側と反対側の開口部図面用高さ
WIDO_N_RSIDE_SURF	建具の外壁側の開口部図面用高さ
WIDO_N_OPRSIDE_SURF	建具の外壁側と反対側の開口部図面用面積
WIDO_VOLUME	建具の開口部体積
WIDO_GROSS_SURFACE	建具の開口部図面用面積
WIDO_GROSS_VOLUME	建具の開口部図面用体積

ランプパラメータ - リストとラベル用のみ

LIGHT_ON	光源がオン
0-光源がオフ、1-光源がオン：［ランプの設定］ダイアログボックスで設定されているとおり（固定パラメータ）	
LIGHT_RED	光源カラーの赤の構成要素
［ランプの設定］ダイアログボックスで設定されているとおり（固定パラメータ）	
LIGHT_GREEN	光源カラーの緑の構成要素
［ランプの設定］ダイアログボックスで設定されているとおり（固定パラメータ）	
LIGHT_BLUE	光源カラーの青の構成要素
［ランプの設定］ダイアログボックスで設定されているとおり（固定パラメータ）	
LIGHT_INTENSITY	照明度
［ランプの設定］ダイアログボックスで設定されているとおり（固定パラメータ）	

ラベルのパラメータ

LABEL_POSITION	ラベルの位置
ラベル位置を定義する 3 つの点の座標を含む配列 [3][2]	
LABEL_CUSTOM_ARROW	［シンボル矢印線の使用］のオン / オフ
1- ［シンボル矢印線の使用］チェックボックスがオンの場合、0- その他の場合	
LABEL_ARROW_PEN	［設定］ダイアログボックスでの矢印線のペン
LABEL_ARROWHEAD_PEN	［設定］ダイアログボックスでの矢印のペン
LABEL_FONT_NAME	［設定］ダイアログボックスのフォント名
LABEL_TEXT_SIZE	［設定］ダイアログボックスのテキストサイズ
LABEL_TEXT_PEN	［設定］ダイアログボックスのテキストのペン
LABEL_FONT_STYLE	［設定］ダイアログボックスのフォントスタイル
0- 標準、1- ボールド、2- イタリック	
LABEL_FRAME_ON	ラベル枠のオン / オフ
1- ラベル枠がオンの場合、0- その他の場合	
LABEL_ANCHOR_POS	ラベル配置位置
0- 中央、1- 上端、2- 下端（設定ダイアログボックスに従う）	
LABEL_ROTANGLE	［設定］ダイアログボックスでの回転角
LABEL_TEXT_ALIGN	［ラベルの設定］ダイアログボックスでのテキストの位置合わせ
1: 左揃え、2: 中央揃え、3: 右揃え、4: 両端揃え	
LABEL_TEXT_LEADING	［ラベルの設定］ダイアログボックスでの行間隔係数
LABEL_TEXT_WIDTH_FACT	［ラベルの設定］ダイアログボックスで設定した幅係数
LABEL_TEXT_CHARSPACE_FACT	文字間隔。［ラベルの設定］ダイアログボックスで設定したとおり

壁のパラメータ－建具のみに使用可能

WALL_ID		**壁ユーザー ID**
WALL_INTGUID		**壁の内部 GUID**
プログラムによって生成される一意の内部 GUID（ユーザーによる制御は不可）		
WALL_RESOL	J~	**曲線壁 3D 解像度**
3D の場合のみ有効		
WALL_THICKNESS	C_	**壁厚さ**
傾斜壁の場合： 開口側の軸から見た壁の厚さ（ローカル Z 軸）		
WALL_START_THICKNESS		**壁始点厚さ**
WALL_END_THICKNESS		**壁終点厚さ**
WALL_INCL		**壁表面の勾配**
傾斜壁の 2 つの表面で構成される角度。一般的な直線壁の場合は 0。		
WALL_HEIGHT	D_	**壁高さ**
WALL_MAT_A	G_	**開口部と反対側の壁の材質**
WALL_MAT_B	H_	**開口部側の壁の材質**
同じ壁に配置された開口ごとに異なってもよい		
WALL_MAT_EDGE	I_	**壁の辺の材質**
WALL_LINETYPE		**壁の線種**
平面図ウィンドウでのみ輪郭に適用される		
WALL_FILL	A~	**壁の塗りつぶし種類**
塗りつぶしインデックス、複合構造の最初の外板		
WALL_FILL_PEN	F_	**壁の塗りつぶしペン**
WALL_COMPS_NAME		**壁の複合構造**
壁の複合構造または断面形状の名前、複合壁のプロファイル属性の名前、複合構造の複合属性の名前、それ以外の場合は空白の文字		
WALL_SKINS_NUMBER		**複合壁またはポリゴン壁の数**
1 ～ 127 の範囲、単一塗りつぶしの場合は 0		
WALL_SKINS_PARAMS		**複合壁またはポリゴン壁のパラメータ**
列が 16 の配列： 塗りつぶし、厚さ、(古い輪郭ペン)、塗りつぶしペン、塗りつぶし背景ペン、芯の状態、上部線ペン、上部線種、下部線ペン、下部線種、終端面ペン、塗りつぶし方向、塗りつぶし種類、面の線種、仕上げ塗りつぶし状態、与えられた方向の塗りつぶし状態、行数は任意		
芯の状態： 0 - 部品以外、1 - 部品、3 - 芯の最後の塗りつぶし、塗りつぶし方向： 0 - グローバル、1 - ローカル、塗りつぶし種類： 0 - 切り取り、1 - 切断面の下、2 - 切断面の上（単純な壁に対して全ての塗りつぶし種類は 0)。平面図の複合壁の D/W では、この変数に全ての切断塗りつぶしのデータが含まれ、平面図の壁の終端では全ての塗りつぶしのデータが含まれます。仕上げ塗りつぶし状態： 0: 仕上げではない、1: 仕上げ、与えられた方向の塗りつぶし状態： 0 - 「塗りつぶし向き」列で設定されたグローバル塗りつぶし向きまたはローカル塗りつぶし向き、1 - 壁の塗りつぶし向きと厚さに一致する塗りつぶし向きとサイズ		
WALL_SECT_PEN	E_	**壁の切断面の輪郭ペン**

平面図ウィンドウと断面／立面ウィンドウの両方で、切断面の輪郭に適用される	
WALL_VIEW_PEN	ビュー上の壁の輪郭ペン
3D ウィンドウ内の全ての辺と断面／立面ウィンドウ内の可視の辺に適用される	
WALL_FBGD_PEN	壁の塗りつぶし背景ペン
WALL_DIRECTION	壁方向
直線壁： 基準線の方向、曲線壁： 円弧の弦の方向	
WALL_POSITION	壁の絶対座標
プロジェクト原点を基準にした壁の開始点の位置	

壁パラメータ－リストとラベル用のみ

WALL_LENGTH_A	基準線側の壁の長さ
WALL_LENGTH_B	基準線と反対側の壁の長さ
WALL_LENGTH_A_CON	基準線側の条件付き壁長さ
WALL_LENGTH_B_CON	基準線と反対側の条件付き壁長さ
WALL_CENTER_LENGTH	壁の中心の長さ
WALL_AREA	壁面積
WALL_PERIMETER	壁外周
WALL_SURFACE_A	基準線側の壁の面積
WALL_SURFACE_B	基準線と反対側の壁の面積
WALL_SURFACE_A_CON	基準線側の条件付き壁面積
WALL_SURFACE_B_CON	基準線と反対側の条件付き壁面積
WALL_GROSS_SURFACE_A	基準線側の壁の総面積
WALL_GROSS_SURFACE_B	基準線と反対側の壁総面積
WALL_EDGE_SURF	壁辺面積
WALL_VOLUME	壁の体積
WALL_VOLUME_CON	屋根の条件付き体積
WALL_GROSS_VOLUME	壁総体積
WALL_VOLUME_A	基準線側の壁仕上げ体積
WALL_VOLUME_A_CON	基準線側の条件付き壁仕上げ体積
WALL_VOLUME_B	基準線と反対側の壁仕上げ体積
WALL_VOLUME_B_CON	基準線と反対側の条件付き壁仕上げ体積

WALL_DOORS_NR	壁のドア数
WALL_WINDS_NR	壁の窓数
WALL_HOLES_NR	単純開口数
WALL_DOORS_SURF	壁内ドア面積
WALL_WINDS_SURF	壁内窓面積
WALL_HOLES_SURF	壁の単純開口面積
WALL_HOLES_SURF_A	基準線側の開口部の分解法面積
WALL_HOLES_VOLUME	壁の開口の分解法体積
WALL_WINDS_WID	壁の窓の組み合わせ幅
WALL_DOORS_WID	壁のドアの組み合わせ幅
WALL_COLUMNS_NR	壁の柱数
WALL_MIN_HEIGHT	壁最小高さ
WALL_MAX_HEIGHT	壁最大高さ
WALL_SKIN_MIN_HEIGHT_A	基準線側の壁仕上げの最小高さ
WALL_SKIN_MAX_HEIGHT_A	基準線側の壁仕上げの最大高さ
WALL_SKIN_MIN_HEIGHT_B	基準線側の壁仕上げの最小高さ
WALL_SKIN_MAX_HEIGHT_B	基準線と反対側の壁仕上げの最大高さ
WALL_SKIN_THICKNESS_A	基準線側の壁仕上げ厚
WALL_SKIN_THICKNESS_B	基準線と反対側の壁仕上げ厚
WALL_INSU_THICKNESS	壁断熱材厚
WALL_AIR_THICKNESS	壁空気層厚
WALL_SECT_PEN	平面図ウィンドウと断面 / 立面ウィンドウの両方で切断面の輪郭に適用される壁の切断面の輪郭ペン
WALL_VIEW_PEN	3D ウィンドウの全ての辺および平面図ウィンドウと断面 / 立面ウィンドウの輪郭辺（切断面の下のビュー上の辺）に適用される、ビュー上の壁の輪郭ペン

柱パラメータ－リストとラベル用のみ

COLU_CORE	芯 / 仕上げの特性
互換用： CPS （Column.Properties）ファイルの特性スクリプトのみで有効	
COLU_HEIGHT	柱の高さ
COLU_MIN_HEIGHT	柱最小高さ
COLU_MAX_HEIGHT	柱最大高さ
COLU_VENEER_WIDTH	柱の仕上げの厚さ
COLU_CORE_X	芯幅
COLU_CORE_Y	芯奥行き
COLU_DIM1	柱の第 1 寸法
COLU_DIM2	柱第 2 寸法
COLU_MAT	柱の材質

注記：壁のラップは、柱の材質を接合壁の材質に置き換えます。

COLU_LINETYPE	柱の線種
平面図ウィンドウでのみ輪郭に適用される	
COLU_CORE_FILL	柱の芯の塗りつぶし
COLU_VENEER_FILL	柱の仕上げの塗りつぶし
COLU_SECT_PEN	柱の切断面の輪郭ペン
平面図ウィンドウと断面 / 立面ウィンドウの両方で断面の輪郭に適用される	
COLU_VIEW_PEN	ビュー上の柱のペン
3D ウィンドウの全ての辺と平面図ウィンドウと断面 / 立面ウィンドウのアウトラインエッジ（切断面の下のビュー上の辺）に適用される	
COLU_CORE_FILL_PEN	柱の芯の塗りつぶしペン
COLU_CORE_FBGD_PEN	柱の芯の塗りつぶし背景ペン
COLU_VENEER_FILL_PEN	柱の仕上げの塗りつぶしペン
COLU_VENEER_FBGD_PEN	柱の仕上げの塗りつぶし背景ペン
COLU_PERIMETER	柱外周
COLU_AREA	柱面積
COLU_VOLUME	柱体積
COLU_GROSS_VOLUME	柱総体積
COLU_CORE_SURF	柱の芯の面積
COLU_CORE_GROSS_SURF	柱の総面積
COLU_CORE_VOL	柱の芯の体積

COLU_CORE_GROSS_VOL	芯総体積
COLU_VENEER_SURF	柱の仕上げの面積
COLU_VENEER_GROSS_SURF	仕上げ総面積
COLU_VENEER_VOL	柱の仕上げの体積
COLU_VENEER_GROSS_VOL	仕上げ総体積
COLU_CORE_TOP_SURF	芯の上部の面積
COLU_CORE_BOT_SURF	芯の下部の面積
COLU_VENEER_TOP_SURF	仕上げ上部の面積
COLU_VENEER_BOT_SURF	仕上げ下部の面積
COLU_CORE_GROSS_TOPBOT_SURF	芯の上部と下部の総面積
COLU_VENEER_GROSS_TOPBOT_SURF	仕上げの上部と下部の総面積

梁パラメータ－リストとラベル用のみ

BEAM_THICKNESS	梁の厚さ
BEAM_HEIGHT	梁の高さ
BEAM_REFLINE_OFFSET	梁の軸を基準にした基準線のオフセット
BEAM_PRIORITY	3D 交差の優先度インデックス
BEAM_MAT_RIGHT	基準線の右側の梁の材質
BEAM_MAT_LEFT	基準線の左側の梁の材質
BEAM_MAT_TOP	壁の上部の材質
BEAM_MAT_BOTTOM	壁の下部の材質
BEAM_MAT_END	両端の梁の材質
BEAM_OUTLINE_LINETYPE	梁の輪郭の線種
BEAM_AXES_LINETYPE	梁の軸の線種
BEAM_FILL	梁の塗りつぶし種類
BEAM_FILL_PEN	梁の塗りつぶしペン
BEAM_SECT_PEN	梁の切断面の輪郭ペン
BEAM_FBGD_PEN	梁の塗りつぶし背景ペン
BEAM_DIRECTION	梁の基準線の方向
BEAM_POSITION	梁の軸の開始点の絶対座標
BEAM_LENGTH_RIGHT	基準線の右側の梁の長さ
BEAM_LENGTH_LEFT	基準線の左側の梁の長さ
BEAM_RIGHT_SURF	基準線の右側の梁の面積
BEAM_LEFT_SURF	基準線の左側の梁の面積
BEAM_TOP_SURF	梁の上部の面積
BEAM_BOTTOM_SURF	梁の下部の面積
BEAM_END_SURF	梁の両端の面積
BEAM_VOLUME	梁の体積
BEAM_VOLUME_CON	梁の条件付き体積
BEAM_HOLES_NR	梁の穴の数
BEAM_HOLES_SURF	梁の穴の総面積
BEAM_HOLE_EDGE_SURF	梁の穴辺の総面積
BEAM_HOLES_VOLUME	梁の穴の総体積

スラブパラメータ－リストとラベル用のみ

SLAB_THICKNESS	スラブの厚さ
SLAB_MAT_TOP	スラブの上面の材質
SLAB_MAT_EDGE	スラブの辺の材質
SLAB_MAT_BOTT	スラブの下面の材質
SLAB_LINETYPE	スラブの線種
SLAB_FILL	スラブの塗りつぶし
塗りつぶしインデックス－複合構造の場合の値は負	
SLAB_FILL_PEN	スラブの塗りつぶしペン
SLAB_FBGD_PEN	スラブの塗りつぶし背景ペン
SLAB_COMPS_NAME	スラブの複合構造
複合構造の名前	
SLAB_SKINS_NUMBER	複合スラブの外板の数
1 ～ 8 の範囲、単一塗りつぶしの場合は 0	
SLAB_SKINS_PARAMS	複合スラブの外板のパラメータ
列が 15 の配列：塗りつぶし、厚さ、(古い輪郭ペン)、塗りつぶしペン、塗りつぶし背景ペン、芯の状態、上部線ペン、下部線ペン、下部線種、終端面ペン、塗りつぶし方向、面の線種、仕上げ塗りつぶし状態、行数は任意	
芯の状態：0 - 部分ではない、1 - 部分、3 - 芯の最終塗りつぶし、塗りつぶし向き：0 - グローバル、1 - ローカル、面の線種：現行の ArchiCAD では、常に 0 - 切り取り、今後、壁に使用可能、仕上げ塗りつぶし状態：0 仕上げではない、1: 仕上げ	
SLAB_SECT_PEN	断面内のスラブの輪郭ペン
平面図ウィンドウと断面 / 立面ウィンドウの両方で断面の輪郭に適用される	
SLAB_VIEW_PEN	スラブのペン
3D ウィンドウ内の全ての辺と断面 / 立面ウィンドウ内の可視の辺に適用される	
SLAB_TOP_SURF	スラブの上部の面積
SLAB_GROSS_TOP_SURF	スラブ上部総面積
SLAB_TOP_SURF_CON	スラブの上部の面積
SLAB_BOT_SURF	スラブの下部の面積
SLAB_GROSS_BOT_SURF	スラブ下部総面積
SLAB_BOT_SURF_CON	スラブの下部の面積
SLAB_EDGE_SURF	スラブの辺の面積
SLAB_GROSS_EDGE_SURF	スラブ辺総面積
SLAB_PERIMETER	スラブ外周
SLAB_VOLUME	スラブ体積
SLAB_GROSS_VOLUME	スラブ総体積

SLAB_VOLUME_CON	スラブの条件付き体積
SLAB_SEGMENTS_NR	スラブの辺数
SLAB_HOLES_NR	スラブの穴数
SLAB_HOLES_AREA	スラブの穴面積
SLAB_HOLES_PRM	スラブの穴外周

屋根パラメータ－リストとラベル用のみ

ROOF_THICKNESS	屋根の厚さ
ROOF_ANGLE	屋根勾配
ROOF_MAT_TOP	屋根の上面の材質
ROOF_MAT_EDGE	屋根の辺の材質
ROOF_MAT_BOTT	屋根の下面の材質
ROOF_LINETYPE	屋根の線種
平面図ウィンドウでのみ輪郭に適用される	
ROOF_FILL	屋根の塗りつぶし
塗りつぶしインデックス－複合構造の場合の値は負	
ROOF_FILL_PEN	屋根の塗りつぶしペン
ROOF_FBGD_PEN	屋根の塗りつぶし背景ペン
ROOF_COMPS_NAME	屋根の複合構造
複合構造の名前	
ROOF_SKINS_NUMBER	複合屋根の外板の数
1 ～ 8 の範囲、単一塗りつぶしの場合は 0	
ROOF_SKINS_PARAMS	複合屋根の外板のパラメータ
列が 15 の配列：塗りつぶし、厚さ、(古い輪郭ペン)、塗りつぶしペン、塗りつぶし背景ペン、芯の状態、上部線ペン、下部線ペン、上部線種、下部線種、終端面ペン、塗りつぶし向き、面の線種、仕上げ塗りつぶし状態、行数は任意	
芯の状態：0 - 部分ではない、1 - 部分、3 - 芯の最終塗りつぶし、塗りつぶし向き：0 - グローバル、1 - ローカル、面の線種：現行の ArchiCAD では、常に 0 - 切り取り、今後、壁に使用可能、仕上げ塗りつぶし状態：0 仕上げではない、1: 仕上げ	
ROOF_SECT_PEN	屋根の切断面の輪郭ペン
平面図ウィンドウと断面／立面ウィンドウの両方で、切断面の輪郭に適用される	
ROOF_VIEW_PEN	ビュー上の屋根ペン
3D ウィンドウの全ての辺と平面図ウィンドウと断面／立面ウィンドウのアウトラインエッジ（切断面の下のビュー上の辺）に適用さ	
ROOF_BOTTOM_SURF	屋根の下部の面積
ROOF_GROSS_BOTTOM_SURF	屋根下部総面積
ROOF_BOTTOM_SURF_CON	屋根の下部の面積
ROOF_TOP_SURF	屋根の上部の面積
ROOF_GROSS_TOP_SURF	屋根上部総面積
ROOF_EDGE_SURF	屋根の辺の面積
ROOF_GROSS_EDGE_SURF	屋根辺総面積
ROOF_PERIMETER	屋根外周
ROOF_VOLUME	屋根の体積
ROOF_GROSS_VOLUME	屋根総体積
ROOF_VOLUME_CON	屋根の条件付き体積

ROOF_SEGMENTS_NR	屋根の辺数
ROOF_HOLES_NR	屋根の穴数
ROOF_HOLES_AREA	屋根の穴面積
ROOF_HOLES_PRM	屋根の穴外周
ROOF_INSU_THICKNESS	屋根の壁断熱材厚
ROOF_RIDGE	屋根棟長さ
ROOF_VALLEY	屋根谷長さ
ROOF_GABLE	屋根破風長さ
ROOF_HIP	屋根隅棟長さ
ROOF_EAVES	屋根軒長さ
ROOF_PEAK	屋根頂点長さ
ROOF_SIDE_WALL	屋根側面壁接続長さ
ROOF_END_WALL	屋根端部壁接続長さ
ROOF_TRANSITION_DOME	ドーム屋根接続長さ
ROOF_TRANSITION_HOLLOW	ボールド屋根接続長さ

