GDLリファレンスガイド

GRAPHISOFT®
A NEMETSCHEK COMPANY

GRAPHISOFT®

特約店および製品在庫情報に関しては、グラフィソフト社ウェブサイト https://www.graphisoft.com 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

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。

まず始めは、簡単な事から徐々にGDLを学んでいきましょう。 それぞれの機能をゆっくりと、その機能の全てを段階的に試しながら、GDLを学習してみてください。 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。

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。
- 『Creating GDL Objects』 e-ガイドは、オブジェクト作成方法の基本的な概要を提供しています。
- 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。
- 最新の学習教材は、初心者にも経験ユーザーにも適したAndrew Watson著の『GDL Handbook』です。
- 『GDL Advanced 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で登録後、ダウンロードすることができます。 https://www.graphisoft.com/support/developer/. 基本的な開発のガイドラインは、このマニュアルの「一般テクニカル標準」をご覧ください。

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。 ライブラリ部品には数量計算の説明も含まれています。

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ー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きます。

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。

上位移行スクリプトには、古い要素の配置されたインスタンスを変換できる変換ロジックを定義できます。

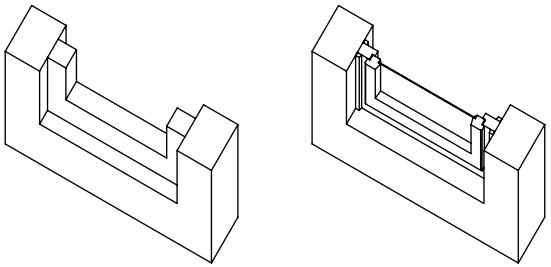
下位移行スクリプトには、要素の古いバージョンへの下位変換を定義できます。

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。

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。

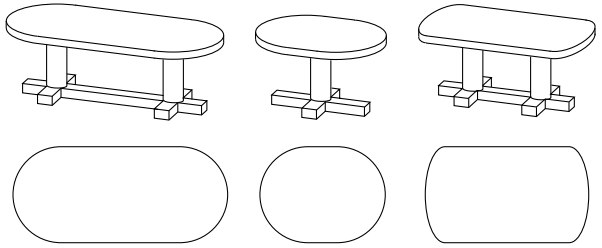


窓のさまざまな表現方法

学習プロセスの早い段階でめげないようにするには、シンプルな寸法が固定されたオブジェクトから始め、形状を作れようにしてください。 基本的なモデリングに慣れてきたら、少しずつレベルを上げて、理想の形状に近づけましょう。 理想としている形状は、必ず

しも複雑なわけではありません。 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：

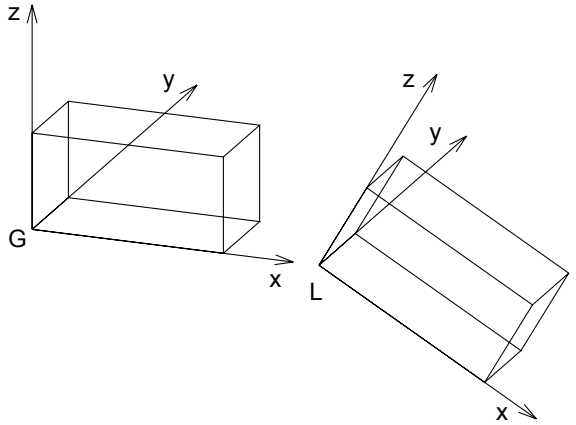
BLOCK, CYLIND, SPHERE, PRISM

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：

初級の座標変換の上位

MULX, MULY, MULZ, ADD, MUL, ROT

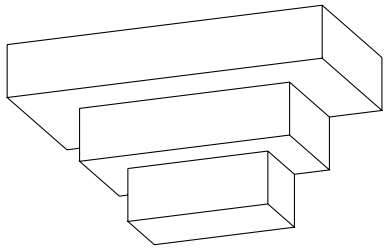
2D GDLコマンド：

初級の座標変換の上位

MUL2

例 ：

```
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, SPLINE2A

フロー制御と条件ステートメント

FOR - TO - NEXT

DO - WHILE, WHILE - ENDWHILE

REPEAT - UNTIL

IF - THEN - ELSE - ENDIF

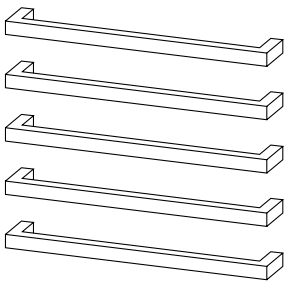
GOTO, GOSUB

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,
         -0.45, -0.1,  15,
         -0.45, 0,     15
   ADDZ 0.2
NEXT i
```



パラメータ

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。 この変換は既に生成されている形状にはまったく影響しません。このコマンドの後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。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、リストプロセスの開始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」が付くスクリプトは、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（「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 / EXIT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） ~ ! : , ; . + - * / ^ = < > <= >= # ( ) [ ] { } ¥ @ & |(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。

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を継続する文字列
\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ASCII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。
- - ASCII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、「グローバル変数」）。 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。 全ての要素は、スクリプト（コールオブジェクトまたはコールされたマクロ）でそれらの値が変更されない限り、Master GDLで設定された値を最初に読み込みます。 異なる解釈のインスタンス間のグローバルデータ交換はありません。 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、31文字を超えてはなりません。 また、パラメータの最大数は1024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2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。

数値型、文字列型、両方とも配列化することが可能です。 1次元もしくは2次元配列まで対応でき、要素にはインデックスを指定することで直接アクセスすることができます。

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。

構造タイプ

変数とパラメータは辞書にすることもできます。互換性：ARCHICAD 23で導入されました。

辞書はキーと値のペアの階層化された集合です。キーには他の辞書、配列、整数、文字列、または浮動小数点型の値を含めることができます。

キーはID（「識別子」）とみなされ、「˜」文字を使用できないことを除き、IDと同じ構文規則が適用されます。

以下の場所では辞書キーは使用できません（単純タイプの場合でも同様です）。

- FOR - TO - NEXTループ変数。
- HOTSPOT2またはHOTSPOT 編集または表示パラメータ。
- UI_... 入力パラメータ、ここで入力パラメータは数式として指定されます。(UI_INFIELD{2}, UI_INFIELD{3}, UI_CUSTOM_POPUP_INFIELD{2}, UI_RADIOBUTTON, UI_PICT_RADIOBUTTON, UI_PICT_PUSHCHECKBUTTON, UI_TEXTSTYLE_INFIELD, UI_LISTITEM{2}, UI_CUSTOM_POPUP_LISTITEM{2}, UI_COLORPICKER{2}, UI_SLIDER{2})
- UI_... 入力パラメータ、ここで入力パラメータは、文字列が辞書タイプのパラメータとして評価される場合、文字列として指定されます。
- VALUESまたはVALUES{2}パラメータ - 値リストは適用できません。
- REQUEST戻り値 - これをサポートする要求では、辞書のルートレベルのみが許可されます。
- APPLICATION_QUERY、SPLIT、INPUT、LIBRARYGLOBAL、またはCALLFUNCTION戻り値
- extra_accuracy_stringを返すSTR{2}
- CALLのRETURNED_PARAMETERS - 返された辞書は辞書のルートレベルにのみ保存できます。

本書で使用する表記規則

[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。

¦

排他またはコマンドのパラメータ間の関係

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では、全ての図形要素は右手系のローカル座標系を基準に生成されます。 GDLは右利きの座標システムを使用しています。例えば、ブロックの１頂点が原点に位置する場合、側面はx−y、x−z、y−z平面上にあります。

任意の位置に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。座標系のある軸方向への移動、またはある軸を中心とした回転、ストレッチなどを総称して変換と呼びます。 スクリプト内で宣言された変換はスタックされていきます。解釈は、常にスタックの最後から逆順に行われます。 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。 それらをスタックから削除することもできます。 スクリプトが終了した時にスタックは解放されます。

2D変換

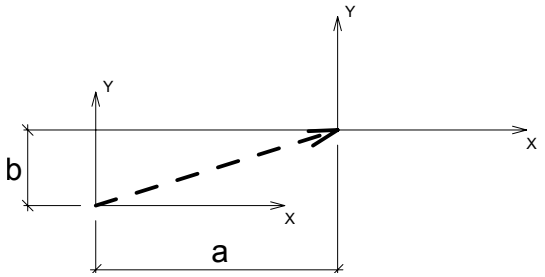
3D座標変換コマンドであるADD、 MUL、ROTZの2D版です。

ADD2

ADD2 x, y

例:

ADD2 a, b

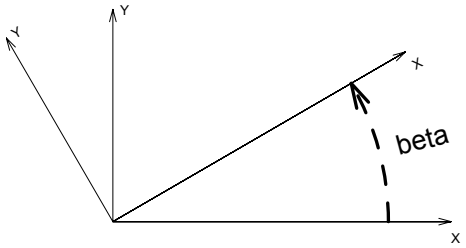


MUL2

MUL2 x, y

ROT2
ROT2 alpha

例:
ROT2 beta



3D変換

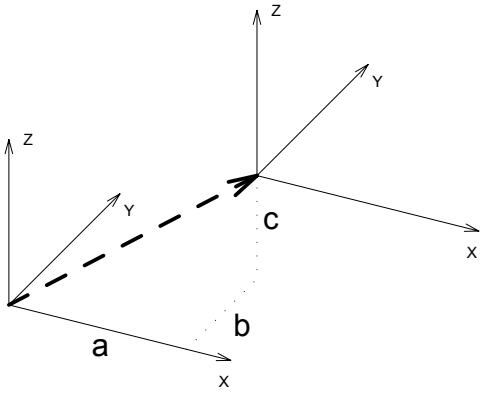
ADDX
ADDX dx

ADDY
ADDY dy

ADDZ
ADDZ dz
それぞれの軸方向（x、y、z）に指定した移動量（dx、dy、dz）だけ、ローカル座標系を移動する。

ADD
ADD dx, dy, dz
シーケンスADDX dx: ADDY dy: ADDZ dzを置換します。

例:
ADD a, b, c



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。

MULX
MULX mx

MULY
MULY my

MULZ
MULZ mz

x、y、z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

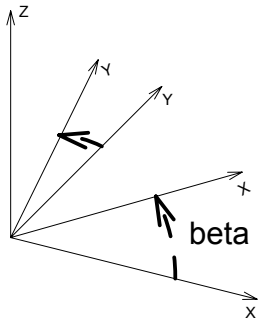
ROTY
ROTY alphay

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

完全な座標変換行列を定義します。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。

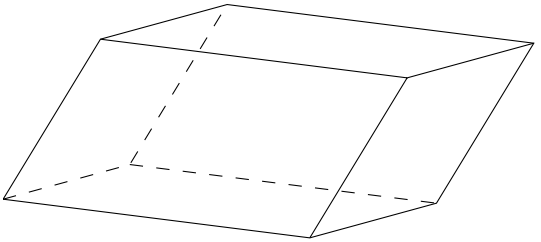
x' = newx_x * x + newy_x * y + newz_x * z + offset_x

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

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

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

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、最後のスタックからn個のエントリを削除し 、座標系はn個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

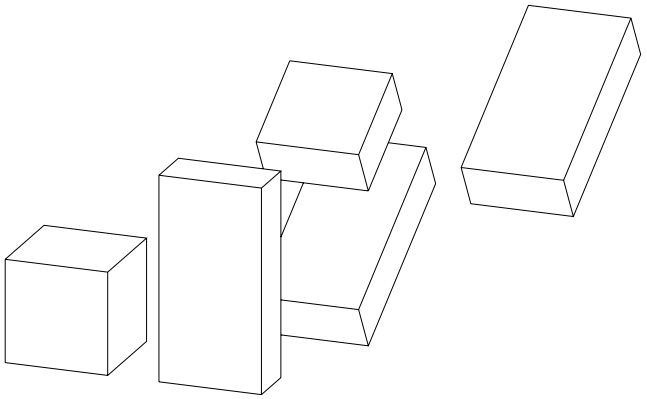
スタックされている変換を全て削除します。

NTR

NTR ()

スタックされている変換の数を返します。

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

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。

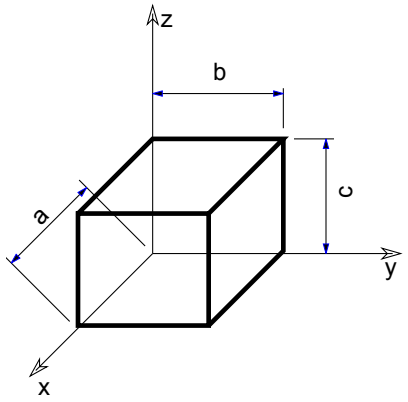
基本形状

BLOCK

BLOCK a, b, c

BRICK

BRICK a, b, c



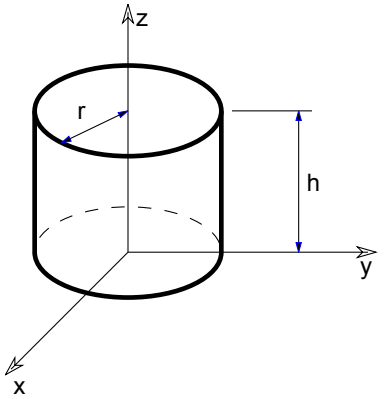
ローカル座標の原点からx、y、z軸方向にそれぞれ長さa、b、cの直方体を生成します。 ゼロの値がある場合は、矩形または直線を生成します。

パラメータの制限:

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

CYLIND

CYLIND h, r

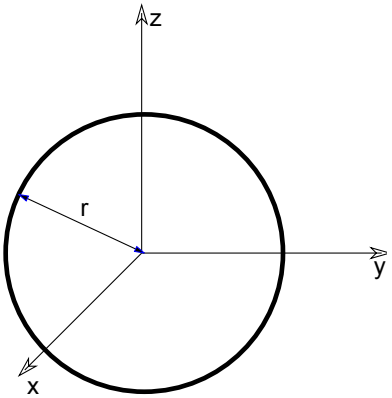


Z軸沿いに、高さがh、半径がrの直円柱。
h = 0の場合、X-Y平面上に円が生成されます。
r = 0の場合、Z軸沿いに直線が生成されます。

SPHERE

SPHERE r

ローカル座標の原点を中心とした、半径rの球。



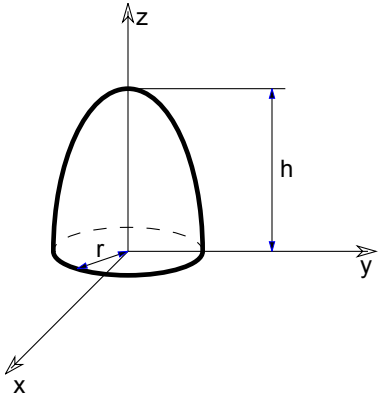
ELLIPS

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。

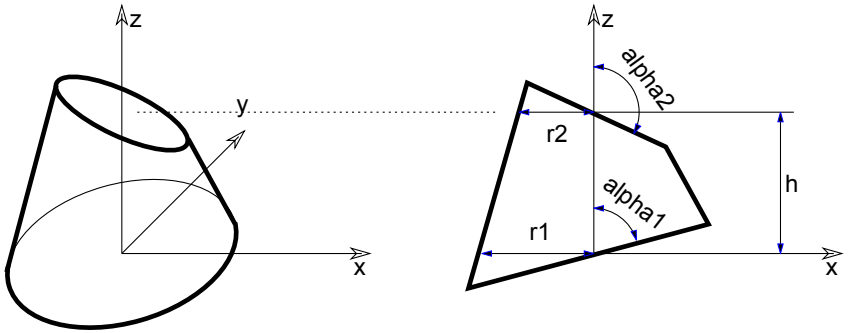
例: 半球

ELLIPS h, r



CONE

CONE h, r1, r2, alpha1, alpha2

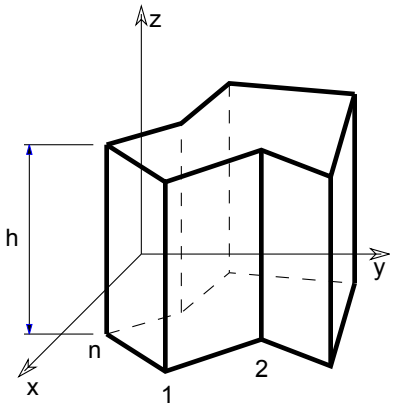


alpha1とalpha2が両端の表面のz軸に対する傾斜角、r1とr2が両端の円の半径、hがz軸沿いの高さである円錐台。
h = 0の場合、alpha1とalpha2の値は意味を持たず、X-Y平面に環が作成されます。
alpha1とalpha2は、度（° ）で指定します。
パラメータの制限:
0 < alpha1 < 180°および0 < alpha2 < 180°

例: 通常の円錐
CONE h, r, 0, 90, 90

PRISM

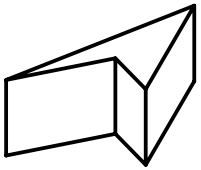
PRISM n, h, x1, y1, ..., xn, yn
基準面がX-Y平面上の角柱（「POLY」および「POLY_」のパラメータを参照）。 Z軸に沿った高さは、abs(h)。hは負の値をとることもできます。負の場合、X−Y平面より下に面が生成されます。
パラメータの制限:
n >= 3



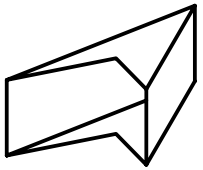
PRISM_

PRISM_ n, h, x1, y1, s1, ..., xn, yn, sn
「PRISM」コマンド 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、ステータスコードを参照してください。

例 1: ソリッドと空洞のある面



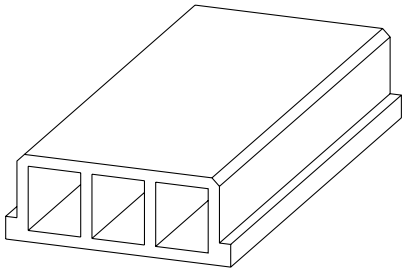
```
PRISM_ 4,1,
    0,0,15,
    1,1,15,
    2,0,15,
    1,3,15
```



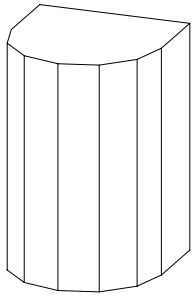
```
PRISM_ 4,1,
    0,0,7,
    1,1,5,
    2,0,15,
    1,3,15
```

例 2: ポリゴン内の穴

```
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
```

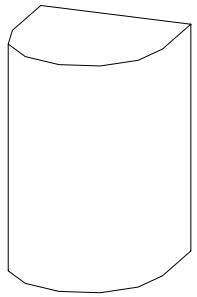


例 3: 曲面



j7 = 0

```
R=1
H=3
PRISM_ 9, H,
      -R,       R,       15,
      COS(180)*R, SIN(180)*R, 15,
      COS(210)*R, SIN(210)*R, 15,
      COS(240)*R, SIN(240)*R, 15,
      COS(270)*R, SIN(270)*R, 15,
      COS(300)*R, SIN(300)*R, 15,
      COS(330)*R, SIN(330)*R, 15,
      COS(360)*R, SIN(360)*R, 15,
      R,        R,       15
```



j7 = 1

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

CPRISM_

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

「PRISM_」コマンドの拡張版。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_」コマンドと同じです。

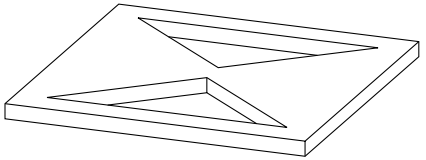
パラメータの制限:

n >= 3

「「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、ステータスコードを参照してください。

例: 名前、インデックス、およびグローバル変数によって定義済みの材質を参照する材質



```
CPRISM_ "Mtl-Iron", 0, SYMB_MAT,
        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_」コマンド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

CPRISM_{3}

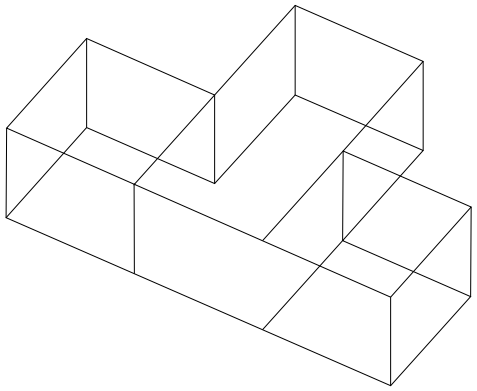
CPRISM_{3} top_material, bottom_material, side_material, mask,
n, h,
x1, y1, alpha1, s1, mat1,
...
xn, yn, alphan, sn, matn

CPRISM_{3}は「CPRISM_{2}」コマンドの拡張版です。生成された多角柱のグローバル表現をコントロールできます。

mask: 生成された多角柱のグローバル表現をコントロールします。
mask = j_1 + 2*j_2 + 4*j_3 + 8*j_4, ここで、 各 jiフラグは0または1をとります.
j_1: 陰線処理をした上部の辺
j_2: 陰線処理をした下部の辺
j_3: 陰線処理をした側面の辺
j_4: 断面形状の曲面の側面の辺と表面は滑らかです。互換性：ARCHICAD 21で導入されました。

例 1:

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



例 2:

```
PEN 1
mat = IND (MATERIAL, "Metal-Aluminium")
!visible side segment edges
mask = 1 + 2 + 4
_secondStat = 15
CPRISM_{3} mat, mat, mat, mask,
 6,  1,
 0,  0,   0, 15,   mat,
 1,  0,   0, _secondStat, mat,
 0.5, 0.5, 0, 900,   mat,
 1,   1,   0, 3015,   mat,
 0,   1,   0, 15,   mat,
 0,   0,   0, -1,   mat
!最初の頂点のステータスコピーを使用した滑らかな辺
mask = 1 + 2 + 4
_secondStat = 15 + 64
CPRISM_{3} mat, mat, mat, mask,
 6,  1,
 0,  0,   0, 15,   mat,
 1,  0,   0, _secondStat, mat,
 0.5, 0.5, 0, 900,   mat,
 1,   1,   0, 3015,   mat,
 0,   1,   0, 15,   mat,
 0,   0,   0, -1,   mat
!マスクを使用した滑らかな辺、最初の辺は滑らかでない
mask = 1 + 2 + 4 + 8
_secondStat = 15
CPRISM_{3} mat, mat, mat, mask,
 6,  1,
 0,  0,   0, 15,   mat,
 1,  0,   0, _secondStat, mat,
 0.5, 0.5, 0, 900,   mat,
 1,   1,   0, 3015,   mat,
 0,   1,   0, 15,   mat,
 0,   0,   0, -1,   mat
```

CPRISM_{4}

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

CPRISM_{4}は「CPRISM_{3}」コマンドの拡張版で、インライン材質定義できます。それはローカルのGDLスクリプトに材質を定義でき、グローバル材質定義で定義された材質と一緒に使用できます。

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指定した半径の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。

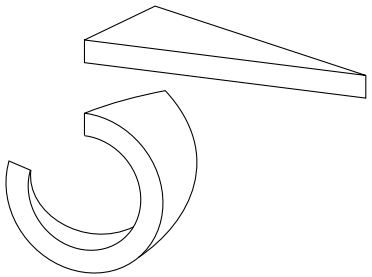
詳細は、「BWALL_」を参照してください。

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。

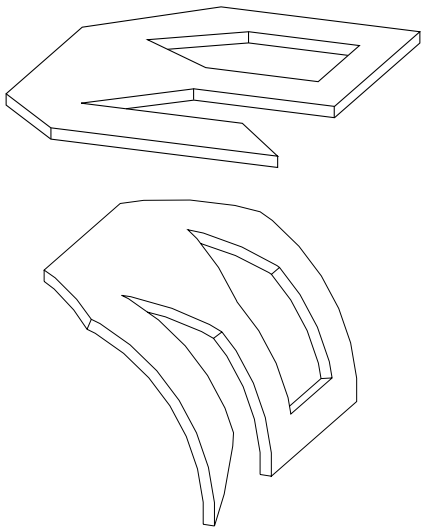
詳細は、ステータスコードを参照してください。

例: 対応する直線部分のある曲面角柱

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



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



FPRISM_

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

「PRISM_」コマンドと似ていますが、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: 丘の側面の材質

angle: 丘の側面の辺の取り付け角。

制限：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 （「RADIUS」,「RESOL」 および「TOLER」 を参照）。

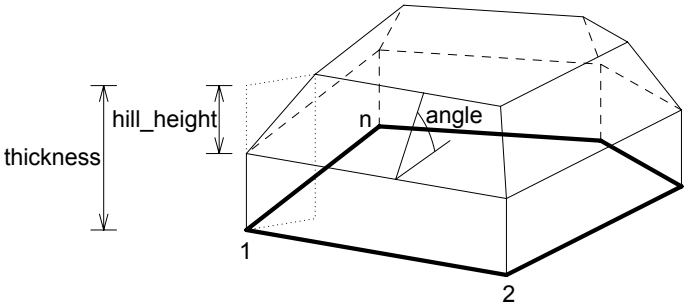
hill_height: 丘の高さ。パラメータthicknessはFPRISM_の全高を表します。

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。

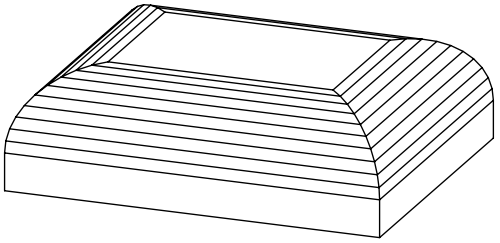
パラメータの制限:

n >= 3, hill_height < thickness

詳細は、ステータスコードを参照してください。

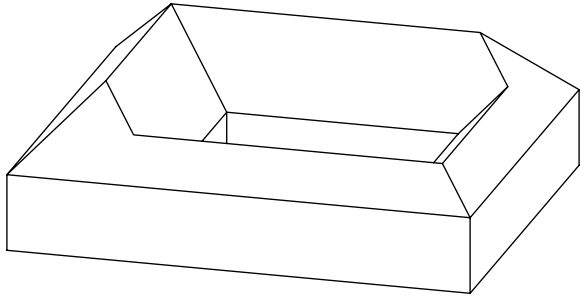


例 1: 曲面傾斜のある角柱



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

例 2: 直線傾斜のある角柱



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

HPRISM_

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

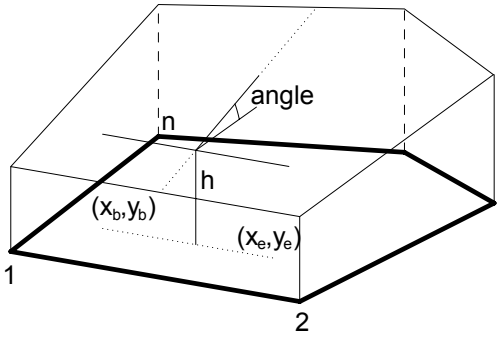
FPRISM_に似ていますが、丘の辺の可視性を制御するパラメータが追加されています。

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_」コマンドの拡張版。X-Y平面と平行でない上部ポリゴンを設定できます。 上部平面は、「CROOF_」 コマンド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。



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）。

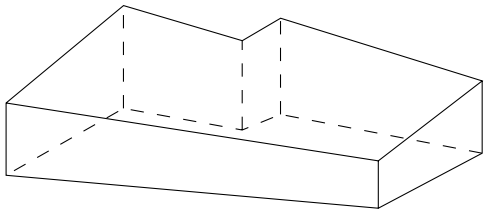
si: ポリゴンの辺と側面の可視性を制御できるようにするステータスコード。 特殊な制約を使用して、穴を定義したり、ポリラインに線分や円弧を作成することもできます。

詳細は、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_」コマンドの拡張版。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_」 コマンドの 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。

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下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: 下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）。

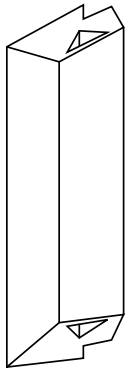
si: ポリゴンの辺と側面の可視性を制御できるようにするステータスコード。 特殊な制約を使用して、穴を定義したり、ポリラインに線分や円弧を作成することもできます。

詳細は、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')
```



SPRISM_{3}

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

CROOF_{2}は 「SPRISM_{2}」 コマンドの拡張版です。生成された多角柱のグローバル表現をコントロールできます。

mask: 生成された多角柱のグローバル表現をコントロールします。

mask = j_1 + 2*j_2 + 4*j_3 + 8*j_4, ここで、 各 jiフラグは0または1をとります.

j_1: 陰線処理をした上部の辺

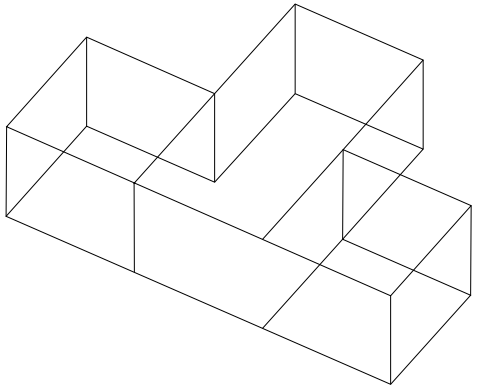
j_2: 陰線処理をした下部の辺

j_3: 陰線処理をした側面の辺

j_4: 断面形状の曲面の側面の辺と表面は滑らかです。互換性：ARCHICAD 21で導入されました。

例:

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

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



SPRISM_{4}

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

SPRISM_{4} は「SPRISM_{3}」コマンドの拡張版で、インライン材質定義できます。それはローカルのGDLスクリプトに材質を定義でき、グローバル材質定義で定義された材質と一緒に使用できます。

SLAB

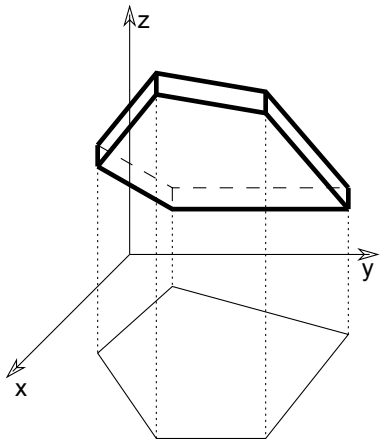
SLAB n, h, x1, y1, z1, ..., xn, yn, zn

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」コマンドに似ていますが、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_」コマンドにも似ています。

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、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_」コマンドの拡張版。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_」コマンドと同じです。

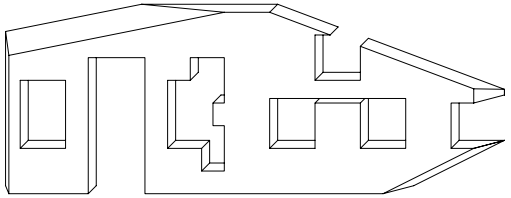
si: ポリゴンの辺と側面の可視性を制御できるようにするステータスコード。 特殊な制約を使用して、穴を定義したり、ポリラインに線分や円弧を作成することもできます。

詳細は、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）。

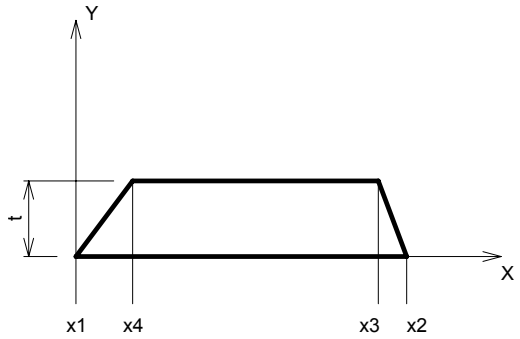


壁の基準線は、常にX軸と一致するように変換されます。壁の側面は、X-Z平面上になります。

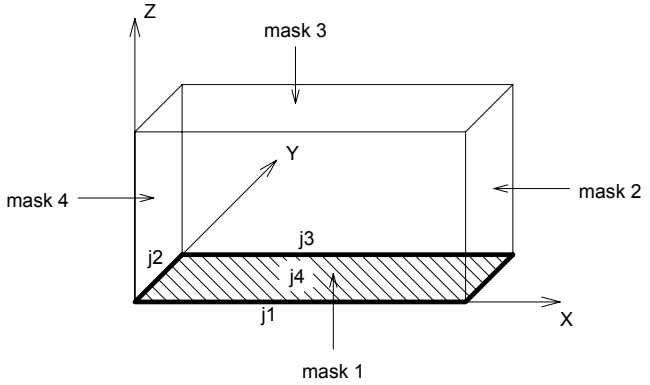
height: 壁の基準面を基準とした壁高さ

x1, x2, x3, x4: 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 = 0、x2 = 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

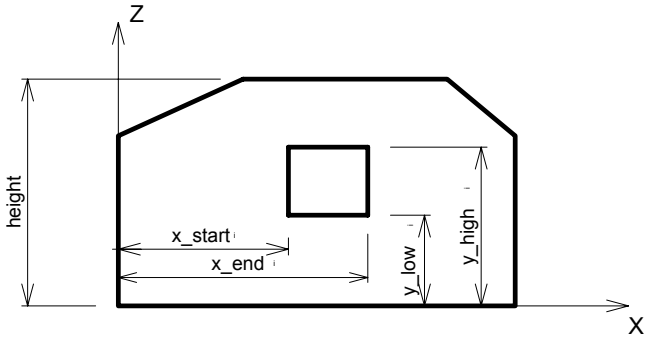


mask1, mask2, mask3, mask4: 辺および側面ポリゴンの可視性を制御します。
mask1, mask2, mask3, mask4 = $j_1 + 2*j_2 + 4*j_3 + 8*j_4$, ここで、 各 jiフラグは0または1をとります.
j1、j2、j3ビットは、側面のポリゴンの辺を表示する（1）か省略する（0）かを表します。 j4ビットは、切り取りから発生した側面のポリゴンの辺を表示する（1）か省略する（0）かを表します。



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。

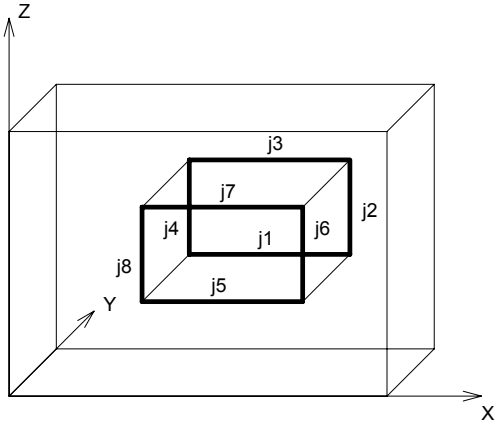


frame_showni:

1: 穴の辺を表示する場合

0: 穴の辺を表示しない場合

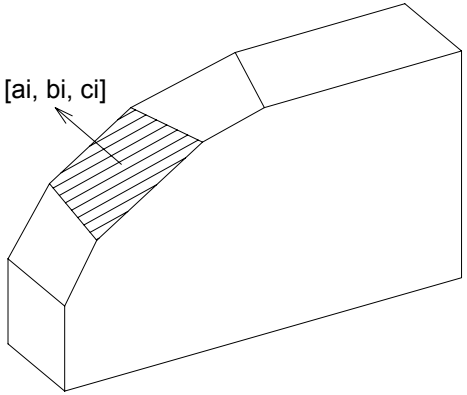
< 0: 開口部の辺の可視性を個別に制御します。frame_showni = -(1*j1 + 2*j2 + 4*j3 + 8*j4 + 16*j5 + 32*j6 + 64*j7 + 128*j8), ここで、j1, j2,..., j8は、0または1をとります。 次の図に示すように、j1からj4までの値で壁表面の左側にある穴の辺の可視性を、j5からj8の値で右側にある辺の可視性をそれぞれ制御します。



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。



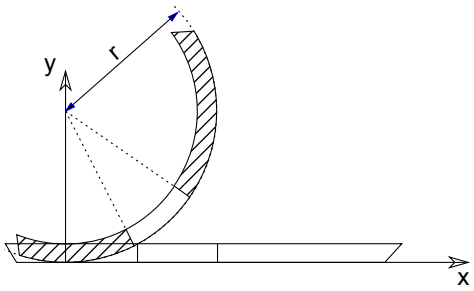
BWALL_

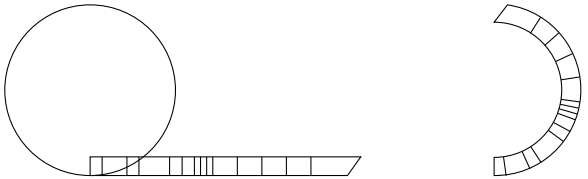
BWALL_ left_material, right_material, side_material,
height, x1, x2, x3, x4, t, radius,
mask1, mask2, mask3, mask4,
n,
x_start1, y_low1, x_end1, y_high1, frame_shown1,
...
x_startn, y_lown, x_endn, y_highn, frame_shownn,
m,
a1, b1, c1, d1,
...
am, bm, cm, dm

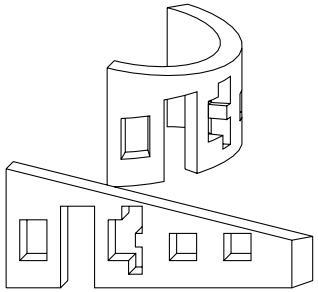
直線壁であるCWALL_要素と同じデータ構造に基づいた、滑らかな曲面壁。半径のパラメータだけが追加されています。 対応するCWALL_の派生形。X-Z平面をこの平面に正接する指定した半径の円柱の上に折り曲げることで作り出されます。 X軸沿いの辺は円弧に変形されますが、Y軸沿いの辺は半径方向になり、垂直な辺は垂直なままです。 曲率は、現在の解像度で設定された辺数によって近似されます（コマンド「RADIUS」、「RESOL」および「TOLER」を参照）。

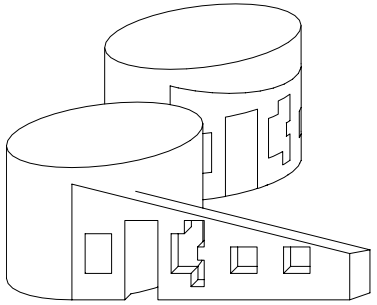
詳細は、「CWALL_」を参照してください。

例 1: BWALL_と対応するCWALL_



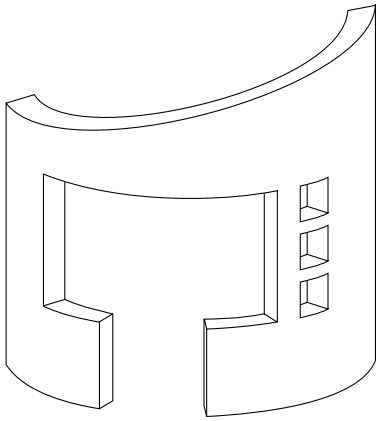






例 2:

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



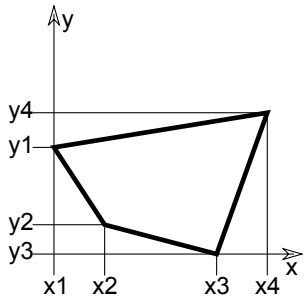
XWALL_

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

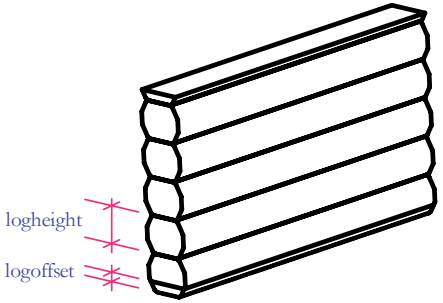
BWALL_要素と同じデータ構造に基づいた壁の定義の拡張版。

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です。

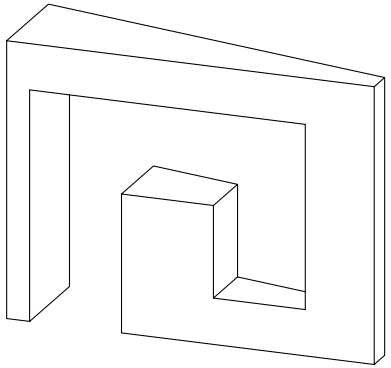


log_height, log_offset: 壁をログで構成できるようにする追加パラメータ。直線壁の場合のみ有効です。



status: ログで構成した壁の挙動を制御します。
status = j_1 + 2*j_2 + 4*j_3 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9, ここで、 各 jiフラグは0または1をとります.
j_1: 水平な辺に右側面材質を適用
j_2: 水平な辺に左側面材質を適用
j_3: ハーフログから開始
j_6: 壁の辺に合わせてテクスチャを調整
j_7: 曲がった側面の二重半径
j_8: 右側面を角ログに
j_9: 左側面を角ログに

例:



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

XWALL_{2}

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

XWALL_要素と同じデータ構造に基づいた壁定義の拡張版。

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: 穴の辺を表示する場合
0: 穴の辺を表示しない場合
< 0: 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 + 256*j9 + 512*j10), ここで j1, j2,..., j10 は、0または1をとります. 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。

XWALL_{3}

XWALL_{3} left_material, right_material, vertical_material, horizontal_material,
height, x1, x2, x3, x4,
y1, y2, y3, y4,
t, radius,
log_height, log_offset,
mask1, mask2, mask3, mask4,
n,
x_start1, y_low1, x_end1, y_high1,
sill_depth1, frame_shown1,
...
x_startn, y_lown, x_endn, y_highn,
sill_depthn, frame_shownn,
m,
a1, b1, c1, d1,
...
am, bm, cm, dm,
status

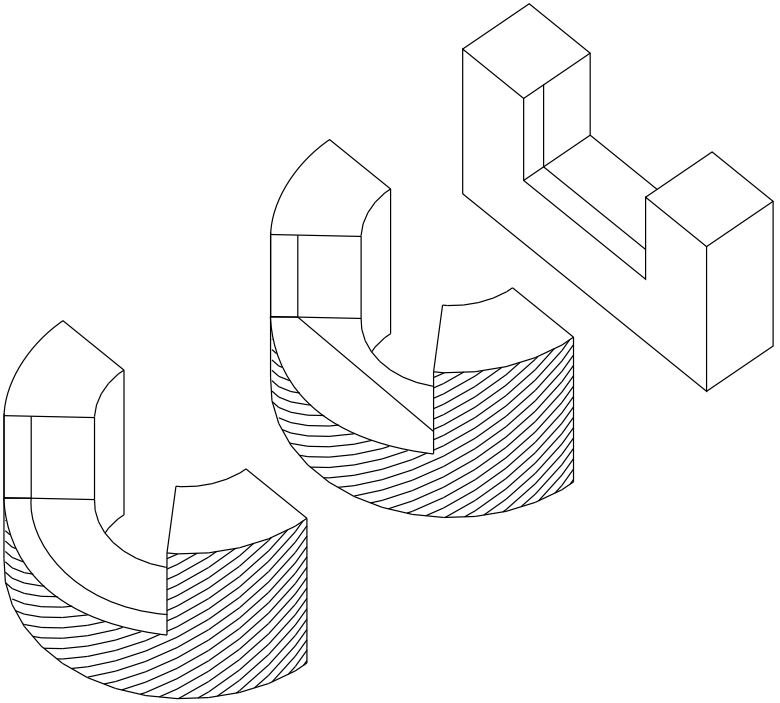
XWALL_{3} はログ壁の辺を全て隠す XWALL_{2} コマンドの拡張機能です。

status: ログで構成した壁の挙動を制御します。

status = j_1 + 2*j_2 + 4*j_3 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9 + 512*j_{10}, ここで、 各 jiフラグは0または1をとります.

j_1: 水平な辺に右側面材質を適用
j_2: 水平な辺に左側面材質を適用
j_3: ハーフログから開始
j_6: 壁の辺に合わせてテクスチャを調整
j_7: 曲がった側面の二重半径
j_8: 右側面を角ログに
j_9: 左側面を角ログに
j_{10}: ログ壁の全ての辺を隠す