塗りつぶしパラメータ－リストとラベル用のみ

FILL_LINETYPE	塗りつぶしの線種
FILL_FILL	塗りつぶしの塗りつぶし種類
FILL_FILL_PEN	塗りつぶしの塗りつぶしパターンペン
FILL_PEN	塗りつぶしペン
FILL_FBGD_PEN	塗りつぶし背景ペン
FILL_SURF	塗りつぶし面積
FILL_PERIMETER	塗りつぶし外周
FILL_SEGMENT_NR	塗りつぶしの辺数
FILL_HOLES_NR	塗りつぶしの穴数
FILL_HOLES_PRM	塗りつぶしの穴外周
FILL_HOLES_AREA	塗りつぶしの穴面積

メッシュパラメータ－リストとラベル用のみ

MESH_TYPE *1-閉じているボディ、2-上面と辺、3-上面のみ*	メッシュタイプ
MESH_BASE_OFFSET	基準レベルに対する下面のオフセット
MESH_USEREDGE_PEN	メッシュのユーザー定義尾根のペン
MESH_TRIEDGE_PEN	メッシュの三角エッジのペン
MESH_SECT_PEN *平面図ウィンドウと断面／立面ウィンドウの両方で、壁の断面の輪郭に適用される*	断面のメッシュの輪郭ペン
MESH_VIEW_PEN *3D ウィンドウ内の全ての辺と断面／立面ウィンドウ内のビューの辺に適用される*	ビュー上の輪郭ペン
MESH_MAT_TOP	メッシュの上面の材質
MESH_MAT_EDGE	メッシュの辺の材質
MESH_MAT_BOTT	メッシュの下面の材質
MESH_LINETYPE *平面図ウィンドウでのみ輪郭に適用される*	メッシュの線種
MESH_FILL	メッシュの塗りつぶし種類
MESH_FILL_PEN	メッシュの塗りつぶしペン
MESH_FBGD_PEN	メッシュの塗りつぶし背景ペン
MESH_BOTTOM_SURF	メッシュの下部の面積
MESH_TOP_SURF	メッシュの上部の面積
MESH_EDGE_SURF	メッシュの辺の面積
MESH_PERIMETER	メッシュの周囲の長さ
MESH_VOLUME	メッシュの体積
MESH_SEGMENTS_NR	メッシュの辺数
MESH_HOLES_NR	メッシュの穴数
MESH_HOLES_AREA	メッシュの穴面積
MESH_HOLES_PRM	メッシュの穴外周

カーテンウォールパラメータ - リストとラベル用のみ

CWALL_ID	カーテンウォールのユーザー ID
CWALL_FRAMES_LENGTH	カーテンウォールのフレームの長さ
CWALL_CONTOUR_FRAMES_LENGTH	カーテンウォールの輪郭のフレームの長さ
CWALL_MAINAXIS_FRAMES_LENGTH	カーテンウォールの主グリッドラインのフレームの長さ
CWALL_SECAXIS_FRAMES_LENGTH	カーテンウォールの副グリッドラインのフレームの長さ
CWALL_CUSTOM_FRAMES_LENGTH	カーテンウォールのその他のフレームの長さ
CWALL_PANELS_SURF	カーテンウォールパネルの面積
CWALL_PANELS_SURF_N	カーテンウォールの北側のパネルの面積
CWALL_PANELS_SURF_S	カーテンウォールの南側のパネルの面積
CWALL_PANELS_SURF_E	カーテンウォールの東側のパネルの面積
CWALL_PANELS_SURF_W	カーテンウォールの西側のパネルの面積
CWALL_PANELS_SURF_NE	カーテンウォールの北東側のパネルの面積
CWALL_PANELS_SURF_NW	カーテンウォールの北西側のパネルの面積
CWALL_PANELS_SURF_SE	カーテンウォールの南東側のパネルの面積
CWALL_PANELS_SURF_SW	カーテンウォールの南西側のパネルの面積
CWALL_SURF	カーテンウォールの面積
CWALL_SURF_BOUNDARY	境界フレームによるカーテンウォール面積境界
CWALL_LENGTH	カーテンウォールの長さ
CWALL_HEIGHT	カーテンウォールの高さ
CWALL_SLANT_ANGLE	カーテンウォールの傾斜角度
CWALL_THICKNESS	カーテンウォールの厚さ
CWALL_PANELS_NR	カーテンウォールパネルの数
CWALL_PATTERN_ANGLE	カーテンウォールのパターン角度

カーテンウォールフレームパラメータ – リストとラベル用のみ

CWFRAME_TYPE *「不可視」、「一般」、「ガラス縁」、またはGDLオブジェクトの名前*	**フレームのタイプ**
CWFRAME_CLASS *0 – マリオン、1 – トランザム、2 – 境界、3 – カスタム*	**フレームのクラス**
CWFRAME_POSITION *0 – 主グリッドライン、1 – 副グリッドライン、2 – 境界、3 – その他*	**フレームの位置**
CWFRAME_DIRECTION *0から90までの角度*	**フレームの傾斜角度**
CWFRAME_WIDTH	**フレーム幅**
CWFRAME_DEPTH	**フレーム奥行き**
CWFRAME_LENGTH	**フレームの長さ**
CWFRAME_MAT	**柱の材質**

カーテンウォールパネルのパラメータ – リストとラベル用のみ

CWPANEL_TYPE *「一般」またはGDLオブジェクトの名前*	メッシュタイプ
CWPANEL_CLASS *0 – メイン、1 – 個別、2 – カスタム*	パネルのクラス
CWPANEL_VERTICAL_DIRECTION *-90から90までの角度*	パネルの外壁側表面の傾斜角度
CWPANEL_HORIZONTAL_DIRECTION *-180から180までの角度*	パネルの外壁側表面のプロジェクトの北からの角度
CWPANEL_WIDTH	パネル幅
CWPANEL_NOMINAL_WIDTH	パネルの有効幅
CWPANEL_HEIGHT	パネルの高さ
CWPANEL_NOMINAL_HEIGHT	パネルの図面用高さ
CWPANEL_THICKNESS	パネルの厚さ
CWPANEL_SURF	パネルの面積
CWPANEL_GROSS_SURF	パネル総面積
CWPANEL_NOMINAL_SURF	パネルの図面用面積
CWPANEL_PERIMETER	パネル外周
CWPANEL_MAT_OUTER	パネルの外壁側表面の材質
CWPANEL_MAT_INNER	パネルの内壁側表面の材質
CWPANEL_MAT_CUT	パネルの辺の材質
CWPANEL_FUNCTION *0 – 固定、1 – ドア、2 – 窓*	パネルの関数
CWPANEL_ORIENTATION *左側／右側*	ドアと窓のパネルの開口部向き

カーテンウォール接続部パラメータ – リストとラベル用のみ

CWJUNC_TYPE *GDLオブジェクトの名前*	接合部タイプ

カーテンウォール付属品パラメータ – リストとラベル用のみ

CWACC_TYPE *GDLオブジェクトの名前*	付属品タイプ

自由なユーザーグローバル変数

GLOB_USER_1	S_	
GLOB_USER_2	T_	
GLOB_USER_3	U_	
GLOB_USER_4	V_	
GLOB_USER_5	W_	
GLOB_USER_6	X_	
GLOB_USER_7	Y_	
GLOB_USER_8	Z_	
GLOB_USER_9	G~	
GLOB_USER_10	I~	1 ～ 10 の自由な変数はデフォルトでは数字に初期化されます。
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		11 ～ 20 の自由な変数はデフォルトでは文字列に初期化されます。

グローバル変数 GLOB_WORLD_ORIGO_... の使用例：

グローバル変数 GLOB_WORLD_ORIGO_...：

```
ADD2 -GLOB_WORLD_ORIGO_OFFSET_X - SYMB_POS_X, -GLOB_WORLD_ORIGO_OFFSET_X -SYMB_POS_Y
LINE2 -0.1, 0.0, 0.1, 0.0
LINE2 0.0, -0.1, 0.0, 0.1
HOTSPOT2 0.0, 0.0, 1
TEXT2 0, 0, "( 0.00 ; 0.00 )"
TEXT2 0, 0.5, "World Origo"
DEL TOP
if ABS(GLOB_WORLD_ORIGO_OFFSET_X) > 0.01 OR ABS(GLOB_WORLD_ORIGO_OFFSET_Y) > 0.01 THEN
       ADD2 - SYMB_POS_X, - SYMB_POS_Y
       LINE2 -0.1, 0.0, 0.1, 0.0
       LINE2 0.0, -0.1, 0.0, 0.1
       HOTSPOT2 0.0, 0.0, 2
```

```
        TEXT2 0, 0, "(" + STR (GLOB_WORLD_ORIGO_OFFSET_X, 9, 4) + "; " +  STR (GLOB_WORLD_ORIGO_OFFSET_Y, 9, 4)
+ " )"
        TEXT2 0, 0.5, "Virtual Origo"
        DEL TOP
ENDIF
if ABS(GLOB_WORLD_ORIGO_OFFSET_X + SYMB_POS_X) > 0.01 OR ABS(GLOB_WORLD_ORIGO_OFFSET_Y + SYMB_POS_Y) > 0.01 THEN
         LINE2 -0.1, 0.0, 0.1, 0.0
         LINE2 0.0, -0.1, 0.0, 0.1
         HOTSPOT2 0.0, 0.0, 3
TEXT2 0, 0, "(" + STR (GLOB_WORLD_ORIGO_OFFSET_X + SYMB_POS_X, 9, 4) + "; " + STR (GLOB_WORLD_ORIGO_OFFSET_Y +
SYMB_POS_Y, 9, 4)+ " )"TEXT2 0, 0.5, "Object Placement"
ENDIF
```

古いグローバル変数

古いグローバル変数名を使用してもかまいませんが、できる限り新規の名前を使用してください。古いグローバル変数は長い名前の新規の変数にそれぞれ対応しています。

```
A_    GLOB_SCALE
B_    GLOB_HSTORY_ELEV
C_    WALL_THICKNESS
D_    WALL_HEIGHT
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

要求

REQ

REQ (parameter_string)

プログラムの現在の状態を問い合せます。そのパラメータ、つまり質問は文字列です。GDL インタプリタは数値で答えます。質問の内容が理解できないときには、答えは負数になります。

現在の質問のリスト：

″GDL_version″
GDL コンパイラ / インタプリタのバージョン番号。

警告： これは、ArchiCAD と同じバージョンではありません。

″ÉvÉçÉOÉâÉÄ″
プログラムのコード（例えば、1: ArchiCAD）

″Serial_number″
キープラグのシリアル番号

″Model_size″
現在の 3D データ構造のバイト単位のサイズ

″Red_of_material name″
″Green_of_material name″
″Blue_of_material name″
与えられた材質のカラー構成要素を 0 ～ 1 の RGB 値で定義します。

″Red_of_pen index″
″Green_of_pen index″
″Blue_of_pen index″
与えられたペンのカラー構成要素を 0 ～ 1 の RGB 値で定義します。

″Pen_of_RGB r g b″
与えられたカラーに最も近いペンのインデックスを定義します。定数 r、g、および b の値は、0 ～ 1 までの範囲となります。

REQUEST

REQUEST (question_name, name | index, variable1 [, variable2,...])

第 1 パラメータは質問の文字列を表し、質問がある場合には第 2 パラメータで質問の対象を表します。パラメータは、文字列タイプでも数値タイプでもかまいません（例えば、質問を「Rgb_of_material」としてその対象を材質の名前としたり、「Rgb_of_pen」としてその対象をペンのインデックスとすることができます)。他のパラメータは、戻り値（答え）が格納される変数の名前です。関数の戻り値は答えの番号です（質問の形式が間違っている場合や存在しない名前を指定した場合には、値は 0 になります)。

REQUEST (″**Name_of_program**″, ″″, program_name)

与えられた変数に ″ArchiCAD″ のようなプログラムの名前を返します。

例：プログラムの名前を印刷する場合

```
n=REQUEST(″Name_of_program″, ″″, program_name)
PRINT program_name
```

REQUEST (″**Name_of_macro**″, ″″, my_name)

REQUEST (″**Name_of_main**″, ″″, main_name)

これらの関数コールを実行した後、my_name 変数にはマクロ名が代入されます。これに対し、main_name にはメインマクロの名前が代入されます（メインマクロがない場合には、空の文字列が代入されます)。

REQUEST (″**ID_of_main**″, ″″, id_string)

平面図に配置されたライブラリ部品の場合、ツールの設定ダイアログボックスで設定されている識別子が id_string 変数に返されます（その他の場合には、空の文字列が返されます)。

REQUEST (″**Name_of_plan**″, ″″, name)

与えられた変数に現在のプロジェクトの名前を返します。

REQUEST (″**Story**″, ″″, index, story_name)

変数 index　tory_name 現在のフロアのインデックスと名前を返します。

REQUEST (″**Home_story**″, ″″, index,
story_name)

変数 index と story_name 変数に配置フロアのインデックスと名前を返します。

REQUEST (″**Story_info**″, expr, nStories,
index1, name1, elev1, height1 [,
index2, name2, ...])

与えられた変数に、フロアの数、フロアのインデックス、名前、高度、次のフロアまでの高さなどのフロア情報を連続的に返します。expr が数式の場合は、フロアのインデックスを意味します。フロアの数と指定されたフロアに関する情報だけが返されます。expr が文字列式の場合は、全てのフロアの情報の要求を意味します。この関数の戻り値は、正常に検索された値の数です。

例：

```
DIM t[]
     n = REQUEST ("STORY_INFO", "", nr, t)
     FOR i = 1 TO nr
     nr = STR ("%.0m", t [4 * (i - 1) + 1])
     name = t [4 * (i - 1) + 2]
    elevation = STR ("%m", t [4 * (i - 1) + 3])
     height = STR ("%m", t [4 * (i - 1) + 4])
     TEXT2 0, -i, nr + "," + name + "," + elevation + "," + height
     NEXT i
```

REQUEST *("**Internal_id**", "", id)*

ïœêîidÇ…ÉâÉCÉuÉâÉäïîïiÇÃì‡îîIDÇšï‘ÇµÇ‹Ç²ÅB

REQUEST ("**Linear_dimension**", "", format_string)

REQUEST ("**Angular_dimension**", "", format_string)

REQUEST ("**Angular_length_dimension**", "" format_string)

REQUEST ("**Radial_dimension**", "", format_string)

REQUEST ("**Level_dimension**", "", format_string)

REQUEST ("**Elevation_dimension**", "", format_string)

REQUEST ("**Window_door_dimension**", "", format_string)

REQUEST ("**Sill_height_dimension**", "", format_string)

REQUEST ("**Area_dimension**", "" format_string)

REQUEST ("**Calc_length_unit**", "", format_string)

REQUEST ("**Calc_area_unit**", "", format_string)

REQUEST ("**Calc_volume_unit**", "", format_string)

REQUEST ("**Calc_angle_unit**", "", format_string)

これらの要求で、ArchiCAD の［**オプション**］→［**環境設定**］→［**寸法**］および［**計算単位**］ダイアログボックスに設定された寸法フォーマットを知ることができます。これらの要求によって返されるフォーマット文字列は、STR () 関数の第 1 パラメータとして使用することができます。

例：

```
format = "" num = 60.55
REQUEST ("Angular_dimension", "",format)!"%.2dd"
```

TEXT2 0, 0, STR (format, num)!60.55

REQUEST ("**Clean_intersections**", "", state)
[包絡表示] 機能の状態を返します（オンの場合は 1、オフの場合は 0)。

REQUEST ("**Zone_category**", "", name, code)

ゾーンの場合は、現在のゾーンカテゴリの名前とコード文字列を返します。

REQUEST ("**Zone_relations**", "", category_name, code, name, number [,category_name2, code2, name2, number2])

与えられた変数に、ゾーンカテゴリ名、ゾーンカテゴリコード、およびこの要求を含むライブラリ部品が位置付けられているゾーンの名前と数を返します。ドアと窓の場合は、最大 2 つまでのゾーンが対象になります。この要求の戻り値は、正常に検索された値の数です（つまり、どのゾーンにもライブラリ部品がない場合は、0 が返されます)。

REQUEST ("**Zone_relations_of_owner**", "", category_name, code, name, number [,category_name2, code2, name2, number2])

与えられた変数に、カテゴリ名とカテゴリコード、およびオブジェクトの所有者が位置づけられているゾーンの名前と数を返します。つまり、ライブラリ部品が所有者（ドア / 窓ラベルおよびドア / 窓マーカーなど）をもつ場合に役立ちます。ドアラベルの場合は、所有者はドアです。
ドアと窓の場合は、最大 2 つまでのゾーンが対象になります。要求の戻り値は、正常に検出された値の数です（オブジェクトに所有者がない場合、または所有者がゾーン内側にない場合は 0 です)。

REQUEST ("**Zone_colus_area**", "", area)

変数 area に、現在のゾーンに配置されている柱の総面積を返します。ゾーンスタンプの場合のみ有効です。

REQUEST ("**Custom_auto_label**", "", name)

変数 name に、ライブラリ部品のカスタム自動ラベルの名前を返します。ライブラリ部品が存在しない場合は、空の文字列を返します。

REQUEST ("**Rgb_of_material**", name, r, g, b)

REQUEST ("**Rgb_of_pen**", penindex, r, g, b)

REQUEST ("**Pen_of_RGB**", "r g b", penindex)

REQ () 関数と同じように（ただし、1 回のコールで)、指定された変数に材質とペンの構成要素 r、g 、b の値または与えられた RGB 値に対応するペンのインデックスを返します。

REQUEST ("**Height_of_style**", name, height [, descent, leading])

与えられた変数に、ミリメートル単位で測定されたスタイルの高さ（メートル単位の高さは、高さ /1000 * GLOB_SCALE)、下降（次の基準線から下降線までのミリメートル単位の距離）および上昇（下降線から上昇線までのミリメートル単位の距離）を返します。

REQUEST ("**Style_info**", name, fontname [, size, anchor, face_or_slant])

1 つ前に定義したスタイルに関する情報を与えられた変数に返します（DEFINE STYLE ステートメントの style パラメータを参照。*219 ページ「DEFINE STYLE」*メインスクリプトに定義されているスタイルに関する情報を収集するマクロで便利です。

REQUEST（"**Name_of_material**", index, name）

与えられた変数に、インデックスで識別された材質名を返します。

REQUEST（"**Name_of_fill**", index, name）

変数 name に、インデックスで識別された塗りつぶしの名前を返します。

REQUEST（"**Name_of_line_type**", index, name）

与えられた変数に、インデックスで識別されたラインの名前を返します。

REQUEST（"**Name_of_style**", index, name）

与えられた変数に、インデックスで識別されたスタイルの名前を返します。
インデックス < 0 の場合、GDL スクリプトまたは MASTER_GDL ファイルに定義されている材質、塗りつぶし、線種、またはスタイルを参照します。index = 0 の要求コールは、変数にデフォルトの材質または線種を返します （塗りつぶしとスタイルの場合は、空の文字列）。
この要求の戻り値は、正常に検出された値の数です（エラーが発生しなかった場合は 1、インデックスが有効でないエラーの時は 0）。

REQUEST ("WINDOW_DOOR_SHOW_DIM", "", show)

9.0 以前のバージョンでは、**［オプション］→［表示オプション］→［ドアと窓］**が［寸法付きで表示］に設定されている場合は、変数 show に 1 を返し、その他の場合は 0 を返していました。9.0 から、ドアと窓がそれぞれ別の表示オプションに分割されました。そこで互換性を保つために、AC はその要求が窓（または窓のマーカー）とドア（またはドアのマーカー）のどちらで使用されるか確認し、自動的に対応する表示オプションを返しています。その他の場合（シンボル、ランプ、ラベル）は、窓オプションが返されます。現在の［表示オプション］に従ってカスタム寸法を表示 / 非表示にするのに使用できます。
9.0 から、「window_show_dim」と「door_show_dim」の別々の要求が使用可能になりました。

REQUEST ("window_show_dim", "", show)

［モデル表示オプション］→［窓］オプションで［マーカー付き］がチェックされている場合は、変数 show に 1 を返し、その他の場合は 0 を返します。

REQUEST ("door_show_dim", "", show)

［モデル表示オプション］→［ドア］オプションで［マーカー付き］がチェックされている場合は、変数 show に 1 を返し、その他の場合は 0 を返します。

REQUEST（"**name_of_listed**", "", name）

変数 name に、この要求を含む特性タイプのライブラリ部品に対応付けられているライブラリ部品の名前を返します。要素（壁、スラブなど）の場合、名前は空白の文字列です。

REQUEST（"**window_door_zone_relev**", " ", out_direction）

ドアと窓にのみ有効。"ZONE_RELATIONS" 要求の補足として使用します。次の場合は変数 out_direction variable に 1 を返します。
「ドア / 窓の開口方向が "ZONE_RELATIONS" 要求で識別された第 1 のゾーン方向である場合」。また次の場合は 2 を返します。
「開口方向が第 2 のゾーン方向である場合」。部屋が 1 つしかなく、開口方向が外側である場合にも 2 を返します。

REQUEST（"**window_door_zone_relev_of_owner**", "", out_direction）

ライブラリ部品の親がドアまたは窓（マーカー、ラベル）である場合のみ有効です。"zone_relations_of_owner" 要求の補足として使用します。建具の開口方向がゾーン関連タイプの要求で識別された第 1 のゾーン方向である場合は、変数 out_direction に 1 を返し、開口方向が第 2 のゾーン方向である場合は 2 を返します。部屋が 1 つしかなく、開口方向が外側である場合にも 2 を返します。

REQUEST（"**matching_properties**", type, name1, name2, ...）

type = 1 の場合は、与えられた変数に、個別に連動された特性ライブラリ部品名を返し、その他の場合は条件によって連動された特性ライブラリ部品名を返します。連動ラベルで使用される場合は、この関数はラベルが連動された要素の特性を返します。

REQUEST（"**Constr_Fills_display**", "", optionVal）

渡された変数に［**ドキュメント**］→［**モデル表示を設定**］→［**モデル表示オプション**］（以前の組み立て要素塗りつぶし）で設定された［切断塗りつぶし表示］オプションの値を入れて返します。有効値は次のとおりです。
1 - 切断塗りつぶし輪郭のみ表示（以前の空き）
2 - 切断塗りつぶし輪郭のみ分離線で表示（以前の塗りつぶしなし）
4 - 切断塗りつぶしパターン： ソリッド（以前のソリッド）
6 - 切断塗りつぶしパターン： 設定による（以前のベクトルハッチング）
この関数の戻り値は、正常に検出された値の数で、エラーが発生した場合は 0 です。

REQUEST（"**Working_length_unit**", "", format_string）

REQUEST（"**Working_angle_unit**", "", format_string）

これらの要求で、［**オプション**］→［**環境設定**］→［**作業単位**］ダイアログボックスで設定された作業単位のフォーマットを知ることができます。これらは、STR () 関数の第 1 パラメータとして使用できるフォーマット文字列を返します。これらの要求を使用できるのは、パラメータまたはユーザーインターフェイススクリプトを解釈するときにかぎられます。

REQUEST（"**Model_length_unit**", "", format_string）

REQUEST（"**Layout_length_unit**", "", format_string）

これらの要求で、[オプション] → [プロジェクト設定] → [作業単位] ダイアログボックスで設定されたレイアウトおよびモデル単位のフォーマットを知ることができます。これらは、STR () 関数の第 1 パラメータとして使用できるフォーマット文字列を返します。これらの要求を使用できるのは、パラメータまたはユーザーインターフェイススクリプトを解釈するときにかぎられます。

REQUEST ("Model_text_size_unit", "", format_string)

REQUEST ("Layout_text_size_unit", "", format_string)

これらの要求によって、ユーザーはレイアウトおよびモデルのテキストサイズフォーマットを得ることができます。これらは、STR () 関数の第 1 パラメータとして使用できるフォーマット文字列を返します。これらの要求を使用できるのは、パラメータまたはユーザーインターフェイススクリプトを解釈するときにかぎられます。

IND (MATERIAL, name_string)
IND (FILL, name_string)
IND (LINE_TYPE, name_string)
IND (STYLE, name_string)
IND (TEXTURE, name_string)

この関数は、材質、塗りつぶし、線種またはスタイル、およびテクスチャの属性の現在のインデックスを返します。結果の数値は、主に、コールするマクロと同じ属性を必要とするマクロに転送するために使用します。結果は、一時的な定義の場合は負で、グローバル定義の場合は正です。

『ArchiCAD ヘルプ』の「設定」章の「属性」および204 ページ「インライン属性定義」も参照してください。

REQUEST ("**ASSOCLP_PARVALUE**", expr, name_or_index, type, flags, dim1, dim2, values)

与えられた変数に、この要求を含むライブラリ部品が対応付けられているライブラリ部品パラメータに関する情報を返します。特性オブジェクト、ラベル、マーカーオブジェクトで使用することができます。

この関数の戻り値は、正常に検出された値の数です。指定されたパラメータが存在しないまたはエラーが発生した場合には、0 です。

要求の対象、対応付けられているライブラリ部品のパラメータ名、またはインデックス式。

1 つ前の式のタイプによって、パラメータのインデックスまたは名前を返します (パラメータ名が指定されている場合はインデックスを、インデックスが指定されている場合は名前を返します)。

タイプ : パラメータタイプ。有効値
- 1: ブール
- 2: 整数
- 3: 実数
- 4: 文字列
- 5: 長さ
- 6: 角度
- 7: ライン

8: 材質
9: 塗りつぶし
10: ペンカラー
11: 光源のスイッチ
12: rgb カラー
13: 照明度
14: 区切り文字
15: タイトル

flags:

flags = j1 + 2 * j2 + 64 * j7 + 128 * j8
ここで、ji は 0 または 1 をとります。
j1 (1): パラメータリスト内の子 / インデント
j2 (2): パラメータリスト内の太字テキスト
j7 (64): 無効（全ての状況でロック）
j8 (128): パラメータリスト内で非表示

dim1, dim2: パラメータの次元を返します。単純タイプの場合は両方とも 0、1 次元の配列の場合は dim1 **>** 0、dim2 = 0 となります。2 次元の配列の場合は、両方とも > 0 となります。dim1 は行の数、dim2 は列の数です。

p_values: パラメータ値または値の配列を返します。配列要素は、これを格納するために指定された変数の次元とは別に、1 次元の配列として行ごとに連続的に返されます。変数が動的配列ではない場合、格納できる最大数の要素が格納されます（単純な変数の場合、1 つ、つまり最初の要素だけです)。値が 2 次元の動的配列の場合、全ての要素は最初の行に格納されます。

REQUEST (″**ASSOCLP_NAME**″, ″″, name)

与えられた変数に、ラベルまたはマーカーオブジェクトに連動されているライブラリ部品の名前を返します。要素（壁、スラブなど）の場合は、名前は空の文字列です。

REQUEST (″**ASSOCEL_PROPERTIES** ″, parameter_string, nr_data, data)

与えられた変数に、自身の特性データまたはこの要求を含むライブラリ部品が（ラベルおよび連動マーカーオブジェクトで）連動されている要素特性を返します。この関数の戻り値は、正常に検出された値の数です。特性データが見つからないまたはエラーが発生した場合には、0 です。この関数は、リストプロセス中は、特性オブジェクトでは作用しません。

特性データレコードの要求されたフィールドを表すカンマで区切られたキーワードの組み合わせ。レコードは、それに応じた順番に並べられます 有効値：

ISCOMP
DBSETNAME
KEYCODE

KEYNAME
CODE
NAME
FULLNAME
QUANTITY
TOTQUANTITY
UNITCODE
UNITNAME
UNITFORMATSTR
PROPOBJNAME

nr_data: データ項目の数を返します。

data: 特性データ、つまりパラメータ文字列によって指定されたフィールドを含み、これらのフィールド別に並べられたレコードを返します。値は、要求されたレコードフィールドを連続的に含む 1 次元の配列として返されます。これは、これを格納するために指定された変数の次元とは無関係です。変数が動的配列ではない場合、格納できる最大数の要素が格納されます（単純な変数の場合、1 つ、つまり最初の要素だけです)。変数が 2 次元の動的配列の場合、全ての要素は最初の行に格納されます。

例：

```
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]  ! 名前
            j = j + 1
        ENDIF
    NEXT i
ENDIF
```

REQUEST ("**REFERENCE_LEVEL_DATA**", "",
name1, elev1, name2, elev2, name3, elev3)

与えられた変数に、**[環境設定]** → **[作業単位]** → **[基準レベル]** ダイアログボックスで設定されているとおりの基準レベルの名前とレベルを返します。
この関数の戻り値は、正常に検出された値の数で、エラーが発生した場合は 0 です。

REQUEST ("**ANCESTRY_INFO**", expr, name [,
guid, parent_name1, parent_guid1,

```
    ...,
    parent_namen, parent_guidn)
```

ライブラリ部品に関する起源情報

expr = 0 の場合、与えられた変数に、この要求関数を含むライブラリ部品の名前とグローバルで一意の識別子を返します。オプションとして、この関数はライブラリ部品の親の名前とグローバルで一意の識別子を返します（parent_namei、paretn_guidi）。親テンプレートがロードされていない場合は、その名前は空の文字列になります。
expr = 1 の場合、この関数を含むテンプレートによって置き換えられるライブラリ部品の情報を返します。この場合、実際にテンプレートに置き換えられない場合は、値は返されません。
この要求の戻り値は、正常に検索された値数です。

例：

```
DIM strings[]
n = REQUEST ("ANCESTRY_INFO", 1, name, guid, strings)
IF n > 2 THEN
    ! data of replaced library part
    TEXT2  0, -1, "replacing: " + name + ' ' + guid
    ! parents
    l = -2
    FOR i = 1 to n - 2 STEP 2
        TEXT2 0, l, strings [i]
        l = l - 1
    NEXT i
ENDIF
```

REQUEST ("**TEXTBLOCK_INFO**",
    textblock_name, width, height)

与えられた変数に、以前に定義した TEXTBLOCK の x および y 方向のサイズを返します。サイズは、TEXTBLOCK の fixed_height パラメータ値に応じて、モデルスペースの mm 単位または m 単位となります(1 の場合はモデルスペースのミリメートル、0 の場合はモデルスペースのメートル)。iù が 0 の場合、要求は計算済みの幅および高さを返します。iù が TEXTBLOCK 定義で指定されている場合、要求はその幅に対応した計算済みの高さを返します。

REQUEST{2} ("**Material_info**", name_or_index,
    param_name, value_or_values)

REQUEST{2} ("**Material_info**", name_or_index,
    extra_param_name,
    value_or_values)

与えられた変数に、指定された材質のパラメータ（または、その他のパラメータ。*222 ページ「追加データ」*を参照）に関する情報を返します。RGB 情報は、3 つの別々の変数に返されます。テクスチャ情報は、テクスチャ定義に応じて file_name、width、height、mask、rotation_angle の各変数に返されます。その他の全てのパラメータ情報は、シングル変数に返されます。材質の定義のパラメータに対応した、考えられる材質パラメータ名は次のとおりです。

gs_mat_surface_rgb（面積R, G, B [0.0..1.0]）
gs_mat_ambient（環境係数 [0.0..1.0]）
gs_mat_diffuse（拡散係数 [0.0..1.0]）
gs_mat_specular（光沢係数 [0.0..1.0]）
gs_mat_transparent（透過係数 [0.0..1.0]）
gs_mat_shining（光沢 [0.0..100.0]）
gs_mat_transp_att（透過減衰 [0.0..4.0]）
gs_mat_specular_rgb（鏡面反射カラー R, G, B [0.0..1.0]）
gs_mat_emission_rgb（放射カラー R, G, B [0.0..1.0]）
gs_mat_emission_att（放射減衰 [0.0..65.5]）
gs_mat_fill_ind (fill index)
gs_mat_fillcolor_ind（塗りつぶしカラーインデックス）
gs_mat_texture（テクスチャインデックス）
例：

```
REQUEST{2} ("Material_info", "Brick-Face", "gs_mat_ambient", a)
REQUEST{2} ("Material_info", 1, "gs_mat_surface_rgb", r, g, b)
REQUEST{2} ("Material_info",
      "Brick-Face", "gs_mat_texture",
      file_name, w, h, mask, alpha)
REQUEST{2} ("Material_info", "My-Material", "my_extra_parameter", e)
```

```
REQUEST ("FONTNAMES_LIST", "", fontnames)
```

渡された変数に、使用しているコンピュータで使用可能なフォント名（キャラクタコードを含めて）を返します。このリスト（またはこのリストの任意の部分）は、フォント名ポップアップを設定する［VALUE］コマンドで使用できます。この関数の戻り値は、正常に検出された値の数で、エラーが発生した場合は 0 です。
例：

```
dim fontnames[]
REQUEST ("FONTNAMES_LIST", "", fontnames)
VALUES "f" fontnames, CUSTOM
```

この VALUES コマンドは、単純な文字列タイプのパラメータ「f」のフォント名ポップアップを集めます。「fontnames」変数は使用可能なフォント名（キャラクタコードを含めて）からなり、手動または REQUEST ("FONTNAMES_LIST", ...) コマンドで設定することができます。CUSTOM キーワードは、他のプラットフォーム / コンピュータ上で欠落したフォントを正しく扱うために必要です。これが指定されていれば、別のプラットフォーム / コンピュータで設定したフォント名が現在の環境で欠落していれば、パラメータ設定でカスタム値として保持されます（指定されていなければ、VALUES コマンドを実行すると、パラメータ設定の欠落した文字列のポップアップ値が現在の最初の文字列の値に置換されます）。

REQUEST ("HomeDB_info", "", homeDBIntId, homeDBUserId, homeDBName, homeContext)

与えられた変数に、内部 ID（整数）、ユーザー ID およびホームデータベース（この要求を含むライブラリ部品が位置づけられている場所）の名前（文字列）を返します。

・平面図に配置されている場合： フロアの内部 ID、文字列のインデックスと名前、homeContext = 1
・断面に配置されている場合： 断面の内部 ID、参照 ID と名前、homeContext = 2
・詳細図に配置されている場合： 詳細図の内部 ID、参照 ID と名前、homeContext = 3
・マスタレイアウトに配置されている場合： レイアウトの内部 ID、空白の文字列と名前、homeContext = 4
・レイアウトに配置されている場合： レイアウトの内部 ID、番号と名前、homeContext = 5

ラベルでは、戻りデータはラベル要素を参照します。プランファイルの異なる ArchiCAD データベースで要素を一意に識別するために、集合データを使用することができます。

REQUEST ("floor_plan_option",""storyViewpointType)

モデル表示オプションで設定されているフロアビューポイントタイプを返します。0 は「平面図」、1 は「天伏図」を意味しています。

REQUEST ("class_of_fill", index, class)

変数 class に、インデックスで識別された塗りつぶしのクラスを返します。

1: ベクトル塗りつぶし
2: シンボル塗りつぶし
3: 透過塗りつぶし
4: 線形グラデーション
5: 円形グラデーション
6: 画像塗りつぶし

REQUEST ("view_rotangle", "", angleViewRotation)

現在のビューの回転角を返します。

REQUEST (extension_name, parameter_string, variable1, variable2, ...)

質問が上記のいずれにも該当しない場合は、REQUEST() 関数によって、機能拡張固有の名前として使用されます。この機能拡張が［アドオン］フォルダにある場合は、与えられている変数名と同じ数の値を取得するために使用されます。パラメータ文字列は機能拡張によって解釈されます。

APPLICATION_QUERY

APPLICATION_QUERY (extension_name, parameter_string, variable1, variable2, ...)

GDL では、個々のアプリケーションでそれぞれの状況に応じた固有の要求関数を使用することができます。このようなクエリオプションは、GDL 構文では定義されていません。それぞれの固有オプションについては、指定されたアプリケーションの GDL 開発者用ドキュメントを参照してください。ArchiCAD には、基本ライブラリドキュメントが適用されます。

LIBRARYGLOBAL

LIBRARYGLOBAL (object_name, parameter, value)

object_name（可能な場合）で定義されたライブラリグローバルオブジェクトの、現在のモデル表示オプションのパラメータ値を入力します。グローバルオブジェクトが現在ライブラリにロードされている場合、または以前ロードされており現在のモデル表示オプションセットに設定が保存されている場合は、ライブラリのグローバル設定が使用可能です。

成功した場合は 1、そうでない場合は 0 を返します。

object_name: ライブラリグローバルオブジェクトの名前

parameter: 要求されたパラメータの名前

value: 要求されたパラメータ値が入力されます。

例:

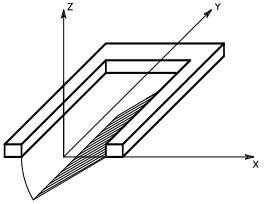
```
success = LIBRARYGLOBAL ("MyGlobalOptions", "detLevel2D", det)
if success > 0 then
  text2 0, 0, det
ELSE[ELSE]
  text2 0, 0, "Not available"
ENDIF
```

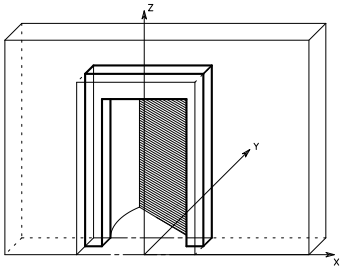
建具

この章では、建具ライブラリ要素の作成に関連する特殊オプションについて解説しています。

概要

建具を壁に挿入すると、ライブラリ部品の座標系のデフォルト位置が回転され、壁面に対して X-Y 平面が垂直方向に、Z 軸が水平方向になります。原点は、開口部の外側の下中央となります。つまり、建具を X-Y 平面上の要素としてモデリングすることになります。下の図を参照してください。





建具タイプのライブラリ部品の動作は特別なため、通常アクセスできない特殊内蔵投影（90 度方向からの上下反転側面図）から 2D シンボルが生成されます。2D シンボルと 3D 形状は、配置枠下側（Y 方向）中央（X 方向）の建具原点に調整されます。ただし、建具を Z 軸方向に自由に配置できるように、Z 軸方向には調整されません。

上記規則を考慮して、正しく機能する建具作成の注意点を以下に列記します。

・平面図ウィンドウで建具を作成する場合、配置する壁の内側から見ていると考えてください。

・GL レベルを、壁の外壁面とします。

・窓枠のように壁の内側に配置する要素は、GL レベルより上に作成します。

・外側に開くドアパネルは、GL レベルより下に作成します。

建具ライブラリ部品の作成

建具タイプのライブラリ部品を作成する際、以下のような状況で調整が必要です。

・直線壁での矩形の建具
・3D の調整
- 直線壁での矩形以外の建具
- 直線壁での矩形の建具
- 曲線壁での矩形以外の建具

・2D の調整
  - カスタム形状の壁開口
    - WALLHOLE2
  - 壁ポリゴンの拡張
    - WALLBLOCK2
    - WALLLINE2
    - WALLARC2

直線壁の矩形の建具

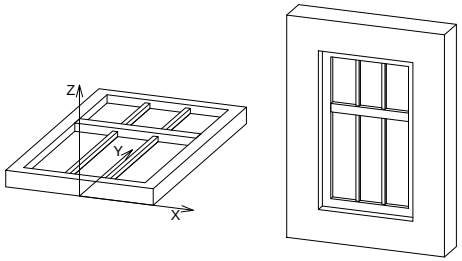
これはドアと窓を作成する最も簡単な方法です。できる限り、PRISM_ や RECT などの単純な GDL コマンドを使用することをお勧めします。

建具要素の表面材質を壁の表面材質と合わせるには、要素の下面を壁の外側、上面を内側に合わせます。これには、建具が配置される壁の材質を表す WALL_MAT_A、WALL_MAT_B および WALL_MAT_EDGE を使用してスクリプトを作成します。2D スクリプトでは、グローバル変数 WALL_SECT_PEN、WALL_FILL_PEN および WALl_FILL ・ を活用します。これらの変数は、建具が配置される壁の平面図での壁輪郭と塗りつぶしのペン番号と、塗りつぶしのインデックス番号となります。複合構造壁の場合は、対応するグローバル変数を使用します。

詳細は、273 ページ「その他」を参照してください。

オブジェクトライブラリには、建具マクロが用意されています。この GDL スクリプトには、ライブラリ内の建具に使用される一般的な建築要素が含まれています。また、一般的に使用される枠、パネルなど建具の一部を生成するためのマクロも用意されています。建具ライブラリ部品を開くと、コールするマクロの種類やマクロが生成する部品のタイプを確認できます。

例：



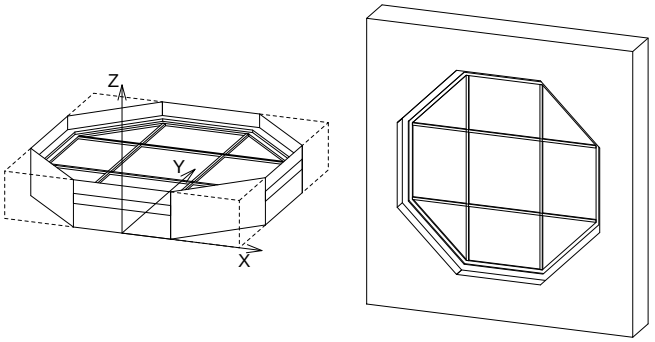