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

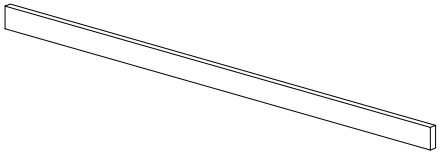
BEAM

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

梁の定義 パラメータはXWALL_要素のものと似ています。

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

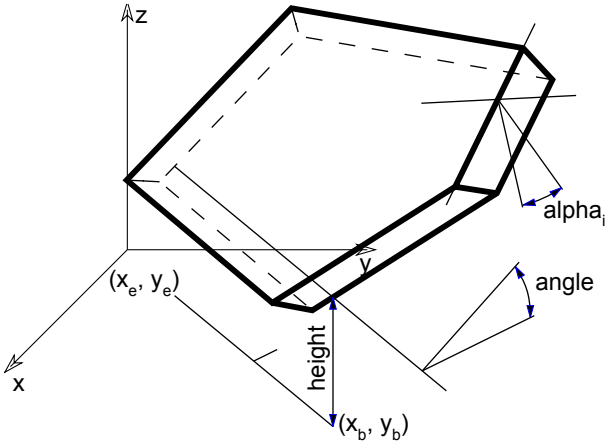
angle: 与えられた方向の基準線を中心とした屋根平面の度数単位の回転角 （反時計回り）

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

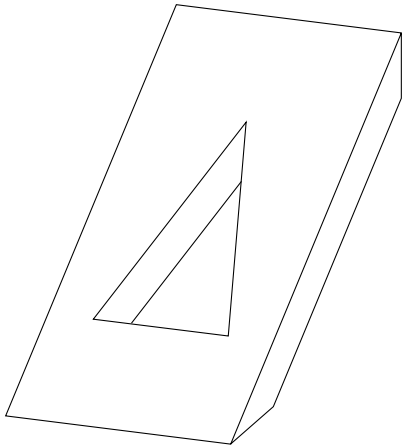
alphai: 屋根の辺iに属している面と屋根平面に垂直な平面との角度。-90 < alphai < 90。 屋根ポリゴンの各辺の正規の方向に対して反時計回りの回転角が正の角度になります。 屋根ポリゴンの辺が正規の向きになるのは、上面図で、輪郭を反時計回り方向、穴を時計回り方向にした場合です。

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、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

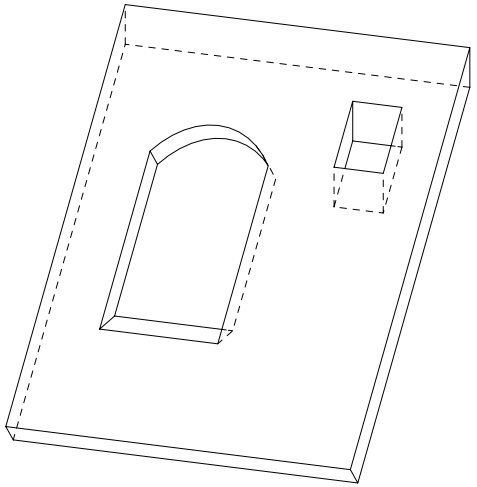
例 1:



```
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
```

例 2:

```
L=0.25
r=(0.6^2+L^2)/(2*L)
a=ASN(0.6/r)
CROOF_ " 屋根 - 瓦 ", " 木材 - マツ横 ", " 木材 - マツ横 ",
        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_」 コマンドの拡張版です。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

CROOF_{3}

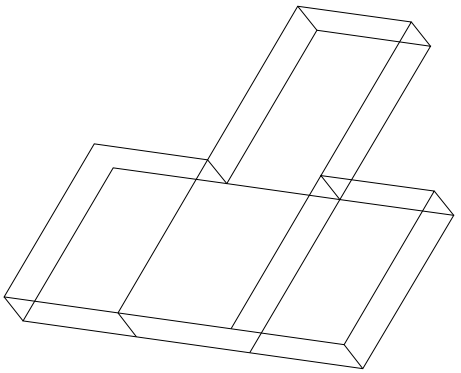
CROOF_{3} top_material, bottom_material, side_material, mask,
n, xb, yb, xe, ye, height, angle, thickness,
x1, y1, alpha1, s1, mat1,
...
xn, yn, alphan, sn, matn

「CROOF_{2}」 コマンドの拡張版です。生成された屋根のグローバル表現をコントロールできます。

mask: 生成された屋根のグローバル表現をコントロールします。
mask = j_1 + 2*j_2 + 4*j_3 + 8*j_4, ここで、 各 jiフラグは0または1をとります.
j_1: 陰線処理をした上部の辺
j_2: 陰線処理をした下部の辺
j_3: 陰線処理をした側面の辺
j_4: 断面形状の曲面の側面の辺と表面は滑らかです。互換性：ARCHICAD 21で導入されました。

例:

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



CROOF_{4}

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

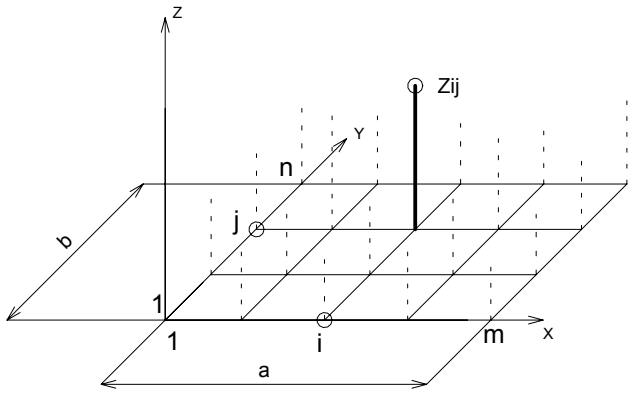
CROOF_{4} は 「CROOF_{3}」コマンドの拡張版で、インライン材質定義できます。それはローカルのGDLスクリプトに材質を定義でき、グローバル材質定義で定義された材質と一緒に使用できます。

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：



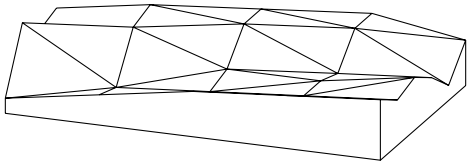
mask:

mask = j_1 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7, ここで、 各 jiフラグは0または1をとります.
j_1: 基準面を表現
j_3: 側面を表現
j_5: 基準面および側面の辺は可視
j_6: 上面の辺は可視
j_7: 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:

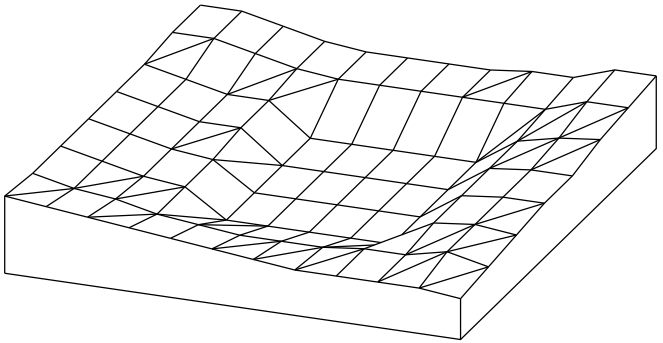
m >= 2, n >= 2

例 1:



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

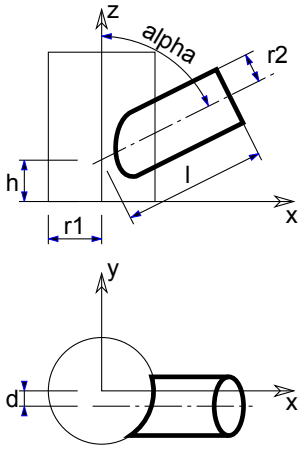
例 2:



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

ARMC

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



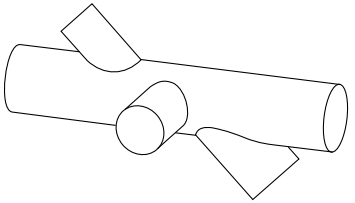
ほかの管から突き出ている管。パラメータについては図を参考にしてください（貫通する曲線についても計算され、描かれます）。alphaは度（°）で指定します。

パラメータの制限:

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

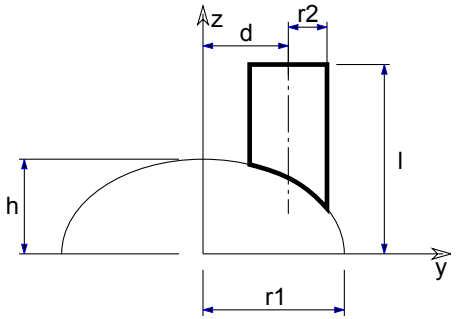
例:

```
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



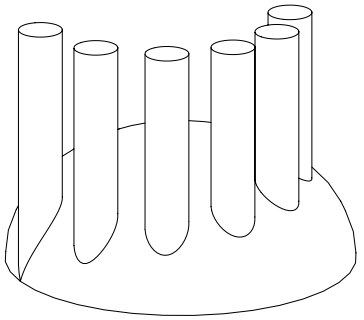
Y-Z平面の楕円から突き出ている管。 パラメータについては図を参考にしてください（貫通する曲線についても計算され、描かれます）。

パラメータの制限:

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

例:

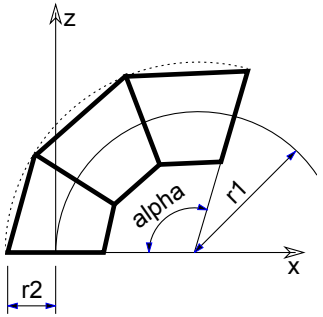
```
ELLIPS 3,4
FOR i=1 TO 6
    ARME 6,4,0.5,3,3.7-0.2*i
    ROTZ 30
NEXT i
```



ELBOW

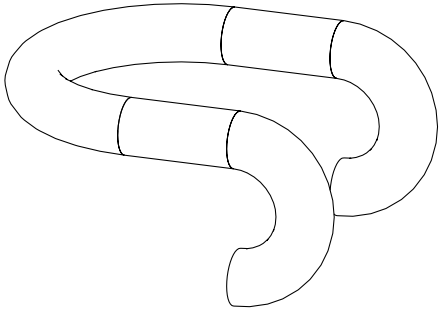
ELBOW r1, alpha, r2

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 [, customDescription]]]]]
点（x, y, z）の3Dホットスポット。

unID: 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。
customDescription: 情報パレット内の表示されたパラメータのカスタム説明文。 このオプションを使用すると、displayParamも設定する必要があります（デフォルトではparamReferenceを使用してください）。
ホットスポットの使用方法については、ホットスポットベースの編集コマンドを参照してください。

HOTLINE

HOTLINE x1, y1, z1, x2, y2, z2, unID
点P1 (x1,y1,z1)と点P2 (x2,y2,z2)の間のステータス線分。

HOTARC

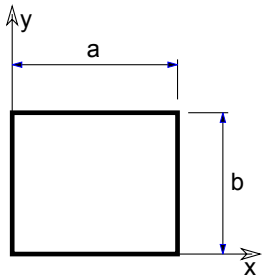
HOTARC r, alpha, beta, unID
x-y平面上の、中心が原点で、alphaで始まりbetaで終わる角度を持つ、半径がrのステータス円弧。
alphaとbetaは度（°）で指定します。

LIN_

LIN_ x1, y1, z1, x2, y2, z2
点P1 (x1,y1,z1)と点P2 (x2,y2,z2)の間の線分。

RECT

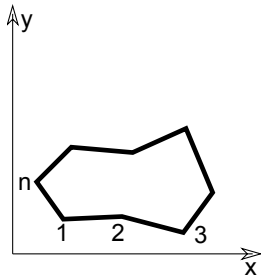
RECT a, b



辺aとbを持つX-Y平面上の矩形。
パラメータの制限:
a >= 0, 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コマンド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: 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。

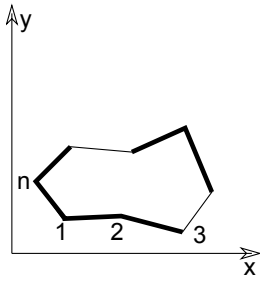
si = 0: の場合、頂点（xi, yi）から延びる辺は省略されます。

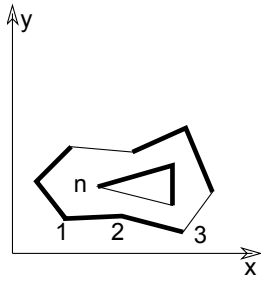
si = 1: の場合、辺は表示されます。

si = -1: は、角柱に直接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、「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

PLANE n, 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、インタプリタはその状態を確認しません。

パラメータの制限:

n >= 3

PLANE_

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

「PLANE」、コマンドとほとんど同じですが、「POLY_」コマンドの場合と同様、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。

「追加ステータスコード」を参照してください。

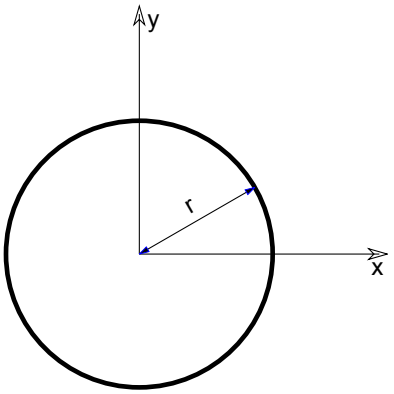
パラメータの制限:

n >= 3

CIRCLE

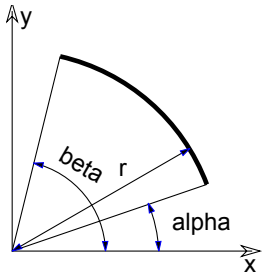
CIRCLE r

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の円弧（ワイヤフレームモード）または扇形（その他のモード）。

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
- TUBE
- TUBE{2}
- 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します（以下を参照）。 2本目のポリライン（RULED、SWEEP、 TUBEとTUBEA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
- COONS{2}

任意の数のポリラインから作成される形状：

- 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かまいません。閉じている場合には、最初の節点をコマンド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:

mask = $j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7$, ここで、 各 jiフラグは0または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。

j_1: 基準面
j_2: 上面
j_3: 側面
j_4: 反対の側面
j_5: 基準面の辺
j_6: 上面の辺

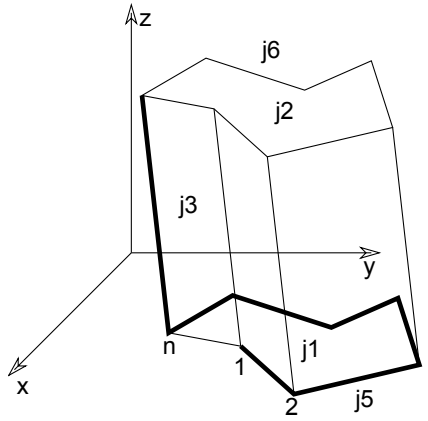
j_7: 横断面/表面の辺は可視、 表面はスムージングされない
全ての面と辺を使用可能にするには、マスク値を127に設定します。

状態

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、「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」 コマンド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

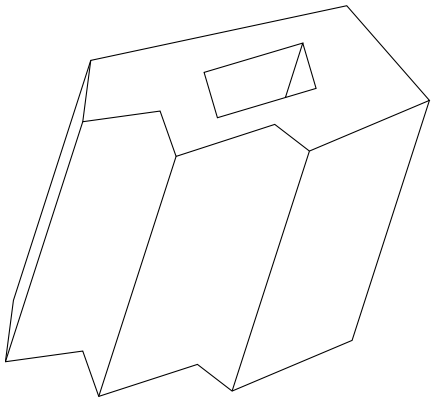
mask: 下面、上面、および（開いているポリラインの場合）側面のポリゴンの有無を制御します。
mask = j_1 + 2*j_2 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8, ここで、 各 jiフラグは0または1をとります.
j_1: 基準面を表現
j_2: 上面を表現
j_3: （閉じている） 側面を表現
j_5: 基準面の辺は可視
j_6: 上面の辺は可視
j_7: 横断面の辺は可視、 表面は明確
j_8: 横断面の辺はシャープで、 表面のスムージングはOpenGLとレンダリングで停止

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。
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、「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 > 2

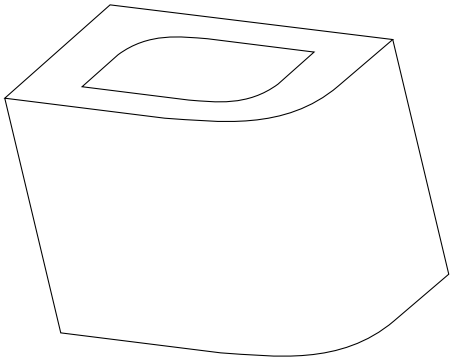
例 1:

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



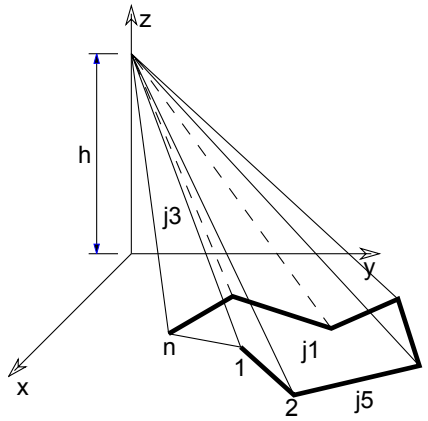
例 2:

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



PYRAMID

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



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。
mask = j_1 + 4*j_3 + 16*j_5, ここで、 各 jiフラグは0または1をとります.
j_1: 基準面を表現
j_3: （閉じている） 側面を表現
j_5: 基準面の辺は可視

si: 側面の辺のステータス
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。

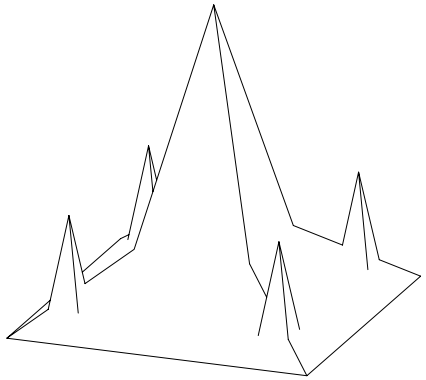
詳細は、「追加ステータスコード」を参照してください。

パラメータの制限:

h > 0およびn > 2

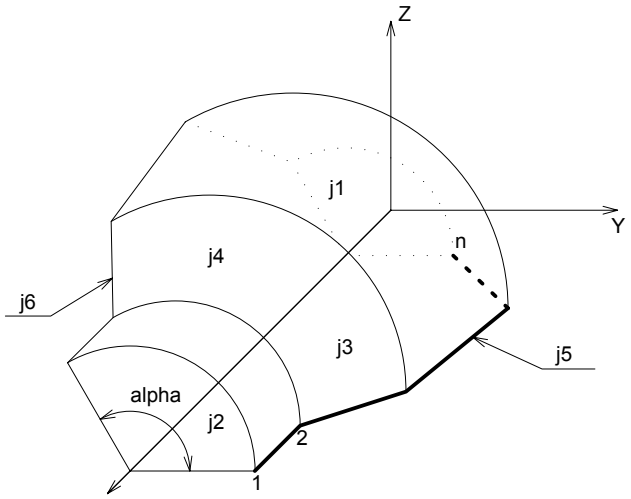
例:

```
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
for i = 1 to 4        ! 4つのピーク
    ADD -1.4, -1.4, 0
    PYRAMID 4, 1.5, 21,
            -0.25, -0.25, 0,
            0.25, -0.25, 0,
            0.25, 0.25, 0,
            -0.25, 0.25, 0
    DEL 1
    ROTZ 90
next i
del 4
```



REVOLVE

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



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曲面。断面形状ポリラインに穴を含めることはできません。

n: ポリラインの節点の数

alpha: グローバル回転角（度単位）

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。
mask = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9, ここで、 各 jiフラグは0または1をとります.
j_1: 基準面を表現
j_2: 端の表面を表現
j_3: 基準面の閉じている側面（断面形状の平面上）を表現
j_4: 最後の閉じている側面（回転した平面上）を表現
j_5: 基準面の辺 （座標x2, y2, z2） は可視
j_6: 最後の辺 （座標xm-1, ym-1, zm-1） は可視
j_7: 横断面の辺は可視、 表面は明確
j_8: 陰線処理をした水平方向の,
j_9: 陰線処理をした垂直方向の辺

si: 縦の円弧のステータス
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。

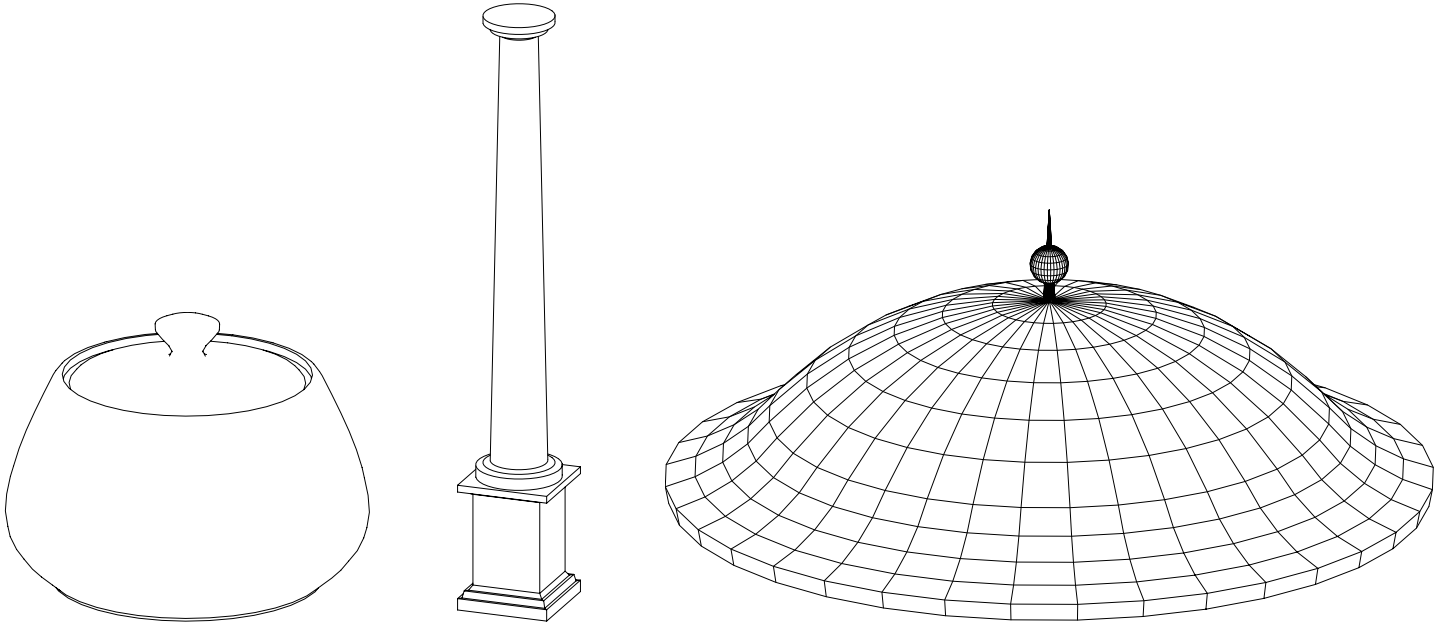
追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。

詳細は、「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:

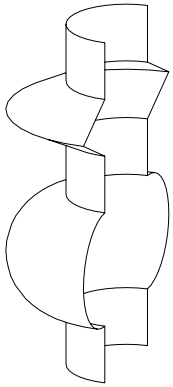
n >= 2
y_i >= 0.0
y_i = 0.0 と y_{i+1} （つまり、 隣接する2つの節点のy値） は同時に0にはできません。

例 1:



```
ROTY -90
REVOLVE 22, 360, 1+64,
        0, 1.982, 0,
        0.093, 2,     0,
        0.144, 1.845, 0,
        0.220, 1.701, 0,
        0.318, 1.571, 0,
        0.436, 1.459, 0,
        0.617, 1.263, 0,
        0.772, 1.045, 0,
        0.896, 0.808, 0,
        0.987, 0.557, 0,
        1.044, 0.296, 0,
        1.064, 0.030, 0,
        1.167, 0.024, 0,
        1.181, 0.056, 0,
        1.205, 0.081, 0,
        1.236, 0.096, 0,
        1.270, 0.1,   0,
        1.304, 0.092, 0,
        1.333, 0.073, 0,
        1.354, 0.045, 0,
        1.364, 0.012, 0,
        1.564, 0,     0
```

例 2:



ステータスコード2を使用しない回避策：

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

ステータスコード2を使用した場合と同じ結果：

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

REVOLVE{2}

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

REVOLVEの上位バージョン。断面形状ポリゴンは常に閉じており、穴がある場合もあります。 開始角度と面の材質が制御可能です。

alphaOffset: 掃引開始角度

alpha: 度単位の掃引角度距離 （負の値も可）

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。
mask = $4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9$, ここで、 各 jiフラグは0または1をとります.
j_3: 基準面の閉じている側面（断面形状の平面上）を表現

j_4: 最後の閉じている側面（回転した平面上）を表現
j_5: 基準面の辺 （座標x2, y2, z2） は可視
j_6: 最後の辺 （座標xm-1, ym-1, zm-1） は可視
j_7: 横断面の辺は可視、 表面は明確
j_8: 陰線処理をした水平方向の,
j_9: 陰線処理をした垂直方向の辺

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

REVOLVE{3}

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

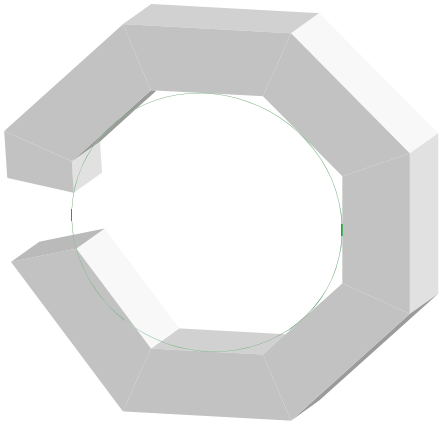
REVOLVE{3} は2つのスナップポジションを可能にする REVOLVE{2} コマンドの拡張機能です。 回転時はベースポリラインの各点のパスはポリラインで、近似する円弧です。 REVOLVE{3} では、ポリラインが正確に円に収まる、2つのスナップロケーションを定義することができます。 REVOLVE{2} では、2つのスナップロケーションは回転の始まりと終わりです。 REVOLVE{3} では、終端ポイントは円上である必要はなく、終わりの面の切断です。

betaOffset: 最初のスナップポジションを定義する角度。定義された角度は回転する範囲である必要はありません。

beta: 最初のスナップポジションを基準に、2つ目のスナップポジションを定義する角度。負の値も可。定義された角度は回転する範囲である必要はありません。

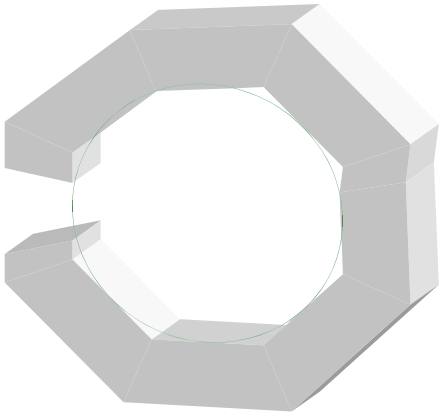
例:

revolve{2} 終端のスナップポジション



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

revolve{3} カスタムのスナップポジション



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

REVOLVE{4}

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

REVOLVE{4} は全ての辺を隠す REVOLVE{3} コマンドの拡張機能です。

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。
mask = $4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9 + 512*j_{10} + 1024*j_{11}$, ここで、 各 jiフラグは0または1をとります.
j_3: 基準面の閉じている側面（断面形状の平面上）を表現
j_4: 最後の閉じている側面（回転した平面上）を表現
j_5: 基準面の辺 （座標x2, y2, z2） は可視
j_6: 最後の辺 （座標xm-1, ym-1, zm-1） は可視
j_7: 横断面の辺は可視、 表面は明確
j_8: 陰線処理をした水平方向の,
j_9: 陰線処理をした垂直方向の辺
j_{10}: 回転の全ての辺を隠す
j_{11}: 断面形状の曲面の側面の辺と表面は滑らかです。互換性：ARCHICAD 21で導入されました。

REVOLVE{5}

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

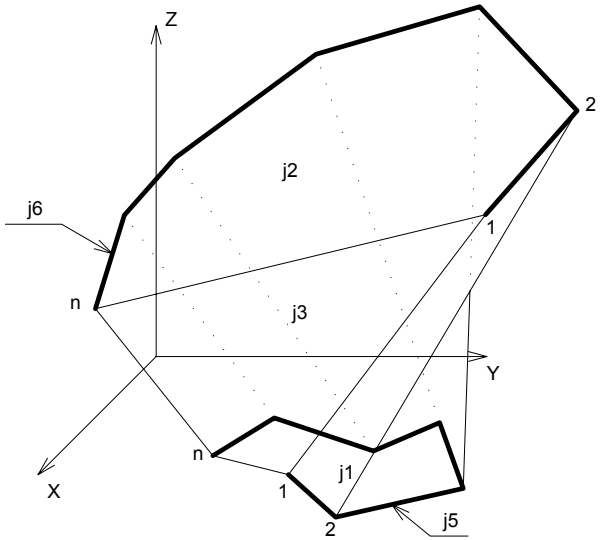
REVOLVE{5} は 「REVOLVE{4}」コマンドの拡張版で、インライン材質定義できます。それはローカルのGDLスクリプトに材質を定義でき、グローバル材質定義で定義された材質と一緒に使用できます。

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サーフェスです。平面曲線ポリラインに穴を含めることはできません。 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ができる唯一の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それらを定義する節点の入力順番（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

ui, vi: 平面曲線の節点の座標

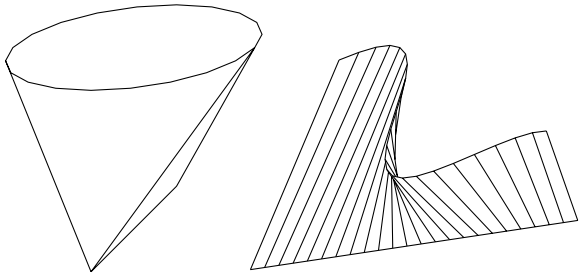
xi, yi, zi: 空間曲線の節点の座標

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。
mask = $j_1 + 2*j_2 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7$, ここで、 各 jiフラグは0または1をとります.
j_1: 基準面を表現
j_2: 上面を表現（上面が平面でない場合は無効）
j_3: 側面を表現（平面四角形あるいは2つの三角形）
j_5: 平面曲線の辺は可視

j_6: 空間曲線の辺は可視
j_7: 表面上の辺は可視。 表面はスムージングされない

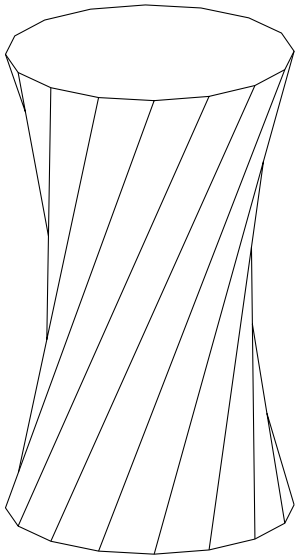
si: 側面の辺のステータス
0: 節点から延びる側面の辺は全て可視
1: 節点から延びる側面の辺を輪郭の表示に使用

パラメータの制限:
n > 1



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



RULEDSEGMENTED

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

互換性：ARCHICAD 21で導入されました。

RULEDSEGMENTEDは、3D空間で任意の2つの形状に基づく表面を作成します。2つのポリラインの頂点の数は同じである必要があります。 RULEDと同様に連続する二重のルールサーフェスを生成しますが、入力ポリラインの制約が少なく、分割をより高品質に行えます。

2つの断面形状の対応する頂点は、直線で接続されます。断面形状のスキュー辺の対応するペアは、二重のルールサーフェス（数学的双曲放物面）で接続されます。両方向に分割されるため、レンダリングと断面がより滑らかになります。

断面形状ポリラインの条件：

- 両方が3Dポリラインであり、同一平面上にある必要はありません。
- それぞれが閉じていますが、どちらにも穴は含まれません。
- 連続する複数の頂点が扇状の表面を形成する場合でも、それぞれに同じ頂点が含まれることがあります。
- 断面形状ポリラインが閉じて同一平面上にあると、閉じたポリゴンが生成されます。

n: 各曲線のポリラインの節点の数

x1i, y1i, z1i: 最初の断面形状ポリライン上の頂点の3D位置

x2i, y2i, z2i: 2番目の断面形状ポリライン上の頂点の3D位置

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。

mask = j_1 + 2*j_2 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7, ここで、 各 jiフラグは0または1をとります.

j_1: ベース表面を表現（最初のポリラインが同一平面上になく、j3が設定されていない場合は無効）

j_2: 上面材質を表現（2番目のポリラインが同一平面上になく、j3が設定されていない場合は無効）

j_3: 閉じた側面材質を表現（最後と最初の頂点間の追加のセグメント上の表面）

j_5: 最初の断面形状ポリライン上の辺は可視

j_6: 2番目の断面形状ポリライン上の辺は可視

j_7: 表面上の辺は可視。 表面はスムージングされない

si: 母線の状態（最初の断面形状ポリラインの1つの頂点と2番目のポリラインの対応する頂点の間の側面の辺）

0: 母線は可視

1: 輪郭の表示に母線を使用

2: 母線は可視でレンダリングの区切りを定義

パラメータの制限:

n > 1

例:

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

RULEDSEGMENTED{2}

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

互換性：ARCHICAD 23で導入されました。

RULEDSEGMENTED{2}は「RULEDSEGMENTED」コマンドの拡張版で、セグメント詳細で生成された表面の材質属性をコントロールし、カスタムテクスチャの投影を適用できます。

追加パラメータ：

top_material: ベース表面の材質属性インデックス（最初のポリラインが同一平面上にあり、j3が設定されている場合）。

bottom_material: 上面の材質属性インデックス（ 2番目のポリラインが同一平面上にあり、j2とj3が設定されている場合）。

textureMode: テクスチャ投影モード
0: 自動、曲面用に最適化、「RULEDSEGMENTED」と同じ。
1: カスタム、「COOR」によって定義。

mati: 生成された材質セグメントiの材質属性インデックス。

例:

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

RULEDSEGMENTED{2} _topMatIndex, _bottomMatIndex,
            4, 1+2+16+32, 0,
            0, 0, 0, 2, _segmentMatIndex_1,
            1, 0, 0, 2, _segmentMatIndex_2,
            1, 1, 0, 2, _segmentMatIndex_1,
            0, 1, 0, 2, _segmentMatIndex_2,
            1, 0, 1,
            1, 1, 1,
            0, 1, 1,
          0, 0, 1
```

SWEEP

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

ポリラインの、空間曲線パスに沿った移動掃引によって生成されるサーフェス。

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回転角の増加量

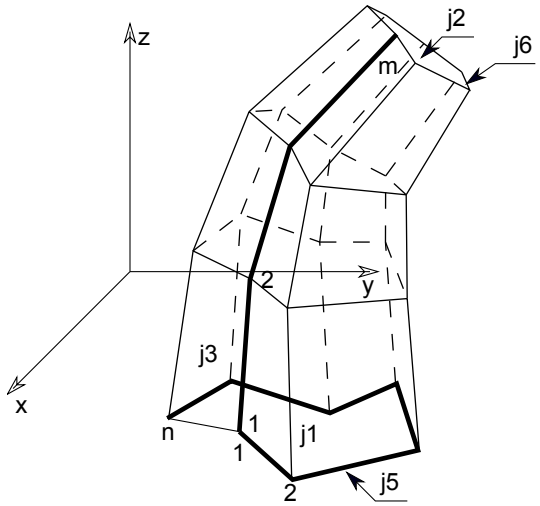
scale: パスのある節点から次の節点へのポリラインのスケール係数の増加量

ui, vi: 基準ポリラインの節点の座標

xi, yi, zi: パス曲線の節点の座標

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
mask = j_1 + 2*j_2 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7, ここで、 各 jiフラグは0または1をとります.
j_1: 基準面を表現

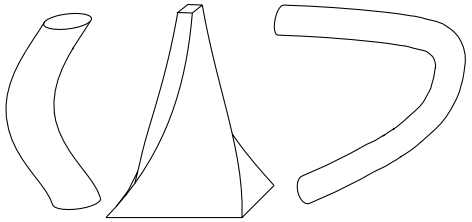
j2: 上面を表現
j3: 側面を表現
j5: 基準面の辺は可視
j6: 上面の辺は可視
j7: 横断面の辺は可視、 表面は明確



si: 側面の辺のステータス
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

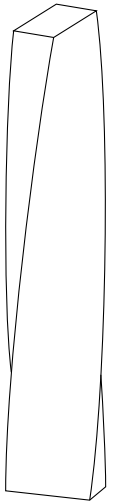
追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。
詳細は、「追加ステータスコード」を参照してください。
パラメータの制限:

n > 1
m > 1
z1 < z2



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サーフェス。 断面は、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 axis: 対応する点における生成元の曲線の正接方向

W axis: 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 axis: 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。

パス線分内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: 基準ポリラインの節点の数

m: パス曲線の節点の数

ui, wi: 基準ポリラインの節点の座標

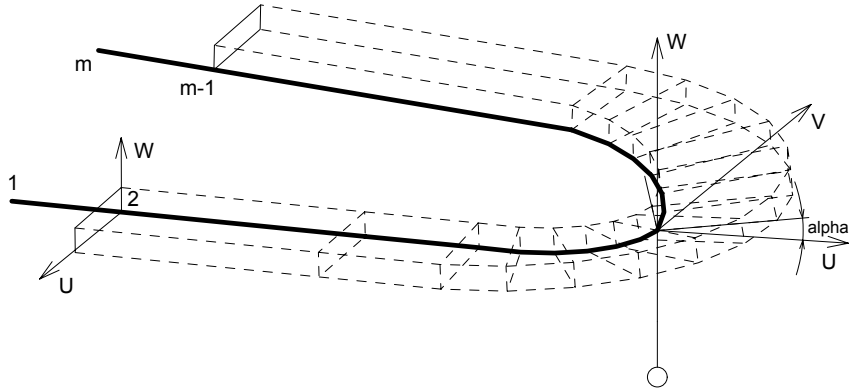
xi, yi, zi: パス曲線の節点の座標

anglei: 切断面の回転角度

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

mask = j_1 + 2*j_2 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8 + 512*j_{10} + 1024*j_{11} + 2048*j_{12} + 4096*j_{13}, ここで、 各 jiフラグは0または1をとります.

j_1: 基準面を表現
j_2: 端の表面を表現
j_5: 基準面の辺（座標x2, y2, z2）は可視
j_6: 最後の辺（座標xm-1, ym-1, zm-1）は可視
j_7: 横断面の辺は可視、 表面は明確
j_8: 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
j_{10}: 陰線処理に参加するベース辺（互換性：ARCHICAD 23で導入されました。）
j_{11}: 陰線処理に参加する終端辺（互換性：ARCHICAD 23で導入されました。）
j_{12}: 陰線処理に参加する縦辺（断面を接続する辺）（互換性：ARCHICAD 23で導入されました。）
j_{13}: 陰線処理に参加する断面の辺（互換性：ARCHICAD 23で導入されました）



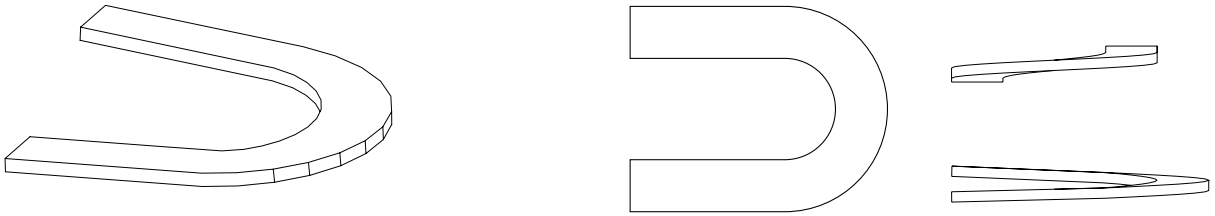
si: 側面の辺のステータス
0: 節点から延びる側面の辺は全て可視
1: 節点から延びる側面の辺を輪郭の表示に使用
2: ARCHICAD またはZバッファレンダリングエンジンを使用してスムージングを設定する場合、この点に属する横の辺は区切りを定義します。この方法は、追加節点の定義と同等です。 計算は、コンパイラによって行われます。ほかのレンダリング方式を使用すると、0を使用した場合と同じ結果になります。

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。

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の最初と最後の断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。

パラメータの制限:
n > 2およびm > 3

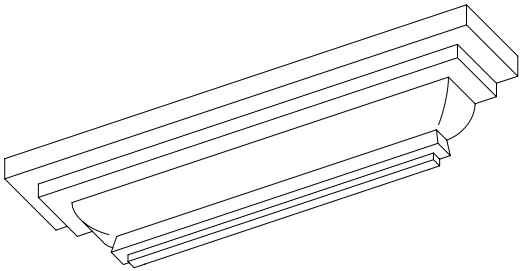
例 1:



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

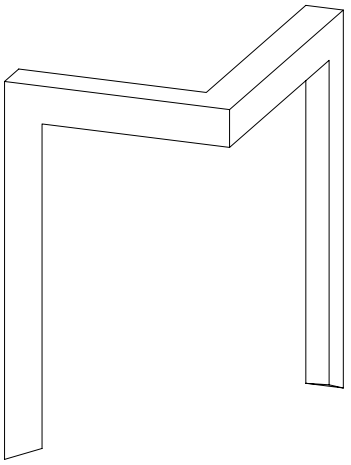
例 2:

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



例 3:

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



TUBE{2}

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

互換性：ARCHICAD 21で導入されました。「TUBE」の拡張バージョンです。

- 輪郭ベースポリゴン内で穴を定義することができます
- 上面、底面ポリゴンと切り取り領域の個別の表面属性
- 同じポリゴン辺に属する側面ポリゴンの個別の表面属性

V axis, W axis, U axis: 「TUBE」での意味と同様です。

top_material: 閉じたポリゴンの表面

bottom_material: 開始ポリゴンの表面

cut_material: 切り取り領域の表面

n, m, ui, wi: 「TUBE」での意味と同様です。

xi, yi, zi, anglei: 「TUBE」での意味と同様です。パスに円弧を含めることはできません（分割は手動です）。

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

mask = j_1 + 2*j_2 + 16*j_5 + 32*j_6 + 256*j_9 + 512*j_{10} + 1024*j_{11} + 2048*j_{12} + 4096*j_{13}, ここで、 各 jiフラグは0または1をとります.

j_1: 基準面を表現

j_2: 端の表面を表現

j_5: 基準面の辺（座標x2, y2, z2）は可視

j_6: 最後の辺（座標xm-1, ym-1, zm-1）は可視

j_9: 断面形状の曲面の側面の辺と表面は滑らかです。

j_{10}: 陰線処理に参加するベース辺（互換性：ARCHICAD 23で導入されました。）

j_{11}: 陰線処理に参加する終端辺（互換性：ARCHICAD 23で導入されました。）

j_{12}: 陰線処理に参加する縦辺（断面を接続する辺）（互換性：ARCHICAD 23で導入されました。）

j_{13}: 陰線処理に参加する断面の辺（互換性：ARCHICAD 23で導入されました）

si: 側面の辺のステータス

-1: ベースポリゴン内の穴の最後の頂点（穴の最初の頂点を複製）、またはベースポリゴンに穴が含まれる場合はポリゴン外側の閉じた頂点を示します。ステータス -1をもつこれらの複製された頂点では、matnパラメータは無視されます。

0, 1, 2: 「TUBE」での意味と同様です。

mati: ベースポリゴンのui, wi頂点から始まる辺に属する側面ポリゴンの個別の表面

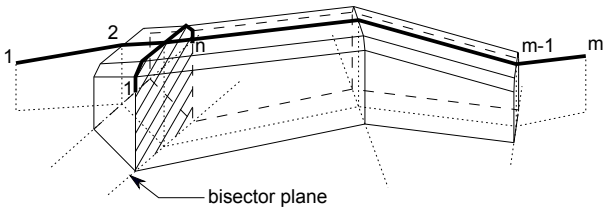
追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。このようなポリゴン辺は、処理中に自動的に分割されます。

例:

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

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

        ! outside contour
        -0.01,  0.01,  0, matOuter,
        -0.01, -0.01,  0, matOuter,
         0.01, -0.01,  0, matOuter,
         0.01,  0.01,  0, matOuter,
        -0.01,  0.01, -1, matOuter,

        ! hole contour
        -0.008,  0.008,  0, matInner,
        -0.008, -0.008,  0, matInner,
         0.008, -0.008,  0, matInner,
         0.008,  0.008,  0, matInner,
        -0.008,  0.008, -1, matInner,

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

TUBEA

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



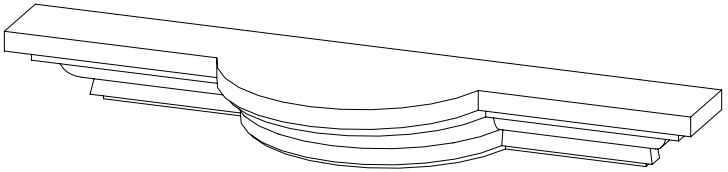
TUBEAは、「TUBE」コマンドとは別のアルゴリズムで、空間曲線パスに沿ってポリラインを掃引することで生成されるサーフェスです。

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サーフェス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。

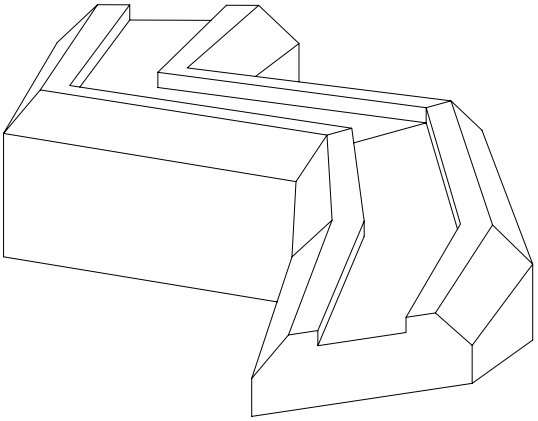
パス線分で切り取られたチューブの横断面は、基準ポリゴンと異なる場合があります。

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、「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。

mask:

mask = $4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7$, ここで、 各 jiフラグは0または1をとります.

j_3: 最初の境界 （x1, y1, z1j7 ） の辺は可視（j7が設定されている場合のみ有効）

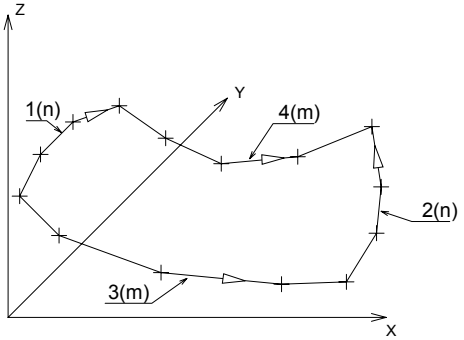
j_4: 2番目の境界 （x2, y2, z2 ） の辺は可視（j7が設定されている場合のみ有効）

j_5: 3番目の境界 （x3, y3, z3 ） の辺は可視（j7が設定されている場合のみ有効）

j_6: 4番目の境界 （x4, y4, z4 ） の辺は可視（j7が設定されている場合のみ有効）

j_7: 表面上の辺は可視、 表面はスムージングされない

表面上の辺が不可視（ビットj7が0に設定されている）の場合、全ての境界辺が可視になり、ビットj3～j6は無効になります。境界辺の可視性を、表面上の辺の可視性に関係なく定義するには、「COONS{2}」を使用します。



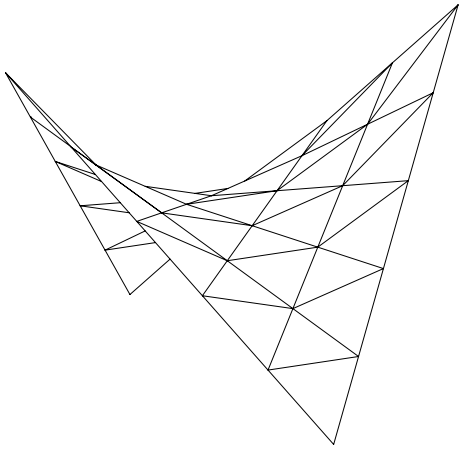
境界の向きは、 曲線1と2は曲線3から4の方向で、曲線3と5は曲線1から2の方向である必要があります。コーナー座標は、各カーブで同じである必要があります。

パラメータの制限:

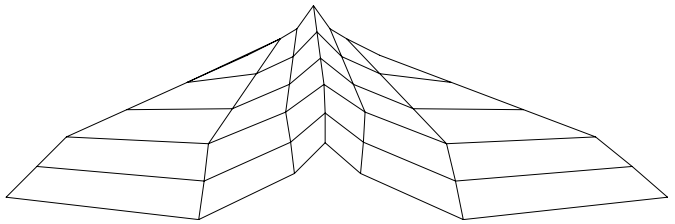
n > 1, m > 1

例 1:

```
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
```



例 2:



```
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
```

COONS{2}

COONS{2} n, m, mask,
x11, y11, z11, ..., x1n, y1n, z1n,
x21, y21, z21, ..., x2n, y2n, z2n,
x31, y31, z31, ..., x3m, y3m, z3m,
x41, y41, z41, ..., x4m, y4m, z4m

COONS{2}は「COONS」コマンドの拡張版で、表面上の辺と境界辺の可視性を個別に設定できます。

mask:
mask = 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7, ここで、 各 jiフラグは0または1をとります.
j_3: 最初の境界 （x1, y1, z1） の辺は可視
j_4: 2番目の境界 （x2, y2, z2） の辺は可視
j_5: 3番目の境界 （x3, y3, z3） の辺は可視
j_6: 4番目の境界 （x4, y4, z4） の辺は可視
j_7: 表面上の辺は可視、 表面はスムージングされない

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

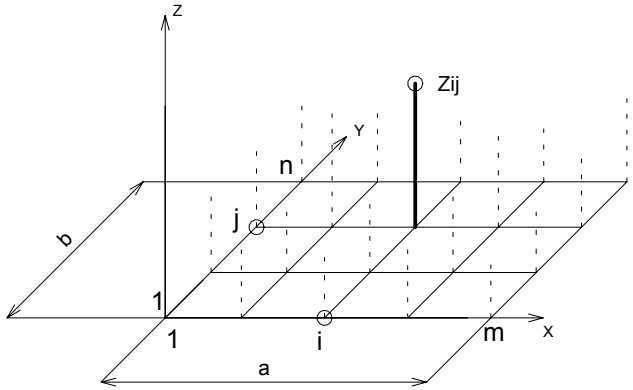
n: マスポリゴンに含まれる節点の数

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

mask:
mask = j_1 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8, ここで、 各 jiフラグは0または1をとります.
j_1: 基準面を表現
j_3: 側面を表現
j_5: 基準面および側面の辺は可視
j_6: 三角形分割辺は可視
j_7: 三角形分割の辺は可視、表面はスムージングされない
j_8: 全ての隆起線はシャープで、表面はスムージングされる



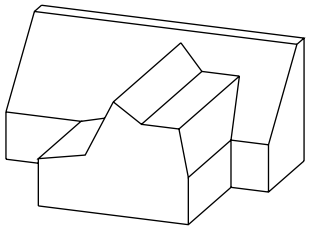
si: 「PRISM_」コマンドと同様。 追加ステータスコードでは、特殊な制約を使用して、平面ポリライン内に線分や円弧を作成することができます。

詳細は、「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, m >= 0

例:



```
MASS " 壁材 - 下塗り ", " 壁材 - 下塗り ", " 壁材 - 下塗り " ,
        15, 12, 117, -5.0,
        0, 12, 0, 15,
        8, 12, 0, 15,
        8, 0, 0, 15,
        13, 0, 0, 13,
        16, 0, 0, 13,
        19, 0, 0, 13,
        23, 0, 0, 13,
        24, 0, 0, 15,
        24, 12, 0, 15,
        28, 12, 0, 15,
        28, 20, 8, 13,
        28, 22, 8, 15,
        0, 22, 8, 15,
        0, 20, 8, 13,
        0, 12, 0, -1,
        0, 22, 8, 0,
        28, 22, 8, -1,
        23, 17, 5, 0,
        23, 0, 5, -1,
        13, 13, 1, 0,
        13, 0, 1, -1,
        16, 0, 7, 0,
        16, 19, 7, -1,
        0, 20, 8, 0,
        28, 20, 8, -1,
        19, 17, 5, 0,
        19, 0, 5, -1
```

MASS{2}

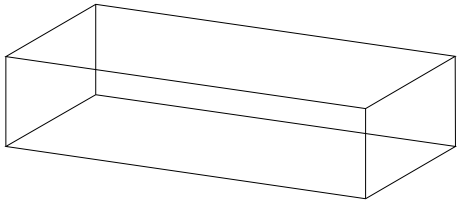
MASS{2} top_material, bottom_material, side_material,
n, m, mask, h,
x1, y1, z1, s1,
...
xn, yn, zn, sn,
xn+1, yn+1, zn+1, sn+1,
...
xn+m, yn+m, zn+m, sn+m

「MASS」の拡張、追加マスクビットにより形状の上部エッジを全てを隠すことが可能。

mask:
mask = j_1 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9 + 512*j_{10}, ここで、 各 jiフラグは0または1をとります.
j_1: 基準面を表現
j_3: 側面を表現
j_5: 基準面および側面の辺は可視
j_6: 上面の辺は可視
j_7: 上面の辺は可視、表面はスムージングされない
j_8: 全ての隆起線はシャープで、表面はスムージングされる
j_9: 陰線処理に参加する辺
j_{10}: 全ての上面の辺は非表示になります。

例:

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



POLYROOF

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

POLYROOFコマンドは多層屋根を作成します。その形状は複数のパラメータ、 最も重要な屋根角度、およびピボットポリゴンと輪郭ポリゴンの2つのポリゴンによって制御します。ピボットポリゴンでは、屋根は屋根角度に傾斜します。 l屋根は、次のレベルの高さに達するか、側面が互いに接してなくな るまで上昇します。 また、輪郭ポリゴンに達するまで下降し、その外側の屋根の一部を切り取ります。 輪郭ポリゴンを使用して、屋根の穴を切り取ることもできます。

defaultMat: 屋根の"内側"の材質のインデックス。 この材質は、屋根を平面で切り取った場合などに、切妻および切断面で可視化されます。

k: レベル数

m: ピボットポリゴンの頂点の数

n: 輪郭ポリゴンの頂点の数

offset: 屋根の厚さのオフセット

thickness: 屋根の厚さ

applyContourInsidePivot: 0に設定した場合、ポリゴンの外輪郭はピボットポリゴン平面の下にだけ適用されます 。 1に設定した場合、ポリゴンの外輪郭はピボットポリゴン平面の上と下の両方に適用されます 。 0の設定を使用すると、輪郭ポリゴンが外側に傾いている切妻を切り取らないようにできます。

z_i: レベルのZ座標

pivotX_i, pivotY_i: ピボットポリゴンの頂点の座標

pivotMask_i:
0: 値 0 は普通の頂点を指し、

-1: 値 -1 は現在のピボットサブポリゴン（外側の輪郭または穴）の終わりを示します。 このような頂点のデータは、サブポリゴンの最初の頂点のデータのコピーである必要があります。 ポリゴンは、その内側に穴がない場合でも、必ずマスク値-1で閉じる必要があります。

roofAngle_i: 指定したレベルにおけるピボットエッジの傾斜角度。この角度が90度以上の場合、屋根の一部は切妻になります。

gableOverhang_i: 切妻の側面で、屋根をそれ自体より低いレベルで拡張することができます。 その量はこのパラメータによって制御でき、少なくとも屋根の2番目のレベルにある切妻（roofAngle >= 90）にのみ影響を与えます。

topMat_i, bottomMat_i: 屋根の上下の材質のインデックス。

contourX_i, contourY_i: 輪郭ポリゴンの頂点の座標

contourMask_i:

0: 値 0 は普通の頂点を指し、

-1: 値-1は現在の輪郭サブポリゴン（外側の輪郭または穴）の終わりを示します。 このような頂点のデータは、サブポリゴンの最初の頂点のデータのコピーである必要があります。 ポリゴンは、その内側に穴がない場合でも、必ずマスク値-1で閉じる必要があります。

edgeTrim_i: 輪郭ポリゴンによってエッジを切り取る方法を指定します。有効値は次のとおりです。

0: 垂直

1: 屋根の平面に対して垂直

2: 水平

3: 屋根の平面に対するカスタム角度

edgeAngle_i: 屋根の平面に対するエッジのカスタム角度。edgeTrimを3 （屋根の平面に対するカスタム角度） に設定した場合のみ効果があります。

edgeMat_i: 屋根のエッジのインデックス。 ここで、 輪郭線が図を切り取ります。

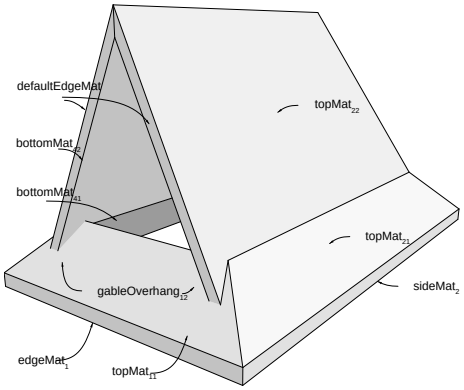

defaultEdgeMat
bottomMat22
bottomMat21
topMat22
topMat21
sideMat2
gableOverhang12
edgeMat1
topMat11

図 1：材質

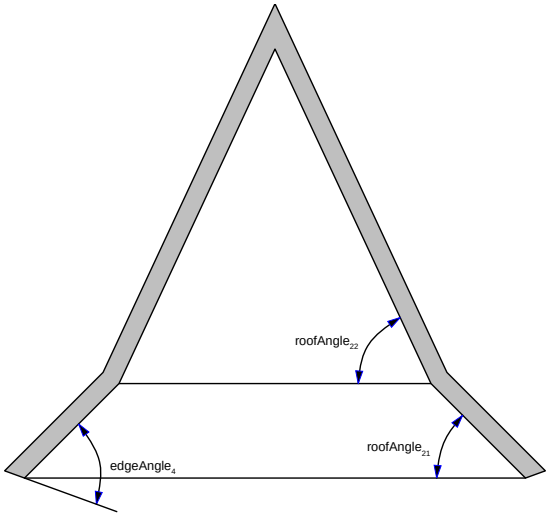

roofAngle22
roofAngle21
edgeAngle4

図 2：角度

例:

```
POLYROOF "Paint-01",
        2, 5, 5,
        0, 0.2, 0,
        ! Start of z values
        2.7,
        3.2,
        ! Start of pivot polygon
        2, 8, 0,
        45, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        90, 0.5, ind(material, "Paint-01"), ind(material, "Paint-01"),
        2, 3, 0,
        45, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        65, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        10, 3, 0,
        45, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        65, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        10, 8, 0,
        45, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        65, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        2, 8, -1,
        45, 0, ind(material, "Paint-01"), ind(material, "Paint-01"),
        90, 0.5, ind(material, "Paint-01"), ind(material, "Paint-01"),
        ! Start of contour polygon
        1.5, 8.5, 0, 0, 0, ind(material, "Paint-01"),
        1.5, 2.5, 0, 0, 0, ind(material, "Paint-01"),
        10.5, 2.5, 0, 0, 0, ind(material, "Paint-01"),
        10.5, 8.5, 0, 0, 0, ind(material, "Paint-01"),
        1.5, 8.5, -1, 0, 0, ind(material, "Paint-01")
```

出力：図1を参照

POLYROOF{2}

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

POLYROOF{2} は屋根の全厚を定義することを可能にする 「POLYROOF」 コマンドの拡張機能です。 このパラメータは、屋の断片の生成が望ましい場合、オフセットおよび厚さと共に考慮すべきです。 この場合、厚みおよびオフセットは断片の厚さにし、断片の上面と屋根全体のそれぞれの間の距離に設定する必要があります。

totalThickness: 屋根の厚さ。

POLYROOF{3}

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

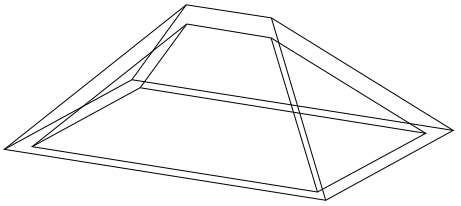
POLYROOF{3}は「POLYROOF{2}」コマンドの拡張版です。生成された屋根のグローバル制御をコントロールできます。

mask: 生成された屋根のグローバル表現をコントロールします。
mask = $j_1 + 2*j_2$, ここで、 各 jiフラグは0または1をとります.

j_1: 陰線処理に参加する辺
j_2: 全ての辺の可視化。

例:

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

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



POLYROOF{4}

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

POLYROOF{4} は 「POLYROOF{3}」コマンドの拡張版で、インライン材質定義できます。それはローカルのGDLスクリプトに材質を定義でき、グローバル材質定義で定義された材質と一緒に使用できます。

EXTRUDEDSHELL

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

最初にポリラインを押し出して、次にそれに厚さを追加することにより、作成されるサーフェス。

topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4: オブジェクトの上面、 下面、 および側面の材質。

defaultMat: オブジェクトの"内側"の材質のインデックス。 この材質は、 オブジェクトを平面で切り取った場合などに、 切断面で可視化されます。

n: 断面形状ベースのポリラインの頂点の数

offset: シェルの厚さのオフセット。負数にできません。

thickness: シェルの厚さ

flipped:
1: シェルを反転する場合は1、
0: 上記以外

trimmingBody:
1: シェルを切り取り目的で閉じる場合は1、
0: 上記以外

x_tb, y_tb, x_te, y_te, topz, tangle: 押し出しの上面を指定。 パラメータの意味は 「SPRISM_{2}」 コマンドと同じです。

x_bb, y_bb, x_be, y_be, bottomz, bangle: 押し出しの下面を指定。 パラメータの意味は 「SPRISM_{2}」 コマンドと同じです。

preThickenTran_i: 厚みをつける前に実行される変換。パラメータの意味については、「XFORM」コマンドを参照してください。

x_i, y_i, s_i: 断面形状ベースのポリラインのXおよびY座標とステータス値。詳細は、「EXTRUDE」を参照してください。 ステータスで側面の可視性を制御することはできません。

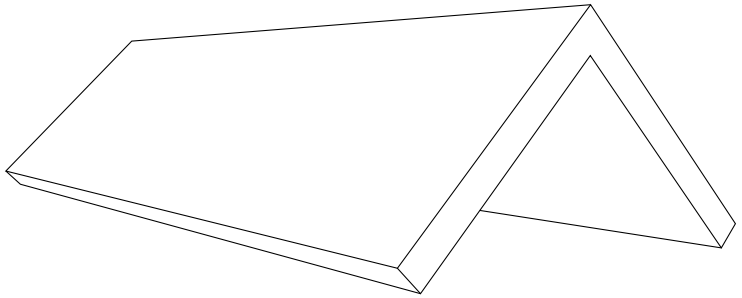
EXTRUDEDSHELL{2}

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

EXTRUDEDSHELL{2} はオリジナルと厚みのあるサーフェスの、全ての辺を隠すことを可能にする 「EXTRUDEDSHELL」 コマンドの拡張機能です。

status: ステータスビット：
status = j_1, ここで、 各 jiフラグは0または1をとります.
j_1: オリジナルと厚みのあるサーフェスの辺を全て非表示にする。

例:



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

EXTRUDEDSHELL{3}

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

EXTRUDEDSHELL{3} は 「EXTRUDEDSHELL{2}」 コマンドの拡張版で、インライン材質定義できます。それはローカルのGDLスクリプトに材質を定義でき、グローバル材質定義で定義された材質と一緒に使用できます。

REVOLVEDSHELL

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

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、次にそれに厚さを追加することによって生成されるサーフェス。

topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4: オブジェクトの上面、 下面、 および側面の材質。

defaultMat: オブジェクトの"内側"の材質のインデックス。 この材質は、 オブジェクトを平面で切り取った場合などに、 切断面で可視化されます。

n: 断面形状ベースのポリラインの頂点の数

offset: シェルの厚さのオフセット。負数にできません。

thickness: シェルの厚さ

flipped:
1: シェルを反転する場合は1、
0: 上記以外

trimmingBody:
1: シェルを切り取り目的で閉じる場合は1、
0: 上記以外

alphaOffset: 掃引開始角度

alpha: 度単位の掃引角度距離 （負の値も可）

preThickenTran_i: 厚みをつける前に実行される変換。パラメータの意味については、「XFORM」コマンドを参照してください。

x_i, y_i, s_i: 断面形状ベースのポリラインのXおよびY座標とステータス値。詳細は、「EXTRUDE」を参照してください。 ステータスで側面の可視性を制御することはできません。

REVOLVEDSHELL{2}

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

REVOLVEDSHELL{2} はサーフェスの辺、オリジナルと厚みのある表面の辺を隠すことを可能にする 「REVOLVEDSHELL」 コマンドの拡張機能です。

status: ステータスビット：

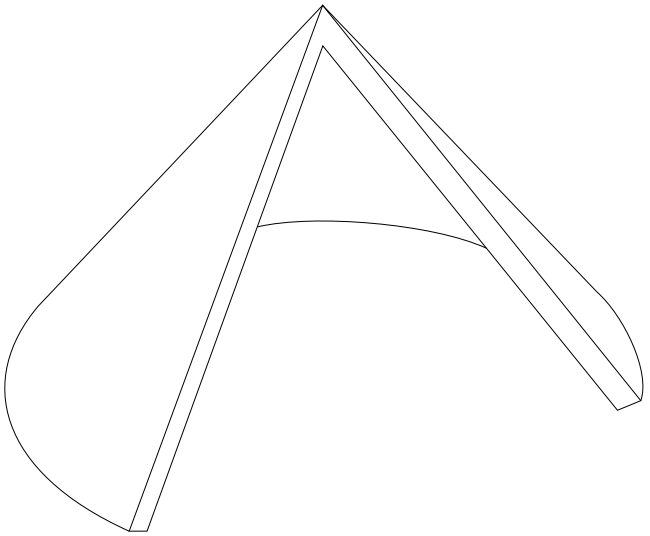
status = j_1 + 2*j_2, ここで、 各 jiフラグは0または1をとります.

j_1: オリジナルと厚みのあるサーフェスの辺を全て非表示にする。

j_2: サーフェスの辺を全て非表示にする。

例:

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



REVOLVEDSHELL{3}

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

REVOLVEDSHELL{3} は 「REVOLVEDSHELL{2}」コマンドの拡張版で、インライン材質定義できます。それはローカルのGDLスクリプトに材質を定義でき、グローバル材質定義で定義された材質と一緒に使用できます。

REVOLVEDSHELLANGULAR

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

「REVOLVEDSHELL」コマンドの角度変数。パラメータは、以下のその他のパラメータの追加と同じです。

segmentationType: 1または2でなければならない。
1: 値1は、 360度の回転がnOfSegmentsセグメントに分割されることを意味します。
2: 値2は、 実際の回転角度 （alphaパラメータにより指定）

nOfSegments: セグメント数、上記のsegmentationType パラメータを参照してください。

REVOLVEDSHELLANGULAR{2}

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

REVOLVEDSHELLANGULAR{2}はサーフェスの辺、オリジナルと厚みのあるサーフェスの辺を隠すことを可能にする「REVOLVEDSHELLANGULAR」 コマンドの拡張機能です。

status: ステータスビット：
status = j_1 + 2*j_2, ここで、 各 jiフラグは0または1をとります.
j_1: オリジナルと厚みのあるサーフェスの辺を全て非表示にする。

j_2: サーフェスの辺を全て非表示にする。

REVOLVEDSHELLANGULAR{3}

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

REVOLVEDSHELLANGULAR{3} は 「REVOLVEDSHELLANGULAR{2}」コマンドの拡張版で、インライン材質定義できます。それはローカルのGDLスクリプトに材質を定義でき、グローバル材質定義で定義された材質と一緒に使用できます。

RULEDSHELL

RULEDSHELL topMat, bottomMat,
sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
n, m, g,
offset, thickness, flipped, trimmingBody,
preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
firstpolyX_1, firstpolyY_1, firstpolyS_1,
...
firstpolyX_n, firstpolyY_n, firstpolyS_n,
secondpolyX_1, secondpolyY_1, secondpolyS_1,
...
secondpolyX_m, secondpolyY_m, secondpolyS_m,
profile2Tran_11, profile2Tran_12, profile2Tran_13, profile2Tran_14
profile2Tran_21, profile2Tran_22, profile2Tran_23, profile2Tran_24
profile2Tran_31, profile2Tran_32, profile2Tran_33, profile2Tran 34
generatrixFirstIndex_1, generatrixSecondIndex_1,
...
generatrixFirstIndex_g, generatrixSecondIndex_g

2つのポリラインを接続することによって作成されるサーフェス。

topMat, bottomMat, sideMat_1, sideMat_2, sideMat_3, sideMat_4: オブジェクトの上面、 下面、 および側面の材質。

defaultMat: オブジェクトの"内側"の材質のインデックス。 この材質は、 オブジェクトを平面で切り取った場合などに、 切断面で可視化されます。

n: 最初の断面形状ベースのポリラインの頂点の数

m: 2番目の断面形状ベースのポリラインの頂点の数

g: 母線の数

offset: シェルの厚さのオフセット。負数にできません。

thickness: シェルの厚さ

flipped:
1: シェルを反転する場合は1、
0: 閉じない場合は0

preThickenTran: 厚くなる前に実行される変換。パラメータの意味については、「XFORM」コマンドを参照してください。

trimmingBody:
1: シェルを切り取り目的で閉じる場合は1、
0: 閉じない場合は0

firstpolyX, firstpolyY, firstpolyS: 最初の断面形状ベースのポリラインのXおよびY座標とステータス値。 詳細は、「REVOLVE」を参照してください。

secondpolyX, secondpolyY, secondpolyS: 2番目の断面形状ベースのポリラインのXおよびY座標とステータス値。詳細は、「REVOLVE」を参照してください。

profile2Tran: 2番目の断面形状で実行される変換。 この変換を使用して、 最初の断面形状を基準として2番目の断面形状を配置します。パラメータの意味については、「XFORM」コマンドを参照してください。

generatrixFirstIndex, generatrixSecondIndex: 頂点のペア。 最初のポリラインの節点番号と、 2 番目のポリラインの節点番号。 指定した番号を持つ頂点は線で接続されます。

RULEDSHELL{2}

RULEDSHELL{2} topMat, bottomMat,
sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
n, m, g, status,
offset, thickness, flipped, trimmingBody,
preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
firstpolyX_1, firstpolyY_1, firstpolyS_1,
...
firstpolyX_n, firstpolyY_n, firstpolyS_n,
secondpolyX_1, secondpolyY_1, secondpolyS_1,
...
secondpolyX_m, secondpolyY_m, secondpolyS_m,
profile2Tran_11, profile2Tran_12, profile2Tran_13, profile2Tran_14
profile2Tran_21, profile2Tran_22, profile2Tran_23, profile2Tran_24
profile2Tran_31, profile2Tran_32, profile2Tran_33, profile2Tran 34
generatrixFirstIndex_1, generatrixSecondIndex_1,
...
generatrixFirstIndex_g, generatrixSecondIndex_g

RULEDSHELL{2} はサーフェスの辺、オリジナルと厚みのあるサーフェスの辺を隠すことを可能にする 「RULEDSHELL」 コマンドの拡張機能です。

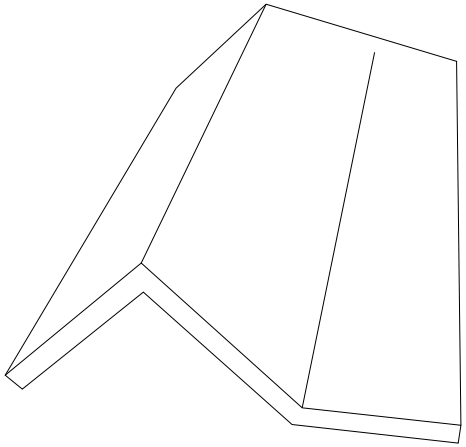
status: ステータスビット：

status = j_1 + 2*j_2, ここで、 各 jiフラグは0または1をとります.

j_1: オリジナルと厚みのあるサーフェスの辺を全て非表示にする。

j_2: サーフェスの辺を全て非表示にする。

例:



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

RULEDSHELL{3}

RULEDSHELL{3} topMat, bottomMat,
sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
n, m, g, status,
offset, thickness, flipped, trimmingBody,
preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
firstpolyX_1, firstpolyY_1, firstpolyS_1,
...
firstpolyX_n, firstpolyY_n, firstpolyS_n,
secondpolyX_1, secondpolyY_1, secondpolyS_1,
...
secondpolyX_m, secondpolyY_m, secondpolyS_m,
profile2Tran_11, profile2Tran_12, profile2Tran_13, profile2Tran_14
profile2Tran_21, profile2Tran_22, profile2Tran_23, profile2Tran_24
profile2Tran_31, profile2Tran_32, profile2Tran_33, profile2Tran 34
generatrixFirstIndex_1, generatrixSecondIndex_1,
...
generatrixFirstIndex_g, generatrixSecondIndex_g

RULEDSHELL{3} は 「RULEDSHELL{2}」 コマンドの拡張版で、インライン材質定義できます。それはローカルのGDLスクリプトに材質を定義でき、グローバル材質定義で定義された材質と一緒に使用できます

ビジュアル化のための要素

LIGHT

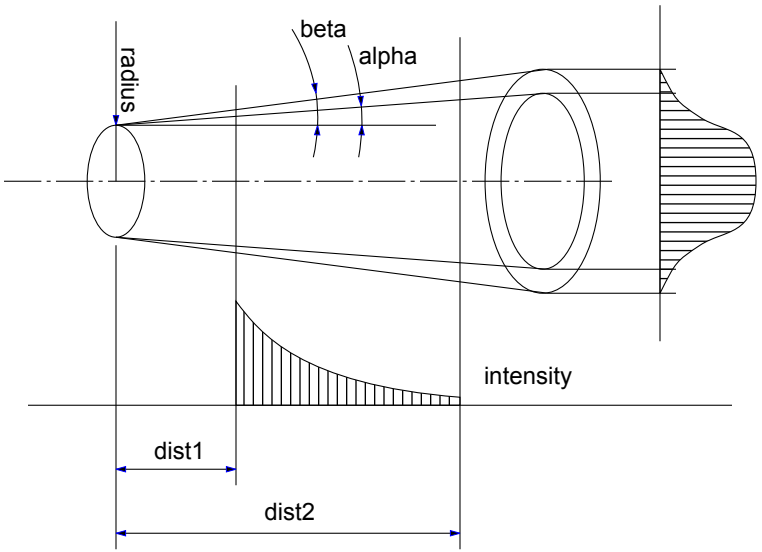
LIGHT red, green, blue, shadow,
radius, alpha, beta, angle_falloff,
distance1, distance2,
distance_falloff [[,] **ADDITIONAL_DATA** name1 = value1,
name2 = value2, ...]

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。

shadow: 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



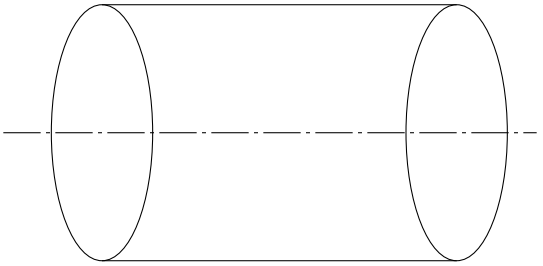
パラメータの制限:

alpha <= beta <= 80°

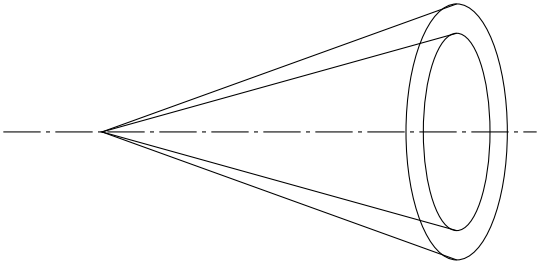
次のようなパラメータの組み合わせには、特別な意味があります。

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。

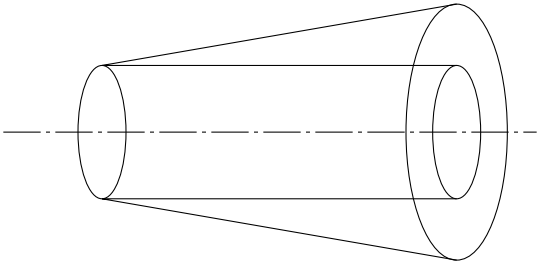
radius > 0, alpha = 0, beta = 0: 平行光線による指向性のある光



r = 0, alpha > 0, beta > 0: 円錐状光線による指向性のある光



r > 0, alpha = 0, beta > 0: 平行光線と円錐状の減光による指向性のある光



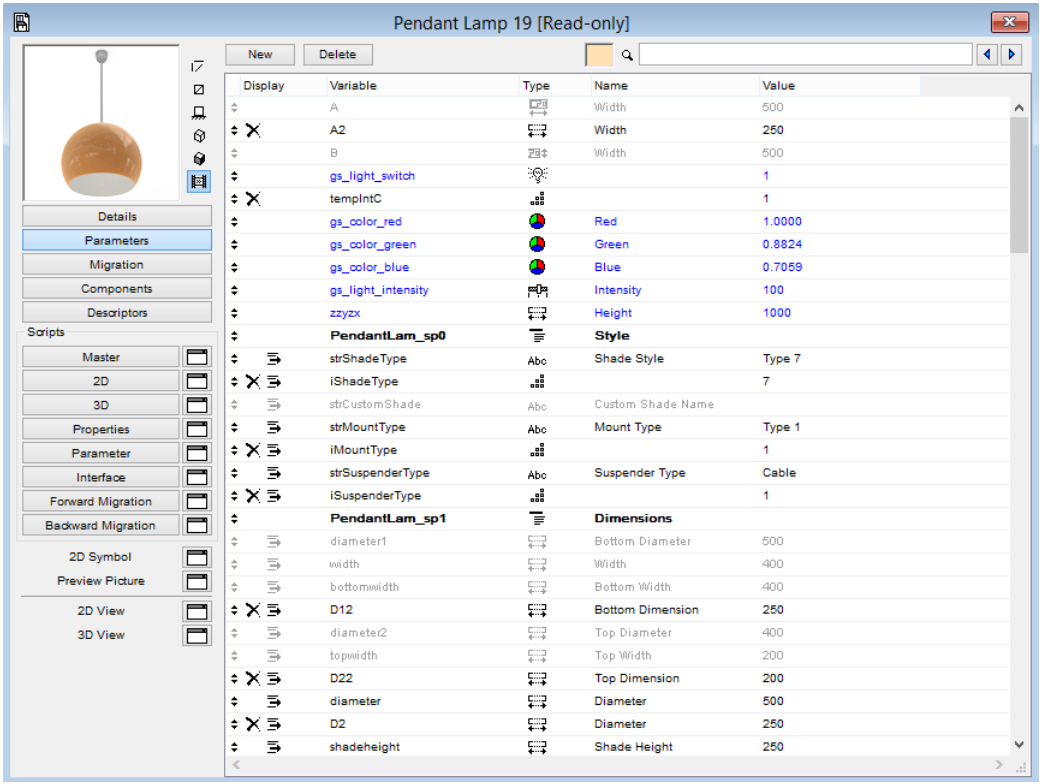
光源には、ADDITIONAL_DATAキーワードの後に省略可能な追加データ定義を含めることができます。 追加データには、名前（namei）と値（valuei）があり、これはいずれかのタイプの式にすることができます。また、配列にすることもできます。 パラメータ名がサブ文字列「_file」で終わっている場合、その値はファイル名とみなされてアーカイブプロジェクトに含められます。
異なる意味の付加データを実行中のアプリケーションから定義し、使用することができる。

例 1:

```
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
```

例 2:

ARCHICADの光に関するライブラリ部品ダイアログボックス



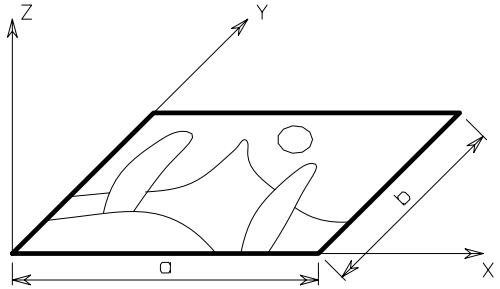
対応するGDLスクリプトの一部：

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

PICTURE

PICTURE expression, a, b, mask

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。

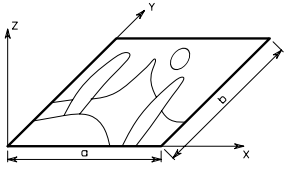
インデックス付きの画像参照は、属性が現在の属性セットに結合されているときには、MASTER_GDLスクリプトでは使用できません。 画像は、ほかのいずれかの3D投影方法でRECTとして扱われる矩形にはめ込まれます。

mask: alpha + distortion

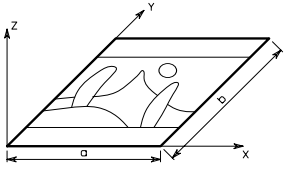
alpha: アルファチャンネルの制御
- 0: アルファチャンネルは使わず、 画像は矩形
- 1: アルファチャンネルを使い、 画像の一部は透過も可

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

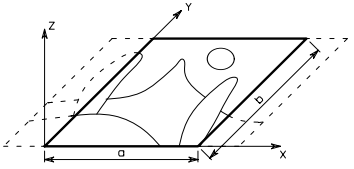
distortion=0



distortion=2



distortion=4

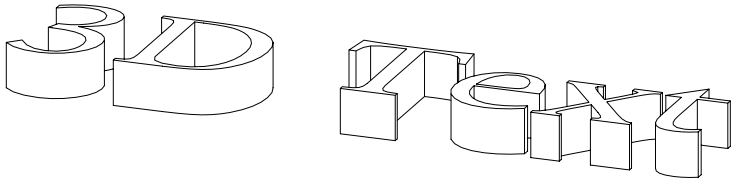


3Dテキスト要素

TEXT

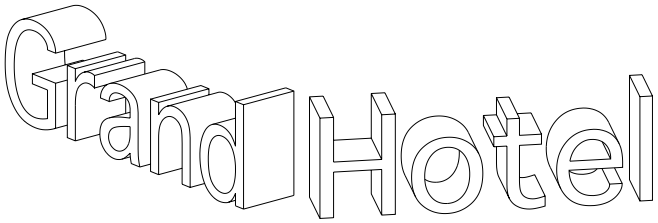
TEXT d, 0, expression
文字列タイプまたは数値タイプの式の値の現在のスタイルでの3D表現。
「[SET] STYLE」および「DEFINE STYLE」を参照してください。
d: 文字の厚さ（メートル単位）
GDLの現在のバージョンでは、2番目のパラメータは常にゼロです。
注記: 2D GDLスクリプトとの互換性を保つため、DEFINE STYLEコマンドでは、文字の高さは常にミリメートルで解釈されます。

例 1:



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

例 2:



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

RICHTEXT

RICHTEXT x, y,
height, 0, textblock_name

あらかじめ定義されたTEXTBLOCKの3D表現。 詳細は、「TEXTBLOCK」を参照してください。

x, y: リッチテキストの位置の座標

height: 文字の厚さ（メートル単位）

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。 ボディは複数のサーフェスと、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、負数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

隣接するポリゴンの辺による検索は、「BODY」コマンドが実行されている間に行われます。

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。

VERT{2}

VERT x, y, z, hard

「VERT」 の拡張機能はノードをハード頂点として宣言することができます。 ハード頂点は滑らかなサーフェスをレンダリングする際の、区切りを定義します。

x, y, z: 節点の座標

hard:

1: もし滑らかなサーフェスのレンダリングする際には、頂点を区切りとして定義する必要があります。
0: 閉じない場合は0

TEVE

TEVE x, y, z, u, v

テクスチャの座標定義を含む、「VERT」コマンドの拡張版。 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」コマンドの代わりに使用することができます（「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: 端点のインデックス。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

status: ステータスビット：
status = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 262144*j_{19}, ここで、 各 jiフラグは0または1をとります.
j_1: 非表示の辺
j_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：
j_3: 曲面の最初の辺 （j2と共にのみ使用）、
j_4: 曲面の最後の辺 （j2と共にのみ使用）、
j_5: 円弧線分の辺、
j_6: 円弧の最初の線分 （j4と共にのみ使用）、
j_7: 円弧の最後の線分 （j4と共にのみ使用）、
j_{19}: 2つの曲線ポリゴン間の鋭いエッジをレンダリング（j2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。

edge1, edge2, ..., edgen: 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反転します。 （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）。

status: ステータスビット：

status = j_1 + 2*j_2 + 16*j_5 + 32*j_6 + 64*j_7 + 4*j_3 + 8*j_4, ここで、 各 jiフラグは0または1をとります.

j_1: 非表示のポリゴン
j_2: 曲面のポリゴン
j_5: 凹形ポリゴン
j_6: 穴の開いたポリゴン
j_7: 凸状の穴 （j6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：

j_3: 曲面の最初のポリゴン （j2と共にのみ使用）
j_4: 曲面の最後のポリゴン （j2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。

PGON{2}

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

最初の3つのパラメータは「PGON」のパラメータと同じです。

wrap: ラッピングモード+投影タイプ
0: グローバルラッピングモードが適用されます。
> 0: 意味は「COOR」の場合と同じです。

edge_or_wrap1, ..., edge_or_wrapn: これらのパラメータの番号と意味は次のラップ定義に基づいています。
edge1, ..., edgen: ラップが0の場合：edgenは「PGON」の場合と同じ意味で、 グローバルに定義されたテクスチャマッピングが適用されます。
x1, y1, z1, x2, y2, z2, x3, y3, z3, x4, y4, z4, edge1, ..., edgen: ラップのラッピングモードが0でない場合： xi, yi, zi 座標がポリゴンに対するテクスチャマッピングの座標系を定義します。
edge1, u1, v1, ..., edgen, un, vn: ラッピングモードが0でラップの投影タイプが0でない場合： ui, viテクスチャ座標は「TEVE」の場合と同じです。 マッピングは現在定義されているポリゴンのみに影響します。

PGON{3}

PGON{3} n, vect, status, wrap_method, wrap_flags, edge_or_wrap1, ..., edge_or_wrapn
このパラメータは「PGON{2}」と同様ですが、wrapは異なり、次の2つのパラメータに分割されます： wrap_methodおよびwrap_flags。意味は「COOR{2}」の場合と同じです。

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」ステートメントと同じです; 残りは 「PGON」ステートメントと同じです。

COOR

COOR wrap, vert1, vert2, vert3, vert4
廃止されました。「COOR{3}」を参照してください。
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

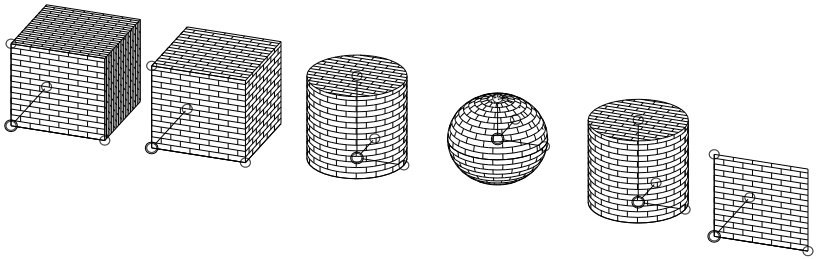
Wrapping modes:
1: 平面ボックス状（廃止予定）
2: ボックス状
3: 円柱状
4: 球体状
5: 円柱状塗りつぶしマッピングと同じ、 ただし上面と下面のレンダリングでは円状マッピング
6: 平面状

Projection types:
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
4096: 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（「TEVE」を参照してください）。

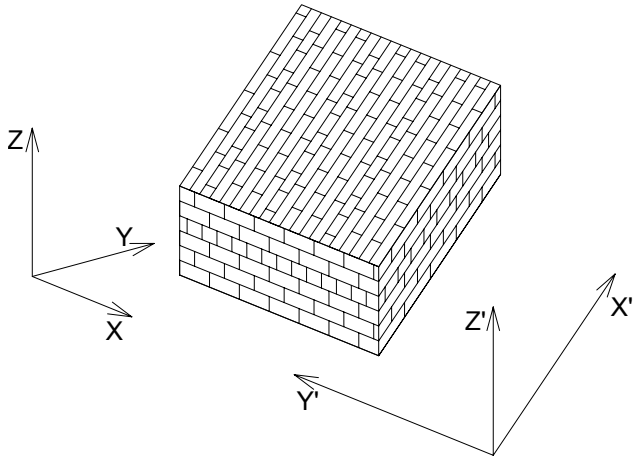
vert1: ローカル座標系の原点を表すVERTのインデックス

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。



例: カスタムテクスチャ軸の：

```
CSLAB_ " レンガ - 赤 ", " レンガ - 赤 ", " レンガ - 赤 ",
        4, 0.5,
        0, 0, 0, 15,
        1, 0, 0, 15,
        1, 1, 1, 15,
        0, 1, 1, 15
BASE
VERT 1, 0, 0 !#1
VERT 1, 1, 1 !#2
VERT 0, 0, 0 !#3
VERT 1, 0, 1 !#4
COOR 2, -1, -2, -3, -4
BODY 1
```



COOR{2}

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

廃止されました。「COOR{3}」を参照してください。

「COOR」と同様ですが、wrapがwrap_methodおよびwrap_flagsの2つのパラメータに分割され、機能が拡張されました。

wrap_method: ラッピングモードは、「COOR」に記載の内容と同じです。投影タイプを適用する代わりにwrap_flagsを使用します。

wrap_flags: Wrapping flags

wrap_flags = $4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8$, ここで、 各 jiフラグは0または1をとります.

j_3: 二次テクスチャ投影（推奨）

j_4: 平均距離に基づく線形テクスチャ投影

j_5: 通常の三角形分割に基づく線形テクスチャ投影

j_8: テクスチャの原点をそれぞれx、y、z方向のグローバル原点の最も近い座標システムに変換。例： j6は原点をx軸方向（v2 - v1ベクトルに沿って）に変換し、x軸の線に対してのグルーバル原点の平行投影になります。すなわち、もし全てのj6、j7、j8が1の場合、原点はグローバル原点に変換されます（投影タイプが256、「COOR」の場合と同じことです）。

注記: j3、j4、j5フラグはwrap_methodが0の場合のみ有効で、そのうち一つだけが1にできます。j6、j7、j8フラグはwrap_methodが0でない場合のみ有効です。これらは、任意の組み合わせで同時に1であることができます。

vert1, vert2, vert3, vert4: 「COOR」と同様。

COOR{3}

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

互換性：ARCHICAD 20で導入されました。

「COOR{2}」と同様。配列パラメータ入力で使用できます（詳細は「壁パラメータ - ドア/窓、リストおよびラベルに使用可能」のWALL_TEXTURE_WRAPを参照）。

投影形状の座標系はCOOR{3}コマンド自体に含まれており、現在のボディに頂点を追加で定義する必要はありません。 NURBS形状と互換性を持ちます（非NURBSプリミティブにはテクスチャ座標系を設定する必要はありません）。

wrap_method: ラッピングモードは、NURBSベースのラッピングモードが追加された「COOR」に記載の内容と同じです。投影タイプを適用する代わりにwrap_flagsを使用します。
1: 平面ボックス状（廃止予定）
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
6: 平面状
7: NURBSベース、NURBS形状の場合のみ、頂点のテクスチャ座標は、材質パラメータから取得されます。

wrap_flags: Wrapping flags
wrap_flags = $4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8$, ここで、 各 jiフラグは0または1をとります.
j_3: 二次テクスチャ投影（推奨）
j_4: 平均距離に基づく線形テクスチャ投影
j_5: 通常の三角形分割に基づく線形テクスチャ投影
j_8: テクスチャの原点をそれぞれx、y、z方向のグローバル原点の最も近い座標システムに変換。例： j6は原点をx軸方向に変換し、x軸の線に対してのグローバル原点の平行投影になります。すなわち、もし全てのj6、j7、j8が1の場合、原点はグローバル原点に変換されます（「COOR」の投影タイプ256の逆の効果）。

注記: j3、j4、j5フラグはwrap_methodが0の場合のみ有効で、そのうち一つだけが1にできます。j6、j7、j8フラグはwrap_methodが0でない場合のみ有効です。これらは、任意の組み合わせで同時に1であることができます。

origin_X, origin_Y, origin_Z: 3つの座標、テクスチャ原点によって定義された、X-Y-Z空間内の節点。

endOfX_X, endOfX_Y, endOfX_Z: 3つの座標、テクスチャマッピングのX方向によって定義された、X-Y-Z空間内の節点。

endOfY_X, endOfY_Y, endOfY_Z: 3つの座標、テクスチャマッピングのY方向によって定義された、X-Y-Z空間内の節点。

endOfZ_X, endOfZ_Y, endOfZ_Z: 3つの座標、テクスチャマッピングのZ方向によって定義された、X-Y-Z空間内の節点。

例: COOR{3}および同等のCOOR{2}パラメータ設定

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

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

BODY

BODY status

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。

status: ステータスビット：

status = j_1 + 2*j_2 + 4*j_3 + 32*j_6 + 64*j_7, ここで、 各 jiフラグは0または1をとります.

j_1: 閉じたボディ（廃止予定）,

j_2: 曲面 （1つ以上） を含むボディ（廃止予定）,

j_3: 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

j_6: 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

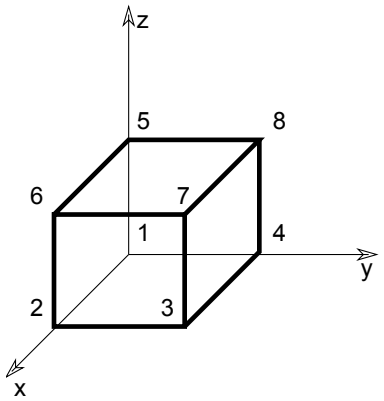
j_7: ボディはシャドウ投射を行わない

j6とj7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。

「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。

NURBSプリミティブ要素

NURBS形状の3Dデータ構造のプリミティブは、「NURBSCURVE2D」、「NURBSCURVE3D」、「NURBSSURFACE」、「NURBSVERT」、「NURBSEDGE」、「NURBSTRIM」、「NURBSTRIMSINGULAR」、「NURBSFACE」、「NURBSLUMP」、「NURBSBODY」です。

NURBSのソリッド形状は、ソリッド領域のNURBS境界面によって表され、NURBSのサーフェス形状はNURBS面自体で表され、NURBSのワイヤ形状はNURBSエッジで表されます。 NURBS形状は、ソリッド、サーフェス、ワイヤの部分を同時に持つことができ、NURBS形状自体はソリッド/サーフェス/ワイヤのどのカテゴリにも分類されません。

NURBSプリミティブは平面形状では使用できず、非NURBSプリミティブはNURBS形状では使用できません。 非NURBSプリミティブステートメントを開始すると、作成中のNURBS形状は終了し、新しい非NURBS形状が開始します（暗黙的なBODYおよびNURBSBODYステートメント）。

同様に、NURBSプリミティブステートメントを開始すると、作成中の非NURBS形状は終了し、新しいNURBS形状が開始します。複合ステートメント（BRICK、CYLIND、PRISMなど）またはMODELステートメントを開始すると、作成中のNURBS形状または非NURBS形状は終了します。 NURBSBODYステートメントによって非NURBS形状が終了した場合、またはBODYステートメントによってNURBS形状が終了した場合、指定されたステータス値は無効になります。

NURBSプリミティブのインデックスは1から開始します。NURBSプリミティブと非NURBSプリミティブ（VERT、TEVE、EDGE、VECT、PGON、PIPG）のインデックスは個別に扱われます。 BASEステートメントは、NURBSプリミティブのカウンタもリセットします。 別のプリミティブで参照されるプリミティブは全て、参照するプリミティブを定義する前に定義しておく必要があります（例えば、エッジの頂点や3D曲線はエッジよりも前に定義しておく必要があります）。

NURBSCURVE2D、NURBSCURVE3D、NURBSSURFACEの各ステートメントは、それ自体は不可視である、NURBS形状内の幾何学的要素のみを作成します。 NURBSエッジは、3D NURBS曲線を参照することでその幾何学的サポートを定義し、同様にNURBSトリムは2D NURBS曲線を、NURBS面はNURBSサーフェスをその幾何学的サポートとして参照します（エッジ、トリムおよび面は幾何学的サポート全体には拡張できません。詳細は各コマンドの説明を参照してください）。

NURBSエッジ、その3D曲線、そのトリム、およびトリムの2D曲線は、常に同じ方向に向きます。 NURBS面とそのサーフェスは、常に同じ方向に向きます。

NURBS面でNURBS立体を構成することができます。立体は、1つまたは複数のシェルによって境界されるソリッド領域を定義したものです。シェルは、閉じて連結された面の集合体で、空間を2つの領域に分割します。立体には、自身を無限の空間から分離する外側シェルがあり、また立体を内側の空洞から分離する空洞シェルを持つ場合もあります。

シェルの面の方向は同じである必要はなく、2つの隣接する面は同じ方向の同じエッジを参照できます。ただし、立体のシェルでは同じ方向にする必要があり、シェルの裏側が立体の内側に向く必要があります。このため、立体は逆方向の面を負数で参照できます。

立体の一部でない面は、たとえその面が閉じたシェルを形成している場合でも、サーフェスとして扱われます。面の一部でないエッジは、ワイヤエッジとして扱われます。1つのNURBS形状には、ソリッドな立体、面、ワイヤエッジを同時に含めることができます。

NURBS形状に2次元多様体のプロパティは必要なく、NURBSエッジは2つ以上の面に接続できます（2つ以上のトリムによって）。シェルが空間を2つの領域に分割する限り、NURBS立体のシェルは1つのエッジで2つ以上の面を持つことができます（つまり、各エッジで指定されたシェルの面の数）。

RADIUS、RESOL、およびTOLERステートメントは、NURBS面およびエッジの滑らかさには影響しません。NURBSプリミティブの滑らかさは自動的に計算され、NURBS形状についてはNURBSBODYコマンドのパラメータによって制限できます（詳細はNURBSBODYを参照）。

NURBSの正しいテクスチャ設定については、「COOR{3}」を参照してください。

NURBS面のトリミング

NURBSサーフェスは3次元空間内の2次元シートで、長方形を空間にマッピングする幾何学的な機能によって定義されます。 NURBS面のジオメトリは常にNURBSサーフェスの一部ですが、それよりもさらに複雑にすることができます。これはトリムによって作成できます。

トリムは、サーフェスの長方形領域での切断、2次元のNURBS曲線による切断を定義します。これは、サーフェスの3次元シートでの切断を意味します。この切断は、面の境界となるNURBSエッジに沿って行われ、サーフェスシートに沿った切断のジオメトリがNURBSエッジのジオメトリに一致する必要があります。

NURBS面は従来のPGONと同じような輪郭を持ちますが、輪郭はNURBSエッジのリストではなく、NURBSトリムのリストです。これは、面を適切に切断するために必要な情報を持つのがトリムであるためです （トリムの2次元曲線はエッジの3次元曲線から計算されますが、これは自己交差サーフェスや特異なサーフェスの場合、またはデータに誤りがある場合には、不正確になったり不明瞭になることさえあります）。

NURBSジオメトリコマンド

以下のコマンドは、NURBS要素の幾何学的な部分である 曲線とサーフェスを記述します。

NURBSCURVE2D

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

NURBSCURVE3D

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

指定された次数、ノットベクトル、制御点、ウェイトを持つ2次元および3次元NURBS曲線。

degree: NURBS曲線の次数、曲線の階数よりも1つ少ない数（階数 = 次数 + 1）、正数

nControlPoints: 制御点の数（n）、曲線の次数よりも大きい（ 階数以上）

knot_i: インデックスiのノット値
- ノット値の数（m、ノットベクトルのサイズ）は、次のように指定されます。m = 次数 + 1 + n
- ノットは昇順にします（knot_i <= knot_{i+1}）。
- 同じノット値が許可されます。多重度は次数まで、最初と最後のノットの場合は次数+1までです。

cPoint_i_x, cPoint_i_y, cPoint_i_z: インデックスiの制御点の座標

weight_i: インデックスiの制御点のウェイト、正数

周期曲線は個別に扱われず、幾何学的に閉じて終端で適切に連続的に接続されたフローティング（ 固定でない）NURBS曲線として記述されます。これは次のように終端で十分な数の制御点とノット間隔を繰り返すことで実現されます。

- 最後の次数個の制御点は、最初の次数個の制御点の複製です（逆の順序でない）。
- ノットベクトルの最初の次数の2倍の数のノット区間（knot_1-knot_0、knot_2-knot_1、...）が、最後のものと同じです（これらは最初（または最後）の次数個の制御点と接続されるノットです）。

使用可能な曲線領域は、knot_{次数 + 1}とknot_{m - 次数}の間の閉区間です。

NURBSSURFACE

NURBSSURFACE degree_u, degree_v, nu, nv,
knot_u_1, knot_u_2, ..., knot_u_mu,
knot_v_1, knot_v_2, ..., knot_v_mv,
cPoint_1_1_x, cPoint_1_1_y, cPoint_1_1_z, weight_1_1,
cPoint_1_2_x, cPoint_1_2_y, cPoint_1_2_z, weight_1_2,
...,
cPoint_1_nv_x, cPoint_1_nv_y, cPoint_1_nv_z, weight_1_nv,
cPoint_2_1_x, cPoint_2_1_y, cPoint_2_1_z, weight_2_1,
...,
cPoint_nu_nv_x, cPoint_nu_nv_y, cPoint_nu_nv_z, weight_nu_nv

u-vパラメータ空間、指定された次数、uおよびv方向のノットベクトル、指定された制御点、 正味ウェイトを持つ3次元NURBSサーフェス。 次数はサーフェスの階数よりも1つ少ない数で（order_u = degree_u + 1）、正数です。

degree_u: サーフェスのuパラメータ方向の次数

degree_v: サーフェスのvパラメータ方向の次数

nu, nv: uおよびv方向の制御点の数、サーフェスの指定された方向の次数よりも大きい（階数以上）

knot_u_i, knot_v_i: インデックスiのuおよびv方向のノット値
- その数値（ノットベクトルのサイズ）は次のように指定されます。mu = degree_u + 1 + nu
- ノットは昇順にします（knot_u_i <= knot_u_{i+1}、not_v_i <= knot_v_{i+1}）
- 同じノット値が許可されます。多重度は次数まで、最初と最後のノットの場合は次数+1までです。

cPoint_i_j_x, cPoint_i_j_y, cPoint_i_j_z: 制御点ネット上の制御点、u方向のインデックスi、v方向のインデックスj

weight_i_j: 制御点cPoint_ijのウェイト、正数
サーフェスは、uまたはvのいずれかの方向、または両方向で周期的にすることができます。 周期サーフェスは個別に扱われず、幾何学的に閉じて終端で適切に連続的に接続されたフローティング（ 固定でない）NURBSサーフェスとして記述されます。これは曲線の場合と同じ方法で実現されます。
使用可能なサーフェス領域は、knot_u_{degree_u + 1}とknot_v_{degree_v + 1}およびknot_u_{mu - degree_u}とknot_v_{mv - degree_v}の間の閉区間のクロス積です。

NURBSトポロジコマンド

次のコマンドは、NURBS要素のトポロジ部分を記述します。

NURBSVERT

NURBSVERT x, y, z, hard, tolerance
頂点、NURBS形状のノード。 「VERT」によって作成される頂点とは異なり、これらとは別にインデックス付けされます。 NURBS形状のみで使用でき、平面形状では使用できません。

x, y, z: 頂点の座標

hard:
1: 滑らかなサーフェスをレンダリングする際に頂点を区切りとして定義する必要がある場合。
0: 上記以外

tolerance: NURBS頂点と、それにトポロジ的に接続されたその他のエントリ（NURBSエッジ、NURBS面）との間の幾何学的な最大距離。 負数の場合、許容値は事前定義されたデフォルト値です。

NURBSEDGE

NURBSEDGE vert1, vert2, curve, curveDomainBeg, curveDomainEnd, status, tolerance
NURBS形状のエッジ。 「EDGE」によって作成されるエッジとは異なり、これらとは別にインデックス付けされます。 NURBS形状のみで使用でき、平面形状では使用できません。

vert1, vert2: NURBSの最初の頂点と最後の頂点のgdlインデックス
- vert1とvert2は同じ値にすることができます。この場合、エッジはループエッジになります（曲線が閉じているか閉じた部分を持ちます）
- リングエッジの場合はvert1とvert2をゼロにすることができます（頂点を持たず、曲線は閉じているか閉じた部分を持ちます）

curve: エッジのジオメトリに対するNURBS曲線のgdlインデックス。正のインデックス、エッジの方向は常に曲線の方向に一致します。

curveDomainBeg, curveDomainEnd: エッジを幾何学的に表現した曲線部分の定義。 curveDomainEndはcurveDomainBegよりも大きい値で、この2つの値は同じではなく、両方の値が使用可能な曲線領域内にある必要があります。

status: エッジのステータス制御：

status = j_1 + 2*j_2 + 4*j_3, ここで、 各 jiフラグは0または1をとります.
j_1: 非表示のエッジ（j2が設定されていない場合のみ設定可能）。
j_2: 輪郭の場合のみエッジを表示（j1が設定されていない場合のみ設定可能）。
j_3: 滑らかなエッジ（エッジは滑らかなサーフェスをレンダリングする際に区切りを定義しません）。
j1とj2の両方が設定されている場合、エッジはエラーを生成し、NURBS形状全体が消えます。

tolerance: NURBSエッジと、それにトポロジ的に接続されたその他のエントリ（NURBS面）との間の幾何学的な最大距離。 負数の場合、許容値は事前定義されたデフォルト値です。

各エンドポイントで評価される曲線は、該当する頂点の位置と一致する必要があります。 エッジは頂点のないリングエッジにすることができます。この場合、[curveDomainBeg, curveDomainEnd]に制限されたエッジは閉じる必要があります。つまり、エッジは各エンドポイントで等しく評価されます。 頂点にはエッジをいくつでもアタッチできます。 NURBSエッジのカラーは、最後のPENステートメントで定義されます。

NURBSTRIM

NURBSTRIM edge, curve, curveDomainBeg, curveDomainEnd, tolerance

NURBSTRIMSINGULAR

NURBSTRIMSINGULAR vertex, curve, curveDomainBeg, curveDomainEnd, tolerance

面の境界エッジ。 面のサーフェスのパラメータ空間で面のトリミングに使用されます。 NURBSTRIMSINGULARは、サーフェスの特異な側に沿って使用されます（その側がサーフェス上の1つの点に縮小されます）。 面をエッジに（または特異な場合は頂点に）接続します。

edge: このトリムがアタッチされるNURBSエッジのgdlインデックス。正のインデックス、エッジとトリムは常に同じ方向を向きます。

vertex: このトリムがアタッチされるNURBS頂点のgdlインデックス（特異な場合）。

curve: 2D NURBS曲線のgdlインデックス。正のインデックス、曲線とトリムは常に同じ方向を向きます。 これは面のサーフェスの領域（u-vパラメータ空間）で定義されます。

curveDomainBeg, curveDomainEnd: トリムを幾何学的に表現した曲線部分の定義。 curveDomainEndはcurveDomainBegよりも大きい値で、この2つの値は同じではなく、両方の値が使用可能な曲線領域内にある必要があります。

tolerance: NURBSトリムの2D曲線と、それにトポロジ的に接続されたその他のエントリ（その他のNURBSトリム）との間の幾何学的な最大距離。 負数の場合、許容値は事前定義されたデフォルト値です。

[curveDomainBeg, curveDomainEnd]の区間に制限される曲線は、面の使用可能なサーフェス領域の範囲内に完全に収まる必要があります（指定された許容値で）。 NURBSTRIMSINGULARの場合、2D曲線は、面の使用可能なサーフェス領域（u-vパラメータ空間）の特異な側に沿って配置される必要があります。

制限された2D曲線とサーフェスの構成で3D曲線が提供され、この3D曲線が制限されたエッジの3D曲線と一致する必要があります。そのため、curveDomainBegとcurveDomainEndで評価される2D曲線は、該当する頂点の位置と一致する必要があります。 特異な場合では、2D曲線とサーフェスの構成で3D点が提供され、これが指定された頂点と一致する必要があります。

特異なトリムと非特異なトリムのインデックスは共通です。

各エッジを参照できるトリムの数に制限はありません（結果的にエッジには面をいくつでもアタッチできます）。 エッジは非2次元多様体にすることができます。

1つのエッジ上の2つのトリムが同じ面に属する場合があります。この場合、エッジはシームエッジと呼ばれます。 例えば、円柱の外側面はシームエッジのある1つの面にすることができます。

NURBSFACE

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

NURBS形状の面。 「PGON」によって作成されるポリゴンとは異なり、これらとは別にインデックス付けされます。 NURBS形状のみで使用でき、平面形状では使用できません。

n: 境界エッジの数（オプションの穴分離記号のゼロを含む）。

surface: 面をサポートするNURBSサーフェスのgdlインデックス。 正のインデックス、面の方向はサーフェスの方向と常に同じです。

trimi: 面の境界となるNURBSトリムのgdlインデックス。

- トリムは、外側の輪郭ループのサーフェスでは反時計回り（正）、穴の輪郭ループでは時計回り（負）の順序でリストされます。
- ゼロの場合は、輪郭の最後を示します（穴分離記号）。
- 負のインデックスは、トリムと輪郭（面の）が反対方向であることを意味します。

tolerance: 負数の場合、許容値は事前定義されたデフォルト値です。

トリムは共通の頂点で接続する必要があります。トリムの最後の頂点は、面の次のトリムの最初の頂点と同じです （非特異なトリムの場合、トリムの頂点は、トリムのエッジの頂点です）。

2D曲線としての連続するトリムは、面の領域（パラメータ空間）にも接続し、そこに1つまたは複数の閉じた輪郭ループを定義します。 最初のループは常に外側のループで、平面上の無限の外側の領域と有限の内側の領域を分離します。 次のループは通常は穴の輪郭です。

各トリムの2D曲線は、面のサーフェスの使用可能な領域内に完全に収まる必要があり、自身や面の他のトリムの曲線と交差してはいけません。 各トリムは1つの面でのみ使用される必要があります。

面の素材と断面属性は、それぞれ最後のMATERIALとSECT_ATTRS（またはSECT_FILL）ステートメントで定義されます。 ポリゴンの分割のために作成された面の内側のエッジのカラーは、最後のPENステートメントによって定義されます。 これは、実際にはこの面の内部から出力されるシルエットに表示されます。

NURBSFACE{2}

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

「NURBSFACE」と同様に、NURBS面で「PGON{3}」のようにテクスチャマッピングを記述する機能を拡張

n, surface, tolerance: 「NURBSFACE」と同様

wrap_method: 「PGON{3}」と同様
- 0: グローバルラッピングモードを適用（x1 ... z4パラメータが必要ですが無視されます）
- > 0: 次と同様：「PGON{3}」

wrap_flags: 投影タイプフラグ（j3、j4およびj5）が無視されることを除いて「PGON{3}」と同様（テクスチャ座標をNURBS面に適用することはできません）

x1, y1, z1 ... x4, y4, z4: NURBS面のテクスチャマッピングの座標システムを定義する座標（これらのパラメータは、wrap_method > 0の場合のみ有効です）

trim1 ... trimn: 「NURBSFACE」と同様

NURBSLUMP

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

ソリッドなNURBS形状のソリッド部分、つまり幾何学的に接続されたサブセットを定義します。

n: 境界面の数（オプションの空洞分離記号ゼロを含む）。

facei: 立体の境界となるNURBS面のgdlインデックス
- ゼロの場合は、シェルの最後で別のシェルの最初であることを示します（空洞分離記号）。
- 負のインデックスは、面が反対方向に使用されていることを意味します。正のインデックスの場合、面の裏側が立体の内側に対応し、負のインデックスの場合、面の表側が内側を向きます。

立体の境界は、次のような複数の閉じたシェルに属する場合があります。立体を無限の外側の空間領域から分離する外側のシェル。立体を空洞領域から分離するゼロ以上の内側（空洞）のシェル。 1つのシェルの面は、連続した面リスト部分で構成する必要があります。別のシェルの別の部分は、ゼロ値で区切る必要があります。最初のシェルは外側のシェルにする必要があります。 シェルの面は、共通のエッジで接続する必要がありますが、リスト内での順番は考慮されません。

シェルの面が、シェル内にない、または別のシェルにある他の面に接続される場合があることに注意してください（エッジは2つ以上の面を持つことができるため）。 各面は、1つの立体でのみ使用する必要があります。 立体のどのシェルも開くことはできません。 開いた形状には立体もシェルもありません。

NURBSBODY

NURBSBODY shadowStatus, smoothnessMin, smoothnessMax
前記のNURBSプリミティブによって定義されたNURBS形状を作成します。

shadowStatus: シャドウ制御のステータス：
shadowStatus = 32*j_6 + 64*j_7, ここで、 各 jiフラグは0または1をとります.
j_6: 前に選択された自動アルゴリズムとは無関係に、NURBS形状は常に影を投影します。
j_7: NURBS形状は影を投影しません。
j6とj7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。 SHADOWコマンドを参照してください。

smoothnessMin, smoothnessMax: 形状のサーフェスと曲線のテッセレーションのために自動計算された滑らかさのパラメータの制限 。 自動計算されるパラメータは、常に0~1の範囲内で、smoothnessMin <= 0は下部限界がないこと、smoothnessMax >= 1は上部限界がないことを意味します。 smoothnessMin > smoothnessMaxの場合、これらの値は自動計算される滑らかさに影響しません。

非NURBSプリミティブステートメント（VERT、TEVE、EDGE、VECT、PGON、PIPG、BODY）または複合ステートメント（BRICK、CYLIND、PRISM、REVOLVEなど）はいずれも、作成中のNURBS形状を終了させます（暗黙的なNURBSBODYステートメント）。この場合、滑らかさの制限は設定されず、shadowStatusはゼロになります（BODYステートメントのステータスパラメータは渡されません）。

点群

POINTCLOUD

POINTCLOUD "data_file_name"
点群の3Dモデルを生成。点群は、各点ごと割り当てられている色やさまざまなメタデータを含む3D点の集合体です。

data_file_name: 点群データを含む、ロードされたライブラリ部品の名前 文字列式にする必要があります。

インターナル3Dエンジンでは点群が表示されません。2Dは投影され、必要のない点は3D切断面を使用してフィルタします。

3Dでの切り取り

CUTPLANE

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

CUTPLANE{2}

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

CUTPLANE{3}

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

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: 切り取る側面の定義
  0: 切断面の上の部分を削除（デフォルト）
  1: 切断面の下の部分を削除。x-y、 x-z、y-z の場合、軸の負の方向の部分を削除

status: 切断面の制御 ステータスが与えられなかった場合、ステータスは 1+2 に自動的に設定されます。
  status = j_1 + 2*j_2 + 4*j_3 + 256*j_9, ここで、 各 jiフラグは0または1をとります.
  j_1: 生成されたポリゴンと辺にボディの属性を使用
  j_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
  j_3: 生成された切断辺を非表示
  j_9: 切断面の頂点を削除されたものとして扱う

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。

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。

注記 1: 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。

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。

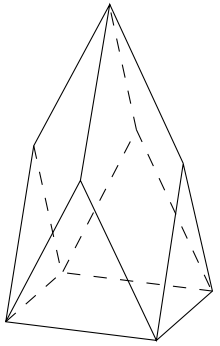
CUTPLANEとCUTENDの間でマクロがコールされた場合、マクロ内の形状が切り取られます。

注記 2: 2つ以上のパラメータを持つ CUTPLANE{2}を使用すると、これが CUTPLANEのように動作します。

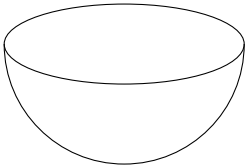
注記 3: CUTPLANE の代わりに CUTPLANE{3} を使用することをお勧めします。 5 つのパラメータを持つ CUTPLANE を使用すると、4 番目のパラメータが省略されます。 CUTPLANE{3}の場合、このパラメータは 5番目のパラメータに関係なく効果があります。

例 1:

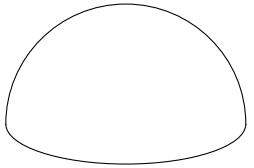
```
CUTPLANE 2, 2, 4
CUTPLANE -2, 2, 4
CUTPLANE -2, -2, 4
CUTPLANE 2, -2, 4
ADD -1, -1, 0
BRICK 2, 2, 4
DEL 1
CUTEND
CUTEND
CUTEND
CUTEND
```



例 2:

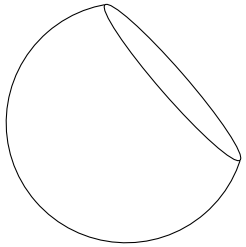


```
CUTPLANE
SPHERE 2
CUTEND
```

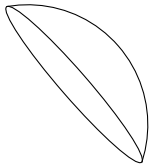


```
CUTPLANE 1, 1, 0, 1
SPHERE 2
CUTEND
```

例 3:

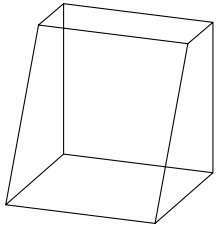


CUTPLANE 1.8, 1.8, 1.8
SPHERE 2
CUTEND

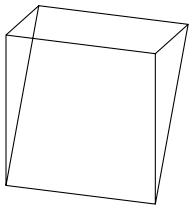


CUTPLANE 1.8, 1.8, 1.8, 1
SPHERE 2
CUTEND

例 4:



CUTPLANE 60
BRICK 2, 2, 2
CUTEND



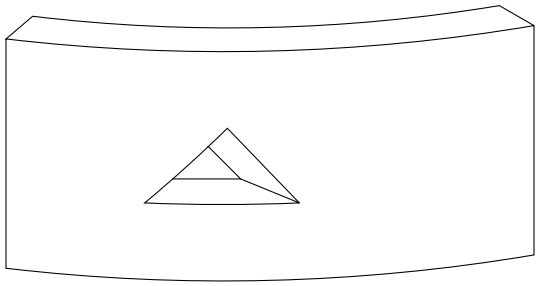
CUTPLANE -120
BRICK 2, 2, 2
CUTEND

CUTPOLY

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

「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。 ミラー変形は予期しない方法での切り取り方向に影響を及ぼします。より簡単な結果を得るには、 「CUTFORM」コマンドを使用します。

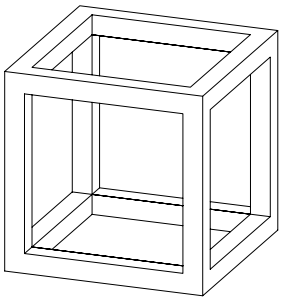
例 1:



```
ROTX 90
MULZ -1
CUTPOLY 3,
	0.5, 1,
	2, 2,
	3.5, 1,
	-1.8, 0, 1
DEL 1
BPRISM_ " レンガ - 赤 ", " レンガ - 赤 ", " レンガ - 通し目地 ",
	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
```

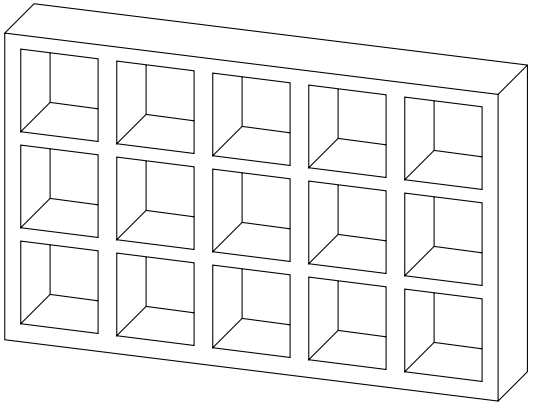
例 2:

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



例 3:

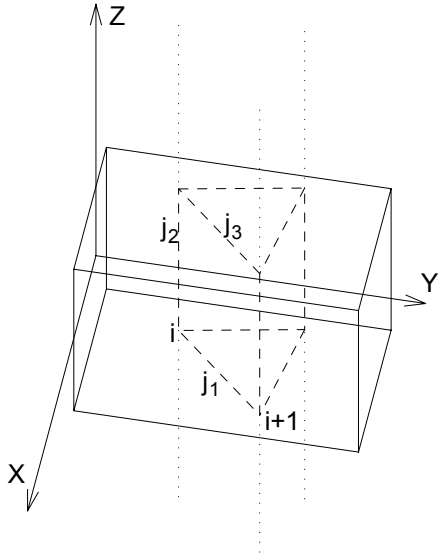
```
ROTX 90
FOR i=1 TO 3
    FOR j=1 TO 5
        CUTPOLY 4,
                0, 0, 1, 0,
                1, 1, 0, 1
        ADDX 1.2
    NEXT j
    DEL 5
    ADDY 1.2
NEXT i
DEL NTR()-1
ADD -0.2, -0.2, 0
BRICK 6.2, 3.8, 1
FOR k=1 TO 15
    CUTEND
NEXT k
DEL TOP
```



CUTPOLYA

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

「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。



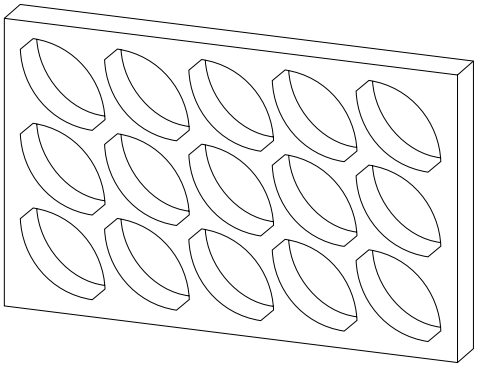
status: 生成された切り取りポリゴンの扱いを制御します。
1: 生成されたポリゴンと辺にボディの属性を使用
2: 生成された切り取りポリゴンは、 通常のポリゴンとして取り扱われる

d: ローカルの原点と半無限チューブの距離
0: の場合、無限チューブで切り取られます。

maski: 「PRISM_」コマンドと同様。

maski = j_1 + 2*j_2 + 4*j_3 + 64*j_7, ここで、 各 jiフラグは0または1をとります.

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

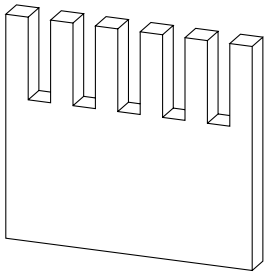
CUTSHAPE d [, status]
[statement1 statement2 ... statementn]
CUTEND

ブロックを"d"の厚さ、無限の長さ（y軸の両端）、半無限の高さ（xy平面の上）で切り取ります。

status: 生成された切り取りポリゴンの扱いを制御します。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 = j_1 + 2*j_2, ここで、 各 jiフラグは0または1をとります.
j_1: 生成されたポリゴンと辺にボディの属性を使用
j_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

例:

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

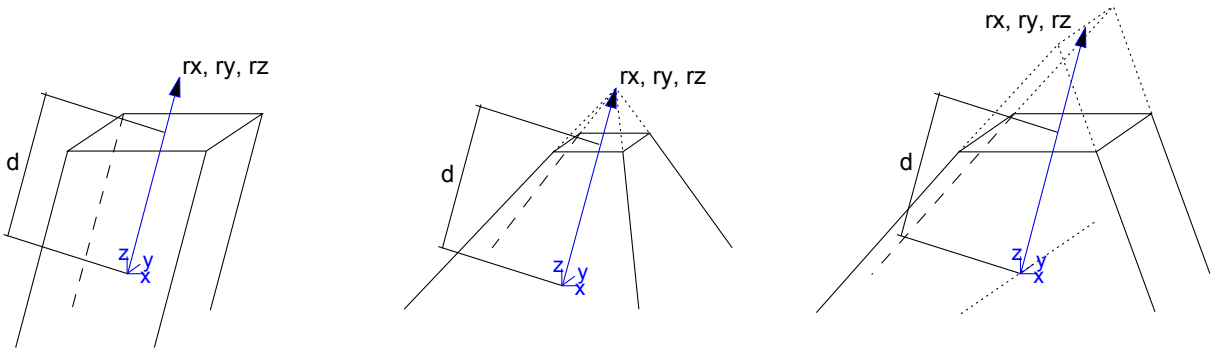


CUTFORM

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

「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: ウェッジ形状の切り取りボディ。 ウェッジ上端の方向は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 = j_1 + 2*j_2 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9, ここで、 各 jiフラグは0または1をとります.
j_1: 生成されたポリゴンと辺にボディの属性を使用
j_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
j_4: 切断の制限を定義（j5と共に）
j_5: 切断の制限を定義（j4と共に）
j_6: 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）。
j_7: 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。
j_8: 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。
j_9: 切り取り形状にはカスタム側面材質 （mati） があります。

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

rx, ry, rz: この3つの座標は、切り取りが角柱形状の場合は切り取り方向を定義します。 この3つの座標は、切り取りが角錐状の場合は角錐の上端を定義します。 切り取りがウェッジ形状の場合、rx-rz座標はウェッジの下端を定義し、ryは無視されます。

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。
mask = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 64*j_7, ここで、 各 jiフラグは0または1をとります.

j_1: 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（ウェッジ形状の切り取りボディで形状の切断をした場合以外），
j_2: 切り取り形式の縦方向の辺は可視
j_3: 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（ウェッジ形状の切り取りボディで形状の切断をした場合以外），
j_4: 切り取り形式の下辺は可視
j_5: 切り取り形式の上辺は可視
j_7: 縦方向の辺の視点に依存する可視性を制御
ウェッジ形の切断形状の値で固形体を切断する場合、Entry-エッジとExit-エッジの可視性のために（j1 および j3） が交換されます。この動作は、互換性のために保持されます。

mati: 切り取り形状の側面材質 （ステータスj9 = 1の場合）

CUTFORM{2}

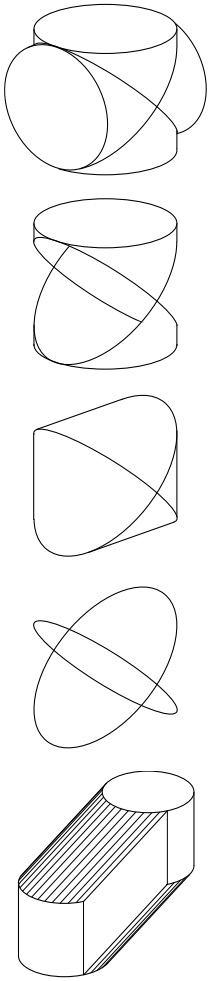
CUTFORM{2} n, method, status,
rx, ry, rz, d,
x1, y1, mask1 [, mat1],
...
xn, yn, maskn [, matn]

CUTFORM{2} は 「CUTFORM」 コマンドの拡張版で、インライン材質定義できます。それはローカルのGDLスクリプトに材質を定義でき、グローバル材質定義で定義された材質と一緒に使用できます

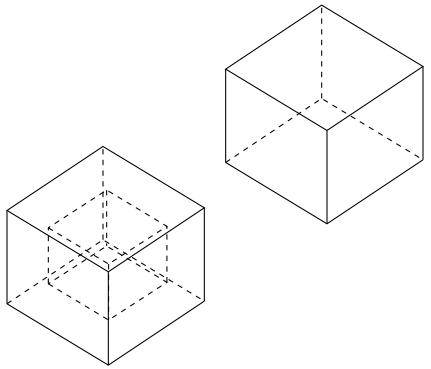
ソリッド図形コマンド

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つのアウターシェルがあり、空洞を含むこともできます （空洞は、立体内の負のインナーシェル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）。

「「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。

表1 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）

表2 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	サーフェスベース（影響なし）	サーフェスベース（影響なし）	ハイブリッドベース（影響なし）	ハイブリッドベース（影響なし）
ワイヤフレームツール	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	ハイブリッドベース（影響なし）	ハイブリッドベース（影響なし）	ハイブリッドベース（影響なし）	ハイブリッドベース（影響なし）

表3 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

表4 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

表5 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 - ENDGROUP

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

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。

ADDGROUP

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

SUBGROUP

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

ISECTGROUP

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

g_expr1: ベースグループのID

g_expr2: ツールグループのID

edgeColor: 0でない場合、新しい辺の色

materialId: 0でない場合、新しい面の材質

materialColor: materialIdが0で、materialColorが0でない場合、新しい面の色

operationStatus: 操作のステータスの制御。
operationStatus = j_1 + 2*j_2, ここで、 各 jiフラグは0または1をとります.
j_1: 生成された新しい辺を非表示
j_2: ツールグループの対応するポリゴンから継承された材質とテクスチャ投影の切り取りポリゴンの結果。

ISECTLINES

ISECTLINES (g_expr1, g_expr2)
グループ操作：和、差、積、交差線。 戻り値は新規のグループで、これは「PLACEGROUP」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。 ADDGROUP、ISECTGROUPおよびISECTLINES操作ではパラメータ設定は対称ですが、SUBGROUPはパラメータの順序が問題になることに注意してください。

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。

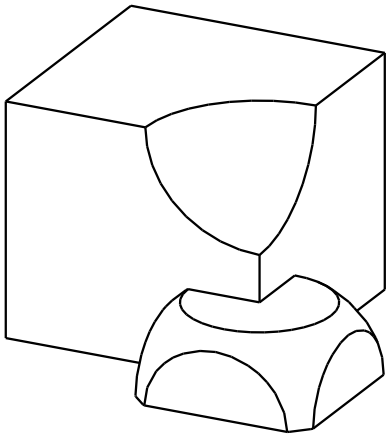
KILLGROUP

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。クリアしたグループの名前を同じスクリプトで再利用することはできません。 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
! "box" から "sphere" を減算
result_1=SUBGROUP("box", "sphere")
! "semisphere" と "brick" を交差
result_2=ISECTGROUP("semisphere","brick") ! 生成されたボディを加算
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 ）。

SWEEPGROUP{3} (g_expr, x, y, z, edgeColor, materialId, materialColor, method)
このバージョンは新しい方法の選択肢をSWEEPGROUP{2} に追加し、材質モデルにも利用できます。

edgeColor: 0でない場合、新しい辺の色

materialId: 0でない場合、新しい面の材質

materialColor: materialIdが0で、materialColorが0でない場合、新しい面の色

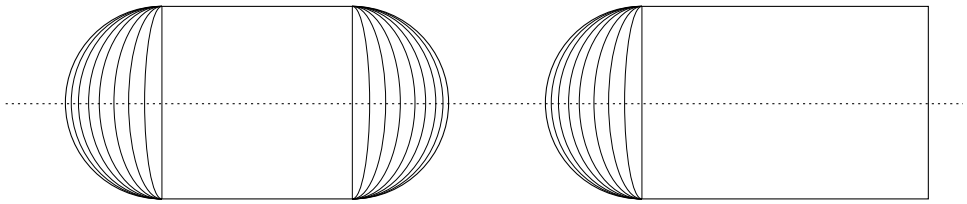
method: 結果ボディの終了形状を制御します。
0: SWEEPGROUP{2} と同じ。両端は発生源のボディから発生
1: 開始は発生源のボディから発生。スウィープエンドはフラット

SWEEPGROUP{4} (g_expr, x, y, z, edgeColor, materialId, materialColor, method, status)
このバージョンは新しいステータスパラメータSWEEPGROUP{3} に追加します。

status: 結果のコントロール属性。
status = $2^{*}j_2$, ここで、 各 jiフラグは0または1をとります.
j_2: 掃引の結果のポリゴンごとのテクスチャマッピングパラメータを保持する（詳細は、「PGON」を参照してください）。

SWEEPGROUP{5} (g_expr, x, y, z, edgeColor, materialId, materialColor, method, status)
SWEEPGROUP{5} はSWEEPGROUP{4} コマンドの拡張版で、インライン材質定義できます。それはローカルのGDLスクリプトに材質を定義でき、グローバル材質定義で定義された材質と一緒に使用できます。
互換性：ARCHICAD 22で導入されました。

例:



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

CREATEGROUPWITHMATERIAL

CREATEGROUPWITHMATERIAL (g_expr, repl_directive, pen, material)
g_exprグループ内のペン及び/又は材質を全て置換することによって作成されたグループを返します。

g_expr: ベースグループを識別するグループ式

repl_directive:
repl_directive = j_1 + 2*j_2 + 4*j_3 + 8*j_4, ここで、 各 jiフラグは0または1をとります.
j_1: ペンを置換
j_2: 材質を置換
j_4: サーフェスの辺を全て非表示にする。

pen: 置換ペンのインデックス

material: 置換材質のインデックス

バイナリ3D

BINARY

BINARY mode [, section, elementID]

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）
0: 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。
1: 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。
2-16: 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。

elementID: バイナリ部品の要素ID。このパラメータはインポート中に生成されました。

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モデルを既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。

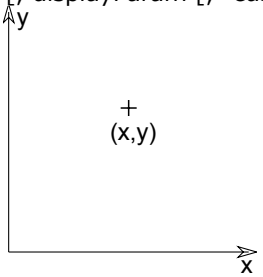
2D形状

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 [, "customDescription"]]]]]



unID: 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。

customDescription: 情報パレット内の表示されたパラメータのカスタム説明文字列。 このオプションを使用すると、displayParamも設定する必要があります（デフォルトではparamReferenceを使用してください）。

HOTSPOT2の使用については、ホットスポットベースの編集コマンドを参照してください。

HOTLINE2

HOTLINE2 x1, y1, x2, y2, unID

2点間のステータス線。ステータス線は、インテリジェントカーソルで認識されるがそれ自体は表示されない線です。 連動寸法のために、ユニークIDでかまいません。

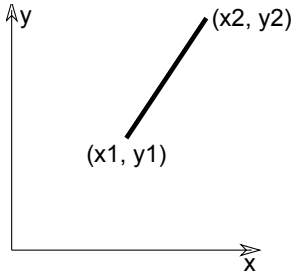
HOTARC2

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

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。 ステータス円弧は、インテリジェントカーソルで認識されるがそれ自体は表示されない円弧です。 連動寸法のために、ユニークIDでかまいません。

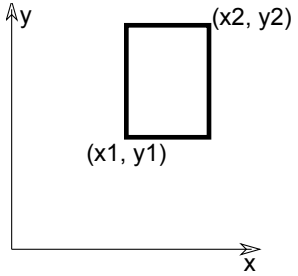
LINE2

LINE2 x1, y1, x2, y2
2点間の線。



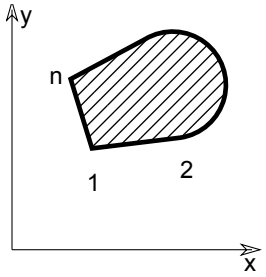
RECT2

RECT2 x1, y1, x2, y2
2つの節点による矩形定義。2つの点は長方形の対角線上にあり、側面は現在のX軸およびY軸に平行です。



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

n: 頂点数

x1, y1, ..., xn, yn: 各頂点の座標

frame_fill:

frame_fill = j_1 + 2*j_2 + 4*j_3, ここで、 各 jiフラグは0または1をとります.

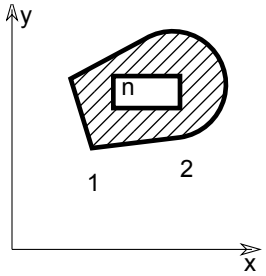
j_1: 輪郭線のみ

j_2: 塗りつぶしのみ

j_3: 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」コマンド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

n: 頂点数

x1, y1, ..., xn, yn: 各頂点の座標

frame_fill:

frame_fill = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 32*j_6 + 64*j_7, ここで、 各 jiフラグは0または1をとります.

j_1: 輪郭線のみ
j_2: 塗りつぶしのみ
j_3: 開いたポリゴンを閉じる
j_4: ローカル塗りつぶし向き
j_6: 切断塗りつぶし （デフォルトは作図塗りつぶし）
j_7: 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）

si: ステータス値：

si = j_1 + 16*j_5 + 32*j_6, ここで、 各 jiフラグは0または1をとります.

j_1: 次の線分を表示
j_5: 次の線分は内側の線 （0の場合は、 一般的な線）
j_6: 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

線のデフォルトの線特性は0（一般的な線）で、「LINE_PROPERTY」 コマンドは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、「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。

fill_pen: 塗りつぶしペンカラー番号

fill_background_pen: 塗りつぶし背景ペンカラー番号

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、「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」の上級版で、ハッチングの原点および方向を定義することができます。

frame_fill:

frame_fill = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7, ここで、 各 jiフラグは0または1をとります.

j_1: 輪郭線のみ
j_2: 塗りつぶしのみ
j_3: 開いたポリゴンを閉じる
j_4: ローカル塗りつぶし向き
j_5: 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）
j_6: 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）
j_7: 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）

fillOrigoX: 塗りつぶし原点のX座標

fillOrigoY: 塗りつぶし原点のY座標

fillAngle: 塗りつぶしの方位角

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、「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_B」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:

frame_fill = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8, ここで、 各 jiフラグは0または1をとります.

j_1-j_7: 前のPOLY2_ コマンドと同様
j_8: 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、「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,
gradientInnerRadius,
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:

frame_fill = j_1 + 2*j_2 + 4*j_3, ここで、 各 jiフラグは0または1をとります.

j_1: 輪郭線のみ
j_2: 塗りつぶしのみ
j_3: 開いたポリゴンを閉じる

fillcategory:

0: 作図
1: 切断
2: 表面

distortion_flags:

distortion_flags = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7, 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。

j_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。
j_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。
j_3: innerRadiusパラメータを使用して円形の歪みを作成
j_4: ローカルの方向を使用、歪み配列（mijパラメータ）を使用
j_5: （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

j_6: （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
j_7: （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。

POLY2_B{6}

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

POLY2_B{5}のアドバンストバージョン。各輪郭辺に対して、輪郭属性（ペンと線種）を個別に制御することができます。

peni: 制御点iから開始する輪郭線のペンインデックス

linetypei: 制御点iから開始する輪郭線の線種インデックス

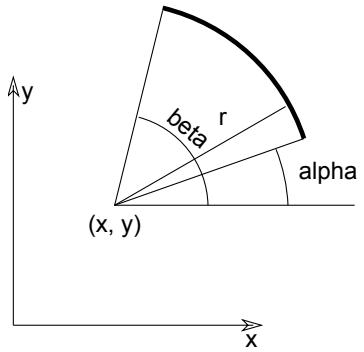
互換性：ARCHICAD 21で導入されました。

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。

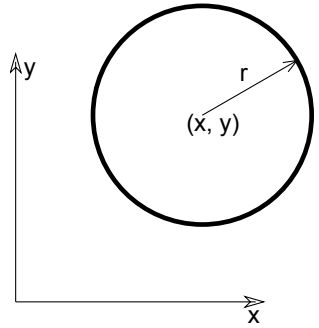
alphaとbetaは度（°）で指定します。



CIRCLE2

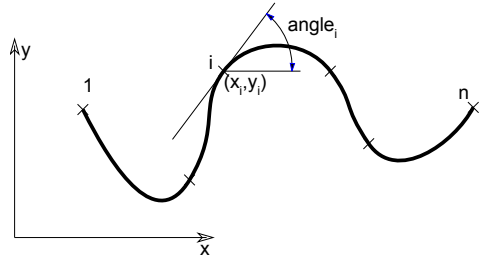
CIRCLE2 x, y, r

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。

パラメータの制限:

n >= 2

si: ステータス値：

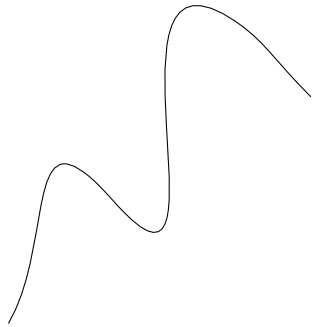
0: デフォルト

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。

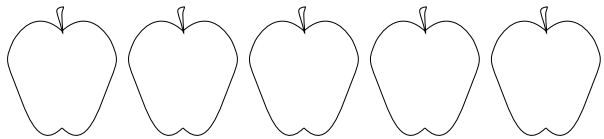
例 1:

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



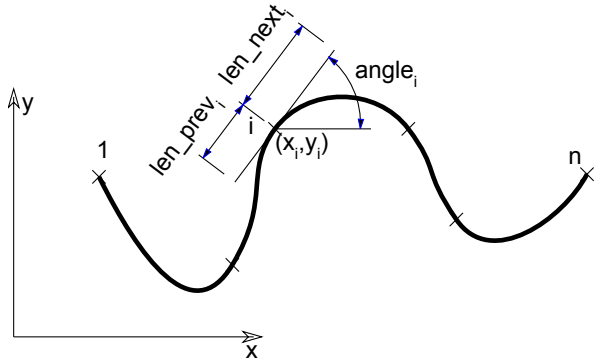
例 2:

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



SPLINE2A

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



「SPLINE2」コマンドの拡張版（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。

si: 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。

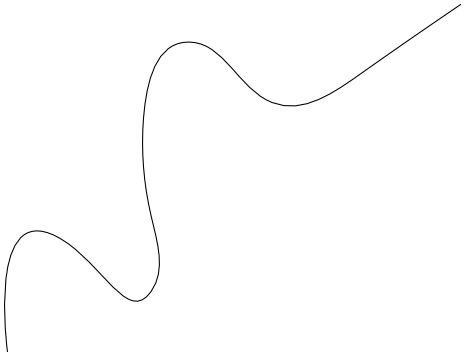
xi, yi: 制御点の座標

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。

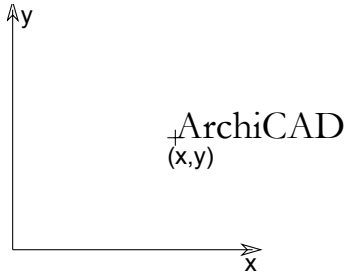
テキスト要素

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。

「[SET] STYLE」および「DEFINE STYLE」を参照してください。



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、「TEXTBLOCK」を参照してください。

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TEXTBLOCKの名前

バイナリ2D

FRAGMENT2

FRAGMENT2 fragment_index, use_current_attributes_flag
FRAGMENT2 ALL, 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。 ALLを指定すると、全ての断片が挿入されます。

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。 空白塗りつぶしを設定するSET FILL 0ショートカットは、この場合は機能しません。実際の空白塗りつぶしを参照する必要があります。

projection_code: 投影のタイプ
- 3: 上面図
- 4: 側面図
- 5: 側面図2
- 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 無効な値が設定された場合または何も設定されていない場合、デフォルトは陰線処理（2）です。
- 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

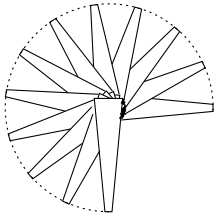
filldirection: 塗りつぶしの方位角

注記: 「[SET] FILL」は、PROJECT2{2}に対して有効です。

例:

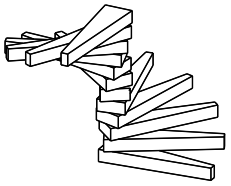
2D