```
A=0.9: B=1.5: C=0.1: D=0.08
E=0.08: F=0.9: G=0.03: H=3
PRISM_ 10,C,
    -A/2, 0, 15, A/2, 0, 15,
    A/2, B, 15, -A/2, B, 15,
    -A/2, 0, -1,
    -A/2+D, D, 15, A/2-D, D, 15,
    A/2-D, B-D, 15, -A/2+D, B-D, 15,
    -A/2+D, D, -1
ADD -A/2+D, F, 0
BRICK A-2*D, E, C
ADD -G/2, -F+D, C/2
GOSUB 1
ADDZ -G
GOSUB 1
DEL 2
MATERIAL "Glass"
ADD0, -F+D, C/2
RECT A-2*D, F-D
ADDY F-D+E
RECT A-2*D, B-F-E-D
END
1: FOR I=1 TO H-1
ADDX (A-2*D)/3
BLOCK G, F-D, G
ADDY F+E-D
BLOCK G, B-F-D-E, G
DEL 1
NEXT I
DEL H-1
RETURN
```

3D の調整

直線壁の矩形以外の建具

建具を作成する際、建具を配置すると壁に矩形の穴が必ず作成される点を理解することが重要です。この穴の大きさは、建具ライブラリ部品のパラメータ A および B によって決定されます。このため、建具が立面的に矩形でない場合、切り取られた矩形の穴全体を建具で埋めることができません。WALLHOLE または WALLNICHE コマンドを使用すれば、壁から切り取るポリゴン形状を定義できます。これには、以下の 2 つの解決方法があります。

・切り取られた矩形部分と建具本体との間を埋める壁部分を 3D スクリプトに追加します。この場合、壁部分の辺の可視性に注意を払って下さい。



・WALLHOLE または WALLNICHE コマンドを使用すれば、壁から切り取るポリゴン形状を定義できます。

WALLHOLE

```
WALLHOLE n, status,
      x1, y1, mask1,
      ...
      xn, yn, maskn
      [, x, y, z]
```

n: ポリゴンの節点の数

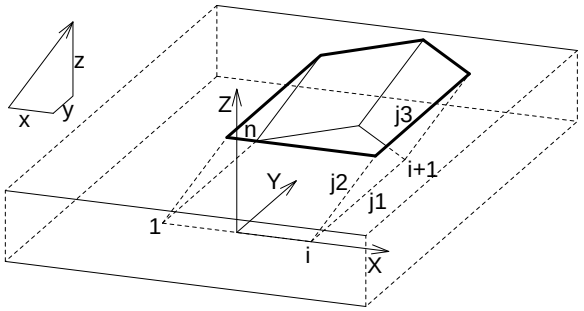
status:
      1: 生成されたポリゴンと辺にボディの属性を使用
      2: 生成された切り取りポリゴンは、通常のポリゴンとして取り扱われる

xi, yi: 切断面ポリゴンの座標

maski: CUTPOLYA ステートメントとほぼ同じ

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

x, y, z: 任意の方向ベクトル（デフォルトは建具の Z 軸）



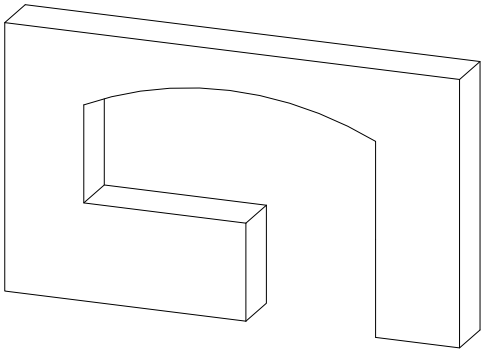
このコマンドを建具の 3D スクリプトに使用して、配置先の壁にカスタム穴を切り取ることができます。現在の壁の 3D 生成中に、全ての建具の 3D スクリプトがモデルを生成しないで解析され、WALLHOLE コマンドをまとめてコールします。WALLHOLE コマンドがある場合、スクリプト内で定義されているポリゴン切断面と方向のチューブ形状で、現在の壁が切り取られます。建具には複数の WALLHOLE を使用できるので、同じ建具に交差するような複数の開口部を切り取ることもできます。建具の 3D スクリプトで WALLHOLE コマンドが見つかると、その建具には矩形の開口部は生成されません。

注記： カスタム穴の場合は、3D 外壁側抱き部分が自動的に生成されません。したがって、スクリプトから生成する必要があります。

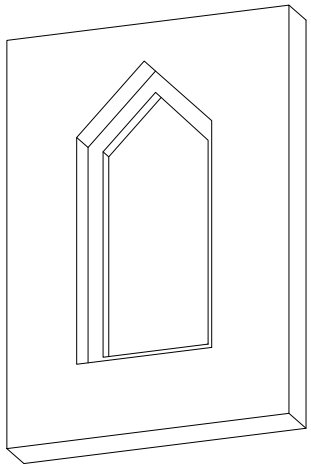
このようにしてカスタマイズされた穴は 3D でのみ可視となります。WALLHOLE コマンドは 2D には影響しません。必要に応じて 2D 表現もスクリプト化します（[平面図の輪郭] はオフにします）。

できる限り凸型のポリゴン切断面を使用してください。凹型のポリゴンを使用すると、シェーディングやレンダリングに問題が生じたり、切り取りエラーが発生する場合があります。凸型ポリゴンを組み合わせて、凹型にします。

例：



```
RESOL 72
L1=2.7: L2=1.2: H1=2.1: H2=0.3: H3=0.9
R=((L1/2)^2+H2^2)/(2*H2)
A=ATN((L1/2)/(R-H2))
WALLHOLE 5,1,
     -L1/2,H3,15,
     L1/2,H3,15,
     L1/2,H1-H2,13,
     0,H1-R,915,
     0,2*A,4015
WALLHOLE 4,1,
     L1/2-L2,0,15,
     L1/2,0,15,
     L1/2,H3,15,
     L1/2-L2,H3,15
```



```
WALLHOLE    5,1,
    -0.45,  0,      15,
    0.45,   0,      15,
    0.45,   1.5,    15,
    0,      1.95,   15,
    -0.45,  1.5,    15

PRISM_      12,     0.1,
    -0.45,  0,      15,
    0.45,   0,      15,
    0.45,   1.5,    15,
    0,      1.95,   15,
    -0.45,  1.5,    15,
    -0.45,  0,      -1,
    -0.35,  0.1,    15,
    0.35,   0.1,    15,
    0.35,   1.45,   15,
    0,      1.80,   15,
    -0.35,  1.44,   15,
    -0.35,  0.1,    -1
```

WALLNICHE

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

CUTFORM 定義と同じ。

method: 切り取りボディの形式を制御します。
1: 角柱形状
2: 角錘形状
3: ウェッジ形状切り取りボディ ウエッジ上端の方向は Y 軸に平行で、その位置は rx, ry, rz (ry は無視)。

status: 切り取りボディの範囲と生成された切り取りポリゴンおよび新規の辺の取り扱いを制御します。

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

j1: 生成されたポリゴンと辺にボディの属性を使用

j2: 生成された切り取りポリゴンは正規ポリゴンとして扱う

j4, j5: 切り取りの制限を定義します:
j4 = 0 および j5 = 0: 有限切り取り、
j4 = 0 および j5 = 1: 半無限切り取り、
j4 = 1 および j5 = 1: 無限切り取り

j6: 切り取りボディを使用して、ブール差ではなくブール積を生成 (CUTFORM コマンドでのみ使用可)。

j7: 切り取りボディの下部で生成される辺は表示されません。

j8: 切り取りボディの上部で生成される辺は表示されません。

rx, ry, rz: 切り取り形式が角柱形状の場合は切り取り方向を、切り取り形式が角錐状の場合は角錐の上端を定義します。

d: 切り取りの終わりまでの rx,ry,rz に沿った距離を定義します。切り取りが無限の場合、このパラメータは無効です。切り取りが有限の場合、切り取りボディの始点はローカル座標系になり、ボディは rx,ry,rz で定義された方向に沿った距離 d で終わります。

切り取りが半無限の場合、切り取りボディの始点は rx,ry,rz で定義された方向に沿った距離 d になり、半無限切り取りの方向は rx,ry,rz で定義された方向の逆になります。

mask: 切り取りボディの辺の可視性を定義します。

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

j1: ポリゴンは、切り取られるボディに入り込むときに可視の辺を作成

j2: 切り取り形式の縦方向の辺は可視

j3: ポリゴンは、切り取られるボディから出るときに可視の辺を作成

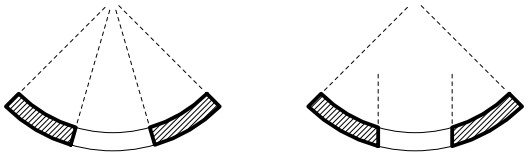
j4: 切り取り形式の下辺は可視

j5: 切り取り形式の上辺は可視

j7: 縦方向の辺の視点に依存する可視性を制御

曲線壁での矩形の建具の作成

建具を曲線壁に配置する時は、壁にあける穴の側面が以下の図のように変化する場合があります。

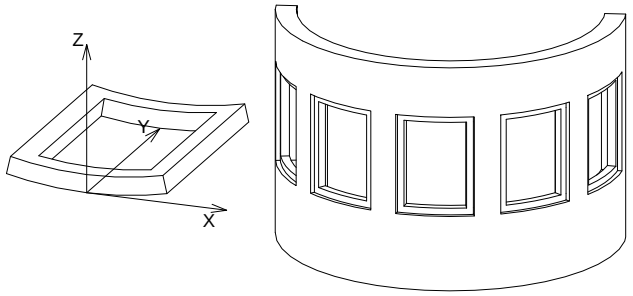


上図左側の壁の穴は、プログラムに自動的に建具用の穴を切り取らせた場合に作成されます。この場合、側面は半径方向と一致します。右側の穴は、建具オブジェクトの 3D スクリプトで WALLHOLE コマンドを使用して切り取られています。これらの点を考慮してオブジェクト自体を作成する必要があります。

その他にも、曲線壁に配置する建具を直線と曲線のどちらにするかも考慮する必要があります。

上図左側の直線の建具の場合、オブジェクトの厚さと幅と壁の厚さが密接に関連しています。一定の寸法を超えると、オブジェクトが壁に収まらなくなるからです。実際に曲線の建具を使用する時は、この問題は発生しません。

例：



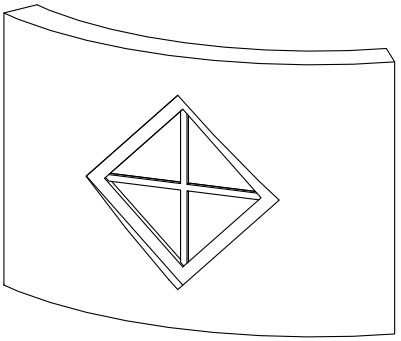

```
RESOL 72
ROTX -90
MULY -1
C= 0.12: Z=(360*A)/(2*R_*PI)
Y= (360*C)/(2*R_*PI)
A1= 270+Z/2: A2=270-Z/2
GOSUB 1
ADDZ B
MULZ -1
GOSUB 1
DEL 2
ADDZ C
GOSUB 2
MULX -1
GOSUB 2
END
1:
PRISM_ 9, C,
 COS(A2)*R_, SIN(A2)*R_+R_, 11,
 COS(A2+Y)*R_, SIN(A2+Y)*R_+R_, 13,
 0, R_, 900,
 0, Z-2*Y, 4009,
 COS(A1)*R_, SIN(A1)*R_+R_, 11,
 COS(A1)*(R_-0.1), SIN(A1)*(R_-0.1)+R_, 11,
 COS(A1-Y)*(R_-0.1),     SIN(A1-Y)*(R_-0.1)+R_,13,
 0, -(Z-2*Y), 4009,
 COS(A2)*(R_-0.1), SIN(A2)*(R_-0.1)+R_, 11
RETURN
```

```
2:
PRISM_ 4, B-2*C,
 COS(A2)*R_, SIN(A2)*R_+R_, 10,
 COS(A2+Y)*R_, SIN(A2+Y)*R_+R_, 15,
COS(A2+Y)*(R_-0.1), SIN(A2+Y)*(R_-0.1)+R_,10,
 COS(A2)*(R_-0.1), SIN(A2)*(R_-0.1)+R_,10
RETURN
```

曲線壁での矩形以外の建具

ここでも、曲線壁に矩形の建具作成の概要が適用されます。

例：



```
C=0.1: D=0.025
Z=A/2-SQR(2)*C: Y=A/2-SQR(2)*C-D
ADDY A/2
WALLHOLE 4, 1,
        0, -A/2, 15,
        A/2, 0, 15,
        0, A/2, 15,
        -A/2, 0, 15
PRISM_ 10, 0.1,
        0, -A/2, 15,
        A/2,  0, 15,
        0, A/2, 15,
        -A/2, 0, 15,
        0, -A/2, -1,
        0, -Z, 15,
        Z, 0, 15,
        0, Z, 15,
```

```
        -Z, 0, 15,
        0, -Z, -1
ADDZ 0.02
GOSUB 1
ADDZ 0.03
GOSUB 1
ADDY -Z
SET MATERIAL "Glass"
ROTZ 45
RECT SQR(2)*Z, SQR(2)*Z
END
1:
PRISM_ 16, 0.03,
        0, -Z, 15,
        D, -Y, 15,
        D, -D, 15,
        Y, -D, 15,
        Z, 0, 15,
        Z, D, 15,
        D, D, 15,
        D, Y, 15,
        0, Z, 15,
        -D, Y, 15,
        -D, D, 15,
        -Y, D, 15,
        -Z, 0, 15,
        -Y, -D, 15,
        -D, -D, 15,
        -D, -Y, 15
RETURN
```

2D の調整

- カスタムの壁の開口部の切り取り

デフォルトでは、建具を配置すると、壁に矩形の穴が開きます。2D のこの穴の大きさは、建具ライブラリ部品のパラメータ A および B によって決定されます。カスタムの抱きを使用したり穴を埋めるには、壁にカスタム形状の穴を開けるか、または平面図で壁を拡張する必要があります。

この問題は、WALLHOLE2、WALLBLOCK2、WALLLINE2 および WALLARC2 のコマンドを使用して対応できます。

WALLHOLE2

WALLHOLE2 n, fill_control, fill_pen, fill_background_pen, fillOrigoX, fillOrigoY, fillAngle,
x1、y1、s1、
...
xn、yn、sn

平面図の表面ポリゴンを伴う壁の開口部の定義。壁の切取り部分のみに作用し、表示壁ポリゴンは影響を受けません。表面ポリゴンには輪郭はありません。

このコマンドは、建具オブジェクトの 2D スクリプトでのみ使用できます。

コマンドのパラメータ設定は、POLY2_B{2} とほぼ同じです。

fill_control = 2*j2 + 8*j4 + 16*j5 + 32*j6 + 64*j7

j2、j4、j5 は 0 または 1 をとります。

j2 (2):

ポリゴンの表面塗りつぶしを描画

j4 (8):

ローカル塗りつぶし向き

j5 (16):

ローカル塗りつぶしは、壁の方向に揃えます（塗りつぶしの原点は壁の原点で、方向は一致します）

j6 (32)、j7 (64):

塗りつぶし種類の判別

0: 作図塗りつぶし

32: 切断塗りつぶし

64: 表面塗りつぶし

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

WALLHOLE2 の上級版で、塗りつぶしの歪みを高度な方法で制御することができます。
これは、図形定義において POLY2_B{5} と同等です。
distortion_flags = j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 128*j8

ここで、各 ji フラグは 0 または 1 をとります。distortion_flags では、0 から 255 までの値が有効です。この範囲以外の値を使用しないでください。

j1-j7: POLY2_B{5} コマンドと同様

j8 (128): ローカル塗りつぶしは、壁の方向に揃えます（塗りつぶしの原点は壁の原点で、方向は一致します)。

j4 が設定されている場合のみ有効です。歪み配列（mij パラメータ）は省略されます。

壁ポリゴンの拡張

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
```

平面図での壁ポリゴン（拡張名）の定義。切り取りポリゴンおよび表示壁ポリゴンは両方とも定義されたポリゴンで切り取られます。別の建具オブジェクトで WALLHOLE2 によって定義された壁の開口部は、WALLBLOCK2 コマンドで生成されたポリゴンを切り取ります。一方、同一オブエクトから生成された壁の穴はポリゴンを切り取りません。
このコマンドは、建具オブジェクトの 2D スクリプトでのみ使用できます。
コマンドのパラメータ設定は、WALLHOLE2 と同じです。

WALLLINE2

WALLLINE2 x1, y1, x2, y2

平面図での 2 点間の壁線（拡張名）の定義。別の建具オブジェクトで WALLHOLE2 によって定義された壁の開口部は、WALLLINE2 コマンドで生成された線を切り取ります。一方、同一オブエクトから生成された壁の穴は線を切り取りません。

このコマンドは、建具オブジェクトの 2D スクリプトでのみ使用できます。

コマンドのパラメータ設定は、LINE2 と同じです。

WALLARC2

WALLARC2 x, y, r, alpha, beta

alpha で始まり beta で終わる角度を持ち半径 r、中心点が（x, y）にある円弧で、配置先の壁によって描画されます。他の建具オブジェクトで WALLHOLE2 によって定義された壁の開口部は、WALLARC2 コマンドで生成された円弧を切り取ります。一方、同一オブジェクトから生成された壁の穴は円弧を切り取りません。

このコマンドは、建具オブジェクトの 2D スクリプトでのみ使用できます。

コマンドのパラメータ設定は、ARC2 と同じです。

平面図から作成される GDL

GDL スクリプトまたはライブラリ部品として平面図を保存すると、以下の GDL 要素となります。これらの GDL スクリプトは、カスタムライブラリ部品のテンプレートとして使用可能です。

キーワード

共通のキーワード

オペレータ、関数

FOR, NEXT
DO, WHILE, ENDWHILE
REPEAT, UNTIL
IF, THEN, ELSE, ENDIF
GOTO
GOSUB
RETURN
END
EXIT
PUT
GET
USE
NSP
CALL, PARAMETERS
PRINT
OPEN
INPUT
VARTYPE
OUTPUT
CLOSE
DIM
BREAKPOINT

予約キーワード

以下のキーワードは予約キーワードです。互換性を確保するために用意されているか、公表されていません。

BAS
BOX
FILTER
GDLBIN
LIN
線
NOD
NODE
ORIGO
PARS
PLOTMAKER
PLOTTER
RECT_
SFLINE
TET
TETRA
TRI
WALL_
VOCA
UI_OK
UI_CANCEL

3D 専用

ADDX, ADDY, ADDZ
ADD
MULX, MULY, MULZ
MUL
ROTX, ROTY, ROTZ
ROT
XFORM

HOTSPOT
LIN_
RECT
POLY, POLY_
PLANE, PLANE_
CIRCLE
ARC

BLOCK, BRICK
CYLIND
SPHERE
ELLIPS
CONE
PRISM, PRISM_, CPRISM_, BPRISM_, FPRISM_, SPRISM_
SLAB, SLAB_, CSLAB_
CWALL_, BWALL_, XWALL_
WALLHOLE[WALLHOLE]
梁
CROOF_
ARMC
ARME
ELBOW
EXTRUDE
PYRAMID
REVOLVE
RULED

SWEEP
TUBE, TUBEA
COONS
メッシュ
MASS
LIGHT
PICTURE
TEXT
VERT, TEVE
VECT
EDGE
PGON, PIPG
COOR
BODY
BASE
BINARY

CUTPLANE
CUTSHAPE
CUTPOLY
CUTPOLYA
CUTEND

DEFINE MATERIAL
DEFINE TEXTURE
[SET] MATERIAL
SHADOW
MODEL

SECT_FILL

2D 専用

ADD2
MUL2
ROT2

HOTSPOT2
LINE2
RECT2
POLY2, POLY2_, POLY2_A, POLY2_B
ARC2
CIRCLE2
SPLINE2, SPLINE2A

PICTURE2
TEXT2
FRAGMENT2

PROJECT2

DEFINE FILL[DEFINE FILL]
DEFINE FILLA
DEFINE LINE_TYPE

[SET] FILL
[SET] LINE_TYPE
DRAWINDEX
DRAWING2
DRAWING3

2D および 3D 用

DEL
[LET]
RADIUS
RESOL
TOLER
PEN
DEFINE STYLE
SET] STYLE

非ジオメトリックスクリプト

特性スクリプト

DATABASE_SET
DESCRIPTOR
COMPONENT
REF
SURFACE3D
VOLUME3D

POSITION

WALLS
COLUMNS
BEAMS
DOORS
WINDOWS
オブジェクト
PITCHED_ROOFS
HIP_ROOFS
LIGHTS
HATCHES
ROOMS
MESHES

DRAWING
BINARYPROP

特性スクリプト

VALUES
PARAMETERS
LOCK

インターフェイススクリプト

UI_DIALOG
UI_PAGE
UI_BUTTON
UI_PREV:
UI_NEXT:
UI_GROUPBOX
UI_SEPARATOR
UI_PICT
UI_STYLE
UI_OUTFIELD
UI_INFIELD
UI_FUNCTION
UI_LINK
UI_CURRENT_PAGE
UI_TOOLTIP

現在の GDL キーワードのアルファベット順リスト

A

ABS (x) x の絶対値（x が整数の場合は整数、それ以外の場合は実数）を返します。

ACS (x) x のアークコサインを返します。(-1.0 <= x <= 1.0; 0° <= ACS(x) <= 180°).

ADD dx, dy, dz

ADD2 x, y:

ADDGROUP (g_expr1, g_expr2)

ADDX dx

ADDY dy

ADDZ dz

AND （あるいは &） 論理積 優先順位 6

ARC r, alpha, beta

ARC2x, y, r, alpha, beta

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

ARME l, r1, r2, h, d

ASN (x) x のアークサインを返します。(-1.0 <= x <= 1.0; -90° <= ASN(x) <= 90°).

ATN (x) x のアークタンジェントを返します。(-90° <= ATN(x) <= 90°).

B

BASE

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

BINARY mode [, section]

BINARYPROP

BITSET (x, b [, expr])

BITTEST (x, b)

BLOCK a, b, c

BODY status

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

BREAKPOINT expression

BRICK a, b, c

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

C

CALL macro_name_string [,] **PARAMETERS ALL** [name1=value1 , ... namen=valuen][[,][**RETURNED_PARAMETERS** r1, r2, ...]

CALL macro_name_string [,] **PARAMETERS** [name1=value1 , ... namen=valuen][[,] **RETURNED_PARAMETERS** r1, r2, ...]

CALL macro_name_string [,] **PARAMETERS** [name1=value1 , ... namen=valuen]

CEIL (x) x よりは小さくない最小の整数値（常に整数）を返します。（例, CEIL(1.23) = 2; CEIL (-1.9) = -1）.

CIRCLE r

CIRCLE2 x, y, r 117

CLOSE 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 94

COOR wrap, vert1, vert2, vert3, vert4

COS (x) x のコサインを返します。

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

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,
...,
xn, yn, alphan, sn

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

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

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

CUTPLANE{2} angle [, 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]

CUTSHAPE d [, status]
[statement1 statement2 ... statementn]
CUTEND

status: 生成された切り取りポリゴンの扱いを制御します。指定がなければ （互換性を確保するため)、デフォルト値は 3 です。

status = j1 + 2*j2
j1: 生成されたポリゴンと辺にボディの属性を使用
j2: 生成された切り取りポリゴンは、通常のポリゴンとして取り扱われる

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

D

DATABASE_SET set_name [descriptor_name, component_name, unit_name, key_name, criteria_name, list_set_name]

DEFINE EMPTY_FILL name [[,] FILLTYPES_MASK fill_types]

DEFINE FILL name [[,] FILLTYPES_MASK fill_types,] pattern1, pattern2, pattern3, pattern4, pattern5, pattern6,
pattern7, pattern8,
spacing, angle, n,
frequency1, direction1, offset_x1, offset_y1, m1,
length11, ... length1m,
...
frequencyn, directionn, offset_xn,
lengthn1, ... lengthnm

DEFINE FILL parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE FILL_A parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE 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 LINE_TYPE name spacing, n,
length1, ... lengthn

DEFINE LINE_TYPE parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE MATERIAL name [,] BASED_ON orig_name [,] PARAMETERS name1 = expr1 [, ...][[,] ADDITIONAL_DATA name1 =
expr1 [, ...]]

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

DEFINE MATERIAL parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE SOLID_FILL name [[,] FILLTYPES_MASK fill_types]]

DEFINE STYLE name font_family, size, anchor, face_code

DEFINE STYLE{2} name font_family, size, face_code

```
DEFINE SYMBOL_FILL name [,][FILLTYPES_MASK fill_types,]
      pat1, pat2, pat3, pat4, pat5, pat6, pat7, pat8,
      spacingx1, spacingy1, spacingx2, spacingy2,
      angle, scaling1, scaling2, macro_name [,]
      PARAMETERS [name1 = value1, ... namen = valuen]

DEFINE SYMBOL_FILL parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE SYMBOL_LINE name dash, gap, macro_name PARAMETERS [name1 = value1, ... namen = valuen]

DEFINE SYMBOL_LINE parameters [[,] ADDITIONAL_DATA name1 = value1, name2 = value2, ...]

DEFINE TEXTURE name expression, x, y, mask, angle

DEL n [, begin_with]

DEl TOP

DESCRIPTOR name [,code, keycode]

DIM var1[dim_1], var2[dim_1][dim_2], var3[ ],
      var4[ ][ ], var5[dim_1][ ],
      var5[ ][dim_2]

DO
      [statment1
      statement2
      ...
      statementn]

DRAWINDEX number

DRAWING2 [expression]

DRAWING3 projection_code, angle, method

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

DRAWING3{3} projection_code, angle, method , parts[, backgroundColor, fillOrigoX, fillOrigoY,
      filldirection][[,]
      PARAMETERS name1=value1 , ... namen=valuen]

DRAWING
```

E

EDGE vert1, vert2, pgon1, pgon2, status

ELBOW r1, alpha, r2

ELLIPS h, r

END / EXIT [v1, v2, ..., vn]

ENDGROUP

EXOR （あるいは @）排他的論理和 優先順位 8

EXP (x) e （e = 2.7182818）の x 乗を返します。

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

F

FILE_DEPENDENCE "name1" [, "name2", ...]

FOR variable_name = initial_value TO end_value [STEP step_value]

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

FRA (x) x の小数部（x が整数の場合は整数 0、それ以外の場合は実数）を返します。（例，FRA(1.23) = 0.23, FRA(-1.23) = 0.77）.

FRAGMENT2 fragment_index,
use_current_attributes_flag

FRAGMENT2 ALL,use_current_attributes_flag

G

GET (n)

GOTO label

GOSUB label

GROUP "name"

H

HIDEPARAMETER name1 [, name2, ..., namen]

HOTARC2 x, y, r, startangle, endangle

HOTLINE2 x1, y1, x2, y2

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

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

```
HPRISM_ top_mat, bottom_mat, side_mat,
      hill_mat,
      n, thickness, angle, hill_height, status,
      x1, y1, s1,
      ...,
      xn, yn, sn
```

I

IF condition **GOSUB** label

IF condition **GOTO** label

IF condition **THEN** label

IND (FILL, name_string)

IND (LINE_TYPE, name_string)

IND (MATERIAL, name_string)

IND (STYLE, name_string)

IND (TEXTURE, name_string)

INPUT (channel, recordID, fieldID, variable1 [, variable2,...])

INT (x) x の整数部（常に整数）を返します。（例， INT(1.23) = 1, INT(-1.23) = -2）.

ISECTGROUP (g_expr1, g_expr2)

ISECTLINES (g_expr1, g_expr2)

K

KILLGROUP g_expr

L

[LET] varnam = n

LGT (x) x の常用対数を返します。

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

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

LINE_PROPERTY expr

LINE2 x1, y1, x2, y2

LOCK name1 [, name2, ..., namen]

LOG (x) x の自然対数を返します。

M

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

MAX (x1,x2, ... xn): 引数の中で最も大きい値を返します。

```
MESH a, b, m, n, mask,
      z11, z12, ... z1m,
```

z21, z22, ... z2m,
...
zn1, zn2, ... znm

MIN (x1,x2, ... xn): 引数の中で最も小さい値を返します。

MODEL SOLID

MODEL SURFACE

MODEL WIRE

MUL mx, my, mz

MUL2 x, y

MULX mx

MULY my

MULZ mz

N

NEXT variable_name

NOT (x) x が真 (<>0) の時は偽 (= 整数 0)、偽 (=0) (論理否定) の時は真 (= 整数 1) を返します。

NSP

NTR ()

O

OPEN (filter, filename, parameter_string)

OR (あるいは |) 論理和 優先順位 7

OUTPUT (ch, recordID, fieldID, var1, var2...)

OUTPUT channel, recordID, fieldID, expression1 [, expression2, ...]

P

PARAGRAPH name alignment, firstline_indent,
left_indent, right_indent, line_spacing [,
tab_size1, ...]
[PEN index]
[[SET] STYLE style1]
[[SET] MATERIAL index]
'string1'
'string2'
...
'string n'
[PEN index]
[[SET] STYLE style2]
[[SET] MATERIAL index]
'string1'
'string2'
...
'string n'
...

PARAMETERS name1 = expression1 [,
name2 = expression2, ...,
namen = expressionn]

PEN n

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

PI 円周率を返します。(p = 3.1415926...).

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

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

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

POLY_ n, x1, y1, s1, ... xn, yn, sn 61

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

POSITION position_keyword

PRINT expression [, expression, ...]

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

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

PROJECT2 projection_code, angle, method

PROJECT2{2} projection_code, angle,method [,backgroundColor, fillOrigoX,
fillOrigoY, filldirection]

PROJECT2{3} projection_code, angle, method , parts[, backgroundColor, fillOrigoX, fillOrigoY,
filldirection][[,]
PARAMETERS name1=value1 , ... namen=valuen]

PUT expression [, expression, ...]

PYRAMID n, h, mask, x1, y1, s1, ... xn, yn, sn

R

RADIUS radius_min, radius_max

RECT a, b

RECT2 x1, y1, x2, y2 114

REF COMPONENT code [, keycode [, numeric_expression]]

REF DESCRIPTOR code [, keycode]

REPEAT
[statement1
statement2
...
statementn]

REQ (parameter_string)

REQ (parameter_string)

REQUEST ("**ANCESTRY_INFO**", expr, name [,
guid, parent_name1, parent_guid1,
...,
parent_namen, parent_guidn)

REQUEST ("**Angular_dimension**", "", format_string)

REQUEST ("**Angular_length_dimension**", "" format_string)

REQUEST ("**Area_dimension**", "" format_string)

REQUEST ("**ASSOCEL_PROPERTIES** ", parameter_string, nr_data, data)

REQUEST ("**ASSOCLP_PARVALUE**", expr, name_or_index, type, flags, dim1, dim2, values)

REQUEST ("**Calculation_angle_unit**", "", format_string)

REQUEST ("**Calculation_area_unit**", "", format_string)

REQUEST ("**Calculation_length_unit**", "", format_string)

REQUEST ("**Calculation_volume_unit**", "", format_string)

REQUEST ("**Constr_Fills_display**", "", optionVal)

REQUEST ("**Elevation_dimension**", "", format_string)

REQUEST (**FONTNAMES_LIST**Åh,ÅhÅh, fontnames)

REQUEST ("**Height_of_style**", name, height [, descent, leading])

REQUEST ("**Level_dimension**", "", format_string)

REQUEST ("**matching_properties**", type, name1, name2, ...)

REQUEST ("**Name_of_program**", "", program_name)

REQUEST ("**Radial_dimension**", "", format_string)

REQUEST ("**REFERENCE_LEVEL_DATA**", "", name1, elev1, name2, elev2, name3, elev3)

REQUEST ("**Sill_height_dimension**", "", format_string)

REQUEST ("**Style_info**", name, fontname [, size, anchor, face_or_slant])

REQUEST ("**Window_door_dimension**", "", format_string)

REQUEST ("**WINDOW_DOOR_SHOW_DIM**", "", show)

REQUEST ("**door_show_dim**", "", show)

REQUEST ("**window_show_dim**", "", show)

REQUEST ("**window_door_zone_relev**", " ", out_direction)

REQUEST ("**Working_angle_unit**", "", format_string)

REQUEST ("**Working_length_unit**", "", format_string)

REQUEST ("**Zone_relations**", "", category_name, code, name, number [,category_name2, code2, name2, number2])

REQUEST ("**Zone_relations_of_owner**", "", category_name, code, name, number [, category_name2, code2, name2, number2])

REQUEST (**TEXTBLOCK_INFO**, textblock_name, width, height)

REQUEST (extension_name, parameter_string, variable1, variable2, ...)

REQUEST (question_name, name | index, variable1 [, variable2,...])

REQUEST (question_name, name | index, variable1 [, variable2,...])

REQUEST{2} ("**Material_info**", name_or_index,
extra_param_name,
value_or_values)

REQUEST{2} ("**Material_info**", name_or_index, param_name, value_or_values)

REQUEST ("HomeDB_info", "", homeDBIntId, homeDBUserId, homeDBName, homeContext)

RESOL n

RETURN

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

RICHTEXT x, y, height, 0, textblock_name

RICHTEXT2 x, y, textblock_name

RND (x) 0.0 から x の間のランダムな値を返します。(x > 0.0) は常に実数。

ROT x, y, z, alpha

ROT2 alpha

ROTX alphax

ROTY alphay

ROTZ alphaz

ROUND_INT (x)

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

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

S

[SET] FILL index

[SET] FILL name_string

[SET] LINE_TYPE index

[SET] LINE_TYPE name_string

[SET] MATERIAL index

[SET] MATERIAL name_string

[SET] STYLE index

[SET] STYLE name_string

SECT_ATTRS fill, fill_background_pen, fill_pen, contour_pen [, line_type]

SECT_FILL fill, fill_background_pen, fill_pen, contour_pen

SGN (x) x が正の時は整数 +1 を返し、負の時は整数 -1 を返します。それ以外の時は整数 0 を返します。

SHADOW keyword_1[, keyword_2]

SIN (x) x のサインを返します。

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

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

SPHERE r

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

SPLINE2A n, status, x1, y1, angle1, length_previous1, length_next1, ...

xn, yn, anglen, length_previousn,
length_nextn 119

SPLIT (string, format, variable1 [, variable2, ..., variablen])

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

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

SQR (x) x の平方根（常に実数）を返します。

STR (numeric_expression, length, fractions)

STR (format_string, numeric_expression)

STR{2}(format_string, numeric_expression [, exta_accuracy_string])

STRLEN (string_expression)

STRSTR (string_expression1, string_expression2)

STRSUB (string_expression, start_position, characters_number)

STW (string_expression)

SUBGROUP (g_expr1, g_expr2)

SURFACE3D ( )

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

SWEEPGROUP (g_expr, x, y, z)

T

TAN (x) x のタンジェントを返します。

TEVE x, y, z, u, v

TEXT d, 0, expression

TEXT2 x, y, expression

TEXTBLOCK name width, anchor, angle, width_factor, charspace_factor, fixed_height, 'string_expr1' [,
'string_expr2', ...]

TOLER d

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

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

U

UI_BUTTON type, text, x, y, width, height [, id [, url]]

UI_CURRENT_PAGE index

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{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} 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,
...,
expression_imagen, textn, value_definitionn]

UI_OUTFIELD expression,x,y,width,height [, flags]]

UI_PAGE page_number

UI_PICT expression, x, y [,width, height][, mask]]

UI_SEPARATOR x1, y1, x2, y2

UI_STYLE fontsize, face_code

UI_TOOLTIP

UI_BUTTON type, text, 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_OUTFIELD expression, x, y, width, height [, flags] [**UI_TOOLTIP** tooltiptext]

UI_PICT expression, x, y [,width, height [, mask]] [**UI_TOOLTIP** tooltiptext]

USE (n)

V

VALUES "fillparam_name" [[,] **FILLTYPES_MASK** fill_types,] value_definition1
[, value_definition2, ...]

VARDIM1(expr)

VARDIM2(expr)

VARTYPE (expression)

VECT x, y, z

VERT x, y, z

VOLUME3D ( )

W

WALLARC2x, 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,
      ...
      xn, yn, maskn

WHILE condition DO
      [statement1
      statement2
      ...
      statementn]
```

X

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

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

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

パラメータ命名規則

サブタイプ階層のため、子ライブラリ部品は自動的に親の全てのパラメータを継承します （サブタイプとパラメータの詳細については、ArchiCAD の『ユーザーガイド』を参照してください)。パラメータはその名前で識別されるので、継承されたパラメータとオリジナルのパラメータは同じ名前を持つことができます。省略されたライブラリ部品名を頭文字として持つ記述的なパラメータ名を使用することで矛盾を避けるのはライブラリ作成者の責任です。

ハンドラパラメータとユーザー定義パラメータのために、GRAPHISOFT 社はライブラリにおけるパラメータ命名規則を導入しました。

注記： ハンドラは、ライブラリ部品に付加的な機能を追加します（例えば、壁およびドアは壁に穴を切り取ります)。

頭文字が **ac_** のパラメータ名は、ArchiCAD ハンドラと対応付けられている特殊パラメータとして予約されています（例えば、**ac_corner_window**)。完全なリストについては、標準の ArchiCAD ライブラリサブタイプテンプレートを確認してください。

GRAPHISOFT 社の標準パラメータ名は、頭文字 **gs_** でマークされています（例えば、**gs_frame_pen**)。参考として、AC ライブラリ部品を確認してみてください。GRAPHISOFT 社のライブラリと完全な互換性があるようにするには、GDL スクリプトでこれらのパラメータを使用してください。

FM_ は、ArchiFM に予約され（例えば、**FM_Type**)、**HVAC_** は、ArchiCAD パラメータ用の HVAC に割り当てられています（例えば、**HVAC_Manufacturer**)。

GDL DATA I/O アドオン

「GDL Data In/Out」アドオンは、GDL コマンドを使用して簡易タイプのデータベースにアクセスできるようにします。
それ以外の点では、このアドオンは「Text GDL In/Out」アドオンと同じです。

データベースの説明

データベースは、レコードが別々の行に格納されたテキストファイルです。データベースは、1 つのキーに基づいてクエリしたり修正することができます。キーとその他の項目は、1 つの文字（OPEN コマンドで指定する）で区切られます。

行の長さは、同じでなくてもかまわず、レコードの列の数が違っていてもかまいません。

データベースを書き込み用に開く場合は、データベースファイルにファイル全体を複写するのに十分なスペースが必要です。

データベースを開いたり閉じたりするのは時間がかかるため、連続してデータベースを開いたり閉じたりしないでください。

大きなデータベース（数十万個以上のレコードをもつデータベース）は、キー値別に並べます。

データベースは、このアドオンで、OPEN、INPUT、OUTPUT、CLOSE の GDL コマンドを使用して、開く、クエリする、修正する、および閉じることができます。

データベースを開く

```
channel = OPEN (filter, filename, paramstring)
```

filter: アドオンの内部名で、ここでは"DATA"。
filename: 開くデータベースファイルの名前
paramstring: 区切り文字パラメータやファイルオープン命令モードパラメータなどのアドオン固有のパラメータ
データベースを開きます。データベースファイルを修正のために開こうとしてそのファイルが存在しないと、新規のファイルが作成されます。データベースファイルを読み取りのために開こうとしてそのファイルが存在しないと、エラーメッセージが表示されます。
戻り値は正の整数で、これで特定のファイルが識別されます。この値が、そのファイルの以降の参照番号となります。
データベースがオープン命令前に開かれていると、チャネル番号のみが生成されます。

paramstring には、次の値を含めることができます。

SEPARATOR: 単引用符（''）の間のキーワードの後には、(読み取り用と書き込み用の両方の）テキストファイルで列の区切り文字として使用する文字を定義することができます。特殊なケースとしては、タブ区切り文字（'¥¥t'）があります。

MODE: このキーワードの後には、オープン命令モードを続けます。
オープン命令モードは、以下の 3 通りのみです。

- RO（読み取り専用）
- WA（読み取り、変更）
- WO（読み取り、変更)。データベースがある場合、これを空にします。

DIALOG: パラメータ「filename」はファイル識別子として扱われ、その他の場合はフルパス名です。ファイル識別子は単純な文字列です。標準の［開く］→［名前を付けて保存］ダイアログ中に、このアドオンによって既存のファイルに対応付けられます。これらの対応は、アドオンによって格納され、ファイルが使用不可の場合を除き、再度問い合わせが行われることはありません。オープン命令モードが読み取り専用の場合、[開く］ダイアログが表示されるので、既存のドキュメントを選択することができます。その他の場合は、警告ダイアログが表示されるので、[作成］と［参照］のどちらかのオプションを選択することができます。

- Browse: 既存のデータファイルを検索します（[開く］ダイアログ）
- Create: 新規のデータファイルを作成します（[名前を付けて保存］ダイアログ)。
SEPARATOR、MODE、DIALOG の間には、常にコンマ（,）を入れてください。

LIBRARY: キーワード LIBRARY がパラメータ文字列内にある場合、データファイルはロードされたライブラリ内になければはなりません。

存在しないキーワードが使用された場合、与えられた区切り文字が間違っている場合、パラメータ文字列が空の場合には、この機能拡張はデフォルトの設定を使用します。

SEPARATOR = '¥¥t', MODE=RO.

例：