```
PROJECT2 3, 270, 2

LINE_TYPE "DASHED"
ARC2 0, 0, A-B/3, 0, E

E = 270
A = 1
B = 0.2

ROT2 E
ADD2 A-B/3, 0
LINE2 0, 0, -0.05, -0.1
LINE2 0, 0,  0.05, -0.1

DEL 2
```



3D

```
n = 12
E = 270
D = 0.2
A = 1
B = 0.2

FOR i=1 TO n
   prism  4, D,
        -B/3, -B/2,
        -B/3,  B/2,
       A-B/3,  B/8,
       A-B/3, -B/8
   ADDZ D
   ROTz E/(n-1)
NEXT i

DEL n*2
```



PROJECT2{3}

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

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 無効な値が設定された場合または何も設定されていない場合、デフォルトは陰線処理（2）です。
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。
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。
16384: 加算変更子：透明材質の透過を有効にする。この場合、透過率が50を超える材質は完全に透明になり、それ以外の材質は非透明になります。

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。

互換性に関する注記：ARCHICAD 19以前では、切り取りポリゴンは、3Dスクリプトの「SECT_FILL」または「SECT_ATTRS」で定義された属性で生成されていました。ARCHICAD 20以降では、切り取りポリゴンの属性は、外側面の表面塗りつぶしによって定義されます（加算変更子32が設定されていない場合）。

parts: 生成する部分を定義 1+2+4+8+16+32 の値は全ての部品を意味します。
parts = $j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6$, ここで、 各 jiフラグは0または1をとります.
j1j2j3、j4、j5、j6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。
j_1: 切り取りポリゴン（シェーディングモードでのみ有効）
j_2: 切り取りポリゴンの辺
j_3: ポリゴンの表示
j_4: ポリゴンの辺の表示
j_5: 3D ホットスポットを静的2Dホットスポットとして投影、
j_6: 3D hotlineとhotarcsを投影（関連する3Dホットスポットを静的2Dホットスポットに変換を含む）。

PROJECT2{4}

PROJECT2{4} projection_code, angle,
useTransparency, statusParts,
numCutplanes,
cutplaneHeight1, ..., cutplaneHeightn,

method1, parts1,
cutFillIndex1,
cutFillFgPen1, cutFillBgPen1,
cutFillOrigoX1, cutFillOrigoY1, cutFillDirection1,
cutLinePen1, cutLineType1,
projectedFillIndex1,
projectedFillFgPen1, projectedFillBgPen1,
projectedFillOrigoX1, projectedFillOrigoY1,
projectedFillDirection1,
projectedLinePen1, projectedLineType1,
...
method(numCutplanes+1)), parts(numCutplanes+1),
cutFillIndex(numCutplanes+1),
cutFillFgPen(numCutplanes+1), cutFillBgPen(numCutplanes+1),
cutFillOrigoX(numCutplanes+1), cutFillOrigoY(numCutplanes+1),
cutFillDirection(numCutplanes+1),
cutLinePen(numCutplanes+1), cutLineType(numCutplanes+1),
projectedFillIndex(numCutplanes+1),
projectedFillFgPen(numCutplanes+1), projectedFillBgPen(numCutplanes+1),
projectedFillOrigoX(numCutplanes+1), projectedFillOrigoY(numCutplanes+1),
projectedFillDirection(numCutplanes+1),
projectedLinePen(numCutplanes+1), projectedLineType(numCutplanes+1)

互換性：ARCHICAD 20で導入されました。

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。 4番目のバージョン、PROJECT2{4}では、X-Y平面と平行に複数の切断面を定義し、線種、ペン、塗りつぶしなど、断片の切断部分と投影部分の属性を制御する機能が追加されています。 切断面の数を0すると、1つの投影断片のみを作成できます（numCutplanes+1）。

useTransparency: は0（透過なし）または正の整数（1：透過が有効）にすることができます。

statusParts: は、生成するステータス部分を定義します（hotline、ホットスポット、hotarc）。1+2の値は、全ての部分を意味します。設定は全ての断片に適用されます。

statusParts = $j_1 + 2*j_2$, ここで、 各 jiフラグは0または1をとります.

j1、j2の数値は、投影したモデルの対応するステータス部分を表示する（1）か省略する（0）かを表します。

j_1: 3D ホットスポットを静的2Dホットスポットとして投影、
j_2: 3D hotlineとhotarcsを投影（関連する3Dホットスポットを静的2Dホットスポットに変換を含む）。

numCutplanes: 定義された切断面の数。0にすることができますが、それより多い数をお勧めします。

cutplaneHeighti: 個別に定義された各切断面の位置。オブジェクトのX-Y平面からの垂直距離として測定されます。

method: 選択された画像処理方式 無効な値が設定された場合または何も設定されていない場合、デフォルトは陰線処理（2）です。
0: 現在の断片は投影の一部でない
1: ワイヤフレーム
2: 陰線処理（分解法）
3: シェーディング
4: ポリゴンによる陰線処理：ポリゴンはシェーディング方式で作成された部分に属するポリゴンまたは線を除外しませんが、他のワイヤフレーム/陰線処理部分に属するポリゴンおよび線を覆うか除外します。最適な結果を得るには、それを空気層に設定してください。このような分解されたポリゴンは、断片の順序に応じて2Dで動作します（シェーディングされた部分を覆いますが、除外しません）。
16: 加算変更子：ベクトルハッチングを描画（隠線処理モード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でポリゴンモードでは陰線処理）
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でポリゴンモードでは陰線処理）
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。

partsi: 生成する部分を定義 1+2+4+8+64の値は全ての部品を意味します。
partsi = $j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 64*j_7$, ここで、 各 jiフラグは0または1をとります.
j1、j2、j3、j4、j7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。
j_1: 切り取りポリゴン（シェーディングモードでのみ有効）
j_2: 切り取りポリゴンの辺
j_3: ポリゴンの表示
j_4: ポリゴンの辺の表示
j_7: プロジェクトの点群

cutFillIndexi: 現在の断片の切断部分の塗りつぶし種類インデックス

cutFillFgPeni: 現在の断片の切断部分の塗りつぶしペン

cutFillBgPeni: 現在の断片の切断部分の塗りつぶし背景ペン

cutFillOrigoXi: 現在の断片の切断塗りつぶし原点のX座標
cutFillOrigoYi: 現在の断片の切断塗りつぶし原点のY座標
cutFillDirectioni: 現在の断片の切断塗りつぶしの方位角
cutLinePeni: 現在の断片の切断線のペンインデックス
cutLineTypei: 現在の断片の切断線の線種
projectedFillIndexi: 現在の断片の投影部分の塗りつぶし種類インデックス
projectedFillFgPeni: 現在の断片の投影部分の塗りつぶしペン
projectedFillBgPeni: 現在の断片の投影部分の塗りつぶし背景ペン
projectedFillOrigoXi: 現在の断片の投影塗りつぶし原点のX座標
projectedFillOrigoYi: 現在の断片の投影塗りつぶし原点のY座標
projectedFillDirectioni: 現在の断片の投影塗りつぶしの方位角
projectedLinePeni: 現在の断片の投影線のペンインデックス
projectedLineTypei: 現在の断片の投影線の線種

リスト内の図面

これらのコマンドは、要素リストが作成された場合のみ、効果があります。

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,
fillOrigoX, fillOrigoY, 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。

method: 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。

method: 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。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。
16384: 加算変更子：透明材質の透過を有効にする。この場合、透過率が50を超える材質は完全に透明になり、それ以外の材質は非透明になります。

ホットスポットベースの編集コマンド

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インタラクティブなグラフィカル編集です。

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

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。

customDescription: 情報パレット内の表示されたパラメータのカスタム説明文字列。 このオプションを使用すると、displayParamも設定する必要があります（デフォルトではparamReferenceを使用してください）。 移動タイプのホットスポットに設定された値のみ表示されます。 同じ基準ホットスポットを有する全ての移動ホットスポットに同じ説明を設定することをお勧めします。

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:

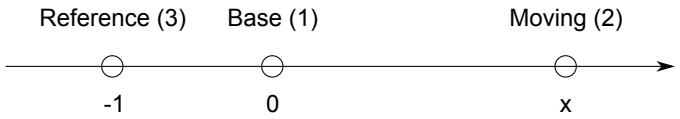
1: 長さタイプの編集、基準ホットスポット
2: 長さタイプの編集、移動ホットスポット
3: 長さタイプの編集、参照ホットスポット（常に非表示）
4: 角度タイプの編集、基準ホットスポット
5: 角度タイプの編集、移動ホットスポット
6: 角度タイプの編集、角度の中心（常に非表示）
7: 角度タイプの編集、参照ホットスポット（常に非表示）

attribute: 属性は、次の値または0の組み合わせが可能です。

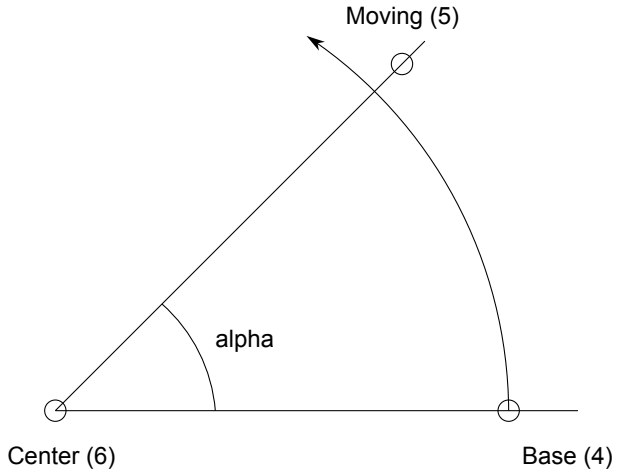
attribute = $128*j_8 + 256*j_9 + 512*j_{10} + 1024*j_{11}$, ここで、 各 jiフラグは0または1をとります.

j_8: ホットスポットを非表示（タイプ 1、2、4、5）、
j_9: 編集可能な基準ホットスポット（タイプ 1、4）、
j_{10}: 2Dで角度を反転（タイプ6）
j_{11}: ペーパースペースでparamReferenceの値をメートルとして使用。

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。 編集線の生の方向は、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、3Dで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）。

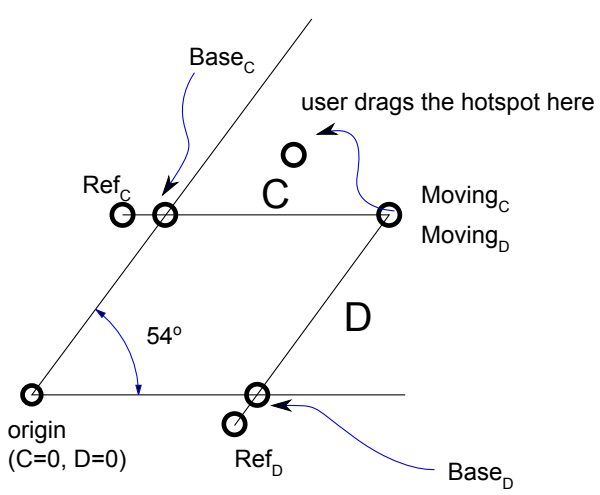
2つの長さタイプのホットスポットのセットを組み合わせて、1回のドラッグで2つのパラメータを編集できます。 2つを組み合わせた場合、ホットスポットの動きは線の制約を受けなくなりますが、長さ編集ホットスポットセットそれぞれの2つの線によって決まる平面の制約を受けます。 3Dでは、長さ編集ホットスポットのセットを3つ組み合わせることで、ホットスポットを空間内の任意の場所に配置できます。 2本の線は、互いに平行であってはならず、3本の線は同一平面上にあってはなりません。 組み合わされたパラメータ編集操作は、選択した点の位置に対応パラメータが異なる2つの編集可能ホットスポット（移動ホットスポットまたは編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。

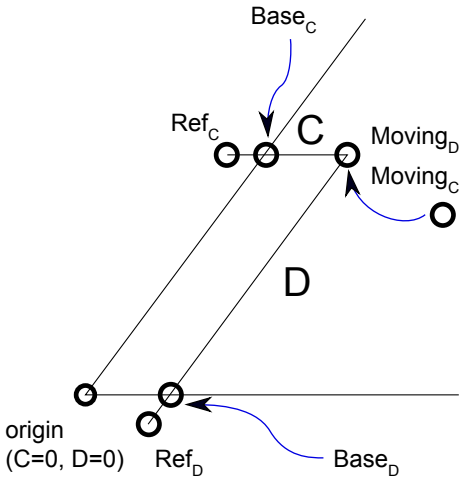
次の図と例2を参照してください。

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
```

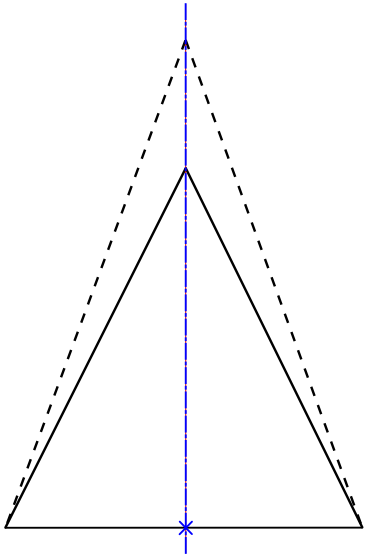
例 2: パラメータが 2 つの組み合わされた長さタイプ編集：






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

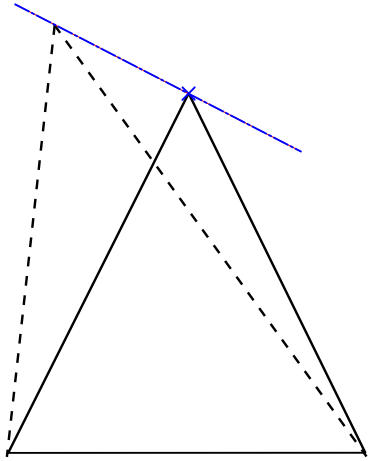
例 3: パラメータが1つの単純な長さタイプ編集：



```
!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, 6, corner_x, 1+128
HOTSPOT2 -1, corner_y, 7, corner_x, 3
HOTSPOT2 corner_x, corner_y, 8, corner_x, 2
LINE2 -1, 0, 1, 0
LINE2 -1, 0, corner_x, corner_y
LINE2 1, 0, corner_x, corner_y
!3D SCRIPT:
HOTSPOT -1, 0, 0, 1
HOTSPOT -1, 0, 0.5, 2
HOTSPOT 1, 0, 0, 3
HOTSPOT 1, 0, 0.5, 4
HOTSPOT corner_x, 0, 0, 5, corner_y, 1+128
HOTSPOT corner_x, -1, 0, 6, corner_y, 3
HOTSPOT corner_x, corner_y, 0, 7, corner_y, 2
HOTSPOT 0, corner_y, 0, 8, corner_x, 1+128
HOTSPOT -1, corner_y, 0, 9, corner_x, 3
HOTSPOT corner_x, corner_y, 0, 10, corner_x, 2
HOTSPOT corner_x, 0, 0.5, 11, corner_y, 1+128
HOTSPOT corner_x, -1, 0.5, 12, corner_y, 3
HOTSPOT corner_x, corner_y, 0.5, 13, corner_y, 2
HOTSPOT 0, corner_y, 0.5, 14, corner_x, 1+128
HOTSPOT -1, corner_y, 0.5, 15, corner_x, 3
HOTSPOT corner_x, corner_y, 0.5, 16, corner_x, 2
PRISM_ 4, 0.5,
   -1, 0, 15,
   1, 0, 15,
   corner_x, corner_y, 15,
   -1, 0, -1
```

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。 POLY2_, POLY2_A, POLY2_B, POLY2_B{2}, POLY2_B{3}, POLY2_B{4}, POLY2_B{5}, POLY_, PLANE_, PRISM_, CPRISM_, BPRISM_, FPRISM_, HPRISM_, SPRISM_, SLAB_, CSLAB_, CROOF_, EXTRUDE, PYRAMID, REVOLVE, SWEEP, TUBE, TUBEA

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
- - 側面の辺および正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: siの番号は、バイナリ整数（0～127）または-1です。

si = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 64*j_7 [+ a_code] , ここで、 各 jiフラグは0または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。

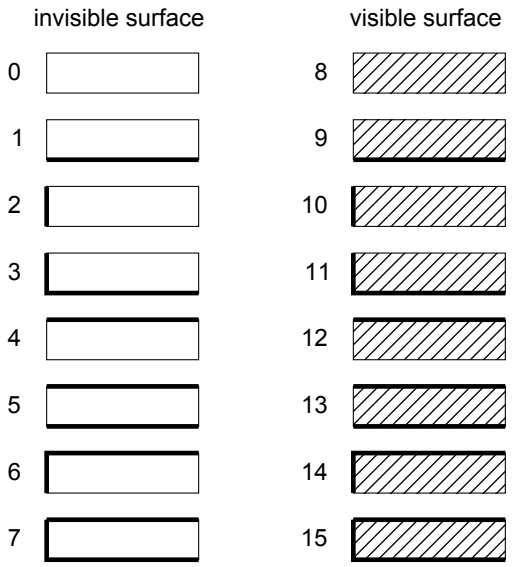
j_1: 水平方向の下の辺
j_2: 垂直方向の辺
j_3: 水平方向の上の辺
j_4: 側面
j_7: j2=1の場合のみ有効な特殊なステータス値で、 現在の垂直方向の辺の表示に応じて視点を制御
a_code: 追加ステータスコード（省略可）。ポリライン内に線分および円弧を作成できるようにします。

j2 = 0 垂直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）。



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。

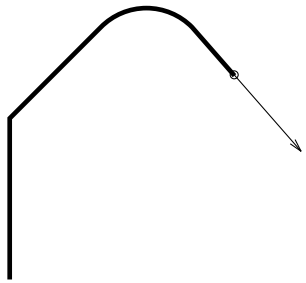
追加ステータスコード

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

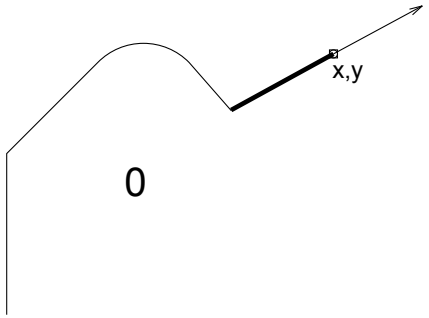
円弧の解像度は、「3Dおよび2Dスクリプトの指示文」の章で説明している指示文によって制御されます。「POLY2_」コマンドの場合、解像度が8 より大きいと、真の円弧が生成され、 それ以下の値にすると、生成される円弧は全て線分化されます。

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。



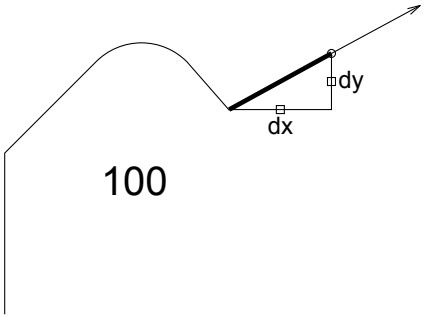
絶対終点による線分

x, y, s
ここで、0 < s < 100



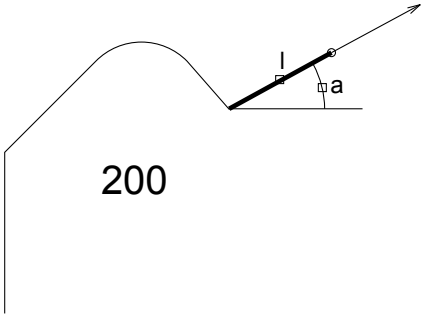
相対終点による線分

dx, dy, 100+s,
ここで、0 < s < 100



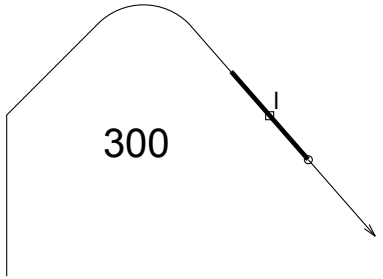
長さと方向による線分

l, a, 200+s,
ここで、0 < s < 100



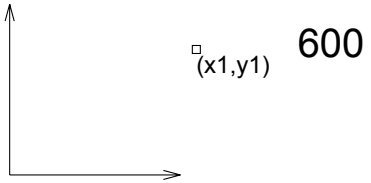
長さによる接線線分

l, 0, 300+s,
ここで、0 < s < 100



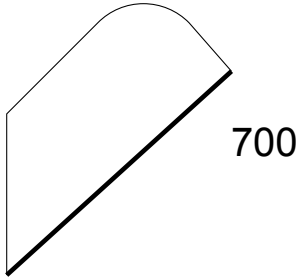
始点の設定

x1, y1, 600,



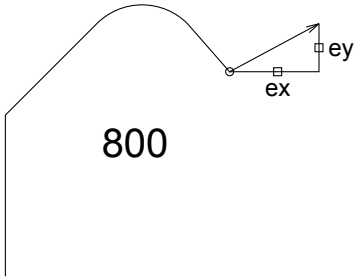
ポリラインを閉じる

0, 0, 700,



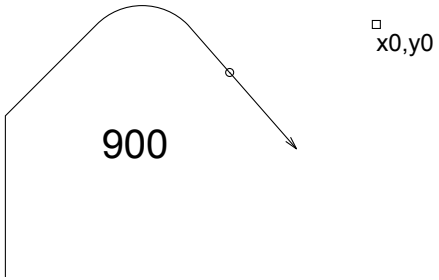
接線の設定

ex, ey, 800,



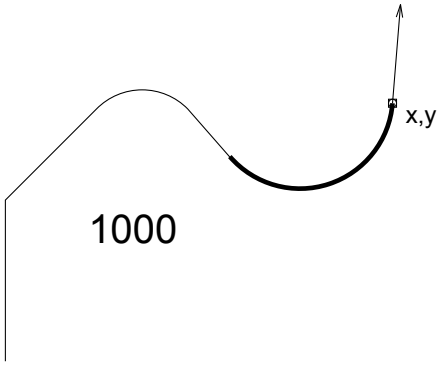
中心点の設定

x0, y0, 900,



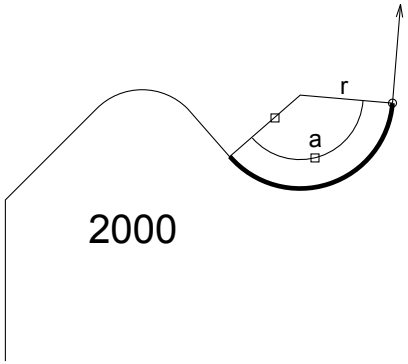
終点までの正接円弧

x, y, 1000+s,
ここで、0 < s < 100



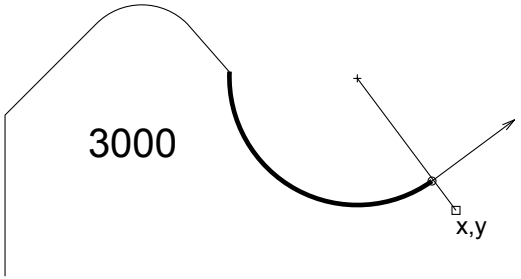
半径と角度による正接円弧

r, a, 2000+s,
ここで、0 < s < 100



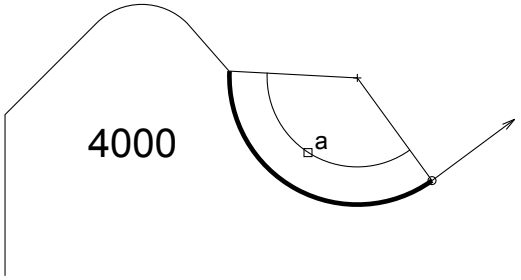
中心点と最終半径上の点による円弧

x, y, 3000+s,
ここで、0 < s < 100



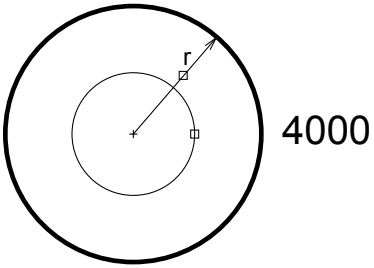
中心点と角度による円弧

0, a, 4000+s,
ここで、0 < s < 100



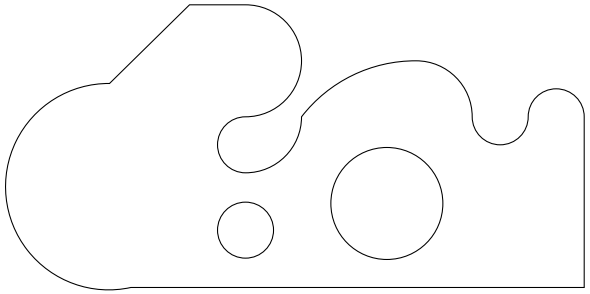
中心点と半径による完全な円

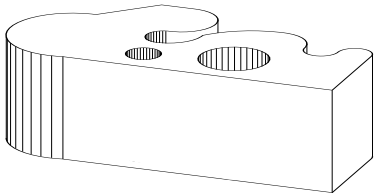
r, 360, 4000+s,
ここで、0 < s < 100



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かまいません。

例 1:

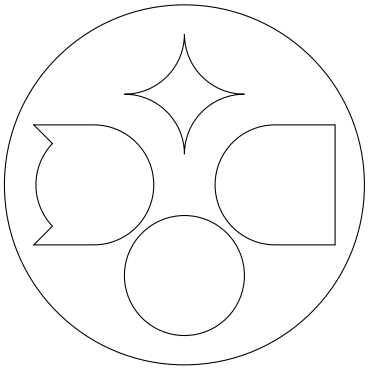




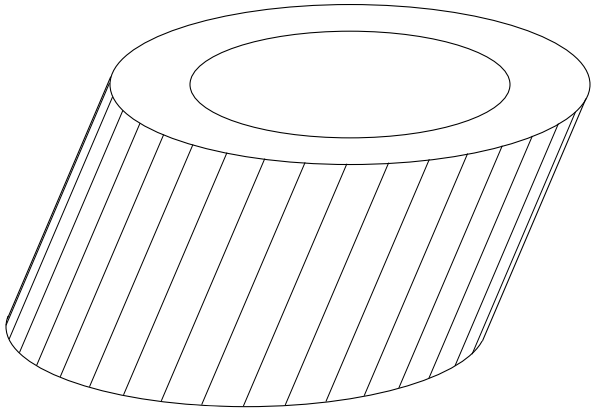
```
EXTRUDE 21, 0, 0, 3, 1+2+4+16+32,
        0, 0, 0,
        7,   0,    0,
        7,   3,    1,
        6,   3,    1000, ! 終点への正接円弧
        5,   3,    1001, ! 終点への正接円弧
        1,   90,   2000, ! 半径と角度による正接円弧
        2,   3,    1001, ! 終点への正接円弧
        1,   3,    900,  ! 終点を設定
        1,   2,    3000, ! 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) によるセグメント
        2,   225,  200,  ! (len, angle) によるセグメント
        -1,  0,    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
```

例 2:

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

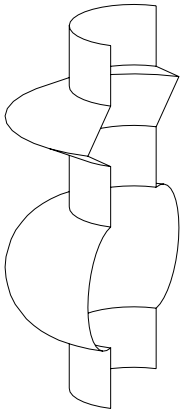


例 3:



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

例 4:



```
ROTY-90
REVOLVE 9, 180, 16+32,
        7, 1, 0,
        6, 1, 0,
        5.5, 2, 0,
        5, 1, 0,
        4, 1, 0,
        3, 1, 900, ! 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コマンド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の求められた値を格納します。

RADIUS

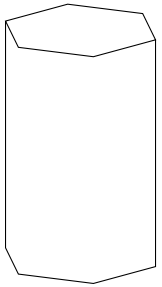
RADIUS radius_min, radius_max
ポリラインの円柱要素と円弧の滑らかさを設定します。
半径がrの円が表現されます。

- 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

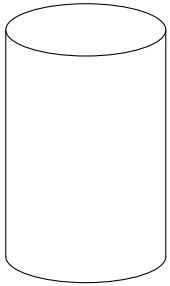
円弧についても同様に変換されます。
RADIUSステートメントの後では、前に使用されたRESOLとTOLERステートメントの効果はなくなります。
パラメータの制限:
r_min <= r_max

例:

RADIUS 1.1, 1.15
CYLIND 3.0, 1.0



RADIUS 0.9, 1.15
CYLIND 3.0, 1.0



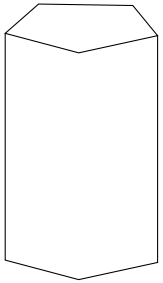
RESOL

RESOL n

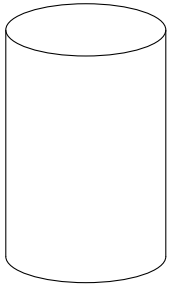
ポリラインの円柱要素と円弧の滑らかさを設定します。円は、辺の数がnの正規ポリゴンに変換されます。
円弧についても同様に変換されます。
RESOLステートメントの後では、前に使用されたRADIUSとTOLERステートメントの効果はなくなります。
パラメータの制限:
　n >= 3
デフォルト：
RESOL 36

例:

RESOL 5
CYLIND 3.0, 1.0



RESOL 36
CYLIND 3.0, 1.0



TOLER

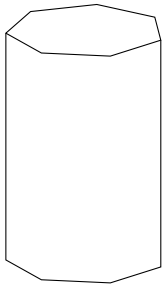
TOLER d

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。

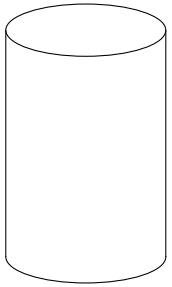
TOLERステートメントの後では、前のRADIUSとRESOLステートメントの効果はなくなります。

例:

TOLER 0.1
CYLIND 3.0, 1.0



TOLER 0.01
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 ポリラインの円弧の滑らかさを設定します。

「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。

expr: 有効値：
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インライン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

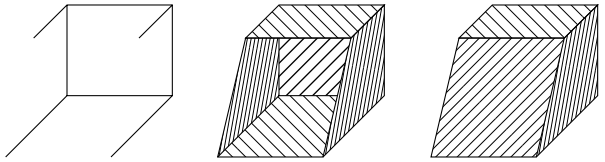
デフォルト：

MODEL SOLID

例: 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あげます。

```
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_or_index

この後に生成される全ての面は、次のMATERIALステートメントまでこの材質を表します。 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、内部データ構造の材質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インライン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。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）

[SET] BUILDING_MATERIAL

[**SET**] **BUILDING_MATERIAL** name_or_index
[, cut_fill_pen [, cut_fill_bkgd_pen, [iOverrideFlag]]]

互換性：ARCHICAD 21で導入されました。

この後に生成される全ての形状は、設定されたビルディングマテリアルの表面、切断塗りつぶしタイプ（ 断面/立面図の場合）、 前景および背景ペンを表します。

cut_fill_pen: 有効なビルディングマテリアル属性のインデックスを上書きするカスタム切断塗りつぶし前景ペン

cut_fill_bkgd_pen: 有効なビルディングマテリアル属性のインデックスを上書きするカスタム切断塗りつぶし背景ペン

iOverrideFlag: "cut_fill_pen"および/または"cut_fill_bkgd_pen"を使用可能にして、次を有効化：
iOverrideFlag = j1 + 2*j2: jがそれぞれ0または1の場合、次のようになります。
j1: cut_fill_penで切断塗りつぶし前景ペンを上書き
j2: cut_fill_bkgd_penで切断塗りつぶし背景ペンを上書き
上書きパラメータはオプションです。"iOverrideFlag"が設定されていない場合、または上書きペンインデックスパラメータにDEFAULTキーワードが使用されている場合は、ビルディングマテリアル属性が有効です。

例: 切断塗りつぶし背景ペンの上書き
BUILDING_MATERIAL buildingMatIndex, DEFAULT, cut_fill_bkgd_pen

この後に生成される全ての形状は、次の BUILDING_MATERIAL、MATERIAL、SECT_FILLまたはSECT_ATTRSステートメントまでビルディングマテリアルの表面を表します。 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。

断面/立面図では、表示される切断塗りつぶし前景および背景ペンは、ビルディングマテリアルの同じ属性に一致します（またはコマンド自体で上書きパラメータが設定されます）。 これは次のBUILDING_MATERIAL、MATERIAL、SECT_FILLまたはSECT_ATTRSステートメントまで続きます。