```
ch1 = OPEN ("DATA", "file1",
     "SEPARATOR=';', MODE = RO", DIALOG)
ch2 = OPEN ("DATA", "file2", "")
ch3 = OPEN ("DATA", "newfile",
     "SEPARATOR = '¥t', MODE = WA")
```

データベースから値を読み取る

```
INPUT (channel, recordID, fieldID, var1
     [, var2, ...])
```

recordID: キー値（数値または文字列）

fieldID: 与えられた
レコード内の列番号（最小番号：1
は、キー
値の後の項目を参照します）

var1,...: 読み取りレコード項目を受け取る変数

キー値に基づいてデータベースに対してクエリを行います。

レコードが見つかると、このレコードの与えられた列から項目の読み取りを開始し、読み取った値を順番にパラメータに入れます。

パラメータリストには、少なくとも1個の値が存在している必要があります。この値は、それぞれに定義されたパラメータタイプにかかわりなく、数値または文字列のタイプとすることができます。戻り値は、正常に読み取られた値の数です。

パラメータが値よりも多い場合、対応する値のないパラメータはゼロに設定されます。空白の列（つまり、区切り文字と区切り文字の間に何もない場合）では、パラメータはゼロに設定されます。

レコードが見つからない場合は、(-1) を返します。

例：

```
nr = INPUT (ch1, "key1", 1, v1, v2, v3)
! 最初の列の 3 つの値の入力
! キー "key1" の入ったレコードの
! （キー後の）
PRINT nr, v1, v2, v3
```

データベースに値を書き込む

OUTPUT channel, recordID, fieldID, expr1 [, expr2, ...]
recordID: キー値（数値または文字列）
fieldID: flag: レコードを削除する場合は 0（または <= 0）を指定、作成または修正する場合は 1（または > 0）を指定。
expr1,...: 見つけられたレコードまたは新規のレコードの新規項目値。削除の場合、この値は無視されます。
レコードの作成または修正の場合は、与えられたキー値に属するレコードを設定します。このレコードは、コマンドに記述されたのと同じ順番で与えられた値を含むことになります。値は、数値タイプまたは文字列タイプを取ることができます。少なくとも 1 個の式が必要です。
削除の場合は、与えられたキーに属するレコードがデータベースから削除されます。式の値は無視されますが、少なくとも 1 つは指定する必要があります。
例：

```
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 ! deletes the record
OUTPUT ch2, "keyB", 1, a, string
```

データベースを閉じる

CLOSE channel
channel: チャネル値
チャネル値によって識別されたデータベースを閉じます。

GDL DateTime アドオン

DateTime 機能拡張は、コンピュータ上で設定されている現在の日時に対してさまざまなフォーマットを設定できるようにします。

このアドオンは、GDL のファイル操作と同じように機能します。チャネルを開き、情報を読み取り、チャネルを閉じる必要があります。

このアドオンは、REQUEST GDL コマンドを使用することでも使用可能です。この場合、OPEN、INPUT、CLOSE のコマンドがこの順番で内部的にコールされます。これが、1 行の GDL コマンドだけで日時情報を獲得する最も簡単な方法です。

REQUEST ("DateTime", format, datetimestring)

要求関数の 2 番目のパラメータは、OPEN 関数のパラメータ paramstring で説明した内容と同じです。

チャネルを開く

channel = OPEN (filter, filename, paramstring)

filter: アドオンの内部名で、ここでは "DateTime"。

filename: 使用されません (システムの日時を取得するためにファイルを開く必要はありません)。

paramstring: 日時の希望の出力形式などのアドオン固有のパラメータ。

戻り値は正の整数で、開かれたチャネルを識別します。この値が、そのファイルの以降の参照番号となります。paramstring には、規則子やその他の文字を入力できます。

規則子は、次のように日時の値に置き換えられます。

%a	省略形の曜日名
%A	省略なしの曜日名
%b	省略形の月名
%B	省略なしの月名
%c	01:35:56 PM Wednesday, March 27, 1996 という形式での日時
%d	10 進数による 1 か月の日付け（01 ～ 31）
%H	時間（24 時間時計）、10 進数で表示（00 ～ 23）
%I	時間（12 時間時計）、10 進数で表示（01 ～ 12）
%j	年間日付け、10 進数で表示（001 ～ 366）
%m	月、10 進数で表示（01 ～ 12）
%M	分、10 進数で表示（00 ～ 59）
%P	12 時間時計の AM/PM 表示
%S	秒、10 進数で表示（00 ～ 61）
%U	1 年の週番号（日曜日を最初の週の最初の日とする）、10 進数で表記
%w	曜日、10 進数で表示（0 （日曜日）～ 6 （土曜日））
%W	1 年の週番号（月曜日を最初の週の最初の日とする）、10 進数で表記（00 ～ 53）
%x	Wednesday, March 27, 1996 形式での日付
%X	01:35:56 PM 形式での時刻
%y	世紀を省略した年数、10 進数で表示（00 ～ 99）
%Y	世紀を表示した年数、10 進数で表示
%Z	GDL はこの規則子は無視します。標準とおりでは、時間帯を決定できる場合は時間帯を印刷します。
%%	% 文字

例：

```
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
```

情報を読み取る

```
n = INPUT (channel, "", "", datetimestr)
```

channel: チャネル値

datetimestr: 文字列タイプ値

OPEN シーケンスで与えられた形式で、日付または時刻、またはこの両方を表す文字列タイプ値を読み取ります。第 2 および第 3 パラメータは未使用です（空の文字列または 0 とすることもできます）。
戻り値は、正常に読み取られた値の数で、この場合は 1 となります。

チャネルを閉じる

```
CLOSE channel
```

チャネル値によって識別されたファイルを閉じます。

GDL File Manager I/O アドオン

「GDL File Manager In-Out」アドオンは、GDL スクリプトから、フォルダに含まれているファイルやサブフォルダをスキャンできるようにします。

OPEN コマンドを使用して、スキャンしたいフォルダを指定します。

INPUT コマンドを使用して、指定したフォルダで、最初のファイル、次のファイル、フォルダの名前を取得します。

CLOSE コマンドを使用して、フォルダのスキャンを終了します。

フォルダを指定する

```
channel = OPEN (filter, filename, paramstring)
```

channel ： フォルダ ID

filter: アドオンの内部名で、ここでは「FileMan」。

filename ： - スキャン対象のフォルダの名前（OS とは無関係のパス）- フォルダ ID 文字列（DIALOG モード - 後述の説明を参照）

paramstring ： アドオン固有のパラメータ

paramString のパラメータはカンマ (,) で区切る必要があります。

1. パラメータ : FILES/FOLDERS

検索対象。
2. パラメータ（オプション）: DIALOG
ファイルパスの代わりにファイル ID 文字列で指定されたフォルダを示します。
この場合、最初（および対応するファイルパスが無効なたび）に、ファイルパスと対応する、保存される ID 文字列を設定するダイアログボックスが表示されます。
例えば、次のようにします。

```
folder = OPEN( "FileMan", "c:¥¥¥
```

上記は、（パソコン）の C ドライブのルートディレクトリをファイルのスキャンのために開きます。

ファイル / フォルダ名を取得する

```
n = INPUT (channel, recordID, fieldID, var1 [,
      var2, ...])
```

channel: フォルダ ID（OPEN コマンドにより返されます）
recordID : 0 （追加の開発のために予約されています）
fieldID : 0 （追加の開発のために予約されています）
var1, ... : ファイル / フォルダの名前を受け取る変数
n: 正常に埋められた変数の数
例えば、次のようにします。

```
n = INPUT (folder, 0, 0, fileName)
```

上記は、指定されたフォルダから次のファイル名を取り出し、1 を返します。
これ以上ファイルまたはサブフォルダがない場合、変数 n はゼロに設定されます。

フォルダのスキャンを終了する

```
CLOSE (channel)
```

チャネル値によって識別されたフォルダを閉じます。

例：1 つのフォルダをリストする

次のコードセグメントは（例えば、オブジェクトの 2D スクリプトセクションと同じように）、MyFavouriteFolder 識別子によって指定されたフォルダ内のファイルをリストします。初めて使用した時は、この識別子に既存のフォルダを割り当てる必要があります。その後、MyFavouriteFoler ID がそのフォルダを表すようになります。

```
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 )
```

GDL TEXT I/O アドオン

GDL Text In/Out アドオンは、外部テキストファイルを読み取り / 書き込み用に開き、GDL スクリプトとの値のやりとりによりファイルを操作できるようにします。

このアドオンは、GDL スクリプトからの OPEN、INPUT、および OUTPUT コマンドのパラメータリスト上の文字列を解釈します。

このアドオンは、[ArchiCAD] 以外にも、[ArchiCAD Data Folder] という名前のフォルダがユーザー定義ファイル用として存在すると想定します （このフォルダの名前は、アドオンのリソースフォークで定義されるので、ローカライズすることができます)。その名前のフォルダが存在しない場合は、アドオンはこれを作成します。フォルダにはサブフォルダを含めることができます。その場合、機能拡張はサブフォルダで既存のファイルを検索します。TEXT タイプのファイルに対して読み取り / 書き込みが実行できます。

ファイルを開く

channel = OPEN (filter, filename, paramstring)

filter: アドオンの内部名で、ここでは「TEXT」

filename: 開くデータベースファイルの名前

paramstring: 区切り文字パラメータやファイルオープン命令モードパラメータなどのアドオン固有のパラメータ

ファイルを開きます。書き込みを行うファイルが存在しない場合、そのファイルを作成します。読み取るファイルが存在しない場合、エラーメッセージが表示されます。

戻り値は正の整数で、これで特定のファイルが識別されます。この値が、そのファイルの以降の参照番号となります。

paramstring には、以下の情報を入力できます。

SEPARATOR: 単引用符 (’’) の間のキーワードの後に、(読み取り用と書き取り用の両方の) テキストファイルで列の区切り文字として使う文字を割り当てることができます。

特殊なケースとしては、タブ (’¥t’) と改行 (’¥n’) 文字があります。

MODE: このキーワードの後には、オープン命令モードを続けます。オープンには次の 3 つのモードしかありません。

RO （読み取り専用）

WA （書き込み専用、ファイルの最後に追加）

WO （書き込み専用、上書き）その前にファイルに保存されていたデータは失われます！

ファイルを同時に読み取り専用と書き込み専用として開くことはできません。

DIALOG: このキーワードを指定すると、ダイアログボックスが表示されてファイル名を入力することができます。

FULLPATH: このキーワードを指定すると、ファイル名はフルパス名として解釈されます。

LIBRARY: このキーワードを指定する場合は、データファイルはロードされたライブラリになければなりません。

キーワードとキーワードの間には、常にコンマ（,）を入れてください。
存在しないキーワードが指定された場合、与えられた区切り文字が間違っている場合、パラメータ文字列が空の場合には、機能拡張によってデフォルト設定が使用されます。SEPARATOR = ‘\t’、MODE = RO。

例：

```
ch1 = OPEN ("TEXT", "file1", "SEPARATOR = ';', MODE = RO")
ch2 = OPEN ("TEXT", "file2", "")
ch3 = OPEN ("TEXT", "file3", "SEPARATOR = '¥n', MODE = WO")
```

値を読み取る

```
INPUT (channel, recordID, fieldID,
     var1 [, var2, ...])
```

channel: チャネル値
recordID: 行番号（数値または文字列）
fieldID: 指定の行の列番号
var1,...: 読み取り　　レコード項目を受け取る変数
チャネル値によって識別されたファイルの与えられた開始位置から、与えられたパラメータの数と同じ数の値を読み取ります。パラメータリストには、少なくとも 1 個の値が存在している必要があります。読み取られた値は、関数によって順番にパラメータに入れられます。値は、値に対して定義されているパラメータタイプとは関係なく、数値タイプまたは文字列タイプであってもかまいません。
戻り値は正常に読み取られた値の数で、ファイルの最後まで達した場合は、(-1) となります。
行と列の番号は、両方とも正の整数である必要があります。そうでない場合、エラーメッセージが表示されます。
行または列の番号が不適切な場合には、入力は実行されません。 (n = 0)
行と列が識別された場合は、与えられた開始位置から、与えられたパラメータの数だけ値が入力されます。値の数よりパラメータの数の方が多い場合は、対応する値のないパラメータはゼロに設定されます。
空の列（つまり、区切り文字と区切り文字の間に何もない場合）では、パラメータはゼロに設定されます。

例：

```
nr = INPUT (ch1, 1, 1, v1, v2, v3) ! 最初の行の最初の列から ! 3 つの値を入力
PRINT nr, v1, v2, v3
```

値を書き込む

OUTPUT channel, recordID, fieldID, expr1 [, expr2, ...]

channel: チャネル値

recordID: 正の場合、出力値の後に新規の行が続きます。

fieldID: 役割はなし。この値は使われません。

expr1: 出力値

チャネル値によって識別されたファイルに、与えられた位置から、定義された式の数と同じ数の値を出力します。少なくとも 1 個の式が必要です。出力値のタイプは、式のタイプと同じです。
テキスト機能拡張の場合、OUTPUT は、与えられた式を連続する位置に配置します。このとき、式と式の間にはファイルを開くときに定義された区切り文字を使用し、オープン命令モードに従って、上書きするかファイルの最後に追加します。この場合、与えられた位置は解釈されません。
recordID は、出力で新規の行を指定するために使います。
recordID が正の場合、出力値の後に新規の行が続きます。その他の場合、最後の値の後に区切り文字が続きます。
例:

string = "Date: 19.01.1996"

a = 1.5

OUTPUT ch2, 1, 0, string ! string に引き続き新規の行

OUTPUT ch2, 0, 0, a, a + 1, a + 2! a + 2 の後に、新規の行ではなく ! 区切り文字

ファイルを閉じる

CLOSE channel
channel: チャネル値
チャネル値によって識別されたデータベースを閉じます。
例：
「f1」ファイルの内容を単純に「f2」と「f3」の両方のファイルにコピーし、「f1」でタブで区切りされている全ての値を「f2」と「f3」の両方のファイルの別々の行に書き込む GDL オブジェクト。

```
ch1 = open ("TEXT", "f1", "mode = ro")
ch2 = open ("TEXT", "f2", "separator = '¥n', mode = wo")
ch3 = open ("TEXT", "f3", "separator = '¥n', mode = wo")
i = 1
1:
n = input (ch1, i, 1, var1, var2, var3, var4)
if n <> -1 then
output ch2, 1, 0, var1, var2, var3, var4
output ch3, 1, 0, var1, var2, var3, var4
i = i + 1
goto 1
else
goto 2
endif
2:
close ch1
close ch2
close ch3
end
```

PROPERTY GDL アドオン

このアドオンの目的は、ArchiCAD 特性データベースを GDL スクリプトからアクセスできるようにすることです。SQL を使用した場合と同じように、データベーステーブルを開き、その内容をクエリできます。単一のレコードおよび複数のレコード

(リスト) をクエリできます。データベースを修正したり、データベースにレコードを追加することはできないという点に注意してください。
特性データベースの詳細については、ヘルプメニューの「ArchiCAD 14 計算ガイド」を参照してください。

OPEN

OPEN ("PROP", "database set name", ["database files"])
Parameters: <database set name> は、以降の OPEN コールでデータベースファイルセットを識別する任意の名前です。
<database files> は、特性データベースの一部であるテキストファイルのリストです。このパラメータは、既に <database set name> を読み取るファイルに割り当てている場合には、省略できます。<key file>、<component file>、<descriptor file>、<unit file> の順番に固定されています。フルパスを指定する必要はありません。ArchiCAD が現在のライブラリでこれらのファイルを検索します。長いファイル名を指定する場合は、引用符 (または ") で囲ってください。
戻り値 : チャネル番号
与えられたデータベースファイルへの通信チャネルを開きます。データベースファイルの内容は、すばやくアクセスできるようにメモリに読み込まれます。データベースが開いている限りは、特性データベースの修正は、このアドオンからは行えません。これは、通常問題にはならないはずです。
例 :
1.

```
channel = OPEN ("PROP", "sample", "AC 8_KEY.txt, AC 8_COMP.txt, AC 8_DESC.txt, AC 8_UNIT.txt")
```

これは、上記のファイル (これらは、ArchiCAD 7.0 特性データベースのファイルです) からなるデータベースを開き、「sample」という名前を付けます。3 番目のパラメータでは、別の引用符文字を使用する必要があることに注意してください (" と ' を使用できます)。
2.

```
channel = OPEN ("PROP", "sample", "")
```

このコマンドは、明示的にデータベースファイルを開いた (例 1 のように) 後、かつこのファイルを閉じる前に発行できます。このコマンドは、Master_GDL スクリプト内の 1 か所で明示的なコマンドを使用し、以後もっと短いバージョンを使用することができるようにします。

CLOSE

Syntax: CLOSE (channel_number)
戻り値 : なし
この前に開いた通信チャネルを閉じます。

INPUT

INPUT (channel_number, "query type", "field list", variable1[,])

パラメータ ：

<channel number> は、1 つ前の［開く］コマンドで与えられた、有効な通信チャネル番号です。

<query type> は、実行するクエリを指定します。このアドオンは、次のキーワードを理解します。

シングルレコード クエリ ：

KEY, <keycode> - キーデータベースでレコードをクエリします。ここでの <keycode> はキーコード属性の値です。有効フィールド：KEYCODE, KEYNAME

UNIT, <unitcode> - 単位データベースでレコードをクエリします。ここでの <unitcode> は単位コード属性の値です。有効フィールド：UNITCODE, UNITNAME, UNITFORMATSTR

COMP, <keycode>, <code> - 単位データベースでレコードをクエリします。ここでの <keycode> はキーコード属性の値、<code> は構成要素コード属性の値です。KEYCODE、KEYNAME、CODE、NAME、QUANTITY、QUANTITYSTR、UNITCODE、UNITNAME、UNITFORMATSTR

DESC, <keycode>, <code> - 単位データベースでレコードをクエリします。ここでの <keycode> はキーコード属性の値、<code> は記述項目コード属性の値です。KEYCODE、KEYNAME、CODE、NAME、NUMOFLINES、FULLNAME

リスト クエリ ：

KEYLIST - キーデータベース内の全てのレコードをリストします。有効フィールド：KEYCODE, KEYNAME

UNITLIST - 単位データベース内の全てのレコードをリストします。有効フィールド：UNITCODE, UNITNAME, UNITFORMATSTR

COMPLIST[, <keycode>] - 構成要素データベース内の全てのレコードをリストします。ただし、<keycode> が与えられている場合は、キーコードが <keycode> と等しいレコードのみをリストします。KEYCODE、KEYNAME、CODE、NAME、QUANTITY、QUANTITYSTR、UNITCODE、UNITNAME、UNITFORMATSTR

DESCLIST[, keycode] - 記述項目データベース内の全てのレコードをリストします。ただし、<keycode> が与えられている場合は、キーコードが <keycode> と等しいレコードのみをリストします。KEYCODE、KEYNAME、CODE、NAME、NUMOFLINES、FULLNAME

COMPDESCLIST[, <keycode>] - 構成要素データベースおよび記述項目データベース内の全てのレコードをリストします。ただし、<keycode> が与えられている場合は、キーコードが <keycode> と等しいレコードのみをリストします。ISCOMP、KEYCODE、KEYNAME、CODE、NAME、QUANTITY、QUANTITYSTR、UNITCODE、UNITNAME、UNITFORMATSTR、NUMOFLINES、FULLNAME

このクエリを使用する時は、十分注意してください。データベースでどちらのフィールドも有効でない場合（例えば、構成要素データベースの FULLNAME）、結果のリストではこの部分は単純に無視されますので、自分で注意する必要があります。

<field list> は、出力したい値を持つデータベース属性をリストします。出力がリストの場合、そのリストはここでリストされた順番でフィールドが並べられて格納されます。

次のフィールドを使用することができます。

KEYCODE - キーコード属性 タイプ： 文字列 使用できるクエリ： KEY、COMP、DESC、KEYLIST、COMPLIST、DESCLIST、COMPDESCLIST

KEYNAME - キー名属性 タイプ： 文字列 使用できるクエリ： KEY、COMP、DESC、KEYLIST、COMPLIST、DESCLIST、COMPDESCLIST

UNITCODE - 単位コード属性 タイプ： 文字列 使用できるクエリ： UNIT、COMP、UNITLIST、COMPLIST、COMPDESCLIST

UNITNAME - 単位名属性 タイプ： 文字列 使用できるクエリ： UNIT、COMP、UNITLIST、COMPLIST、COMPDESCLIST

UNITFORMATSTR - 単位の GDL 形式文字列 タイプ： 文字列 使用できるクエリ： UNIT、COMP、UNITLIST、COMPLIST、COMPDESCLIST

CODE - 構成要素または記述項目コード属性（クエリによる）タイプ： 文字列 使用できるクエリ： COMP、DESC、COMPLIST、DESCLIST、COMPDESCLIST

NAME - 構成要素の名前または記述項目レコードの最初の行 タイプ： 文字列 使用できるクエリ： COMP、DESC、COMPLIST、DESCLIST、COMPDESCLIST

QUANTITY - 数としての構成要素の数量（計算用）タイプ： 数字 使用できるクエリ： COMP、COMPLIST、COMPDESCLIST

QUANTITYSTR - 文字列形式での構成要素の数量 タイプ： 文字列 使用できるクエリ： COMP、COMPLIST、COMPDESCLIST

NUMOFLINES - 記述項目レコード内の行数 タイプ： 数字 使用できるクエリ： DESC、DESCLIST

FULLNAME - 記述項目レコード全体 タイプ： 文字列（1 つ以上）使用できるクエリ： DESC、DESCLIST

ISCOMP - 次のレコードが構成要素であるか記述項目であるかを示します。タイプ： 数値（構成要素の場合は 1、記述項目の場合は 0）使用できるクエリ： COMPDESCLIST

<variables> は、クエリの完了時にその結果を保持します。必要な数が正確にわかっている場合は、複数の変数をリストできます（例えば、シングルクエリで）。また、動的配列を指定することもできます。レコードは、順次リストされます。

例：

1.