前のBUILDING_MATERIALステートメントは、SECT_FILLまたはSECT_ATTRSステートメントを使用した後に生成された形状をもう制御していません。 次のステートメントを使用した後に生成された形状は、BUILDING_MATERIAL設定を保持しています。 MATERIALステートメントは、生成された形状の表面のみを上書きします。 SECT_ATTRS{2}ステートメントは、断面図ビューの輪郭ペンと線種の表示のみを制御します。 その他の属性は引き続きビルディングマテリアル自体に制御されます。

インデックスは、内部データ構造のビルディングマテリアルスタックを参照する定数です。 ビルディングマテリア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ときのみ推奨します。

index 0には特別な意味があります。生成される断面には、塗りつぶしに基づく線がありません。

デフォルト：
BUILDING_MATERIAL 0
スクリプトにBUILDING_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、BUILDING_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

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

断面/立面図ウィンドウの3D要素の切断部分で使用する属性を定義します。 互換性：ARCHICAD 19以前では、「PROJECT2{3}」も影響を受けます。 インラインのfillおよびline_type属性（マスタースクリプトまたは3Dスクリプトで定義される）は受け入れられません。

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

SECT_ATTRS{2}

SECT_ATTRS{2} contour_pen [, line_type]

互換性：ARCHICAD 21で導入されました。

断面/立面図の3D要素の切断部分で使用する輪郭ペンと線種を定義します。 BUILDING_MATERIALステートメントと組み合わせると、全ての断面/立面属性を処理することができます。 インラインのline_type属性（マスタースクリプトまたは3Dスクリプトで定義される）は受け入れられません。

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 casting [, catching]

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。

casting: ON、AUTO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。

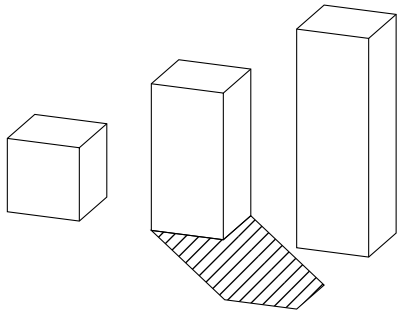
catching: ONあるいはOFF

このオプションパラメータは、面上のシャドウの表現を（他のボディから）制御します。

シャドウ投射が指定されていない場合、デフォルトはAUTOです。

例:

```
SHADOW OFF
! 水平表面
PRISM 4, 0.2,
      0, 0,
      6, 0,
      6, 6,
      0, 6

ADDX 0.5
ADDY 2.5

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

DEL 4
```



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。 マスタースクリプトまたは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。ただし、（パラメータによってではなく）名前またはインデックスで使用される場合に限ります。 マスタースクリプトまたは3Dスクリプトで定義される塗りつぶし種類と線種には、3Dスクリプトでアクセスできません。

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,
surface_red, surface_green, surface_blue
[, ambient_ce, diffuse_ce, specular_ce, transparent_ce,
shining, transparency_attenuation
[, specular_red, specular_green, specular_blue,
emission_red, emission_green, emission_blue, emission_att]]
[, fill_index [, fillcolor_index, texture_index]]

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、「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」で説明しています。
0: 一般定義、n = 16
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 1: ソリッドカラーの材質

```
DEFINE MATERIAL "water" 0,
	0.5284, 0.5989, 0.6167,! 表面 RGB [0.0..1.0]
	1.0,		! 環境係数 [0.0..1.0]
	0.5,		! 拡散係数 [0.0..1.0]
	0.5,		! 鏡係数 [0.0..1.0]
	0.9,		! 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]
```

例 2: 塗りつぶしの材質

```
DEFINE MATERIAL "Brick-Red" 10,
	0.878294, 0.398199, 0.109468,
	0.58, 0.85, 0.0, 0.0,
	0,
	0.0,
	0.878401, 0.513481, 0.412253,
	0.0, 0.0, 0.0,
	0,
	IND(FILL, "common brick")	! 塗りつぶしインデックス
```

例 3: 塗りつぶしとテクスチャの材質

```
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", ...) 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: (surface 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

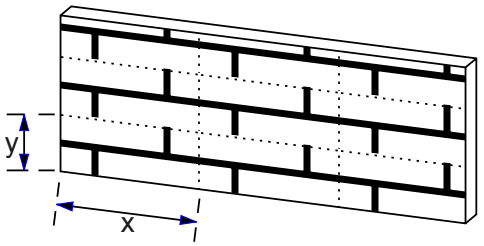
mask:

mask = $j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9$, ここで、 各 jiフラグは0または1をとります.

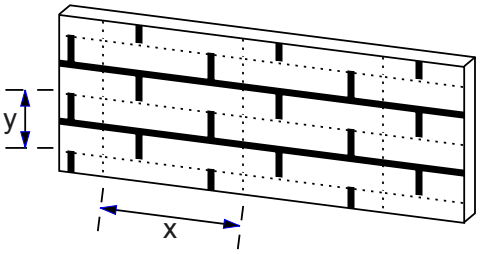
アルファチャネルの制御（j1... j6）：

j_1: アルファチャネルによってテクスチャの透過が変化
j_2: 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。
j_3: 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
j_4: 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
j_5: 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
j_6: 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

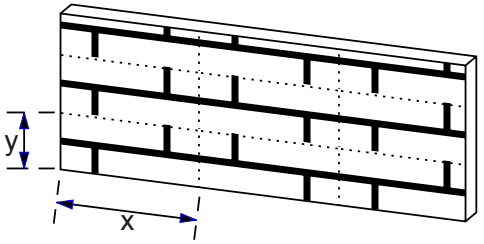
接続の制御（j7... j9）：（値がゼロの場合、標準モードが選択されます。）



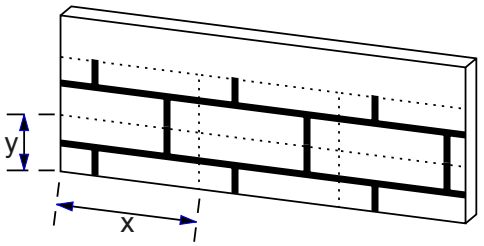
j_7: テクスチャはランダムにシフトされます。



j_8: ‘x’方向にミラー



j_9: ‘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

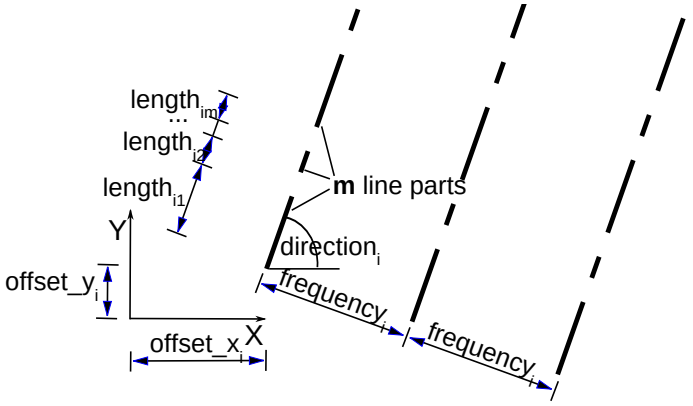
塗りつぶし

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 1: 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、「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の2D要素でのみ使用できます。



name: 塗りつぶしの名前

fill_types:

fill_types = j_1 + 2*j_2 + 4*j_3, ここで、 各 jiフラグは0または1をとります.

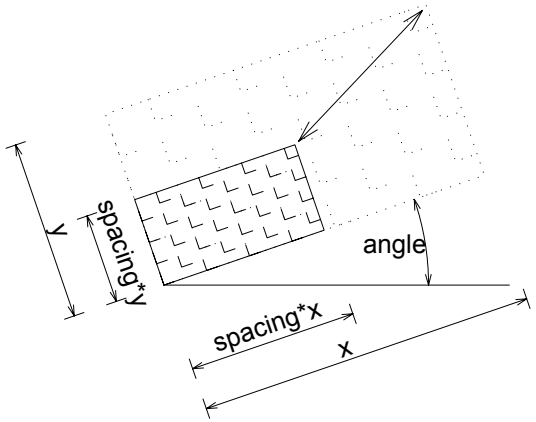
j_1: 切断塗りつぶし

j_2: 表面塗りつぶし

j_3: 作図塗りつぶし

jビットがセットされている場合、指定されているタイプに対応して、定義された塗りつぶしを使用できます。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 2: 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。

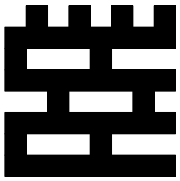
例:

```
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
```

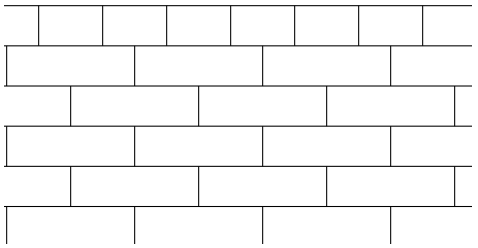
ビットマップパターン：

```
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：



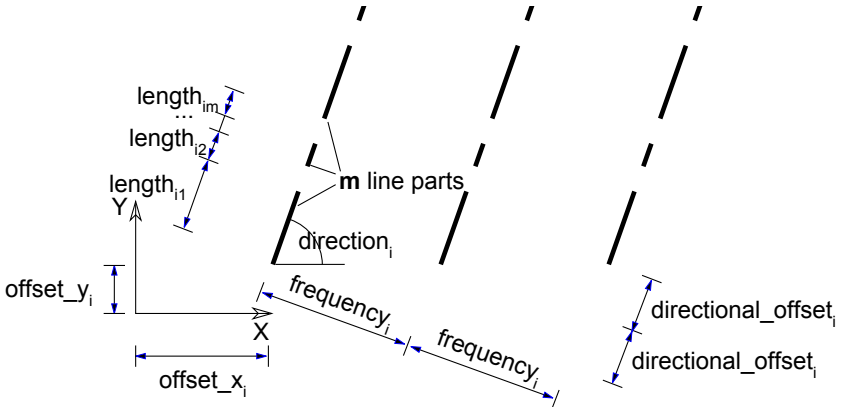
ベクトルハッチング：



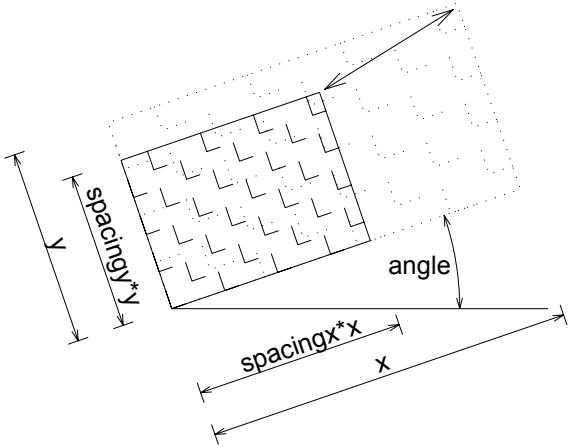
DEFINE FILLA

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、「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。



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、定義された間隔によって調節されます。

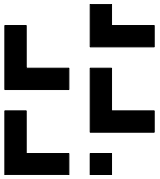
例:

```
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
```

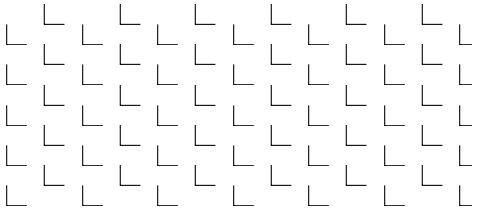
ビットマップパターン：

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：



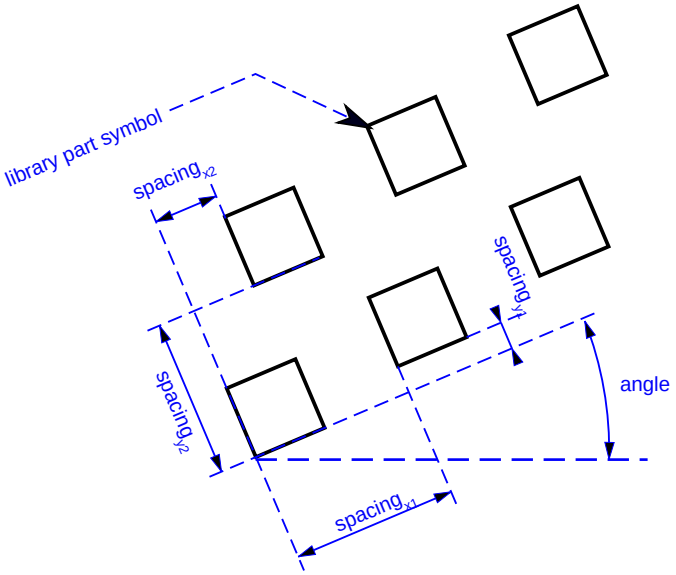
ベクトルハッチング：



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、「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。

spacingx1, spacingx2: 水平方向の間隔

spacingy1, spacingy2: 垂直方向の間隔

scaling1: 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、「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、「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: 前景色の不透明度のパーセント。 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_mask = $1024*j_{11} + 2048*j_{12}$, ここで、 各 jiフラグは0または1をとります.
表面上の画像のレイアウトに関する詳細は、「DEFINE TEXTURE」を参照してください。
j_{11}: ‘x’ 方向にミラー
j_{12}: ‘y’ 方向にミラー

image_rotangle: 標準座標系からのパターンの回転角度。

線種

DEFINE LINE_TYPE

DEFINE LINE_TYPE name spacing, n,
length1, ..., lengthn

注記 1: 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、「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

lengthi: 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 2: このコマンドでは、単純な線種、つまり線分とスペースのみで構成された線種を定義できます。シンボル線種は「DEFINE SYMBOL_LINE」で定義できます。

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、「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。

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

テキストスタイルとテキストブロック

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

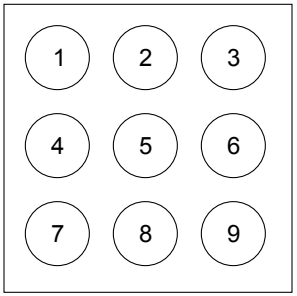
font_family: 使用されているフォントファミリの名前 （Garamondなど）

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：

face_code = j_1 + 2*j_2 + 4*j_3, ここで、 各 jiフラグは0または1をとります.

j_1: 太字

j_2: 斜体

j_3: 下線

face_code = 0の場合、スタイルは標準です。

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：

face_code = j_1 + 2*j_2 + 4*j_3 + 32*j_6 + 64*j_7 + 128*j_8, ここで、 各 jiフラグは0または1をとります.

j_1: 太字
j_2: 斜体
j_3: 下線
j_6: 上付き文字
j_7: 下付き文字
j_8: 取り消し線

face_code = 0の場合、スタイルは標準です。

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 name alignment, firstline_indent,
left_indent, right_indent, line_spacing [,
tab_position1, ...]
[PEN index]
[[SET] STYLE style1]
[[SET] MATERIAL index]
'string1'
'string2'
...
'string n'
[PEN index]
[[SET] STYLE style2]
[[SET] MATERIAL index]
'string1'
'string2'
...
'string n'
...
ENDPARAGRAPH

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。文字列または整数を使用できます。 整数識別子は「TEXTBLOCK_」でのみ機能します。

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）。 特殊文字「\t」でのみ機能します。

stringi: テキスト部分。定数文字列か文字列タイプパラメータのどちらかを使用できます。

TEXTBLOCK

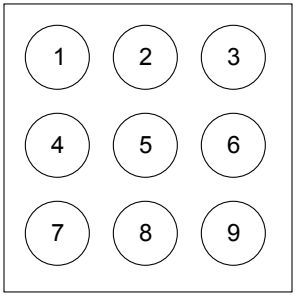
TEXTBLOCK name width, anchor, angle, width_factor, charspace_factor, fixed_height,
'string_expr1' [, 'string_expr2', ...]

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、「RICHTEXT2」と「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", ...) 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。

name: テキストブロックの名前、文字列タイプ値

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: 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: 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）。

TEXTBLOCK_

TEXTBLOCK_ name width, anchor, angle, width_factor, charspace_factor, fixed_height, n,
'expr_1' [, 'expr_2', ..., 'expr_n']

「TEXTBLOCK」と同様。 全てのパラメータは意味は同じですが、次の追加項目があります：

expr_i: 段落名は1つのテキストブロック内の文字列または整数のいずれかのタイプにすることができます。

n: リストされたexpr_i namesの数

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。 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。 異なる意味の付加データを実行中のアプリケーションから定義し、使用することができる。

追加データ定義は、以下のコマンドで使用可能です。

DEFINE MATERIAL parameters [[,] **ADDITIONAL_DATA** name1 = value1, name2 = value2, ...]
DEFINE MATERIAL name [,] **BASED_ON** orig_name [,] PARAMETERS name1 = expr1 [, ...]
[[,] **ADDITIONAL_DATA** name1 = expr1 [, ...]]
DEFINE FILL parameters [[,] **ADDITIONAL_DATA** name1 = value1, name2 = value2, ...]
DEFINE FILLA parameters [[,] **ADDITIONAL_DATA** name1 = value1, name2 = value2, ...]
DEFINE SYMBOL_FILL parameters
[[,] **ADDITIONAL_DATA** name1 = value1, name2 = value2, ...]
DEFINE SOLID_FILL name [[,] **FILLTYPES_MASK** fill_types]
[[,] **ADDITIONAL_DATA** name1 = value1, name2 = value2, ...]
DEFINE EMPTY_FILL name [[,] **FILLTYPES_MASK** fill_types]
[[,] **ADDITIONAL_DATA** name1 = value1, name2 = value2, ...]
DEFINE LINEAR_GRADIENT_FILL name [[,] **FILLTYPES_MASK** fill_types]
[[,] **ADDITIONAL_DATA** name1 = value1, name2 = value2, ...]
DEFINE RADIAL_GRADIENT_FILL name [[,] **FILLTYPES_MASK** fill_types]
[[,] **ADDITIONAL_DATA** name1 = value1, name2 = value2, ...]
DEFINE TRANSLUCENT_FILL name [[,] **FILLTYPES_MASK** fill_types]
pat1, pat2, pat3, pat4, pat5, pat6, pat7, pat8,
percentage [[,] **ADDITIONAL_DATA** name1 = value1, name2 = value2, ...]

DEFINE IMAGE_FILL name image_name [[,] **FILLTYPES_MASK** fill_types]
part1, part2, part3, part4, part5, part6, part7, part8,
image_vert_size, image_hor_size, image_mask, image_rotangle
[[,] **ADDITIONAL_DATA** name1 = value1, name2 = value2, ...]
DEFINE LINE_TYPE parameters [[,] **ADDITIONAL_DATA** name1 = value1, name2 = value2, ...]
DEFINE SYMBOL_LINE parameters
[[,] **ADDITIONAL_DATA** name1 = value1, name2 = value2, ...]

外部ファイルの依存性

FILE_DEPENDENCE

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オブジェクトの表現を定義する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ユーザーインターフェイススクリプト、 古いインスタンスを実際の要素に合わせて移行する方 法を 定義 す るた めの 上位 移行 スク リプ ト、 その 要素 を古 い要 素に 後退 して移 行す る方 法を 定 義 する た めの 下位 移行 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。

position_keyword: 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、ARCHICAD_LibraryMaster objectの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。 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。数値パラメータポップアップメニューアイテム値は、VALUES{2} コマンドを利用することで、文字列として定義することができます。

VALUES

VALUES "parameter_name" [,]value_definition1 [, value_definition2, ...]

VALUES "fill_parameter_name" [[,] **FILLTYPES_MASK** fill_types], value_definition1
[, value_definition2, ...]
VALUES "profile_parameter_name" [[,] **PROFILETYPES_MASK** profile_types], value_definition1
[, value_definition2, ...]

パラメータの値制限を定義（辞書タイプを除く）。このコマンドは塗りつぶしタイプと断面形状タイプのパラメータの特別な構文。配列パラメータで利用すると、この制限は個別の全ての項目に適用される。

parameter_name: 既存パラメータの名前

fill_parameter_name: 既存の塗りつぶしパターンのパラメータの名前

fill_types:

fill_types = j_1 + 2*j_2 + 4*j_3, ここで、 各 jiフラグは0または1をとります.
j_1: 切断塗りつぶし
j_2: 表面塗りつぶし
j_3: 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。 このパラメータの塗りつぶしポップアップには、1に設定されたビットによって指定される塗りつぶしのタイプのみが含まれます。 デフォルトは全ての塗りつぶし（0）です。

profile_parameter_name: 既存の断面形状パターンのパラメータの名前

profile_types:

profile_types = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5, ここで、 各 jiフラグは0または1をとります.
j_1: 壁、
j_2: 梁、
j_3: 柱、
j_4: 手摺り、
j_5: その他。

断面形状タイプパラメータにのみ設定できます。 断面形状タイプパラメータの値リストには、プランファイルの既存の全ての断面形状が自動的に含まれます。個別にVALUESを定義する必要はありません。 マスキング（0）なしでVALUESを使用すると、まったく同じ結果になります。 マスキングありでVALUESを使用すると値リストにフィルタを適用して、ビットが1に設定された対応する断面形状のみを残すことができます。 通常の値定義として個別の断面形状インデックスをリストすることもできます。

value_definitioni: 値の定義は、次のように行います。

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。
RANGE: 範囲定義、オプションのステップを含む

RANGE left_delimiter[lower_limit], [upper_limit]right_delimiter [STEP step_start_value, step_value]

left_delimiter: [または(。[と>=、(と>、はそれぞれ同義。 lower_limit：下限値。 upper_limit：上限値。 right_delimiter：]または)。]と<=、)と<、はそれぞれ同義。 step_start_value：開始値; step_value：ステップ値

VALUES{2}

VALUES{2} "parameter_name" [,]num_expression1, description1,
[, num_expression2, description2, ...]
VALUES{2} "parameter_name" [,]num_values_array1, descriptions_array1
[, num_values_array2, descriptions_array2, ...]

parameter_name: 既存の角度の名前、長さ、実数、整数タイプパラメータ

num_expressioni, num_values_arrayi: 数値パラメータの単純値定義または、複数の数値を含む配列式。VALUES{2} のみ使用できます。

descriptioni, descriptions_arrayi: 数値の説明文字列 i または、num_values_arrayi で定義された値の複数の説明文字列を含む配列式（配列寸法は一致する必要があります）。VALUES{2} のみ使用できます。

例 1: 単純な値リスト

```
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
```

例 2: 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[]
! パラメータデータファイル名
filename = "ProjectNotes.txt"
ch1 = OPEN ("text", filename, "MODE=RO, LIBRARY")
i = 1
j = 1
sarray[1] = ""
! の全ての文字列を収集
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
! ファイルから読み取った文字列を持つパラメータポップアップ
VALUES "RefNote" sarray
```

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。

namen: 文字列式、ロックするパラメータ名。

互換性：ARCHICAD 22以降、選択したARCHICADインターフェースコントロールのロック/非表示が拡張されています。

拡張機能は、[特定の固定名オプションパラメータの非表示/ロックを有効化]設定（ライブラリ部品エディタでオブジェクトの[詳細/互換性オプション]ダイアログを参照）で有効にできます。拡張された選択内容には、次に対応する固定名オプションパラメータが含まれます。

- [テキストスタイル]設定ダイアログパネルの標準のテキスト処理コントロール（「テキスト処理用パラメータ」を参照）
- ラベルツールの[テキストスタイル]設定ダイアログパネルの拡張されたスタイルラベルコントロール（「ラベルのパラメータ」を参照）
- [ポインタ]設定ダイアログパネルの選択済みラベルポインタコントロール（「ラベルのパラメータ」を参照）

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

ALLキーワードの後にリストされたパラメータを除いて、設定ダイアログボックス内の全てのパラメータをロックします。

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。

namen: 文字列式、非表示にするパラメータ名。

互換性：ARCHICAD 22以降、選択したARCHICADインターフェースコントロールのロック/非表示が拡張されています。詳細は、「LOCK」を参照してください。

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

設定ダイアログボックスで、ALLキーワードの後にリストされたパラメータ（およびその子）以外の全てのパラメータとその子パラメータを非表示にします。

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。デフォルトタイトルは「カスタム設定」です。 現在では、使用できる領域のサイズは444×296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 [, parent_id, page_title [, image]]

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。デフォルトのページ番号は1から始まりますが、任意の開始番号を用いることができます。 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表示されます。 ページ間の移動は、異なる方法で定義することができます：

- 最も簡単な方法はARCHICADでやることです： オブジェクトエディタで、ユーザーインターフェイススクリプト ウィンドウ下部にある、"階層ページ"ボタンをクリックし、UI_PAGEコマンドのオプションパラメータを入力してください。 この場合、ツリーから選択されたページの page_number は gs_ui_current_page パラメータを通して、ライブラリ部品に渡されます。ページングパラメータの値のリストを設定する必要はありません： ARCHICADは、オブジェクトのUIスクリプトを事前に読み込むことで、UI_PAGEコマンドのパラメータから全ての有効なpage IDを収集しソートします。
- もう一つの方法は、UI_NEXTおよびUI_PREVコマンドで生成された2つのボタンを使用することです。それらを各ページに配置し、"gs_ui_current_page"パラメータの値を操作します。 詳細は、「UI_BUTTON」を参照してください。

- 新規階層ページ設定が必要でない場合は、動的なページの取り扱いには、「UI_INFIELD{3}」を使用します。
  "gs_ui_current_page"パラメータの値のリストを設定し、その値を使用しポップアップを全てのページに配置します。

page_number: ページ番号、正の整数。次のインターフェイスの要素がページに配置されます。

parent_id: 正の整数、そのページの parent id。特殊な値 -1 の値は親ルートを意味します。「階層ページ」が設定されている場合のみ、テストされます。

page_title: ページのタイトル文字列、ページのトップとツリービューポップアップに表示されます。「階層ページ」が設定されている場合のみ、テストされます。

image: 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。指定されていて、空または0出ない場合、ページに関連するアイコンがページのトップと、タイトルの横のツリービューポップアップに表示されます。 「階層ページ」が設定されている場合のみ、テストされます。

警告：単純なページ指定の方法では、ページ番号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。 この制限は「UI_CURRENT_PAGE」を使用して回避できます。

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]]

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。 ボタンにはテキストを設定できます。

type: 次にあ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ブブラウザで開きます。

text: テキストがボタンに表示されます。

x, y: ボタンの位置

width, height: ボタンの幅と高さ（ピクセル単位）。 指定がなければ（互換性を確保するため）、デフォルト値は幅60ピクセル、高さ20ピクセルです。

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スクリプト
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,
        "https://www.graphisoft.co.jp"
! パラメータスクリプト
if GLOB_UI_BUTTON_ID = 3 then
        parameters gs_ui_current_page = 9, ...
endif
```

UI_PICT_BUTTON

UI_PICT_BUTTON type, text, picture_reference,
x, y, width, height [, id [, url]]
「UI_BUTTON」と同様。ただし、このボタンタイプには画像を設定できます。

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。 画像のピクセル透過が可能です。

text: 画像ボタンには影響しません。

UI_SEPARATOR

UI_SEPARATOR x1, y1, x2, y2
矩形の区切りを生成します。x1 = x2またはy1 = y2の場合、区切りは一本の（縦または横の）区切り線となります。

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キャプションテキストを含む矩形の区切りです。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 （単位： ピクセル）

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: アルファ+歪み
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: は「DEFINE STYLE」に似ていますが、組み合わせに値を使用できません。
0: 標準
1: 太字
2: 斜体
4: 下線

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: テキストボックスの幅と高さ。 省略すると、テキストボックスは指定されたフォントでテキストとの間隔ができるだけ狭くなるように囲みます。

flags:

flags = j_1 + 2*j_2 + 4*j_3, ここで、 各 jiフラグは0または1をとります.
j_1: 水平整列（j2と）
j_2: 水平整列（j1と）
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: Not used,
j_3: 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,
...
[picIdxArray, textArray, valuesArray,
...]
expression_imagen, textn, value_definitionn]

UI_INFIELD{4}

UI_INFIELD{4} "name", x, y, width, height [,
method, picture_name,
images_number,
rows_number, cell_x, cell_y,
image_x, image_y,
expression_image1, text1, value_definition1,
...
[picIdxArray, textArray, valuesArray,
...]
expression_imagen, textn, value_definitionn]

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。 異なるサムネイ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、サムネイ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とバージョン4のinfieldはサムネイル項目を関連するパラメータ値のリスト項目にバインドする、値の関連付けを定義します。 サムネイ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。 同一のサイズの配列は、同様に定義のラインのために使用することができます。

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。

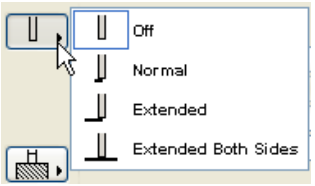
name: 文字列式としてのパラメータ名（4つ全てのバージョンのコマンド）、 UI_INFIELD{2}とUI_INFIELD{3}の場合はパラメータ名オプション、 UI_INFIELD{4}の場合はテキスト配列値オプションとしてのパラメータ名を含む。

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 （単位： ピクセル）

method: コントロールのタイプ
1: リストビューコントロール

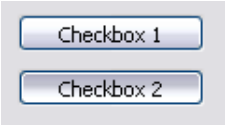
2: ポップアップメニューコントロール



3: ポップアップアイコンラジオコントロール（画像の矢印）

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

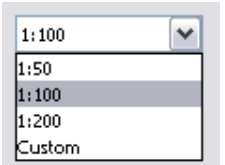


6: 画像付きプッシュボタン

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

9: ポップアップアイコンラジオコントロール（画像の横の矢印）

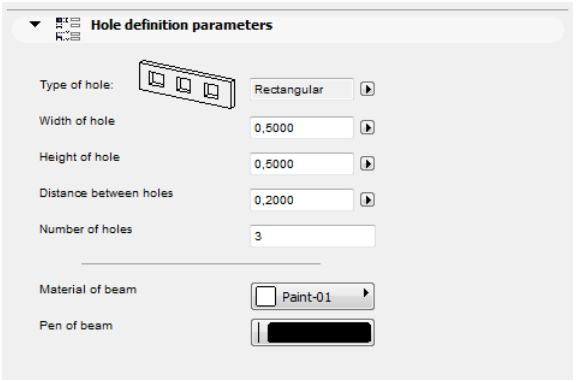


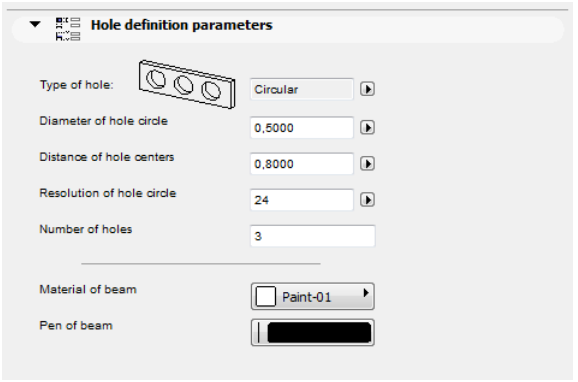
picture_name: 連結画像のマトリクス、または空の文字列を含む共通画像ファイル名
images_number: マトリクス内の画像の数、ブールパラメータの場合は0または2
rows_number: マトリクス内の行の数
cell_x, cell_y: 画像とテキストを含む、サムネイルビューフィールド内のセルの幅と高さ
image_x, image_y: セル内の画像の幅と高さ
expression_imagei: マトリクス内のi番目の画像のインデックス、または個々のファイル名 共通の画像ファイル名が指定されている場合は、ここではインデックスを使用する必要があります。 インデックスと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。
picIdxArray: 画像名の動的配列（文字列）または、セル内の頂点（整数）。配列のタイプを混合して使用しないでください。
textArray: セル内の動的配列文

valueArray: セル内の動的配列パラメータ

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:

```
! パラメータスクリプト：
VALUES "myParameter" "Two", "Three", "Five", CUSTOM

! インターフェイススクリプト：
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
```

例 3:

```
! パラメータスクリプト :
VALUES "myParameter" "Two", "Three", "Five", CUSTOM

! インターフェイススクリプト :
px = 80
py = 60
cx = px + 3
cy = py + 25

paramNameVar = "myParameter"
UI_INFIELD{4} paramNameVar, 10, 10, 4 * cx + 21, cy + 5,
        1, "myPicture", 6,
        1, cx, cy, px, py,
        1, "1 - one", "One",
        2, "2 - two", "Two",
        3, "3 - three", "Three",
        4, "4 - four", "Four",
        5, "5 - five", "Five",
        6, "custom value", CUSTOM
```

例 4:

```
! Master Script
dim picIdxValuesUI[]
dim textValuesUI[]
dim parameterValues[]

if myTypeParameter = 1 then
    picIdxValuesUI[1] = 6
    picIdxValuesUI[2] = 7
    picIdxValuesUI[3] = 8

    textValuesUI[1] = "6 - six"
    textValuesUI[2] = "7 - seven"
    textValuesUI[3] = "8 - eight"

    parameterValues[1] = "Six"
    parameterValues[2] = "Seven"
    parameterValues[3] = "Eight"
else
    picIdxValuesUI[1] = 6
    picIdxValuesUI[2] = 7

    textValuesUI[1] = "6 - six"
    textValuesUI[2] = "7 - seven"

    parameterValues[1] = "Six"
    parameterValues[2] = "Seven"
endif
```

```
    ! パラメータスクリプト :
VALUES "myTypeParameter" 1, 2
VALUES "myStringParameter" "Two", "Three", "Five", parameterValues, CUSTOM

! インターフェイススクリプト :
px = 80
py = 60
cx = px + 3
cy = py + 25

paramNameVar = "myStringParameter"
UI_INFIELD{4} paramNameVar, 10, 10, 4 * cx + 21, cy + 5,
        1, "myPicture", 6,
        1, cx, cy, px, py,
        1, "1 - one", "One",
        2, "2 - two", "Two",
        3, "3 - three", "Three",
        4, "4 - four", "Four",
        5, "5 - five", "Five",
        picIdxValuesUI, textValuesUI, parameterValues,
        9, "custom value", CUSTOM
```

UI_CUSTOM_POPUP_INFIELD

UI_CUSTOM_POPUP_INFIELD "name", x, y, width, height,
storeHiddenId, treeDepth,
groupingMethod, selectedValDescription,
value1, value2, valuesArray1, ... valuen, valuesArrayn

UI_CUSTOM_POPUP_INFIELD{2}

UI_CUSTOM_POPUP_INFIELD{2} name, x, y, width, height,
storeHiddenId, treeDepth,
groupingMethod, selectedValDescription,
value1, value2, valuesArray1, ... valuen, valuesArrayn

互換性：ARCHICAD 20で導入されました。

パラメータスクリプトの使用を回避するために、ユーザーインターフェイススクリプトで定義されたパラメータの値リストを表示するポップアップを生成します。

パラメータスクリプトで要求できないリストに最適です。 パラメータスクリプトの制限事項については、「REQUESTオプション」を参照してください。

name: UI_CUSTOM_POPUP_INFIELDの文字列式としてのパラメータ名、またはUI_CUSTOM_POPUP_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、ポップアップの位置

width, height: 幅と高さ （単位： ピクセル）

storeHiddenId, treeDepth: 自動ツリーまたは手動ツリーを設定します。

storeHiddenId = 0, treeDepth = 0: 配列パラメータでのみ機能します。

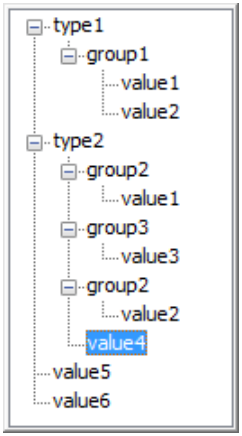
"treeDepth"パラメータは、配列の2番目の次元（列数）によって自動的に設定されます。

storeHiddenId = 1, treeDepth > 0: 1つのパラメータでのみ機能します。

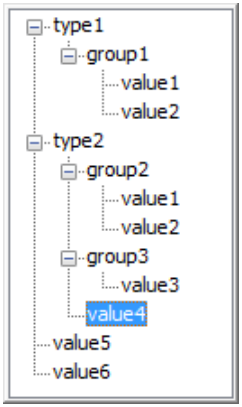
n * (1 + treeDepth)値を定義する必要があります （最初の値は保存されたID用、残りはカスタムツリーの定義用）。

groupingMethod: ツリーをソートするグループ化方式

1: 同じ親に属するグループおよび値をソートしません。



2: 同じ親に属するグループおよび値をソートします。

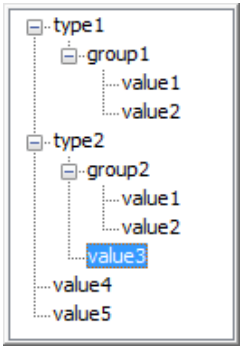


selectedValDescription: フィールドに書き込むテキスト、空白の文字列の場合、テキストは選択された項目の保存されたIDになります。

valuei, valuesArrayi: ツリー値を1つずつおよび/または1次元配列で定義します。

例:

```
UI_CUSTOM_POPUP_INFIELD "stParameterName", x, y, width, height,
    1, 3, 2, "",    ! storeHiddenId, treeDepth, groupingMethod, selectedValDescription
    "hiddenID1",    "type1",    "group1",   "value1",
    "hiddenID2",    "type1",    "group1",   "value2",
    "hiddenID3",    "type2",    "group2",   "value1",
    "hiddenID4",    "type2",    "group2",   "value2",
    "hiddenID5",    "type2",    "",         "value3",
    "hiddenID6",    "",         "",         "value4",
    "hiddenID7",    "",         "",         "value5"
```



UI_RADIOBUTTON

UI_RADIOBUTTON name, value, text, x, y, width, height

UI_RADIOBUTTON{2}

UI_RADIOBUTTON{2} "name", value, text, x, y, width, height

バージョン{2}の互換性：ARCHICAD 20で導入されました。

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: UI_RADIOBUTTONの場合はパラメータ名または文字列式としての名前、UI_RADIOBUTTON{2}の場合は文字列式としてのパラメータ名（またはインデックス付きテキスト配列の値）。

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。

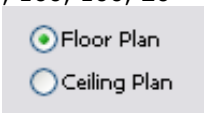
text: ラジオボタンの隣に表示されるテキスト

x, y: ボタンの位置

width, height: 幅と高さ （単位： ピクセル）

例:

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



UI_PICT_RADIOBUTTON

UI_PICT_RADIOBUTTON name, value, text,
picture_reference, x, y, width, height [UI_TOOLTIP tooltip]

UI_PICT_RADIOBUTTON{2}

UI_PICT_RADIOBUTTON{2} "name", value, text,
picture_reference, x, y, width, height [UI_TOOLTIP tooltip]

互換性：ARCHICAD 22で導入されました。

ラジオボタングループのアイコン付きのラジオボタンを1つ生成します。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: UI_PICT_RADIOBUTTONの場合はパラメータ名または文字列式としての名前、UI_PICT_RADIOBUTTON{2}の場合は文字列式としてのパラメータ名（またはインデックス付きテキスト配列の値）。

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: このテキストは、画像を宣言しない場合にボタンに表示されます。

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。 画像のピクセル透過が可能です。

x, y: ラジオコントロールの位置（左上の配置位置）。

width, height: ボタンの幅と高さ（ピクセル単位）。画像サイズは個別に宣言しません。画像は収まるように自動的にストレッチされないので、ボタンに合わせ、ボタンの中央に配置する必要があります。

UI_PICT_PUSHCHECKBUTTON

UI_PICT_PUSHCHECKBUTTON name, text, picture_reference,
frameFlag, x, y, width, height [UI_TOOLTIP tooltip]

UI_PICT_PUSHCHECKBUTTON{2}

UI_PICT_PUSHCHECKBUTTON{2} "name", text, picture_reference,
frameFlag, x, y, width, height [UI_TOOLTIP tooltip]

互換性：ARCHICAD 22で導入されました。

ブールパラメータ用のアイコン付きのプッシュチェックボタンを1つ生成します。

メソッド6を使用する「UI_INFIELD{3}」と同様に、追加のオプションを使用してボタンフレームの可視性を制御します。

name: UI_PICT_PUSHCHECKBUTTONの場合はパラメータ名または文字列式としての名前、UI_PICT_PUSHCHECKBUTTON{2}の場合は文字列式としてのパラメータ名（またはインデックス付きテキスト配列の値）。

text: このテキストは、画像を宣言しない場合にボタンに表示されます。

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。 画像のピクセル透過が可能です。

frameFlag: 1 - フレームを表示、0 - フレームを非表示。このオプションを使用して、コントロールをスタイルの他のユーザーインターフェースに合わせます。

x, y: ボタンの位置（左上の配置位置）。

width, height: ボタンの幅と高さ（ピクセル単位）。画像サイズは個別に宣言しません。画像は収まるように自動的にストレッチされないので、ボタンに合わせ、ボタンの中央に配置する必要があります。

UI_TEXTSTYLE_INFIELD

UI_TEXTSTYLE_INFIELD name, faceCodeMask, x, y,
buttonWidth, buttonHeight[, buttonOffsetX]

UI_TEXTSTYLE_INFIELD{2}

UI_TEXTSTYLE_INFIELD{2} "name", faceCodeMask, x, y,
buttonWidth, buttonHeight [, buttonOffsetX]

互換性：ARCHICAD 22で導入されました。

整数パラメータを使用して、一般的なプログラムインターフェースに表示されるのと類似した外観で、フォントスタイルの設定専用のプッシュチェックボタンの行を生成します。 設定される値の形式は、「DEFINE STYLE{2}」の入力パラメータと一致します。 アイコンとツールチップはどちらも、ローカライズバージョンに従ってARCHICAD自体から参照されます。 有効なボタンは一行に配列されて表示されます。

name: UI_TEXTSTYLE_INFIELDの場合はパラメータ名または文字列式としての名前、UI_TEXTSTYLE_INFIELD{2}の場合は文字列式としてのパラメータ名（またはインデックス付きテキスト配列の値）。

faceCodeMask: 使用されるビットで、一致するフォントスタイルオプションをコントロールに追加します。
faceCodeMask = j_1 + 2*j_2 + 4*j_3 + 32*j_6 + 64*j_7 + 128*j_8, ここで、 各 jiフラグは0または1をとります.
j_1: 太字
j_2: 斜体
j_3: 下線
j_6: 上付き文字
j_7: 下付き文字
j_8: 取り消し線
faceCodeMask = 0の場合、有効な全てのフォントスタイルボタンが表示されます。無効なfaceCodeMaskの場合、[スクリプトを確認]が警告（「無効なマスク値が使用されています」）を返します。

x, y: 最初のボタンの位置（左上の配置位置）。

buttonWidth, buttonHeight: 1つのボタンの幅と高さ（ピクセル単位）。必要に応じて、faceCodeMask、buttonWidth、buttonOffsetXの値を使用して全体の幅を計算できます。

buttonOffsetX: 行の隣接するボタン間の距離（単位：ピクセル）。設定しない場合は自動。

UI_LISTFIELD

UI_LISTFIELD fieldID, x, y, width, height [, iconFlag [, description_header [, value_header]]]

任意のパラメータの設定ができるスクロールリストを作成します。 設定項目はアイコン、説明、パラメータ値の3つです。 リストの線はUI_LISTITEM コマンドを利用して定義することができます。UI_LISTFIELD と UI_LISTITEM 定義は任意の順番でスクリプトすることができます。 空のlistfieldsは（リストアイテムのない） 表示されません。

fieldID: listfieldのユニークID。このIDは UI_LISTITEM コマンドでも使われ、listitemsが属するlistfield を指定します。 ユーザーインターフェイス内でのスクリプトの複製はできません。

x, y: listfieldの左上角の位置。

width, height: 幅と高さ （単位： ピクセル）

iconFlag:

iconFlag = 0: このlistfieldのアイコン列は作成されません。

iconFlag = 1: このlistfieldのアイコン列は作成されます（指定されていない場合はディフォルト値）。

もしカスタム設定パネルに1つしかコントロールがない場合、このコントロールは listfieldです。x、y、幅、高さのパラメータには影響しません。 この場合、listfieldの幅はカスタム設定パネルと同じです。

description_header: 説明列のタイトル。

value_header: 値列のタイトル。

description_header と value_headerが両方とも空文字列または指定されていない場合、listfieldはヘッダーなしで生成されます。文字列に1つ以上スペースが含まれる場合、、listfieldは空のヘッダーで生成されます。

UI_LISTITEM

UI_LISTITEM itemID, fieldID, "name" [, childFlag [, image [, paramDesc]]]

UI_LISTITEM{2}

UI_LISTITEM{2} itemID, fieldID, name [, childFlag [, image [, paramDesc]]]

listitemをfieldIDパラメータで定義された listfield に追加。

itemID: listfieldのユニークID。Listitemsを任意の順番にitemIDでソートしてスクリプトが可能です。 listfield内での listitem ID を複製することは許可されていません。

fieldID: このlistitemを含むlistfieldのユニークID。

name: UI_LISTITEMの文字列としてのパラメータ名、またはUI_LISTITEM{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。

childFlag:

childFlag = 0: listitemはgroupitemです（指定されていない場合はディフォルト値）。
childFlag = 1: listitemはchilditemです。親項目は上の最初のgroupitemです。

image: 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。 有効な場合、関連するlistitemの行の、最初の列にアイコンが表示されます。

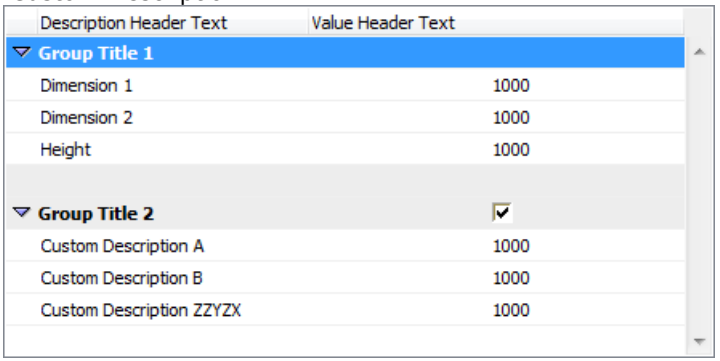
paramDesc: 説明列のlistitemに表示される名前。 空の場合、説明文はライブラリ部品のパラメータリスト説明文から自動的に埋まります。 説明文がない場合、パラメータ名が代わりに表示されます。

"名前" 文字列が空の場合、listitemは太字のフォントでグループになります。 "name" 文字列とparamDescの両方が空の場合、listitemはセパレータです。 HIDEPARAMETER コマンドはリストアイテムには無効です、スクリプトは項目を追加しません。 LOCKコマンドは使用でき、リストアイテムに有効です。

listfield では、異なるパラメータ、グループやセパレータに対して、異なる itemID を定義することをお薦めします。

例:

```
! 列アイコンなしのヘッダー付きリスト

ui_listfield 1, 10, 35, 432, 220, 0, "Description Header Text", "Value Header Text"

ui_listitem 1, 1, "", 0, "", "Group Title 1" ! Group Line
ui_listitem 2, 1, "A", 1
ui_listitem 3, 1, "B", 1
ui_listitem 4, 1, "ZZYZX", 1

ui_listitem 5, 1, "" !separator
ui_listitem 6, 1, "AC_show2DHotspotsIn3D", 0, "", "Group Title 2" ! Group Parameter Line
ui_listitem 7, 1, "A", 1, "", "Custom Description A"
ui_listitem 8, 1, "B", 1, "", "Custom Description B"
ui_listitem 9, 1, "ZZYZX", 1, "", "Custom Description ZZYZX"
```



UI_CUSTOM_POPUP_LISTITEM

UI_CUSTOM_POPUP_LISTITEM itemID, fieldID, "name", childFlag, image, paramDesc,
storeHiddenId, treeDepth,
groupingMethod, selectedValDescription,
value1, value2, valuesArray1, ... valuen, valuesArrayn

UI_CUSTOM_POPUP_LISTITEM{2}

UI_CUSTOM_POPUP_LISTITEM{2} itemID, fieldID, name, childFlag, image, paramDesc,
storeHiddenId, treeDepth,
groupingMethod, selectedValDescription,
value1, value2, valuesArray1, ... valuen, valuesArrayn

互換性：ARCHICAD 20で導入されました。

"UI_CUSTOM_POPUP_INFIELD"および"UI_CUSTOM_POPUP_INFIELD{2}"と同様

パラメータスクリプトの使用を回避するために、ユーザーインターフェイススクリプトで定義されたパラメータの値リストをポップアップ表示するlistitemを生成します。

パラメータスクリプトで要求できないリストに最適です。 パラメータスクリプトの制限事項については、「REQUESTオプション」を参照してください。

itemID: listfieldのユニークID。Listitemsを任意の順番にitemIDでソートしてスクリプトが可能です。 listfield内での listitem ID を複製することは許可されていません。

fieldID: このlistitemを含むlistfieldのユニークID。

name: UI_CUSTOM_POPUP_LISTITEMの文字列式としてのパラメータ名、またはUI_CUSTOM_POPUP_LISTITEM{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。

childFlag:

childFlag = 0: listitemはgroupitemです（指定されていない場合はディフォルト値）。

childFlag = 1: listitemはchilditemです。親項目は上の最初のgroupitemです。

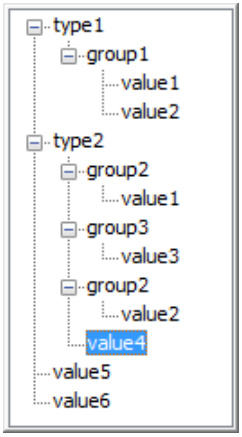
image: ライブラリ部品に格納される画像ファイル名、またはインデックス番号。 有効な場合、関連するlistitemの行の、最初の列にアイコンが表示されます。

paramDesc: 説明列のlistitemに表示される名前。 空の場合、説明文はライブラリ部品のパラメータリスト説明文から自動的に埋まります。 説明文がない場合、パラメータ名が代わりに表示されます。

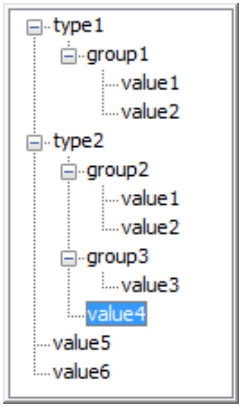
storeHiddenId, treeDepth: 自動ツリーまたは手動ツリーを設定します。
storeHiddenId = 0, treeDepth = 0: 配列パラメータでのみ機能します。

"treeDepth"パラメータは、配列の2番目の次元（列数）によって自動的に設定されます。
storeHiddenId = 1, treeDepth > 0: 1つのパラメータでのみ機能します。
n * (1 + treeDepth)値を定義する必要があります （最初の値は保存されたID用、残りはカスタムツリーの定義用）。

groupingMethod: ツリーをソートするグループ化方式
1: 同じ親に属するグループおよび値をソートしません。



2: 同じ親に属するグループおよび値をソートします。

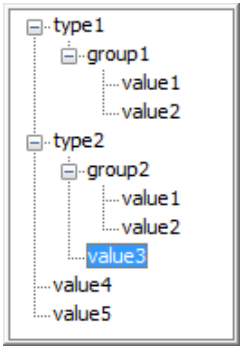


selectedValDescription: フィールドに書き込むテキスト、空白の文字列の場合、テキストは選択された項目の保存されたIDになります。

valuei, valuesArrayi: ツリー値を1つずつおよび/または1次元配列で定義します。

例:

```
UI_CUSTOM_POPUP_LISTITEM itemID, fieldID, "stParameterName", 0, "", "",
    1, 3, 2, "",    ! storeHiddenId, treeDepth, groupingMethod, selectedValDescription
    "hiddenID1",    "type1",    "group1",   "value1",
    "hiddenID2",    "type1",    "group1",   "value2",
    "hiddenID3",    "type2",    "group2",   "value1",
    "hiddenID4",    "type2",    "group2",   "value2",
    "hiddenID5",    "type2",    "",         "value3",
    "hiddenID6",    "",         "",         "value4",
    "hiddenID7",    "",         "",         "value5"
```



UI_TOOLTIP

UI_BUTTON type, text, x, y, width, height [, id [, url]] [**UI_TOOLTIP** tooltiptext]
UI_PICT_BUTTON type, text, picture_reference,
x, y, width, height [, id [, url]] [**UI_TOOLTIP** tooltiptext]
UI_INFIELD "name", x, y, width, height [, extra parameters ...]
[**UI_TOOLTIP** tooltiptext]
UI_INFIELD{2} name, x, y, width, height [, extra parameters ...]
[**UI_TOOLTIP** tooltiptext]
UI_INFIELD{3} name, x, y, width, height [, extra parameters ...]
[**UI_TOOLTIP** tooltiptext]

UI_INFIELD{4} "name", x, y, width, height [, extra parameters ...]
[**UI_TOOLTIP** tooltiptext]
UI_CUSTOM_POPUP_INFIELD "name", x, y, width, height , extra parameters ...
[**UI_TOOLTIP** tooltiptext]
UI_CUSTOM_POPUP_INFIELD{2} name, x, y, width, height , extra parameters ...
[**UI_TOOLTIP** tooltiptext]
UI_RADIOBUTTON name, value, text, x, y, width, height [**UI_TOOLTIP** tooltiptext]
UI_OUTFIELD expression, x, y, width, height [, flags] [**UI_TOOLTIP** tooltiptext]
UI_PICT expression, x, y [, width, height [, mask]] [**UI_TOOLTIP** tooltiptext]
UI_LISTFIELD fieldID, x, y, width, height [, iconFlag [, description_header [, value_header]]]
[**UI_TOOLTIP** tooltiptext]
UI_LISTITEM itemID, fieldID, "name" [, childFlag [, image [, paramDesc]]]
[**UI_TOOLTIP** tooltiptext]
UI_LISTITEM{2} itemID, fieldID, name [, childFlag [, image [, paramDesc]]]
[**UI_TOOLTIP** tooltiptext]
UI_CUSTOM_POPUP_LISTITEM itemID, fieldID, "name", childFlag , image , paramDesc,
extra parameters ...
[**UI_TOOLTIP** tooltiptext]
UI_CUSTOM_POPUP_LISTITEM{2} itemID, fieldID, name, childFlag , image , paramDesc,
extra parameters ...
[**UI_TOOLTIP** tooltiptext]

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 button、infield、outfield、listfield、listitem、およびpictureコマンドで使用できます。ただし、ユーザーが実行中のコンテキスト（ARCHICADのヘルプメニューなど）でこれを無効にしている場合は使用できません。

項目が宣言されていない場合、listfieldのツールチップは含まれるlistitemの全てに表示されます。 listitemのツールチップは、listfieldインラインのツールチップの影響を引き継ぎます（存在する場合） 。

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

UI_COLORPICKER

UI_COLORPICKER "redParamName", "greenParamName", "blueParamName", x0, y0 [, width [, height]]

UI_COLORPICKER{2}

UI_COLORPICKER{2} redParamName, greenParamName, blueParamName, x0, y0 [, width [, height]]

カラーピッカーダイアログ r, g, b でコンポーネントの色、与えられたパラメータにそれらを格納します。これらの値は後ほどLIGHT コマンドで使用できます。

redParamName, greenParamName, blueParamName: UI_COLORPICKERの文字列式としてのパラメータ名、または UI_COLORPICKER{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。

x0, y0: カラーピッカーの左上角の位置。

width, height: 幅と高さ （単位： ピクセル）

UI_SLIDER

UI_SLIDER "name", x0, y0, width, height [, nSegments [, sliderStyle]]

UI_SLIDER{2}

UI_SLIDER{2} name, x0, y0, width, height [, nSegments [, sliderStyle]]

範囲で定義された整数パラメータのためのスライダーコントロールを生成します。 整数パラメータの未定義の範囲の下限値と上限値は-32768 (minimum signed short) と32767 (maximum signed short)です。

name: 文字列式のパラメータとしてパラメータ名または、省略可能なインデックス値を持つUI_SLIDER{2}。

x0, y0: スライダーの位置

width, height: スライダー幅と高さ （単位： ピクセル） 幅が高さ以上の場合、スライダーは水平です。逆の場合は、垂直です。

nSegments: スライダーのオプションセグメント数。0の場合、セグメントは表示されず、省略または負数の場合は、セグメント数は下限値と上限値の範囲とパラメータで設定されたステップで計算されます。

sliderStyle: オプションスライダースタイル（ディフォルトは0）

0: 下までのスライダーポイント（水平スライダー） または、右まで（垂直スライダー）。
1: 上までのスライダーポイント（水平スライダー） または、左まで（垂直スライダー）。

上位移行スクリプト

より新しいライブラリで要素を完全に変更する場合、移行ロジックを定義することにより互換性を維持することができます。 詳細については、「上位移行スクリプト」をご覧になってください。

例:

```
actualGUID = FROM_GUID

! ============================================================================
! Subroutines
! ============================================================================

    _startID    = "AAAA-AAAA-...AAA"
    _endID      = "BBBB-BBBB-...BBB"
gosub "migrationstepname_FWM"

! ============================================================================
! Set Migration GUID
! ============================================================================

setmigrationguid actualGUID

! ============================================================================
end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end
! ============================================================================

! ============================================================================
! migrationstepname
! ============================================================================
"migrationstepname_FWM":
    if actualGuid = _startID then
        newParameter = oldParameter
        parameters newParameter = newParameter
        actualGuid = _endID
    endif
return
```

FROM_GUID は、移行が実行されるオリジナルのオブジェクトのメインIDを保持するグローバル変数です。
スクリプトが成功した場合は、インスタンスは更新されたパラメータを含む新規要素に置換されます。

SETMIGRATIONGUID

SETMIGRATIONGUID guid

このコマンドは、どの要素が現在のオブジェクトに一致する移行要素になるかを実行環境に通知します。返されたIDが現在の要素に含まれる場合、オブジェクトの移行は完了します。

STORED_PAR_VALUE

STORED_PAR_VALUE ("oldparname", outputvalue)

移行されるオブジェクトに存在するが、存在または新しいバージョンのオブジェクト削除されたパラメータの値を検索します。 このコマンドフォームは、新しいオブジェクトに存在する物のパラメータにお勧めです。 古い配列パラメータの値を取得するには、outputvalueパラメータを配列として初期化する必要があります（dimコマンドを使用します）。

oldparname: 文字列式、古いパラメータリストのパラメータの名前。

outputvalue: パラメータの値を保存するための出力変数。

戻り値：1は成功、 それ以外は 0 （古いオブジェクトのパラメータリストにその名前のパラメータがない場合など）。 スクリプトのチェック中は、 古いパラメータセクションが認識されないため、 戻り値は常に0です。

DELETED_PAR_VALUE

DELETED_PAR_VALUE ("oldparname", outputvalue)

移行されるオブジェクトに存在するが、存在または新しいバージョンのオブジェクト削除されたパラメータの値を検索します。 このコマンドフォームは、新しいオブジェクトに存在する物のパラメータにお勧めです。 古い配列パラメータの値を取得するには、outputvalueパラメータを配列として初期化する必要があります（dimコマンドを使用します）。

oldparname: 文字列式、古いパラメータリストのパラメータの名前。

outputvalue: パラメータの値を保存するための出力変数。

戻り値：1は成功、 それ以外は 0 （古いオブジェクトのパラメータリストにその名前のパラメータがない場合など）。 スクリプトのチェック中は、 古いパラメータセクションが認識されないため、 戻り値は常に0です。

下位移行スクリプト

下位移行スクリプトを使用して、新しいオブジェクトのインスタンスを古いオブジェクトのインスタンスに変換する逆方向の変換ロジックを定義することができます。 詳細については、「下位移行スクリプト」をご覧になってください。

例:

```
targetGUID = TO_GUID

! ============================================================================
! Subroutines
! ============================================================================

gosub "migrationstepname_BWM"

! ============================================================================
! Set Migration GUID
! ============================================================================

setmigrationguid targetGUID

! ============================================================================
end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end
! ============================================================================


! ============================================================================
! migrationstepname
! ============================================================================
"migrationstepname_BWM":
    if targetGUID # "" then
        bMigrationSuccess = 1
        if bMigrationSuccess = 1 then
            oldParameter = newParameter
            parameters oldParameter = oldParameter
        else
            targetGuid = ""
        endif
    endif
return
```

TO_GUID は、変換時にターゲットの要素のメインIDを保持するグローバル変数です。
targetGUIDの設定には 「SETMIGRATIONGUID」 を使用します。

NEWPARAMETER

NEWPARAMETER "name", "type" [, dim1 [, dim2]]

新しいパラメータを、下位移行スクリプトのライブラリ部品のパラメータに追加します。パラメータの作成は、スクリプトが完全に解釈された後にのみ行われます。 指定した名前のパラメータが既にパラメータリストに存在する場合、エラーが発生します。

name: 文字列式、作成されるパラメータ名。

type: 文字列式、パラメータのタイプ。有効値は次のとおりです。
- Integer
- Length
- Angle
- RealNum
- LightSwitch
- ColorRGB
- Intensity
- LineType
- Material
- FillPattern
- PenColor
- String
- Boolean
- BuildingMaterial（互換性：ARCHICAD 22で導入されました。）
- Profile（互換性：ARCHICAD 22で導入されました。）
- Dictionary（互換性：ARCHICAD 24で導入されました。）

dim1, dim2: dim1は、0でない場合、パラメータの第1寸法です。dim2は、0でない場合、パラメータの第2寸法です。
- dim1 = 0, dim2 = 0: の場合パラメータはスカラーパラメータ、
- dim1 > 0, dim2 = 0: の場合パラメータは 1次元配列、
- dim1 > 0, dim2 > 0: の場合パラメータは 2次元配列、

パラメータの制限:
- If dim2 > 0, then dim1 > 0.

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。

DICT

DICT variableName1[, variableName2...]

互換性：ARCHICAD 23で導入されました。

GDLは辞書をサポートしています。変数は上記の宣言ステートメントの後で辞書として宣言されます（配列や単純タイプに変更することはできません。逆の場合も同様です）。ライブラリ部品パラメータを辞書にすることもできます。その場合は、パラメータリストで辞書タイプを選択します。

DICTキーワードの後に、任意の数の変数名をコンマで区切って入力できます。各変数に階層構造のキーと値のペアが含まれます。辞書のキーはドット表記で参照できます。キーのフルパスは255文字を超えてはいけません（配列インデックスは1文字としてカウントします）。

辞書と単純タイプの値：

- 単純タイプ（文字列、整数、浮動小数点型）の値を辞書キーに割り当てることはできません。
- 宣言は不要で、キーの値タイプは現在の値によって設定されます。

```
DICT myDictionary

myDictionary.element1 = 1
myDictionary.element1 = "hello"

print myDictionary
```

辞書と派生タイプ値：

- 辞書は配列タイプ（1次元のみ）と辞書タイプのキーをネストすることができます。
- 辞書内部の配列には、名称未設定の辞書または単純タイプ（インデックスで参照）を含めることができます。
- ただし、単独の配列タイプパラメータ/変数に辞書タイプ要素を含めることはできません。
- ネストされた配列キーはそれを直接参照することで初期化でき、この場合はDIMで宣言する必要はありません。
  ```
  DICT myDictionary
  myDictionary.myArray[1] = 1
  myDictionary.myArray[2] = 5

  print myDictionary
  ```
- ネストされた配列の参照されていないインデックスは、最初に参照された配列要素のタイプ（文字列キーは""、数値キーは0、辞書キーは{}）に従って自動的に初期化されます。
  ```
  DICT myDictionary
  DICT dictForNesting
  dictForNesting.elem1 = "hello"
  dictForNesting.elem2 = "world"

  myDictionary.myArray[2] = dictForNesting

  print myDictionary

  DICT myDictionary2
  myDictionary2.myArray[3] = 33
  myDictionary2.myArray[4] = 44

  print myDictionary2
  ```
- ネストされた配列の値は、同じタイプ（全て文字列、全て整数、全て浮動小数点型、または全て辞書タイプ）にする必要があります。これは配列の仕組みとは異なりますので、特に注意が必要です。
  ```
  DICT myDictionary2
  myDictionary2.myArray[1] = 1
  myDictionary2.myArray[2] = 1.0 ! GDLエラー
  ```
- ネストされた配列の値タイプを変更するには、まずリセットする必要があります。つまり、空の配列を作成して、ネストされた配列をこの新しい空の配列で上書きします。次に参照される値のタイプは、リセット後の配列のタイプに自動的に設定されます。
  ```
  DICT myDictionary
  myDictionary.myArray[1] = "hello"
  print myDictionary

  DIM arrayForReset[]
  myDictionary.myArray = arrayForReset
  print myDictionary

  myDictionary.myArray[1] = 10000
  print myDictionary
  ```

- ネストされた配列の最初の値のタイプを変更しても、配列にその値しか含まれていない場合は、配列のタイプは変更されません。
```
DICT myDictionary
myDictionary.myArray[1] = "hello"
print myDictionary

myDictionary.myArray[1] = 10000 ! GDLエラー
print myDictionary
```

初期化とコピー：

- 辞書の最初の参照はDICT dictNameにする必要があり、辞書の名前を含むサブルーチンをその前に配置することはできません（DIM配列の場合と同じ）。
- 初めてキーを使用する場合は、まず初期化する必要があります。これはキーに割り当てることで明示的に、または適切な深さのキーを含む辞書を割り当てることで暗黙的に行います。
```
DICT myDictionary

myDictionary.level1.a = 1
myDictionary.level1.b = 2
myDictionary.level2 = myDictionary.level1

print myDictionary.level2.b
```
- 実際の内部キーを指定せずに辞書名を記述すると、状況によっては受け入れられる辞書構造全体が参照されます（CALL、PRINT、LETのステートメントの場合）。
- 構造の一部を記述すると、そのキーの下のサブツリーが辞書として参照されます。
```
DICT myDictionary
myDictionary.point1.x = 1
myDictionary.point1.y = 1
myDictionary.point1.type = 0
print myDictionary

DICT myPoint
myPoint = myDictionary.point1
print myPoint

myDictionary.point2 = myDictionary.point1
print myDictionary
```
- 辞書の全てまたは一部を割り当てると、右側または左側のディープコピーが作成されます。

```
DICT myDictionary, tempPoint
tempPoint.x = 1
tempPoint.y = 1
myDictionary.line.point1 = tempPoint

tempPoint.x = 2
tempPoint.y = 2
myDictionary.line.point2 = tempPoint

DICT myLine
myLine = myDictionary.line
myDictionary.line.point2.x = 0
print myLine
```

マクロコールと要求：

- マクロコールでは、受信側に辞書タイプパラメータがある場合、辞書タイプの値をマクロに送信できます。
- RETURNED_PARAMETERSを辞書で使用できます。この場合、受信側（呼び出しオブジェクト）で空のDICTを宣言する必要があります。次のコードでは、myDictionaryはマクロ内の辞書タイプパラメータで、_myDictionaryは呼び出しオブジェクト側の辞書タイプ変数です。

```
! 呼び出しオブジェクトのマスタースクリプト
DICT _myDictionary

_myDictionary.element1 = 1
_myDictionary.element2 = 2

DICT _dictForReceivedData

call "macroname" parameters all myDictionary = _myDictionary,
    returned_parameters _dictForReceivedData

print _dictForReceivedData

! マクロオブジェクトのマスタースクリプト
myDictionary.element1 = myDictionary.element1 * 2
myDictionary.element2 = myDictionary.element2 * 2

end myDictionary
```

- REQUESTオプションには、現時点では辞書をサポートする要求はありません。ただし、今後追加される可能性があります。
- LIBRARYGLOBAL要求は辞書タイプの値を返すことができません。

ビジュアリゼーションと関数：

- 辞書タイプパラメータは、[設定]ダイアログの[全てのパラメータ]ページには表示されません。
- 辞書タイプパラメータは、リスト表示またはIFCマッピングには使用できません。

- テキスト形式の表示は、「PRINT」（[スクリプトを確認]警告およびJSON形式でのレポートウィンドウへの出力）でのみ可能です。
- TEXT2、RICHTEXT2などの一般的なテキスト処理コマンドは、完全な辞書をサポートしていません。
- ただし、ネストされた非辞書タイプの値は、これらで表示できます。
  ```
  DICT myDictionary
  myDictionary.myArray[1] = "hello"
  myDictionary.myArray[2] = "world"

  text2 0, 0, myDictionary.myArray[1] + " " + myDictionary.myArray[2]
  print myDictionary
  ```
- 辞書タイプパラメータの値は、パラメータスクリプトでのみ設定できます（ユーザーがパラメータリストまたはUIコントロールから直接入力することはできません）。「VALUES」はこのタイプでは無効です。
- ただし、ユーザー入力に非辞書タイプパラメータを使用することはできます。次のコードでは、myDictionaryは辞書タイプパラメータで、stTextInputは文字列タイプパラメータです（これはユーザーインターフェースで使用でき、[全てのパラメータ]ページに表示され、GLOB_MODPAR_NAMEと一緒に使用できます）。
  ```
  ! パラメータスクリプト
  myDictionary.text1 = stTextInput
  parameters myDictionary = myDictionary

  ! マスタースクリプト
  print myDictionary

  ! 2Dスクリプト
  TEXT2 0, 0, myDictionary.text1
  ```
- LP_XMLConverterツールを使用した値の置換は、現時点では辞書タイプパラメータには使用できません。

HASKEY

HASKEY (dictionary.key)
キーが以前に辞書で定義されているかどうかをブールで返します（キーにはサブキーを含めることができます）。

例:

```
DICT myDictionary
myDictionary.point[1].x = 1
myDictionary.point[1].y = 1

print HASKEY(myDictionary.point)        ! true
print HASKEY(myDictionary.point[2])     ! false
print HASKEY(myDictionary.point[1].z)   ! false
```

REMOVEKEY

REMOVEKEY (dictionary.key)

この関数は、参照されたキーを、割り当てられた値とともに辞書から削除します。削除が正常に完了した場合、戻り値は1で、そうでない場合は0です（キーが存在しないまたは既に削除されている場合）。

例:

```
DICT myDictionary
myDictionary.myText[1] = "hello"
myDictionary.myOtherText[1] = "world"

print myDictionary

_dummy = REMOVEKEY(myDictionary.myOtherText)
print myDictionary, _dummy

_dummy2 = REMOVEKEY(myDictionary.myNonExistentText)
print myDictionary, _dummy2
```

DIM

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

GDL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、次元が指定される上記の宣言ステートメントの後で配列になります（辞書タイプの変数を配列として宣言しなおすことはできません。その逆も同様です）。

キーワードDIMの後に、コ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

VARDIM1 (expr)

VARDIM2

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次元の配列の場合、VARDIM2は0を返します。

例 1: 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
```

例 2: 文字列式の例：

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

例 3: 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、3要素の配列
v2[2][3] = 23 ! その他全ての要素(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
```

例 4: 辞書値を使用する式の例：

```
DICT _exampleDict

! DICT simple key types

_exampleDict.false = (1 = 2) ！論理的な偽（内部的に整数）
_exampleDict.true = (1 = 1) ！論理的な真（内部的に整数）
_exampleDict.int = 2 ！整数
_exampleDict.float = 1 / 3 ！浮動小数点型
_exampleDict.string = "Custom text" ！文字列

! DICT配列キータイプ
 ！その都度配列を初期化
_exampleDict.array[1] = _exampleDict.float
 ！その都度配列に追加
_exampleDict.array[2] = _exampleDict.float * 2
 ！中間要素の自動初期化で配列に追加
_exampleDict.array[4] = _exampleDict.float * 3

! DICT array of DICTs

DIM array[]
DICT _element
_element.a = "A"
_element.b = 1

_exampleDict.array = array ！既存の配列を空に配列
_exampleDict.array[2] = _element ！以前と異なる値タイプ

! print DICT array of DICTs
print "¥n¥t",
  "Print DICT array of DICTs",
  "¥n--------------------------------------------------------------------------------¥n¥t",
  vartype(_exampleDict.array), vardim1(_exampleDict.array),  "¥n",
  _exampleDict.array,
  "¥n--------------------------------------------------------------------------------"
```

PARVALUE_DESCRIPTION

PARVALUE_DESCRIPTION (parname [, ind1 [, ind2]])

この関数は、 「VALUES」 コマンドンを使用して、指定された数値パラメータのパラメータ値の説明文字列を返します。 説明文が指定されていない場合、戻り値は空の文字列になります。

parname: パラメータ名

ind1, ind2: 実際のインデックスパラメータが配列の場合。

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（除算の余り） x MOD y = x - y * INT (x/y)	優先順位 3
+	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または0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。

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のアークコサインを返します。（-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。 クエリコールもGDL機能拡張との通信に使用することができます。

REQ

REQ (parameter_string)
プログラムの現在の状態を問い合わせます。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。

parameter_string: 質問文字列、以下のいずれかです。
"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。
"Program": 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。
これらのクエリの戻り値は常に、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クエリで部分的に定義されます。 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。
ARCHICADは、コマンドのバージョンまたは要求オプションの正確な名前（定数文字列として）のいずれかによって、入力パラメータの順序と数値を識別します。 現在使用できるバリエーションは次のとおりです。

- n = REQUEST - デフォルトの要求。文字列または数値型の1つの入力パラメータで構成されます
- n = REQUEST{2} - 2つの入力パラメータで構成されます：文字列または数値、文字列型
- n = REQUEST{3} - 2つの入力パラメータで構成されます：文字列、文字列または数値配列型
- n = REQUEST{4} - 3つの入力パラメータで構成されます：文字列または数値、数値、文字列型 互換性：ARCHICAD 21で導入されました。

使用できるオプションのリストは、「REQUESTオプション」を参照してください。

IND

IND (**MATERIAL**, name_string)
IND (**BUILDING_MATERIAL**, name_string)
IND (**FILL**, name_string)
IND (**LINE_TYPE**, name_string)
IND (**STYLE**, name_string)
IND (**TEXTURE**, name_string)
IND (**PROFILE_ATTR**, name_string, index)
この関数は、材質、ビルディングマテリアル、塗りつぶし、線種またはスタイル、テクスチャまたは断面形状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。
IND関数は属性のインデックス（整数）値を返します。 結果は、インライン定義（スクリプト内でまたはMaster_GDLファイルから）の場合は負でグローバル定義（プロジェクト属性から）の場合は正です。
「「インライン属性定義」」も参照してください。

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クエリ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。 「「アプリケーションクエリオプション」」も参照してください。

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。定数文字列にする必要があります。 警告：文字列変数またはパラメータをオブジェクト名として使用すると、このライブラリグローバルオブジェクトをクエリするオブジェクトの2Dおよび3D表示は自動的に更新されません。

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
    text2 0, 0, "Not available"
endif
```

文字列関数

STR

STR (numeric_expression, length, fractions)
STR (format_string, numeric_expression)
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。

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）。

パラメータの制限:
length >= -100, length <= 100
fractions <= 20, fractions < length

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
```

STR{2}

STR{2} (format_string, numeric_expression [, extra_accuracy_string])
STRの2番目のフォームの拡張版。 format_stringに追加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: "%[0 or more flags][field_width][.precision] conv_spec"

flags: （m、mm、cm、dm、
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': 表示された小数点以下の範囲内で.5に丸め処理、正確な文字列を戻さない、（面積計算に使用）、
'*6': 表示された小数点以下の範囲内で.25に丸め処理、正確な文字列を戻さない、（面積計算に使用）、
'*7': fiまたはffiの場合、extra_accuracy_stringにnumeric_expressionの小数部を埋めますが、ファンクションの返された式には小数部を含みません。
'#': は0を非表示にします（m、mm、cm、dm、
ffi、fdi、fi、df、di、sqm、sqcm、sqf、sqi、dd、fr、rad、cum、l、cucm、cumm、cuf、cui、cuy、galの場合）、
'0': 0 インチを表示（ffi、fdi、fi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）。
（m、mm、cm、dm、fdi、df、di、sqm、sqcm、sqf、sqi、dd、fr、rad、cum、l、cucm、cumm、cuf、cui、cuy、galの場合）、

'^': 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）。
'[1*j1+2*j2+4*j3]': 分数の前に0フィートと0インチを表示、 '0' フラグが指定されてない場合に有効（ffi、fdi、fiの場合）
j1: 分数の前に0インチを表示（1'-0 3/4"）
j2: 0インチを表示（1'-0"）
j3: 分数の前に0フィートを表示（0 3/4"）

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
dm: デシメートル
互換性：デシメートルはARCHICAD 22で導入されました。
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:

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

```
alpha = 88.657
TEXT2 0,  5, STR ("%+10.3dd", alpha) !+88.657°
TEXT2 0,  4, STR ("%.1dms", alpha)   !88°39'
TEXT2 0,  3, STR ("%.2dms", alpha)   !88°39'25"
TEXT2 0,  2, STR ("%10.4gr", alpha)  ! 98.5078G
TEXT2 0,  1, STR ("%rad", alpha)     !1.55R
TEXT2 0,  0, STR ("%.2surv", alpha)  !N 1°20'35" E

nr  = 1'-0 3/4"
TEXT2 0, -1, STR ("%[1].16ffi", nr) !1'-0 3/4"
nr = 1'-0"
TEXT2 0, -2, STR ("%[5].16ffi", nr) !1'
nr = 0 3/4"
TEXT2 0, -3, STR ("%#[7].16ffi", nr) !0 3/4"

nr = 0.34278
TEXT2 0, 0, STR ("%*5 .4m", nr)  !0.3430

! 整数部分と小数部分に分割
extra_accuracy_string = ""
nr  = 1'-0 3/4"

TEXT2 0, -3, STR{2}("%*7.16ffi", nr,  extra_accuracy_string) !1'-
TEXT2 0, -4, extra_accuracy_string        !3/4"
```

SPLIT

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

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または%^nは数値を表します。「^」フラグが存在する場合、実際の数値を一致させるときに、小数区切り文字や桁集合化文字についての現在のシステム設定が考慮されます。

variablei: 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" "Gabriola", 180000 / GLOB_SCALE, 1, 0
SET STYLE "own"
string = "abcd"
width = STW (string) / 1000 * GLOB_SCALE
n = REQUEST ("Height_of_style", "own", height)
height = height / 1000 * GLOB_SCALE
TEXT2 0,0, string
RECT2 0,0, width, -height
```

STRLEN

STRLEN (string_expression)
文字列の（整数の）長さ（文字数）を返します。

STRSTR

STRSTR (string_expression1, string_expression2[, case_insensitivity])
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。

注記：*string_expression2*が空白の文字列の場合、関数は1を返します。

case_insensitivity:
0 or not set: 大文字と小文字の区別
1: 大文字と小文字の区別

例 1:

```
szFormat = ""
n = REQUEST ("Linear_dimension", "", szFormat)
unit = ""
IF STRSTR (szFormat, "m")  > 0 THEN unit = "m"
IF STRSTR (szFormat, "mm") > 0 THEN unit = "mm"
IF STRSTR (szFormat, "cm") > 0 THEN unit = "cm"
IF STRSTR (szFormat, "dm") > 0 THEN unit = "dm"
TEXT2 0, 0, STR (szFormat, a) + " " + unit !1.00 m
```

例 2:

```
STRSTR ("abcdefg", "BCdEf") = 0
STRSTR ("abcdefg", "BCdEf", 0) = 0
STRSTR ("abcdefg", "BCdEf", 1) = 2
```

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:

```
string = "Flowers.jpeg"
len = STRLEN (string)
iDotPos = STRSTR (string, ".")
TEXT2 0, -1, STRSUB (string, 1, iDotPos - 1) !Flowers
TEXT2 0, -2, STRSUB (string, len - 4, 5)     !.jpeg
```

STRTOUPPER

STRTOUPPER (string_expression)
大文字に変換した文字列を返します。

例:

```
_oldString = "flower"
_newString = STRTOUPPER (_oldString)    ! _newString will be "FLOWER"
```

STRTOLOWER

STRTOLOWER (string_expression)
小文字に変換した文字列を返します。

例:

```
_oldString = "FLOWER"
_newString = STRTOLOWER (_oldString)    ! _newString は "flower"に変換されます
```

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 - TO - NEXT

FOR variable_name = initial_value **TO** end_value [**STEP** step_value] **NEXT** variable_name

FOR ループの最初のステートメント。

NEXT 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。

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。

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。

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 1:

```
FOR i=1 TO 10 STEP 2
   PRINT i
NEXT i
```

例 2:

```
! 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

! 2nd
2:
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」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 - WHILE

DO [statment1
   statement2
   ...
   statementn]
WHILE condition

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 - ENDWHILE

WHILE condition **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 - UNTIL

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 - GOTO

IF condition **THEN** label
IF condition **GOTO** label
IF condition **GOSUB** label

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。THEN、GOTOまたはTHEN GOTOはこのコンテキストでは同じです。

例:

```
IF a THEN 28
IF i > j GOTO 200+i*j
IF i > 0 GOSUB 9000
```

IF - THEN - ELSE - ENDIF

IF condition **THEN** statement [**ELSE** statement]
IF condition **THEN**
    [statement1
    statement2
    ...
    statementn]
[**ELSE**
    statementn+1
    statementn+2
    ...
    statementn+m]
ENDIF

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 label

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 label

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 [v1, v2, ..., vn]
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。

注記：戻り要素の数は32767項目に制限されます。

「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。

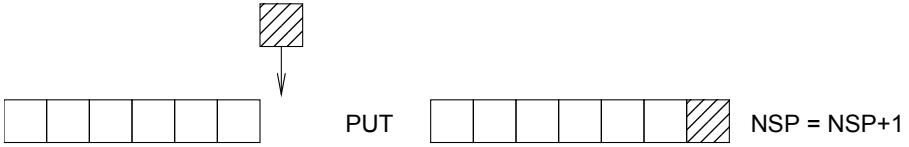
BREAKPOINT

BREAKPOINT expression

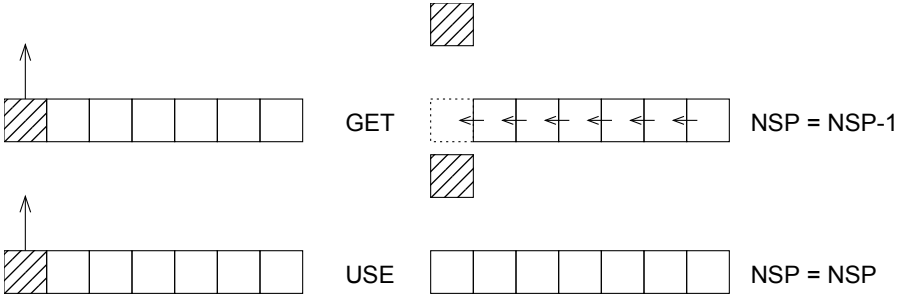
このコマンドでは、GDLスクリプト内にブレークポイントを指定することができます。 GDLデバッガは、パラメータの値（数式）が真（1）の場合はこのコマンドで停止し、デバッガの[ブレークポイントを有効に]オプションがオンにされます。 通常の実行モードでは、GDLインタプリタは単にこのコマンドを無視します。

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、コンマで区切られた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。

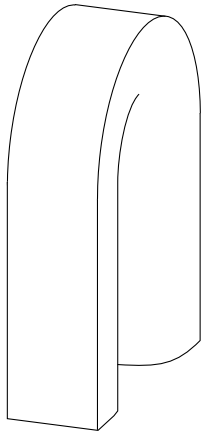
GET
GET (n)
内部パラメータバッファから次のn個の値を取り出して使い、その後それを削除します。

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。

例: 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。 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。

CALL

CALL macro_name_string [,]
PARAMETERS [**ALL**][name1=value1, ..., namen=valuen][[,]
RETURNED_PARAMETERS r1, r2, ...]

macro_name_string: 文字列、既存のライブラリ部品の名前

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。 GDLに依存性について知らせるには、可能な各マクロ名に対して「FILE_DEPENDENCE」を使用してください。 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。

PARAMETERS: 実際のマクロのパラメータリストを続けます

コールされたマクロのパラメータ名を、記号「=」と実際の値の両方を使用して、任意の順序でリストできます。 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。

ALL: コールする側の全てのパラメータをマクロに渡します

このキーワードが存在する場合、パラメータを1つずつ指定する必要はありません。 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。

RETURNED_PARAMETERS: 変数リストに続く、マクロをの戻り値を集めることができます。

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てください。 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。 注記：戻り要素の数は32767項目に制限されます。 「END / EXIT」の戻りパラメータの構文を参照してください 。

CALL macro_name_string [,]**PARAMETERS**
value1 or **DEFAULT** [, ..., valuen or **DEFAULT**]

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、 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n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。

CALL macro_name_string [, parameter_list]

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、 パラメータリストに1文字の数値パラメータ（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。 この方式では、文字列タイプの式または配列は使用できません。 パラメータリストは単純な数値のリストです。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。パラメータリストで指定されたA～Z値より少ない場合は、不足している値には自動的に0が使用されます。 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。