```
INPUT (channel, "KEY, 001", "KEYNAME", keyname)
```

これは単純なクエリです。コードが'001'であるキーの名前は、keyname 変数に入れられます。

2.

```
INPUT (channel, "DESC, 004, 10", "NUMOFLINES, FULLNAME", desc_txt)
```

キーコードが 004 でコードが 10 である記述項目レコードが処理され、記述項目テキストの行数とテキスト自体が desc_txt 配列に入れられます。結果は、次のとおりです。

```
desc_txt[1] = <numoflines> (数値)
desc_txt[2] = <first row of description> (文字列)
…
desc_txt[<numoflines+1>] = <last row of description
```

3.
INPUT (channel, "COMPLIST", "NAME, KEYNAME, QUANTITY", comp_list)
構成要素リストを作成し、名前フィールドでソートを行い、次にキー名で、最後に数量フィールドでソートを行い、それをcomp_list 配列に入れます。結果は、次のとおりです。
complist[1] = <name1> (文字列)
complist[1] = <name1> (文字列)
complist[3] = <quantity1> (数値)
complist[4] = <name2> (文字列)
... etc.
4.
INPUT (channel, "COMPDESCLIST, 005", "ISCOMP, KEYNAME, NAME, QUANTITY", x_list)
共通の構成要素と記述項目のリストを作成します。つまり、両方のテーブルの<keycode>が 005 であるレコードがリストされます。出力は、次のとおりです。
x_list[1] = 0 (数値、0 >、これは記述項目です)
x_list[2] = <name1> (文字列 > 記述項目には<keyname>フィールドがないので無視されます)
x_list[3] = 0 (数値、記述項目には数量フィールドがありません)
…
x_list[(n*2)-1] = 1 (数値 > n-1 個の記述項目がリストされていて、次が構成要素になります)
x_list[n*2] = <keyname_n> (文字列)
… etc.

OUTPUT

このコマンドは、特性データベースが読み取り専用であるために、このアドオンには実装されていません。

GDL XML 拡張機能

この機能拡張は、XML ファイルの読み取り、書き込み、編集を可能にします。この機能は、Document Object Model (DOM) インターフェイスのサブセットを実装します。XML は、HTML と同じように、データを階層システムに構造化するのにタグを使用するテキストファイルです。XML ドキュメントは、階層ツリー構造によってモデル化することができます。このツリーのノードにはドキュメントのデータが含まれています。この機能拡張では、次のノードタイプが既知です。

- *Element:* ドキュメント内の開始タグと終了タグとの間にあるもの。空白要素の場合は、空白要素タグとなる場合もあります。要素には名前があり、属性を持つ場合もあります。ただし、通常は内容がある必要はありません。つまり、要素タイプのノードは子ノードを持つことができます。属性は、属性リスト内に保持されます。このリストでは、属性は、それぞれ異なる名前とテキスト値を持っています。
- *Text:* 文字シーケンス 子ノードを持つことはできません。
- *Comment:* コメント区切り文字間のテキスト。<!-- コメント自体 --> 。コメントのテキストでは、'-' 文字の後に必ず '-' 以外の文字が続いていなければなりません。また、次のような例は不正となります。<!-- コメント ---> 。コメントタイプのノードは、子ノードを持つことはできません。

・*CDATASection:* CDATA セクション区切り文字間のテキスト。<![CDATA[テキスト自体]]>。CDATA セクションでは、XML ドキュメントで特殊な意味を持つ文字を拡張する必要はなく、拡張してはなりません。認識されるマークアップは、終了の ']]>' のみです。CData セクションのノードは子ノードを持つことはできません。
・*Entity-reference:* 定義済みの構成要素への参照。このようなノードは読み取り専用のサブツリーを持つことができます。このサブツリーは参照された構成要素の値を与えます。ドキュメントの解析中に選択されて構成要素参照がテキストノードに変換されるようにすることができます。

最上位レベルでは、要素タイプノード（ルート）は必ず 1 つでなければなりませんが、コメントタイプのノードは複数個あってもかまいません。DOM インターフェイスのドキュメントタイプのノードはこの機能拡張のインターフェイスを介しては使用不可です。
ツリーの各ノードに対し、名前と対応付けられた文字列があります。これらの意味はノードのタイプによって異なります。

	名前：	**値：**
要素：	タグの名前	"" (空白の文字列)
テキスト：	"#text"	ノードのテキストコンテンツ
コメント：	"#comment"	ノードのテキストコンテンツ
CDATA セクション：	"#cdata-section"	ノードのテキストコンテンツ
構成要素参照：	参照された構成要素の名前	"" (空白の文字列)

この機能拡張は、各ノードタイプに対して文字列キーワードを定義します。このキーワードは、特定の命令で機能拡張にパスすることができます。

要素：	ELEM
テキスト：	TXT
コメント：	CMT
CDATAセクション:	CDATA
構成要素参照：	EREF

OPEN、INPUT、または OUTPUT コマンドの正常終了コードまたはエラーコードは、INPUT コマンドの GetLastError 命令で検索することができます。

XML ドキュメントを開く

OPEN コマンドは、次のとおりです。

channel = **OPEN** (filter, filename, parameter_string)

filter: ファイル拡張子。これは、'XML' にしてください。

filename: 開く（または作成する）ファイルの名前とパス。ファイルがダイアログボックスを使用して開かれ、ファイルの位置がユーザーによって与えられる場合は、識別子名。

parameter_string: オープン命令モードを決定する文字フラグの順番。

- **'r'**: 読み取り専用モードで開きます。通常、INPUT コマンドのみが使用できます。
- **'e'**: 構成要素参照は、ツリーのテキストノードには変換されません。このフラグを指定しないと、ドキュメント構造に構成要素参照はなくなります。
- **'v'**: 読み取りおよび書き込み中に妥当性確認が行われます。ドキュメント内に DTD がある場合、ドキュメントの構造はこれと一致する必要があります。このフラグを指定しなくてもきちんした構造にできますが、無効なドキュメントがエラーメッセージなしで読み取られたり、書き込まれたりします。
- **'n'**: 新規のファイルを作成します。ファイルが存在する場合、開く操作は失敗します （OPEN の後では、CreateDocument 命令が最初に実行されなければなりません)。
- **'w'**: ファイルがある場合は、このファイルを空のドキュメントで上書きします。ファイルがない場合は、新規のファイルが作成されます （OPEN の後では、CreateDocument 命令が最初に実行されなければなりません)。
- **'d'**: ファイルは、ダイアログボックスを介してユーザーによって取得されます。以降の実行では、OPEN コマンドのパラメータ filename に与えられた識別子に対応付けられます （識別子が既にファイルに対応付けられている場合は、ユーザーに対してダイアログボックスは表示されません)。
- **'f'**: パラメータ filename はフルパスを含みます。
- **'l'**: ファイルは、ロードされたライブラリ部品内にあります。

channel: 以降の入出力コマンドにおける接続を識別するのに使用されます。

既存の XML ファイルを修正のために開く場合は、パラメータ文字列に「r」、「n」、「w」のフラグはいずれも設定してはなりません。「d」、「f」、「l」のフラグのいずれか 1 つを設定してください。これらのフラグのどれも設定しないと、filename はユーザーのドキュメントフォルダを基準としたパスとみなされます。

XML ドキュメントを読み取る

DOM はオブジェクト指向のモデルで、GDL のような BASIC に似た言語には直接適合させることはできません。階層ツリーのノードを表すためには、位置記述項目を定義します。ツリーのノード間を移動するには、まず機能拡張に新規の位置記述項目を要求する必要があります。新規の記述項目は、最初はルート要素を指しています。記述項目は、実際には 32 ビットの識別番号で、その値は GDL スクリプトでは意味がありません。記述項目が参照する位置は、ツリー内をあるノードから別のノードへと移動するにつれて変更できます。

INPUT コマンドは、次のとおりです。

INPUT (ch, recordID, fieldID, var1, var2...)

ch: OPEN コマンドによって返されたチャネル
recordID: 命令名とパラメータ
fieldID: 通常は位置記述項目 var1, var2,...: 戻りデータを受け取る変数のリスト（省略可能）
INPUT の命令：

1 **GetLastError:** 最後の操作の結果を検索します。
recordID: "GetLastError"
fieldID: 無視されます。
return values:
var1: エラーコード/OK
エラーまたはOKの説明テキスト
2 **NewPositionDesc:** 新規の位置記述項目の要求
recordID: "NewPositionDesc"
fieldID: 無視されます。
return value: var1: 新規の位置記述項目（最初はルートを参照）
3 **CopyPositionDesc:** 開始ノードが別の記述項目から取られた新規の位置記述項目の要求
recordID: "CopyPositionDesc"
fieldID: 既存の位置記述項目
return value: 新規の位置記述項目（最初は、fieldIDに与えられた記述項目が参照する場所を参照）
4 **ReturnPositionDesc:** 位置記述項目が必要なくなったとき
recordID: "ReturnPositionDesc"
fieldID: 位置記述項目
。
この命令は、NewPositionDescまたはCopyPositionDesc命令から受け取った位置記述項目を使わなくなったときにコールします。
5 **MoveToNode:** 記述項目の位置を変更します （また、新規のノードのデータを検索します)。
この命令は、ツリー階層を移動するのに使用することができます。
recordID: "MoveToNode searchmode nodename nodetype nodenumber"
fieldID: 位置記述項目
searchmode (or movemode): パラメータnodenameは、xmlドキュメント内の要素タイプまたは構成要素参照タイプのノードを決定するパスを含んでいなければなりません。
パスは、fieldIDに与えられたノードを基準とします。区切り記号は「.」文字です（さもなければ、要素の名前で受け付けられる文字です。このため、全ての場合に有効ではありません)。「..」パス内の文字列「..」は、親ノードへのステップを意味します。開始ノードは、要素タイプまたは構成要素参照タイプのノードでなくてもかまいませんが、その場合は、元に戻るためにはパスが「..」で始まる必要があります。 同一レベルに同一名の要素ノードが複数ある場合は、最初のものが選択されます。
以下の移動モードの場合、ほかのパラメータは指定できません。
ToParent: fieldIDに与えられたノードの親に移動します。
ToNextSibling: 同一レベルの次のノードに移動します。
ToPrevSibling: 同一レベルの1つ前のノードに移動します。

ToFirstChild: fieldID ノードの最初の子に移動します。

ToLastChild: fieldID ノードの最後の子に移動します。

以下は、検索モードです。ほかのパラメータを指定してもかまいません。指定しない場合、デフォルト値を持つことになります。

FromNextSibling: 検索は、同一レベルの次のノードから開始され、順方向に移動します。

FromPrevSibling: 検索は、fieldID の前のノードから開始され、同一レベルを逆方向に移動します。

FromFirstChild: 検索は、fieldID ノードの最初の子から開始され、順方向に行われます。

FromLastChild: 検索は、fieldID ノードの最後の子から開始され、逆方向に行われます。

nodename: 検索は、名前または値が nodename と一致するノードのみを対象として行われます。* および？は、ワイルドカード文字と解釈されます。要素タイプと構成要素参照タイプのノードの場合は名前が比較され、文字、コメント、CDATA セクションのタイプのノードの場合は値が比較されます。デフォルト値 ： *

nodetype: 検索は、タイプが nodetype によって許可されているノードのみを対象として行われます。* は、全てのタイプを意味します。その他の場合は、タイプキーワードを + 文字と組み合わせて nodetype を形成することができます (TXT+CDATA のようにスペースなしの 1 ワードとする必要があります)。デフォルト値は * です。

nodenumber: 一致するノードが複数ある場合、このパラメータが一致するノードのシーケンスの中で検索するノードの数を与えます。（1 から始まります）デフォルト値：1

戻り値：

var1: ノードの名前

var2: ノードの値

var3: ノードのタイプキーワード

例：

同一レベルを逆方向に、要素または構成要素参照であり、名前が K で始まる 2 番目のノードまで移動したい場合は、次のようにします。

```
INPUT (ch, "MoveToNode FromPrevSibling K* ELEM+EREF 2", posDesc, name, val, type)
```

6 GetNodeData: 与えられたノードのデータを検索します。

recordID: "GetNodeData"

fieldID: 位置記述項目

戻り値：

var1: ノードの名前

var2: ノードの値

var3: ノードのタイプキーワード

7 NumberofChildNodes: 与えられたノードの子ノードの数を与えます。
recordID: "NumberofChildNodes nodetype nodename"
fieldID: 位置記述項目
次の省略可能パラメータで、対象となる子ノードのセットの範囲を狭めることができます。
nodetype: MoveToNode 命令で定義されているノードタイプが許可されます。
nodename: MoveToNode 命令で定義されているノードの名前または値が許可されます。
戻り値：
var1: 子ノードの数

8 NumberofAttributes: 要素ノードの属性の数を返します。
recordID: "NumberofAttributes attrname"
fieldID: 位置記述項目（要素ノードを参照しなければなりません）
attname: 指定した場合、対象となる属性のセットを、名前（値は対象とはならない）が attrname と一致する属性のみに狭めることができます。attrname 内の * および？文字は、ワイルドカード文字とみなされます。
戻り値：
var1: 属性の数

9 GetAttribute: 要素ノードの属性のデータを返します。
recordID: "GetAttribute attrname attrnumber"
fieldID: 位置記述項目（要素ノードを参照しなければなりません）
省略可能なパラメータ：
attrname: 属性の名前を与えます。* および？は、ワイルドカード文字とみなされます。デフォルト値 ：*
attnumber: attrname と一致する属性が複数ある場合は、attrnumber が一致する属性のシーケンス内から属性を選択します （カウントは 1 から始まります)。デフォルト値：1
戻り値：
var1: 属性の値
var2: 属性の名前

10 Validate: ドキュメントの妥当性を確認します。
妥当性は、ドキュメント修正命令の作業中は確認されません。確認は、オープン命令モード文字列にフラグ v が設定されている場合には、ファイルがディスクに書き出される間に行われます。妥当性確認は、Validate 命令でいつでも強制的に行えますが、時間とメモリを非常に多く消費するので、修正を行うたびにこの確認を行うことはお勧めできません。
recordID: "Validate"
fieldID: 無視されます。

XML ドキュメントを修正する

OUTPUT (ch, recordID, fieldID, var1, var2...)

ch: OPEN コマンドによって返されたチャネル
recordID: 命令名とパラメータ
fieldID: 通常は位置記述項目
var1, var2,...: 追加入力データ

OUTPUT 命令：
OUTPUT 命令のほとんどは、読み取り専用モードで開かれているファイルに対しては無効です。

1 CreateDocument:

recordID: "CreateDocument"
fieldID: 無視されます。
var1: ドキュメントの名前 これは、ルート要素の tagname にもなります。

CreateDocument は、ファイルが新規ファイルモードまたは上書きモードで開かれている場合にのみ許可されます。これらのモードでは、この命令が最初に実行されないと、XML ドキュメントは作成されません。

2 NewElement: ドキュメントに新規の要素タイプノードを挿入します。

recordID: "NewElement insertpos"
fieldID: 新規のノードが挿入される場所を基準とする位置記述項目
var1: 新規の要素の名前（要素のタグ名）
insertpos は次のようにすることができます。
AsNextSibling: 新規の要素は、fieldID に与えられた位置の後に挿入されます。
AsPrevSibling: 新規の要素は、fieldID に与えられた位置の前に挿入されます。
AsFirstChild: 新規の要素は、fiedlID で与えられたノードの最初の子として挿入されます（要素ノードでなければなりません）。
AsLastChild: 新規の要素は、fiedlID で与えられたノードの最後の子として挿入されます（要素ノードでなければなりません）。

3 NewText: ドキュメントに新規の CDATA セクションノードを挿入します。

recordID: "NewText insertpos"
fieldID: 位置記述項目
var1: 挿入されるテキスト

「NewElement 命令」も参照してください。

4 NewComment: ドキュメントに新規のコメントノードを挿入します。

recordID: "NewComment insertpos"
fieldID: 位置記述項目

var1: 挿入されるコメントのテキスト

「NewElement 命令」も参照してください。

5 NewCDATASection: ドキュメントに新規の CDATA セクションノードを挿入します。

recordID: "NewCDATASection insertpos"

fieldID: 位置記述項目

var1: 挿入される CDATA セクションのテキスト

「NewElement 命令」も参照してください。

6 Copy: あるノードの下に、ドキュメントのサブツリーのコピーを作成します。

recordID: "Copy insertpos"

fieldID: サブツリーが挿入される場所を基準とした位置記述項目

var1: コピーされるサブツリーのノードを与える位置記述項目

insertpos: NewElement 命令と同じ

コピー元のサブツリーは変更されません。コピー元のサブツリーのあるノードを指す位置記述項目は、コピー後も同じノードを指します。

7 Move: ドキュメントのあるサブツリーをほかの場所に入れ換えます。

recordID: "Move insertpos"

fieldID: サブツリーが挿入される場所を基準とした位置記述項目

var1: 移動されるサブツリーのノードを与える位置記述項目

insertpos: NewElement 命令と同じ

オリジナルのサブツリーは削除されます。移動されたサブツリーのあるノードを指す位置記述項目は、サブツリーの新規の位置にある同じノードを指します。

8 Delete: ノードとそのサブツリーをドキュメントから削除します。

recordID: "Delete"

fieldID: 削除するノードを与える位置記述項目

削除されたサブツリーのあるノードを指す全ての位置記述項目は無効となります。

9 SetNodeValue: ノードの値を変更します。

recordID: "SetNodeValue"

fieldID: 位置記述項目。テキスト、コメント、または CDATA セクションのタイプのノードを参照しなければなりません。

var1: ノードの新規のテキスト値

10 SetAttribute: 要素ノードの属性を変更するか、新規のノードを作成します。

recordID: "SetAttribute"

fieldID: 位置記述項目（要素タイプのノードを参照しなければなりません）

var1: 属性の名前

var2: 属性のテキスト値

要素が既にこの名前の属性を持っている場合は、その値が変更されます。その他の場合は、新規の属性が要素の属性リストに追加されます。

11 RemoveAttribute: 要素ノードの属性を削除します。

recordID: "RemoveAttribute"

fieldID: 位置記述項目（要素タイプのノードを参照しなければなりません）

var1: 削除する属性の名前

12 Flush: 現在のドキュメントをファイルに書き戻します。

recordID: "Flush"

fieldID: 無視されます。

ファイルが妥当性確認モードで開かれている場合は、有効なドキュメントのみが保存されます。

13 ChangeFileName: 別のファイルを現在のドキュメントに対応付けます。

recordID: "ChangeFileName"

fieldID: 新規のファイルパス

var1: fieldID の解釈方法を指定します。var1 が空の文字列の場合、fieldID はユーザーのドキュメントフォルダを基準とするパスを含みます。「d」は、ファイルの位置がファイルダイアログボックスでユーザーが指定したことによって得られたことを意味します（コマンドオープンモードフラグ *369 ページ「OPEN」*を参照）。「l」は、ファイルがロードされているライブラリから取得されることを意味します。「f」は、fieldID がフルパスを含んでいることを意味します。

この命令は、ファイルが読み取り専用モードで開かれている場合でもコールできます。その場合、この命令の実行後にはドキュメントの読み取り専用属性は失われるので、修正して新規のファイル位置に保存することができます。

エラーコードとメッセージ：

0: ″OK″
-1: ″アドオンの初期化に失敗しました″
-2: ″メモリが不足しています″
-3: ″誤ったパラメータ文字列″
-4: ″ファイルダイアログエラー″
-5: ″ファイルが存在しません″
-6: ″XML 解析エラー″
-7: ″ファイル操作エラー″
-8: ″ファイルが既に存在します″
-9: ″このチャネルは開いていません″
-10: ″構文エラー″
-11: ″オープンエラー″
-12: ″無効な位置記述項目″
-13: ″この操作のノードタイプが無効です″
-14: ″そのようなノードは見つかりません″
-15: ″内部エラー″

-16: ″パラメータエラー″
-17: ″そのような属性は見つかりません″
-18: ″無効な XML ドキュメント″
-19: ″処理不能な例外″
-20: ″読み取り専用ドキュメント″
-21: ″CreateDocument は許可されていません″
-22: ″ドキュメントの作成に失敗しました″
-23: ″NodeValue の設定に失敗しました″
-24: ″移動は許可されていません″
-25: ″削除は許可されていません″
-26: ″SetAttribute は許可されていません″
-27: ″ファイル形式エラー″
-28: ″挿入（またはコピー）は許可されていま
-29: ″ノードの作成に失敗しました″
-30: ″誤った文字列″
-31: ″無効な名前″

ポリゴン操作拡張機能

ポリゴン操作拡張機能を使って、ポリゴンを操作することができます。

チャネルを開く

ch = INITADDONSCOPE (″PolyOperations ″, ″″, ″″)

チャネルを開きます。戻り値は、開かれたチャネルの ID です。

ポリゴンコンテナの管理

PREPAREFUNCTION ch, ″CreateContainer″, ″myContainer″, ″″

ポリゴンコンテナを新規に作成します。

PREPAREFUNCTION ch, ″DeleteContainer″, ″myContainer″, ″″

既存のポリゴンコンテナを削除します。

PREPAREFUNCTION ch, ″EmptyContainer″, ″myContainer″, ″″

既存のポリゴンコンテナを空にします。

PREPAREFUNCTION ch, ″SetSourceContainer″, ″mySourceContainer″, ″″

コンテナをソースコンテナとして設定します。

PREPAREFUNCTION ch, "SetDestinationContainer", "myDestinationContainer", ""

コンテナを出力先コンテナとして設定します。

ポリゴンの管理

PREPAREFUNCTION ch, "Store", "poly1", nVertices, nContours, vertArray, contourArray [, defaultInhEdgeInfo, inhEdgeInfosArray]

指定されたパラメータを持つポリゴン「poly1」を、現在のソースコンテナに保存します。

「poly1」: 保存されるポリゴンの名前

nVertices: 頂点の総数

nContours: 輪郭線の総数

vertArray: ポリゴンの全ての輪郭線を記述する nVertices 項目を全て含んだ配列。レコードの 2 次元配列（x、y、角度)。ここで、x、y、および角度は実数値です。曲線の辺の場合、角度パラメータはビュー角度（たわみ）となります。これは符号付きの値で、方向を示しています。値 0 は、直線の辺を示します。

contourArray: i 番目の輪郭線の最後の頂点のインデックスを含む配列。nContours 項目を全て含んでいる必要があります。

defaultInhEdgeInfo: 継承された辺情報のうちの 1 つ。この情報は、後から行った操作で（分割で作成されたのではなく）まったく新規に作成された辺に添付されます。辺情報を使用することにより、複雑な操作の後でも新規に作成した辺を簡単にトレースすることができます。（オプション）

inhEdgeInfosArray: 辺に添付された情報を含む配列。nVertices 整数タイプ項目を全て含んでいる必要があります。辺を複数の新規の辺に分割した場合、この情報は新規に作成された全ての辺にそのまま継承されます。例えば、屋根の側面の角度を保存する場合などに使用できます。（オプション）

PREPAREFUNCTION ch, "Dispose", "poly1", "myContainer"

ポリゴン「poly1」を、コンテナ「myContainer」から削除します。

ポリゴン操作の設定

PREPAREFUNCTION ch, "HalfPlaneParams", "", ca, cb, cc

「PolyCut」操作で使用する 2D に、半平面の記述を設定します。

半平面の不等式を定義します。ca * x + cb * y > cc

ca: x の係数

cb: y の係数

cc: 定数

PREPAREFUNCTION ch, "OffsetParams", "", itemIdx, offsetValue

「OffsetEdge」および「ResizeContour」操作で使用するオフセットパラメータを設定します。

itemIdx: 変換する辺のインデックス（「OffsetEdge」操作の場合）。サイズ変更可能な輪郭線のインデックス（「ResizeContour」操作の場合）。

offsetValue: 変換の距離。負と正のオフセット値によって、辺がそれぞれ内側と外側に移動します。オフセット値が大きい場合、隣接する頂点が切り取られることがあります。

ポリゴン操作

以下のポリゴン操作では、「poly1」および「poly2」ソースポリゴンが、ソースポリゴンコンテナ内にあります。

この操作で作成されたポリゴンは、「resPolygonID」で始まる一意の名前で、出力先ポリゴンコンテナに保存されます。ここでの「ID」は番号です。

dim resPolyIDArray[]

nPgon = CALLFUNCTION (ch, "poly1 OP poly2", "", resPolyIDArray)

「poly1」および「poly2」ポリゴンで「OP」操作を実行し、新規の値を指定されたパラメータに入力します。戻り値は、生成されたポリゴンの総数です。

「OP」では以下のことが行えます。

+ ポリゴンの加算

- ポリゴンの減算

/ ポリゴンの交差

resPolyIDArray: 生成されたポリゴン ID の配列

dim resPolyIDArray[]

nPgon = CALLFUNCTION (ch, "CopyPolygon", "poly1", resPolyIDArray)

ソースコンテナから出力先コンテナにポリゴンをコピーします。

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "Regularize", "poly1", resPolyIDArray)
```

ポリゴンを正規化します。ポリゴンを幾何学的に有効にします。
以下の場合に、ポリゴンは有効になります。
1 最初の境界に他の全てが含まれている
2 正しく方向付けられている（メイン輪郭線が正、その他が負）
3 自己交差していない
4 面積が 0 でない
5 長さが 0 の辺がない

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "PolyCut", "poly1", resPolyIDArray)
```

半平面を持つポリゴンと交差します。半平面は、「HalfPlaneParams」コマンドで設定されている必要があります。結果は正規化されます。

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "OffsetEdge", "poly1", resPolyIDArray)
```

ポリゴンの辺を、その方向に対して垂直に変換します。辺のインデックスと変換オフセットは、「OffsetParams」コマンドで設定されている必要があります。結果は正規化されます。

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "ResizeContour", "poly1", resPolyIDArray)
```

ポリゴンの輪郭線を拡大または縮小します。輪郭線のインデックスと変換オフセットは、「OffsetParams」コマンドで設定されている必要があります。結果は正規化されます。

結果として生成されたポリゴンを取得する

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "GetSourcePolygons", "", resPolyIDArray)
```

現時点のソースコンテナから全てのポリゴン名を取得します。

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "GetDestinationPolygons", "", resPolyIDArray)
```

現時点の出力先コンテナから全てのポリゴン名を取得します。

```
dim resVertices[]
nVertices = CALLFUNCTION (ch, "GetVertices", polygonID, resVertices)
```

ポリゴン操作をコールした後に、生成されたポリゴンの頂点を返します。生成されたポリゴンは「polygonID」という名前で出力先ポリゴンコンテナに入ります。

```
dim contArr[]
nContours = CALLFUNCTION (ch, "GetContourEnds", polygonID, contArr)
```

ポリゴン操作をコールした後に、生成されたポリゴンの輪郭線終端インデックスを返します。生成されたポリゴンは「polygonID」という名前で出力先ポリゴンコンテナに入ります。

```
dim inhEdgeInfosArr[]
nEdgeInfos = CALLFUNCTION (ch, "GetInhEdgeInfos", polygonID, inhEdgeInfosArr)
```

ポリゴン操作をコールした後に、生成されたポリゴンの辺情報を返します。生成されたポリゴンは「polygonID」という名前で出力先ポリゴンコンテナに入ります。

チャネルを閉じる

```
CLOSEADDONSCOPE (ch)
```

チャネル「ch」を閉じます。保存されている全てのポリゴンを削除します。

索引

数字

2D シンボル
　～での建具の生成 310
2D スクリプト 13
3D スクリプト 13
3D スクリプトの 2D シンボルへの投影 169
3D テキスト要素 122
3D での節点の定義
　テクスチャ原点のある～ 124
3D での平面形状 79
3D での節点の定義 124

A

ABS 249
ACS 250
ADD 31
ADD2 29
ADDGROUP 143, 149
ADDITIONAL_DATA 222
ADDX 30
ADDY 30
ADDZ 30
AND 248
ARC 82
ARC2 162
ArchiCAD 13
　～の構成要素リスト 227
　～の要素リスト 173
ArchiFM 13
ARMC 75
ARME 76
ASN 250
ATN 250

B

BackgroundColor 169
BASE 132
BINARY 13, 152
BINARYPROP 13, 227
BITSET 251
BITTEST 251
BLOCK 35
BODY 130
BPRISM_ 44
BREAKPOINT 264
BRICK 35
BWALL_ 59

C

CALL 268
CEIL 249
CIRCLE 81
CIRCLE2 162
CLOSE 271
COMPONENT 226, 227
CONE 38
COONS 109
COONS パッチの生成 109
COOR 127
COS 250
CPRISM_ 43
CROOF_ 68
CSLAB_ 55
CUSTOM 229
CUTFORM 142
CUTPLANE 132
CUTPLANE{2} 132
CUTPOLY 135
CUTPOLYA 139
CUTSHAPE 141
CWALL_ 55
CYLIND 36

D

DATABASE_SET 225
DEFINE FILL 223, 337
DEFINE FILL_A 223
DEFINE FILLA 213
DEFINE LINE_TYPE 204, 218, 223
DEFINE MATERIAL 204, 223
DEFINE STYLE 219
DEFINE SYMBOL_FILL 215, 223, 338
DEFINE SYMBOL_LINE 218, 223
DEL 33
DEL TOP 33
DESCRIPTOR 226
DIALOG 364
DIM 245
DO 260
DRAWINDEX 203
DRAWING 228
DRAWING2 173
DRAWING3 173
DRAWING3{2} 173

E

EDGE 126
ELBOW 78
ELLIPS 37
ELSE 262
END 25, 264
ENDGROUP 149
ENDIF 262
ENDWHILE 260
EXIT 25
EXOR 248
EXP 250
EXTRUDE 84

F

FILE_DEPENDENCE 223, 270, 339
FILL 203, 210, 269
FILLA 213
FOR 259
FPRISM_ 47
FRA 249
FRAGMENT2 13, 168
FULLPATH 364

G

GDL スクリプトで使用可能な文字 25
GDL スクリプトの引用符 26
GDL スクリプトの角括弧 28
GDL スクリプトの感嘆符 25
GDL スクリプトのコロン 25
GDL スクリプトのコンマ 25
GET 265
GOSUB 262, 263
GOTO 262, 263
GROUP 149

H

HIDEPARAMETER 231
HOTARC 180
HOTARC2 180
HOTLINE 180
HOTLINE2 180
HOTSPOT 79, 175
HOTSPOT2 155, 175
HPRISM_ 49

I

IF 262
IND 252, 303
INPUT 271
INT 249
ISECTGROUP 144, 149
ISECTLINES 144, 149

K

KILLGROUP 150

L

LET 195
LGT 250
LIGHT 116
LIN_ 79
LINE2 156
LINE_PROPERTY 159, 199, 341
LINE_TYPE 204, 218, 269
LOCK 231
LOG 251

M

MASS 113
MASTEREND_GDL 22
MASTER_GDL 22, 27, 204, 228
MATERIAL 200, 204, 269
MAX 251
MIN 251
MOD 248
MODE 364
MODEL 199, 269
MODEL SURFACE 147
MUL 31
MUL2 30
MULX 31
MULY 31
MULZ 31

N

NEXT 259
NOT 251
NSP 265
NTR 34

O

OPEN 270
OR 248
OUTPUT 271

P

PARAMETERS 230
PEN 269
PGON 126
PI 250
PICTURE 13, 121
PICTURE2 13, 166
PIPG 127
PLACEGROUP 150
PLANE 80
PLANE_ 81
POLY 80
POLY_ 80
POLY2 157
POLY2_ 158
POLY2_A 159
POLY2_B 159
POSITION 227
PRINT 270
PRISM 39
PRISM_ 40
PROJECT2 168
PROJECT2{2} 169
prompt（プロンプト） 28
PYRAMID 88

R

RADIUS 196, 269
RANGE 229
RECT 79
RECT2 156

REF 226
REQ 252, 297
REQUEST 252
RESOL 269
RETURN 263
REVOLVE 90
REVOLVE{2} 91
RICHTEXT 123, 347
RICHTEXT2 167, 347
RND 251
RO 364
ROT 32
ROT2 30
ROTX 32
ROTY 32
ROTZ 32
RULED 94
RULED{2} 95, 166

S

SECT_FILL 201
SEPARATOR 364
SET FILL 203
SET MATERIAL 200
200, 204
SETSTYLE 199
SGN 249
SHADOW 202, 269
SIN 250
SLAB_ 54
SOLID 199
SPHERE 36
SPLINE2 163
SPLINE2_A 165
SPLIT 255
SPRISM_ 50
SQR 249
STEP 229, 259
STR 252
STR{2} 252
STRLEN 257
STRSTR 257
STRSUB 258
STW 257
STYLE 199, 219, 269
SUBGROUP 143, 149
SURFACE 199
SURFACE3D 227
SWEEP 98
SWEEPGROUP 144, 152
SWEEPGROUP (2) 152

T

TAN 250
TEVE 124
TEXT 122
TEXT2 167
TEXTURE 208
THEN 262
TOLER 198, 269
TUBE 102
TUBEA 107

U

UI_BUTTON 232
UI_DIALOG 231
UI_GROUPBOX 233
UI_INFIELD 235
UI_OUTFIELD 235
UI_PAGE 231
UI_SEPARATOR 233, 234
UI_STYLE 234
UNTIL 261
USE 265

V

VALUES 229
VARDIM1 246
VARDIM2 246
variable（変数） 28
VARTYPE 271
VECT 124
VERT 124

W

WA 364
WALLARC2 324
WALLBLOCK2 323
WALLHOLE 313
WALLHOLE2 322
WALLLINE2 324
WALLNICHE 317
WHILE 260
WIRE 199
WO 364

X

XFORM 32
XWALL_ 62
XWALL_{2} 65

あ

値の割り当て 195
値リスト 23, 27

え

円弧の定義 162
円周率 250
円錐台の定義 38
円柱の定義 36
円柱要素の滑らかさの定義
　解像度による～ 197
　近似による～ 198
　半径による～ 196
円の定義 162

お

押し出し角柱の生成 84
オブジェクトの 3D 形状の体積 227
オブジェクトの 3D 形状の表面積 227

か

カーテンウォールのパネルのパラメータ 293
カーテンウォールパラメータ 291
カーテンウォールフレームパラメータ 292
回転した表面の定義 90
角錐の定義 88
角柱の定義
　丘のある～ 47
　押し出された一般的な～ 84
　拡張～ 43
　傾斜した～ 54
　傾斜した拡張～ 54
　平行でない上部ポリゴンがある～ 50
壁の定義
　拡張～ 62
画像ポリゴンの定義 127
画像要素の定義 166
壁の終端 277
壁の定義
　曲面～ 59
から 259
管の定義
　楕円から突き出ている～ 76
　他の管から突き出ている～ 75
　曲げた～ 78

き

記述項目 13
　～の定義 226
構成要素
　～の定義 226
基本構文要素 25
曲線壁の定義 59
曲面角柱の定義 44

く

矩形の建具
　直線壁の～ 311
矩形以外の建具
　直線壁の～ 313
矩形の定義 156
グローバル原点 21
グローバル変数 23, 27

こ

光源の定義 116
記述項目
　～参照 226
構成要素 13
　～参照 227
コメント 13

さ

サーフェス 145
サーフェスベース 145
材質の設定 200
材質の定義 205

　要素リストの～の定義 173
作図
　2D スクリプト内の～参照 228
座標系の回転 32
座標系の定義
　プリミティブのローカル～ 127
座標変換 29
　基本～ 15
　中級～ 18
サブルーチン 262, 263

し

識別子 26
自然対数 251
シャドウ投射の制御 202
自由なユーザーグローバル変数 293, 294
主座標系 21
上級コマンドと機能 19
上級スクリプト 21
条件 260
状態 83
常用対数 250
初級コマンド 15

す

数式 27
スクリプトタイプ 12
スクリプト定義の終了 264
スクリプトの行 25
スクリプトのブレークポイント定義 264
図形プリミティブ 21
ステートメント 25
ステートメントの書き出し 270
作図 173
スプラインの定義
　ベジェタイプ～ 165
　163
スラブ 54

せ

線種 218
　～の定義 216, 218
線の定義 156

そ

属性
　～の定義 23
ソリッド 145
ソリッド図形コマンド 143
ソリッドベース 145

た

球の定義 36
単純な形状 15

ち

中級コマンド 16
頂点 123

て

データベースの定義 225
テキストスタイル
　～の設定 199
線種
　～の設定 204
テキストの定義
　2D の～ 167
　3D での～ 122
テクスチャの定義 208

と

特殊文字 28
特性スクリプト 13

ぬ

塗りつぶしパターンの設定
　2D 表示の～ 203
塗りつぶしパターンの定義
　単純な～ 210

は

バイナリ 2D データ 13
バイナリ 3D 152
バイナリ 3D データ 13
バイナリデータ参照 227
バイナリ特性データ 13
ハイブリッド 145
パラメータ 13, 27
　～値のロック 231
　～バッファ 264
　GDL スクリプトの～ 20
　GDL 内の～値の修正 230
　単純タイプ 27
　派生タイプ 27
パラメータスクリプト 13
パラメータの配列 27
パラメータバッファへの値の格納 264
梁の定義 68
半楕円球体の定義 37

ひ

ビットマップパターン 212
描画順序の定義 203

ふ

ファイルからの値のインポート 271
ファイル操作 270
ファイルへの値のエクスポート 271
ファイルを閉じる 271
ファイルを開く 270
ブール差 145
複雑な変換マトリクス 32
プリミティブによるボディの定義 130
プリミティブ要素 123
プレビュー画像 13
フロー制御ステートメント 20
プログラミング言語 12
ブロックの定義 35

へ

平面ポリライン
　相対終点による線分 184
　中心点と半径による完全な円 189
　～の定義 124
ベクトル 123, 124
ベクトルハッチング 212
辺 123
ペンカラーの設定 198
変換
　～の削除 33
変数 26
辺の定義 126

ほ

ボディ 123
ポリゴン 123
　詳細な～ 159
ポリゴンの定義 126, 157, 159
ポリラインから生成される形状 82
ポリラインで生成された表面
　曲線パスを掃引した～ 98
　空間曲線パスに沿って掃引された～ 102, 107
　平面曲線および空間曲線からの～ 95
ポリラインの一般的な制限 83

ま

マクロオブジェクト 268
マクロコール 21
　～の定義 268
　建具の～ 311
マスキング 83
マスキングの規則 182
　角柱の～ 181
　メッシュの～ 72
マスタースクリプト 12

め

メッシュ 72
メッシュの生成 113
メッシュの定義
　等間隔の～ 72

も

文字列 26
文字列式
　～の相互の位置 257
　～の長さ 257
　～の幅 257
　～の分割 255
　サブ文字列 258
　数式からの～の作成 252
　フォーマット文字列 253
モデリングモードの設定 199

や

屋根の定義
　傾斜した～ 68

ゆ

ユーザーインターフェイススクリプト 13
ユーザーグローバル変数 27

よ

要求コール 252

ら

ライブラリ 364
ライブラリ部品 12
ラベル 25

り

リスト内の要素タイプの変更 227

る

ループ 259

ろ

ローカル座標系 22
ローカル座標系の移動 30
ローカル座標系のスケール変更 31
ローカル変数 26
ログ壁のパラメータ 62

わ

ワイヤフレーム 145
ワイヤフレームベース 145
ワイヤフレームモデリングモード 199

角柱の定義 39
梁 68