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
```

警告ボックスまたはレポートウィンドウへの出力

PRINT

PRINT expression [, expression, ...]

作業環境に応じて、ダイアログボックスまたはレポートウィンドウで、引数の全てを書き込みます（「GDL警告」を参照してください）。 引数は、コンマで区切られた、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取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、「GDL Text I/Oアドオン」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。

「「GDL Text I/Oアドオン」」も参照してください。

OPEN

OPEN (filter, filename, parameter_string)

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を識別します、アドオンが欠落している場合は-2、ファイルが欠落している場合は-1。 正数の場合、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」コマンドをファイル名と組み合わせて使用します。

filter: 文字列。既存の機能拡張の名前。

filename: 文字列。ファイル名。

parameter_string: 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。

INPUT

INPUT (channel, recordID, fieldID, variable1 [, variable2, ...])

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。

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。

VARTYPE

VARTYPE (expression)

式のタイプを返します。

- 1 - 数値
- 2 - 文字列
- 3 - グループ（「ADDGROUP」などの結果として）
- 4 - 辞書

「INPUT」コマンドを使って変数の値を読み取る際に役立ち、現在の値に従ってタイプ1とタイプ2を切り替えることができます。 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, ...]
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。

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。

CLOSE

CLOSE channel
channel値で識別されたファイルを閉じます。

決定的アドオンの使用

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。

「「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)
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。

extension: 文字列。既存の機能拡張の名前。

parameter_string1: 内容は機能拡張によって解釈されます。

parameter_string2: 内容は機能拡張によって解釈されます。

PREPAREFUNCTION

PREPAREFUNCTION channel, function_name, expression1 [, expression2, ...]
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。

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

CALLFUNCTION

CALLFUNCTION (channel, function_name, parameter, variable1 [, variable2, ...])
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。

channel: チャネル値、接続を識別するのに使用されます。
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。内容は機能拡張によって解釈されます。
variablei: 出力パラメータ

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。 固定パラメータ名のリストと詳しい情報は 「固定名パラメータ」をご覧ください。

スクリプトの互換性

パラメータスクリプトの実行の機会と結果のパラメータ値がコンテキスト依存になり、プランファイル内で一貫性がなくなるのを避けるために、**パラメータスクリプト（または、パラメータスクリプトとして実行しているマスタースクリプト）でビューまたはプロジェクト依存のグローバル変数を使用しないでください。**

ARCHICAD 19以前の互換性：このようなグローバル変数がパラメータスクリプトで誤って使用されると、GDL警告が生成されます。

ARCHICAD 20以降の互換性：このようなグローバル変数がパラメータスクリプトで使用されると、GDL警告が生成され、静的なデフォルト値のみが含められます（型が一致）。

ARCHICAD 22以降の互換性：ビュー依存のグローバル変数をプロパティスクリプトで使用しないでください（そうしないと、マスタースクリプトがプロパティスクリプトとして実行されます）。このようなインスタンスによってスクリプトで警告が発生します。ただし、プロジェクト依存の変数は、たいていの場合プロパティスクリプトで有効です。

互換性の詳細については、グローバル変数記述を確認してください。示されている場合、スクリプトタイプの制限が適用されます。

凡例

✅	制限なしに機能します
⚠	機能します（追加の警告あり）
⛔	ダミーのデフォルト値を含む（追加の警告あり）

一般環境情報

GLOB_SCRIPT_TYPE — 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 - マスタースクリプト
- 7 - 上位移行スクリプト
- 8 - 下位移行スクリプト

GLOB_VIEW_TYPE — 現在のビューのタイプ（ビュー依存、パラメータ/プロパティスクリプトで使用しない）。

2D		3D		UI		Parameter		Property		テ#フォルト	-

- 2 - 2D（平面図）
- 3 - 3D
- 4 - 断面図
- 5 - 立面図
- 6 - 3Dドキュメント
- 7 - 詳細図
- 8 - レイアウト
- 9 - 計算

正確な必要な値を使用。将来的な値の拡張を踏まえると、範囲の使用は推奨されません。

GLOB_PREVIEW_MODE — 現在のプレビューのタイプ（ビュー依存、パラメータ/プロパティスクリプトで使用しない）

- 0 - なし
- 1 - ダイアログ
- 2 - リスト
- 3 - お気に入りの保存

正確な必要な値を使用。将来的な値の拡張を踏まえると、範囲の使用は推奨されません。

GLOB_FEEDBACK_MODE — 編集の進行中の表示（ビュー依存、パラメータ/プロパティスクリプトで使用しない）

0 - オフ、1 - 編集フィードバックモード

GLOB_SEO_TOOL_MODE — ソリッド編集の進行中の表示（ビュー依存、パラメータ/プロパティスクリプトで使用しない）

0 - オフ、1 - ソリッド編集フィードバックモード

GLOB_DIAGNOSTICS_MODE GDL診断用のライブラリ開発者（59）メニューコマンド

互換性：ARCHICAD 23で導入されました。
0 - オフ、1 - オン
ライブラリ部品のデバッグコンテンツを可視化するために、スクリプトで条件ステートメントとして使用します。

GLOB_SCALE 図面のスケール（ビュー依存、パラメータ/プロパティスクリプトで使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	100

現在のウィンドウに基づく

GLOB_DRAWING_BGD_PEN 図面の背景色のペン（ビュー依存、パラメータ/プロパティスクリプトで使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	19

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_FILL_INDEX_SOLID テンプレートに従った塗りつぶしタイプ[ソリッド]のインデックス（プロジェクト依存、パラメータスクリプトで使用しない）

2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	デフォルト	16

適用した塗りつぶしタイプ[ソリッド]のインデックスを含む
互換性：ARCHICAD 22で導入されました。

GLOB_FILL_INDEX_BACKGROUND テンプレートに従った塗りつぶしタイプ[背景]のインデックス（プロジェクト依存、パラメータスクリプトで使用しない）

2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	デフォルト	16

適用した塗りつぶしタイプ[背景]のインデックスを含む
互換性：ARCHICAD 22で導入されました。

GLOB_NORTH_DIR プロジェクトの北の方向（プロジェクト依存、パラメータスクリプトを使用しない）。

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	デフォルト	90

[プロジェクトの場所]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_PROJECT_LONGITUDE プロジェクト経度（プロジェクト依存、パラメータスクリプトを使用しない）。
GLOB_PROJECT_LATITUDE プロジェクト緯度（プロジェクト依存、パラメータスクリプトを使用しない）。
GLOB_PROJECT_ALTITUDE プロジェクト高度（プロジェクト依存、パラメータスクリプトを使用しない）。

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	デフォルト	0

[プロジェクトの場所]ダイアログで指定された設定に従ったプロジェクト原点の地理的な座標

GLOB_PROJECT_DATE プロジェクト日付（プロジェクト依存、パラメータスクリプトを使用しない）。

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	テ#フォルト	[0, 0, 0, 0, 0, 0]

次の6つの値の配列：1 - 年、2 - 月、3 - 日、4 - 時間、5 - 分、6 - 秒。 この変数にはプロジェクトの現在の日付が含まれ、EcoDesigner STAR®アドオンでのみ設定されます（その他の場合は全ての値が0に設定されます）。 この変数の値は、ソーラ解析ルーチンの実行時にアドオンによって変更され、特定のGDLオブジェクト（例えば落葉樹）を季節によって異なって表示されるようにすることができます。

GLOB_WORLD_ORIGO_OFFSET_X （プロジェクト依存、パラメータスクリプトで使用しない）

GLOB_WORLD_ORIGO_OFFSET_Y （プロジェクト依存、パラメータスクリプトで使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	テ#フォルト	0

ワールド原点を基準にしたプロジェクト原点の位置 グローバル変数GLOB_WORLD_ORIGO_...の使用例：を参照してください。

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

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 （プロジェクト依存、パラメータスクリプトで使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	テ#フォルト	[1.,, 3.0, -0.1, -0.1]

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系の絶対限度 詳細は、『ARCHICADヘルプ』の[平面図の切断面]ダイアログを参照してください。

GLOB_STRUCTURE_DISPLAY 躯体表示詳細（プロジェクト依存、パラメータスクリプトを使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	テ#フォルト	0

躯体表示オプション設定に関する情報（整数）： 0 - モデル全体、1 - 躯体のみ、2 - 仕上げなし

GLOB_ISSUE_SCHEME バージョンスキームで定義されたカスタムデータのリスト

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	-

全てのコンテキストで使用可能です。対応するGUID（第二行目）に関するバージョンのスキーム（最初の行）で定義されたフィールドの名前を含む2行の文字列配列。 最初の5つの列は固定されています：改訂ID、バージョンID、バージョン名、発行日、発行者。

例：

改訂ID	**バージョンID**	**バージョン名**	**発行日**	**発行者**	**受信者**	**状態**	**...**
{RevIdGUID}	{IssueIdGUID}	{IssueNameGUID}	{IssueDateGUID}	{IssuedByGUID}	{Custom1GUID}	{Custom2GUID}	

LAYOUT_REVISION_HISTORY　　list of the current Layout's Revision History

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	-

レイアウト変更コンテキストのみ使用可能です。文字配列、GLOB_ISSUE_SCHEMEと同じ構造に改訂ごとに1行を含む。 最初の5つの列は固定されています：改訂ID、バージョンID、バージョン名、発行日、発行者。
例：

01	1	最初のバージョン	2013-06-30	ユーザー1	全員	SD	...
02	3	一般アップデート	2013-07-31	ユーザー2	メカニカル	DD	
03	5	構造アップデート	2013-08-31	ユーザー1	構造	DD	
...							

GLOB_CHANGE_SCHEME　　変更スキームで定義されたカスタムデータのリスト

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	-

全てのコンテキストで使用可能です。対応するGUID（第二行目）に関する変更スキーム（最初の行）で定義されたフィールドの名前を含む2行の文字列配列。 最初の5つの列は固定されています：改訂ID、変更ID、変更名、最終修正日、修正者。
例：

改訂ID	**変更ID**	**変更の説明**	**最終修正**	**最終修正者**	**作成者**	**承認者**	**...**
{RevIdGUID}	{ChIdGUID}	{ChDescGUID}	{ModiTimeGUID}	{ModiByGUID}	{Custom1GUID}	{Custom2GUID}	

LAYOUT_CHANGE_HISTORY　　現在のレイアウト上に表示されている改訂履歴の変更を全てリスト

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	-

レイアウト変更コンテキストのみ使用可能です。文字配列、GLOB_ISSUE_SCHEMEと同じ構造に変更ごとに1行を含む。 最初の5つの列は固定されています：改訂ID、変更ID、変更名、最終修正日、修正者。
例：

2	Ch-13	キッチン	2013-07-13	ユーザー1	設計者 1	担当設計者 1	...
2	Ch-15	空調	2013-07-16	ユーザー2	設計者 2	担当設計者 1	
3	Ch-18	構造柱	2013-08-03	ユーザー2	設計者 1	担当設計者 2	
3	Ch-19	梁断面	2013-08-12	ユーザー1	設計者 3	担当設計者 2	
b	Ch-23	ドア番号	2013-10-01	ユーザー3	設計者 2	担当設計者 1	
...							

LAYOUT_CURRENTREVISION_OPEN （現在のレイアウトの進行中の作業状態（プロジェクト依存、パラメータスクリプトを使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	デ#フォルト	0

レイアウト変更コンテキストのみ使用可能です。 0 - 現在のレイアウトに開かれている改訂がない、1 - 現在のレイアウトに開かれている改訂が（進行中の作業のレイアウトに）

フロア情報

GLOB_HSTORY_ELEV 配置フロアの高度（プロジェクト依存、パラメータスクリプトを使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	デ#フォルト	0

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 配置フロアの高さ（プロジェクト依存、パラメータスクリプトを使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	デ#フォルト	3.1

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 現在のフロアの高度（プロジェクト依存、パラメータスクリプトを使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	デ#フォルト	0.0

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 現在のフロアの高さ（プロジェクト依存、パラメータスクリプトを使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	デ#フォルト	3.1

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 現在のフロアと配置フロアの相対位置（プロジェクト依存、パラメータスクリプトを使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	デ#フォルト	0.0

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

アニメーションでの現在のフレーム番号（ビュー依存、パラメータ/プロパティスクリプトを使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	-1

アニメーションにのみ有効、静止画像の場合は-1

GLOB_FIRST_FRAME

フライスルーでの最初のフレームインデックス（ビュー依存、パラメータ/プロパティスクリプトを使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	0

アニメーションにのみ有効、静止画像の場合は0

GLOB_LAST_FRAME

フライスルーでの最後のフレームインデックス（ビュー依存、パラメータ/プロパティスクリプトを使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	0

アニメーションにのみ有効、静止画像の場合は0

GLOB_EYEPOS_X

現在のカメラポジション（x）（ビュー依存、パラメータ/プロパティスクリプトを使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	-5.0

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

現在のカメラポジション（y）（ビュー依存、パラメータ/プロパティスクリプトを使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	-5.0

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

現在のカメラポジション（z）（ビュー依存、パラメータ/プロパティスクリプトを使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	1.7

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				現在のターゲットポジション（x）（ビュー依存、パラメータ/プロパティスクリプトを使用しない）							
2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	0.0
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				現在のターゲットポジション（y）（ビュー依存、パラメータ/プロパティスクリプトを使用しない）							
2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	0.0
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				現在のターゲットポジション（z）（ビュー依存、パラメータ/プロパティスクリプトを使用しない）							
2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	1.7
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				太陽方位（プロジェクト依存、パラメータスクリプトを使用しない）							
2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	デフォルト	240
[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（プロジェクト依存、パラメータスクリプトを使用しない）							
2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	✓	デフォルト	35
[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　要素の基準高さ

- ドア/窓 オブジェクト：現在の設定に基づいた下端高さ
- スラブ：設定に基づいた、選択されたスラブの参照面の高度
- 他の要素/オブジェクト：設定に基づいた、基準高さ

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、13 - カーテンウォール、14 - カーテンウォールフレーム、15 - カーテンウォールパネル、16 - カーテンウォール接続部、 17 - カーテンウォール付属品、18 - シェル、19 - 天窓、20 - モルフ、 21 - 階段、22 - 階段 踏面、23 - 階段 蹴上、24 - 階段構造、25 - 手摺り、 26 - 開口部、 27 - 柱セグメント、28 - 梁セグメント。

オブジェクト、ランプ、ドア、窓、壁の終端、天窓のパラメータ

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

SYMB_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　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　ライブラリ部品のデフォルトの材質属性インデックス

SYMB_POS_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　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軸とZ軸の向きについては、「建具」を参照してください。

SYMB_POS_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軸とZ軸の向きについては、「建具」を参照してください。

SYMB_ROTANGLE　　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　　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。

ホットリンクの反転ステータスは含まれず、モジュールのライブラリ部品は、モジュールを基準とした反転ステータスを受信します（モジュールを保存した最初のプランのように）。

オブジェクト、ランプ、ドア、窓、壁の終端、天窓、カーテンウォール付属品のパラメータ - リストとラベル用のみ

SYMB_A_SIZE　　ライブラリ部品公称長さ/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　　ライブラリ部品公称幅/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　　ライブラリ部品公称高さ/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公称高さ、それ以外の場合は0

開口部パラメータ - リストとラベル用のみ

OPENING_HEIGHT　　開口部の長さ、公称高さ

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	1

互換性：ARCHICAD 23で導入されました。

OPENING_WIDTH　　開口部の長さ、公称幅

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	1

互換性：ARCHICAD 23で導入されました。

OPENING_HEADERHEIGHT_VALUES 辞書、特定の基準レベルからの開口部のヘッダー（または回転している場合は一番上の点）の相対高度

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	{}

互換性：ARCHICAD 23で導入されました。

```
{
    "toHomeStory": 1.5,
    "toProjectZero": 1.5,
    "toWallBottom": 1.5,
    "toWallTop": 1.5
}
```

OPENING_CENTERHEIGHT_VALUES 辞書、特定の基準レベルからの開口部の中心の相対高度

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	{}

互換性：ARCHICAD 23で導入されました。

```
{
    "toHomeStory": 1.0,
    "toProjectZero": 1.0,
    "toWallBottom": 1.0,
    "toWallTop": 2.0
}
```

OPENING_SILLHEIGHT_VALUES 辞書、特定の基準レベルからの開口部の下端（または回転している場合は一番下の点）の相対高度

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	{}

互換性：ARCHICAD 23で導入されました。

```
{
    "toHomeStory": 0.5,
    "toProjectZero": 0.5,
    "toWallBottom": 0.5,
    "toWallTop": 0.5
}
```

開口シンボルのパラメータ

OPENING_SYMBOL_DISPLAY　整数、平面図の切断面に応じた開口シンボルの可視性（ビュー依存、パラメータ/プロパティスクリプトで使用しない）

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	1

互換性：ARCHICAD 23で導入されました。
1 - 可視、2 - 非表示、3 - 上部線

OPENING_SYMBOL_GEOMETRY　辞書、シンボルのジオメトリを含む（ビュー依存、パラメータ/プロパティスクリプトで使用しない）。

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	{}

互換性：ARCHICAD 23で導入されました。

- *.boundingBox2D{}*:（辞書）開口部ポリゴンの境界線定義、常に開口シンボル座標系に合わせて整列
- *.boundingBox2D.xmin*
- *.boundingBox2D.xmax*
- *.boundingBox2D.ymin*
- *.boundingBox2D.ymax*
- *.polygon2D{}*:（辞書）親要素のジオメトリによって切断された開口部の投影されたポリゴン
- *.polygon2D.isClosed*:（ブール）ポリゴンの場合は常に1（0 - 今後の開発のために予約されています）
- *.polygon2D.contour{}*：（辞書）ポリゴン輪郭のデータを含みます。デフォルトのポリゴンは、.boundingBox2D{}の長方形に対応します。
- *.polygon2D.contour.edges[n]*：（配列）ポリゴンの各辺に対する埋め込み辞書を含みます。
- *.polygon2D.contour.edges[n].type*：（整数）0 - 直線、1 - 曲線（ 円弧）
- *.polygon2D.contour.edges[n].begPoint{}*:（辞書）辺の開始点の座標に対する埋め込み辞書
- *.polygon2D.contour.edges[n].arcAngle*:（角度）曲線辺の中心角、正は反時計回り、負は時計回り（直線辺には設定されません）
- *.polygon2D.holes[m]*:（配列）内部の穴のデータを含み、.contourと同様に、穴が存在する場合のみ設定されます

```
OPENING_SYMBOL_GEOMETRY: {
   "boundingBox2D": {
     "xmin": 0,
     "xmax": 1,
     "ymin": 0,
     "ymax": 1
   },
   "polygon2D": {
     "isClosed": 1,
     "contour": {
       "edges": [
         {
           "type": 1,
           "begPoint": {
             "x": 1,
             "y": 0.5
           },
           "arcAngle": 90
         },
         {
           "type": 1,
           "begPoint": {
             "x": 0.5,
             "y": 1
           },
           "arcAngle": 90
         },
         {
           "type": 1,
           "begPoint": {
             "x": 0,
             "y": 0.5
           },
           "arcAngle": 90
         },
```

```
      {
        "type": 1,
        "begPoint": {
          "x": 0.5,
          "y": 0
        },
        "arcAngle": 90
      }
    ]
  },
 "holes": {
    "edges": [
      {
        "type": 0,
        "begPoint": {
          "x": 0.4,
          "y": 0.4
        },
      },
      {
        "type": 0,
        "begPoint": {
          "x": 0.4,
          "y": 0.6
        },
      },
      {
        "type": 0,
        "begPoint": {
          "x": 0.6,
          "y": 0.6
        },
      },
```

```
      {
        "type": 0,
        "begPoint": {
          "x": 0.6,
          "y": 0.4
        },
      }
    ]
  }
}
```

窓、ドア、壁終端のパラメータ

WIDO_REVEAL_ON	窓/ドアの内蔵抱きオン
0-外壁側抱きオフ、1-外壁側抱きオン	
WIDO_SILL	建具の下端奥行き
曲線壁用：公称サイズの開口の角からの半径方向	
WIDO_SILL_HEIGHT	建具の公称下端高さ
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	窓/ドア内蔵右側抱き
WIDO_LEFT_JAMB	窓/ドア内蔵左側抱き
WIDO_THRES_DEPTH	窓/ドア内蔵下端の高さ
WIDO_HEAD_DEPTH	窓/ドア内蔵上額縁の奥行き
WIDO_HEAD_HEIGHT	建具の公称上額縁の高さ
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	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	窓/ドア枠厚さ
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	窓/ドアオフセット
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
WIDO_SUBFL_THICKNESS	サブフロア厚さ（下端補正）
WIDO_PREFIX	建具の下端高さ頭文字
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	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	手すり壁のインセット

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	具の外壁側の開口部表面積
WIDO_OPRSIDE_SURF	具の外壁側と反対側の開口部表面積
WIDO_N_RSIDE_WIDTH	建具の外壁側の開口部公称幅
WIDO_N_OPRSIDE_WIDTH	建具の外壁側と反対側の開口部公称幅
WIDO_N_RSIDE_HEIGHT	建具の外壁側の開口部公称高さ
WIDO_N_OPRSIDE_HEIGHT	建具の外壁側と反対側の開口部公称高さ
WIDO_N_RSIDE_SURF	建具の外壁側の開口部公称高さ
WIDO_N_OPRSIDE_SURF	建具の外壁側と反対側の開口部公称面積
WIDO_VOLUME	建具の開口部体積

WIDO_GROSS_SURFACE	建具の開口部公称表面積
WIDO_GROSS_VOLUME	建具の開口部公称体積

ランプパラメータ - リストとラベル用のみ

LIGHT_ON 0-光源がオフ、1-光源がオン	光源がオン
LIGHT_RED	光源カラーの赤の構成要素
LIGHT_GREEN	光源カラーの緑の構成要素
LIGHT_BLUE	光源カラーの青の構成要素
LIGHT_INTENSITY	照明度

マーカーパラメータ（詳細図、ワークシート、変更マーカー）

MARKER_HEAD_ROT_MODE

整数タイプグローバル。設定ダイアログのマーカーパネルの[マーカー角度]機能：[画面に対する固定角度]/[モデルに対する固定角度]に従って設定

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	✓	デ#フォルト	1

GDLシンボル側でARCHICADのマーカーヘッドの回転設定に反応させる場合に使用できます
1 - 画面に対する固定角度、2 - モデルに対する固定角度
互換性：ARCHICAD 22で導入されました。

MARKER_HEAD_ANGLE

角度タイプグローバル。設定ダイアログのマーカーパネルでユーザーが設定

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	✓	デ#フォルト	0

マーカーのGDLシンボル部分の、マーカーヘッドのシンボル回転データ
互換性：ARCHICAD 22で導入されました。
詳細図/ワークシートマーカーオブジェクトで、SYMB_ROTANGLE値はこの新しい角度データとの互換性を維持します。
- [画面に対する固定角度]モード：1 - SYMB_ROTANGLEは、ビューの回転と反対の角度
- [モデルに対する固定角度]モード：2 - SYMB_ROTANGLEはMARKER_HEAD_ANGLEと同じ

変更マーカーオブジェクトで、SYMB_ROTANGLE値は全ての場合において以前と同様に0を維持します。

ラベルのパラメータ

LABEL_POSITION — ラベルの位置

2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	デフォルト	0

ラベルGDLシンボルのラベルポインタと開始位置を定義する3つの点の座標を含む配列[3][2]。
互換性：パラメータおよびプロパティスクリプトの制限がARCHICAD 22で導入されました。
[固定角度]がオンの場合がビュー依存の値。[ラベル向き]を[平行]または[垂直]に設定した場合プロジェクト依存。親要素は移動されます。そのため、変数の値は変更されますが、ラベル自体のパラメータスクリプトを実行しません（パラメータとして格納できません）。

LABEL_ASSOC_ELEM_ORIENTATION — 関連する要素の向き

2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	デフォルト	0

- 直線要素：基準線の方向
- 曲面要素：円弧の弦の方向
- 点状の要素：要素の回転角度

互換性：パラメータおよびプロパティスクリプトの制限がARCHICAD 22で導入されました。
プロジェクト依存：親要素を移動できるため、変数の値は変更されますが、ラベル自体のパラメータスクリプトを実行しません（パラメータとして格納できません）。

LABEL_ROTANGLE — GDLシンボルタイプラベルの絶対回転角度データ。

2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	デフォルト	0

角度は、ラベル向き、固定角度、および読み取り可能な設定に従って計算されます。
互換性：パラメータおよびプロパティスクリプトの制限がARCHICAD 22で導入されました。
[固定角度]がオンの場合がビュー依存の値。[ラベル向き]を[平行]または[垂直]に設定した場合プロジェクト依存：親要素を移動できるため、変数の値は変更されます。

LABEL_ARROWHEAD_PEN — [設定]ダイアログボックスでの矢印のペン

LABEL_HAS_POINTER — ブール

1 - [ラベル設定/ポインタ]パネルで[ポインタの追加/削除]がオン、0 - それ以外。
互換性：ARCHICAD 22で導入されました。類似する逆動作グローバル変数「LABEL_CUSTOM_ARROW」はARCHICAD 22以降廃止される見込みです。

壁パラメータ - ドア/窓、リストおよびラベルに使用可能

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 曲線壁3D解像度

3Dの場合のみ有効

WALL_THICKNESS 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 壁表面傾き

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_FLIPPED 壁の[基準線位置：基準線で壁を反転]オプションに従って設定されるブール値

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	✓	デフォルト	0

現在の壁の反転した状態に関する情報
0 - デフォルトの壁の位置、1 - 基準線で壁を反転
互換性：ARCHICAD 22で導入されました。

WALL_HEIGHT 壁高さ

WALL_MAT_A 開口部と反対側の壁の材質の属性インデックス

WALL_MAT_B 開口部側の壁の材質属性インデックス

同じ壁に配置された開口ごとに異なってもよい

WALL_MAT_EDGE 壁の辺の材質の属性インデックス

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 壁の塗りつぶし種類

塗りつぶしインデックス、複合構造の最初の層

WALL_FILL_PEN 壁の塗りつぶしペン

WALL_COMPS_NAME 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列の名前

WALL_BMAT_NAME	壁のビルディングマテリアル名
壁のビルディングマテリアル名、壁の複合構造または断面形状の場合は空白	
WALL_BMAT	壁のビルディングマテリアルのインデックス
互換性：ARCHICAD 21で導入されました。 壁のビルディングマテリアルのインデックス、壁の複合構造または断面形状の場合は0。	
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

任意の数の行を含む19列の配列：

- [1] 塗りつぶし
- [2] 厚さ
- [3] （古い輪郭ペン）
- [4] 塗りつぶしのペン
- [5] 塗りつぶし背景のペン
- [6] 躯体の状態
- [7] 上部線ペン
- [8] 上部線種
- [9] 下部線ペン
- [10] 下部線種
- [11] 終端面ペン
- [12] 塗りつぶし向き
- [13] 面の種類
- [14] 終端面線種
- [15] 仕上げ塗りつぶし状態
- [16] 与えられた方向の塗りつぶし状態
- [17] 台形/二面傾斜状態
- [18] ビルディングマテリアルのインデックス
- [19] 層の辺材質のインデックス（壁の辺材質上書きを考慮） 互換性：ARCHICAD 22で導入されました。

躯体の状態：0 - 部分ではない、1 - 部分、3 - 躯体の最終層；

塗りつぶしの方向：0 - グローバル、1 - ローカル、

面の種類：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面の種類は0）。

台形/二面傾斜：0 - この層には全ての状況において平行な面がある 1 - この層には、 台形または二面傾斜壁の幅の違いを調整するための平行でない面がある場合がある。 壁面が平行な場合でも、このフラグをオンにすることができます。

仕上げ塗りつぶし状態：0 - 仕上げ塗りつぶしではない、1 - 仕上げ塗りつぶし

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

複雑な壁の場合、この変数には、平面図で切断された層のデータ（2D - 平面図の切断面の高さに関する）またはD/W下端/壁終端の高さで切断された層のデータ（3D）のみ含まれます。

WALL_SKINS_BMAT_NAMES　　複合壁またはポリゴン壁のビルディングマテリアル名

列が1の配列：任意の数の行を有する層のビルディングマテリアル名

3Dウィンドウの D/Wと壁の終端では、D/Wまたは壁の終端によって実際に切り取られた塗りつぶしのデータが含まれます。

WALL_SECT_PEN　　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	
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
列が3の配列：プロジェクト原点を基準にした壁の開始点の位置	
WALL_TEXTURE_WRAP	VERTとCOOR{2}、またはCOOR{3}コマンドで使用される壁のテクスチャラッピングデータ 壁のテクスチャ座標は、壁が接続されたオブジェクトのローカル座標系に一致するように変換されます（追加の変換は不要）。

14行の配列：

- [1]: wrapping_method
- [2]: wrap_flags
- [3]-[4]-[5]: origin_X, origin_Y, origin_Z（vert 1のノード）
- [6]-[7]-[8]: endOfX_X, endOfX_Y, endOfX_Z（vert 2のノード）
- [9]-[10]-[11]: endOfY_X, endOfY_Y, endOfY_Z（vert 3のノード）
- [12]-[13]-[14]: endOfZ_X, endOfZ_Y, endOfZ_Z（vert 4のノード）

壁パラメータ-リストとラベル用のみ

WALL_LENGTH_A	基準線側壁長さ
WALL_LENGTH_B	基準線反対側壁長さ
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	壁中心長さ
WALL_AREA	壁面積
WALL_PERIMETER	壁外周
WALL_SURFACE_A	基準線側壁表面積
WALL_SURFACE_B	基準線反対側壁表面積
WALL_SURFACE_A_CON	基準線側の条件付き壁表面積
WALL_SURFACE_B_CON	基準線と反対側の条件付き壁表面積

WALL_GROSS_SURFACE_A	基準線側壁総表面積
WALL_GROSS_SURFACE_B	基準線反対側壁総表面積
WALL_EDGE_SURF	壁辺表面積
WALL_VOLUME	壁の体積
WALL_VOLUME_CON	壁の条件付き体積
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	壁内ドア表面積
WALL_WINDS_SURF	壁内窓表面積
WALL_HOLES_SURF	壁内開口表面積
WALL_HOLES_SURF_A	基準線側開口部分解法表面積
WALL_HOLES_SURF_B	反対側開口部分解法表面積
WALL_HOLES_VOLUME	壁内開口部分解法体積
WALL_WINDS_WID	壁内窓総幅
WALL_DOORS_WID	壁内ドア総幅
WALL_COLUMNS_NR	壁の柱数
WALL_CROSSSECTION_TYPE 0 - 断面形状、1 - 矩形、2 - 傾斜、3 - 台形	壁の断面タイプ
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

柱パラメータ-リストとラベル用のみ

互換性： ARCHICAD 23以降、柱要素は柱セグメントの集合になりました。GLOB_ELEM_TYPEは柱セグメント用の新しい値27を持ち、柱要素の値は6のままです。

各グローバル変数の使用可能性（意味のあるデータを含むかどうか）が表にアイコンで表示され、GLOB_ELEM_TYPEグローバル変数の値が括弧内に表示されます。

COLU_SEGMENT_INDEX	COLU_SEGMENT_INFO.segments[]配列での柱セグメントのインデックス

✅柱セグメント（27）	⛔単一セグメントの柱（6）	⛔複数セグメントの柱（6）

互換性：ARCHICAD 23で導入されました。

COLU_SEGMENT_INFO — 辞書、全ての柱セグメントの形状設定を含み、他のグローバル変数を使用可能かどうかを定義できます。パラメータやプロパティスクリプトでは使用できません。

列セグメントに配置した場合、COLU_SEGMENT_INFO.segments[COLU_SEGMENT_INDEX]にはラベル付きセグメントに関する情報が含まれます。

- *.segments[n].tapered*: （整数）：0 - 均一断面、1 - テーパード断面
- *.segments[n].crossSection{}*: （辞書）セグメントの断面データ
- *.segments[n].crossSection.type*: （整数）断面タイプ：1 - 矩形、2 - 円形、3 - 断面形状
- *.segments[n].crossSection.startWidth*: （長さ）セグメントの開始点の境界断面幅。セグメントの方向はその基準線によって定義されます。
- *.segments[n].crossSection.startHeight*: （長さ）セグメントの開始点の境界断面高さ。セグメントの方向はその基準線によって定義されます。
- *.segments[n].crossSection.endWidth*: （長さ）セグメントの終了点の境界断面幅。セグメントの方向はその基準線によって定義されます。
- *.segments[n].crossSection.endHeight*: （長さ）セグメントの終了点の境界断面高さ。セグメントの方向はその基準線によって定義されます。

✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）

互換性：ARCHICAD 23で導入されました。

COLU_CORE — 躯体/仕上げの特性（テーパード柱セグメントの場合は0）

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

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

✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）

COLU_MIN_HEIGHT — 柱最小高さ

✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）

COLU_MAX_HEIGHT — 柱最大高さ

✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）

COLU_VENEER_WIDTH — 柱の仕上げの厚さ

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

COLU_CORE_X　躯体の幅（テーパード柱セグメントの場合は0）

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

COLU_CORE_Y　躯体の奥行き（テーパード柱セグメントの場合は0）

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

COLU_DIM1　柱第1寸法（テーパード柱セグメントの場合は0）- ラベルの場合のみ

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

COLU_DIM2　柱第2寸法（テーパード柱セグメントの場合は0）- ラベルの場合のみ

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

COLU_MAT　柱の材質属性インデックス

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

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

✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）

平面図ウィンドウでのみ輪郭に適用される

COLU_CORE_FILL　柱の躯体の塗りつぶし

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

COLU_CORE_BMAT_NAME　柱の躯体のビルディングマテリアル名

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

COLU_CORE_BMAT　柱の躯体のビルディングマテリアルのインデックス

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

互換性：ARCHICAD 21で導入されました。

COLU_VENEER_FILL 柱の仕上げの塗りつぶし

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

COLU_VENEER_BMAT_NAME 柱の仕上げのビルディングマテリアル名

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

COLU_VENEER_BMAT 柱の仕上げのビルディングマテリアルのインデックス

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

互換性：ARCHICAD 21で導入されました。

COLU_SECT_PEN 柱の切断面の輪郭ペン

✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）

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

✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）

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 柱の躯体の塗りつぶしペン

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

COLU_CORE_FBGD_PEN 柱の躯体の塗りつぶし背景ペン

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

COLU_VENEER_FILL_PEN 柱の仕上げの塗りつぶしペン

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

COLU_VENEER_FBGD_PEN 柱の仕上げの塗りつぶし背景ペン

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

COLU_PERIMETER	柱外周	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_AREA	柱面積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_VOLUME	柱体積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_GROSS_VOLUME	柱総体積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_CORE_SURF	柱の躯体の表面積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_CORE_GROSS_SURF	柱の総表面積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_CORE_VOL	柱の躯体の体積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_CORE_GROSS_VOL	躯体総体積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_VENEER_SURF	柱の仕上げの表面積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_VENEER_GROSS_SURF	仕上げ総表面積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_VENEER_VOL	柱の仕上げの体積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）

COLU_VENEER_GROSS_VOL	仕上げ総体積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_CORE_TOP_SURF	躯体の上部の表面積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_CORE_BOT_SURF	躯体の下部の表面積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_VENEER_TOP_SURF	仕上げ上部の表面積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_VENEER_BOT_SURF	仕上げ下部の表面積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_CORE_GROSS_TOPBOT_SURF	躯体の上部と下部の総表面積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_VENEER_GROSS_TOPBOT_SURF	仕上げの上部と下部の総表面積	
✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）
COLU_PROFILE_NAME	複雑な場合は、柱の断面形状の名前	
✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

梁パラメータ-リストとラベル用のみ

互換性： ARCHICAD 23以降、梁要素は梁セグメントの集合になりました。GLOB_ELEM_TYPEは梁セグメント用の新しい値28を持ち、梁要素の値は12のままです。

各グローバル変数の使用可能性（意味のあるデータを含むかどうか）が表にアイコンで表示され、GLOB_ELEM_TYPEグローバル変数の値が括弧内に表示されます。

BEAM_SEGMENT_INDEX — BEAM_SEGMENT_INFO.segments[]配列での梁セグメントのインデックス

✅ 梁セグメント（28）	⛔ 単一セグメントの梁（12）	⛔複数セグメントの梁（12）

互換性：ARCHICAD 23で導入されました。

BEAM_SEGMENT_INFO — 辞書、全ての梁セグメントの形状設定を含み、他のグローバル変数やその他の梁セグメント形状データを使用可能かどうかを定義できます。パラメータやプロパティスクリプトでは使用できません。

梁セグメントに配置した場合、BEAM_SEGMENT_INFO.segments[BEAM_SEGMENT_INDEX]にはラベル付きセグメントに関する情報が含まれます。

- *.segments[n].curvature*：軸の曲率（整数）：0 - 直線、1 - 水平湾曲、2 - 垂直湾曲
- *.segments[n].tapered*:（整数）：0 - 均一断面、1 - テーパード断面
- *.segments[n].refLineLength*: 梁セグメント基準線の3D長さ（長さ）
- *.segments[n].crossSection{}*:（辞書）セグメントの断面データ
- *.segments[n].crossSection.type*:（整数）断面タイプ：1 - 矩形、2 - 円形、3 - 断面形状
- *.segments[n].crossSection.startWidth*:（長さ）セグメントの開始点の境界断面幅。セグメントの方向はその基準線によって定義されます。
- *.segments[n].crossSection.startHeight*:（長さ）セグメントの開始点の境界断面高さ。セグメントの方向はその基準線によって定義されます。
- *.segments[n].crossSection.endWidth*:（長さ）セグメントの終了点の境界断面幅。セグメントの方向はその基準線によって定義されます。
- *.segments[n].crossSection.endHeight*:（長さ）セグメントの終了点の境界断面高さ。セグメントの方向はその基準線によって定義されます。

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅複数セグメントの梁（12）

互換性：ARCHICAD 23で導入されました。

BEAM_THICKNESS — 梁の厚さ（ テーパード梁セグメントの場合は0）

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔複数セグメントの梁（12）

BEAM_HEIGHT — 梁の高さ（ テーパード梁セグメントの場合は0）

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔複数セグメントの梁（12）

BEAM_REFLINE_OFFSET — 梁の軸を基準にした基準線のオフセット

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅複数セグメントの梁（12）

BEAM_ELEVATION_TOP 梁の形状の一番上の点の高度が計算されます

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅複数セグメントの梁（12）

互換性：ARCHICAD 22で導入されました。

BEAM_ELEVATION_BOTTOM 梁の形状の一番下の点の高度が計算されます

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅複数セグメントの梁（12）

互換性：ARCHICAD 22で導入されました。

BEAM_PRIORITY 3D交差の優先度インデックス

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔複数セグメントの梁（12）

BEAM_MAT_RIGHT 基準線の右側の梁の材質属性のインデックス

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔複数セグメントの梁（12）

BEAM_MAT_LEFT 基準線の左側の梁の材質属性のインデックス

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔複数セグメントの梁（12）

BEAM_MAT_TOP 壁の上部の材質属性インデックス

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔複数セグメントの梁（12）

BEAM_MAT_BOTTOM 壁の下部の材質属性インデックス

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔複数セグメントの梁（12）

BEAM_MAT_END 壁の終端の材質属性インデックス

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔複数セグメントの梁（12）

BEAM_OUTLINE_LINETYPE 梁の輪郭の線種

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅複数セグメントの梁（12）

BEAM_AXES_LINETYPE 梁の軸の線種

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅複数セグメントの梁（12）

BEAM_FILL	梁の塗りつぶし種類	
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔ 複数セグメントの梁（12）

BEAM_BMAT_NAME	梁のビルディングマテリアル名	
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔ 複数セグメントの梁（12）

BEAM_BMAT	梁のビルディングマテリアルのインデックス	
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔ 複数セグメントの梁（12）

互換性：ARCHICAD 21で導入されました。

BEAM_FILL_PEN	梁の塗りつぶしペン	
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔ 複数セグメントの梁（12）

BEAM_SECT_PEN	梁の切断面の輪郭ペン	
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅ 複数セグメントの梁（12）

BEAM_FBGD_PEN	梁の塗りつぶし背景ペン	
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔ 複数セグメントの梁（12）

BEAM_DIRECTION	梁の基準線の方向	
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅ 複数セグメントの梁（12）

BEAM_POSITION	梁の軸の開始点の絶対座標	
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅ 複数セグメントの梁（12）

BEAM_LENGTH_RIGHT	基準線の右側の梁の長さ（テーパード梁または垂直湾曲梁の場合は0）	
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔ 複数セグメントの梁（12）

互換性：ARCHICAD 23でリスト用が廃止されました。

BEAM_LENGTH_LEFT		基準線の左側の梁の長さ（テーパード梁または垂直湾曲梁の場合は0）
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔ 複数セグメントの梁（12）
互換性：ARCHICAD 23でリスト用が廃止されました。		
BEAM_RIGHT_SURF		基準線の右側の梁の表面積
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅ 複数セグメントの梁（12）
BEAM_LEFT_SURF		基準線の左側の梁の表面積
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅ 複数セグメントの梁（12）
BEAM_TOP_SURF		梁の上部の表面積
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅ 複数セグメントの梁（12）
BEAM_BOTTOM_SURF		梁の下部の表面積
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅ 複数セグメントの梁（12）
BEAM_END_SURF		梁の両端の表面積
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅ 複数セグメントの梁（12）
BEAM_VOLUME		梁の体積
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅ 複数セグメントの梁（12）
BEAM_VOLUME_CON		梁の条件付き体積
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅ 複数セグメントの梁（12）
BEAM_HOLES_NR		梁の穴の数
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅ 複数セグメントの梁（12）
BEAM_HOLES_SURF		梁の穴の総表面積
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅ 複数セグメントの梁（12）

BEAM_HOLE_EDGE_SURF		梁の穴辺の総表面積
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅複数セグメントの梁（12）
BEAM_HOLES_VOLUME		梁の穴の総体積
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅複数セグメントの梁（12）
BEAM_PROFILE_NAME		複雑な場合は、梁の断面形状の名前
✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔複数セグメントの梁（12）

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_ELEVATION_TOP	スラブ上端高度
SLAB_ELEVATION_BOTTOM	スラブ下端高度
SLAB_MAT_TOP	スラブ上部の材質の属性インデックス
SLAB_MAT_EDGE	スラブの辺の材質の属性インデックス
SLAB_MAT_BOTT	スラブ下部の材質の属性インデックス
SLAB_LINETYPE	スラブの線種
SLAB_FILL 塗りつぶしインデックス-複合構造の場合の値は負	スラブの塗りつぶし
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	スラブの複合構造の名前
SLAB_BMAT_NAME	スラブのビルディングマテリアル名、複合構造スラブの場合は空白
SLAB_BMAT 互換性：ARCHICAD 21で導入されました。	スラブのビルディングマテリアルのインデックス、複合構造スラブの場合は0
SLAB_SKINS_NUMBER 1～8の範囲、単一塗りつぶしの場合は0	複合スラブの層の数

SLAB_SKINS_PARAMS	複合スラブの層のパラメータ

任意の数の行を含む18列の配列：

- [1] 塗りつぶし
- [2] 厚さ
- [3] （古い輪郭ペン）
- [4] 塗りつぶしのペン
- [5] 塗りつぶし背景のペン
- [6] 躯体の状態
- [7] 上部線ペン
- [8] 上部線種
- [9] 下部線ペン
- [10] 下部線種
- [11] 終端面ペン
- [12] 塗りつぶし向き
- [13] 面の種類
- [14] 終端面線種
- [15] 仕上げ塗りつぶし状態
- [16] 与えられた方向の塗りつぶし状態
- [17] 躯体の状態（躯体が存在しない場合は、最も厚い塗りつぶし）
- [18] ビルディングマテリアルのインデックス

躯体の状態：0 - 部分ではない、1 - 部分、3 - 躯体の最終層、塗りつぶし向き：0 - グローバル、1 - ローカル、 面の種類：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_SKINS_BMAT_NAMES	複合スラブの層のビルディングマテリアル名

列が1の配列：任意の数の行を有する層のビルディングマテリアル名

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	スラブの上部の表面積

穴表面積を減算しない

SLAB_GROSS_TOP_SURF	開口を含まないスラブ上部総表面積

開口表面積を減算

SLAB_TOP_SURF_CON 指定された値より大きい穴表面積を減算	スラブの条件付き上表面積
SLAB_BOT_SURF 穴表面積を減算しない	開口を含まないスラブの下部の表面積
SLAB_GROSS_BOT_SURF 開口表面積を減算	スラブ下部総表面積
SLAB_BOT_SURF_CON 指定された値より大きい穴表面積を減算	スラブの条件付き下表面積
SLAB_EDGE_SURF 穴表面積を減算しない	スラブの辺の表面積
SLAB_GROSS_EDGE_SURF 開口表面積を減算	開口を含まないスラブ辺総表面積
SLAB_PERIMETER	スラブ周囲長さ
SLAB_VOLUME 穴体積を減算しない	スラブ体積
SLAB_GROSS_VOLUME 穴体積を減算	開口を含まないスラブ総体積
SLAB_VOLUME_CON 指定された値より大きい穴体積を減算	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
SLAB_GROSS_TOP_SURF_WITH_HOLES	スラブ上部総表面積
SLAB_GROSS_BOT_SURF_WITH_HOLES	スラブ下部総表面積
SLAB_GROSS_EDGE_SURF_WITH_HOLES	スラブ辺総表面積
SLAB_GROSS_VOLUME_WITH_HOLES	スラブ総体積

階段構成要素のパラメータ

階段の一般的な変数 - リストおよびラベルに使用可能

互換性：ARCHICAD 21で導入されました。

STAIR_AREA				階段の投影2D領域							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
STAIR_VOLUME				全ての3D部分を含む階段の領域							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
STAIR_HEIGHT				Z座標の最大と最小の差							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
STAIR_WALKLINE_LENGTH				階段の移動線の投影2D長さ							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
STAIR_DEFAULT_WIDTH				階段のデフォルト幅（ [階段のデフォルト設定]/[形状と位置]パネルの設定）							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
STAIR_DEFAULT_GOING_DEPTH				踏面のデフォルト長さ（[階段のデフォルト設定]/[形状と位置]パネルの設定）							
2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
STAIR_DEFAULT_RISER_HEIGHT				蹴上のデフォルト幅（[階段のデフォルト設定]/[形状と位置]パネルの設定）							
2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
STAIR_DEFAULT_TREAD_THICKNESS				階段のデフォルトの踏面厚さ（[階段のデフォルト設定]/[形状と位置]パネルの設定）							
2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
STAIR_NR_OF_TREADS_IN_FLIGHTS				階段の各フライトの踏面数を示す1次元の整数配列（[n]） （n = フライトの数）							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
STAIR_NR_OF_RISERS_IN_FLIGHTS				階段の各フライトの蹴上数を示す1次元の整数配列（[n]） （n = フライトの数）							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0

STAIR_NR_OF_RISERS											階段全体の蹴上数
2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	デフォルト	0

STAIR_NR_OF_TREADS											階段全体の踏面数
2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	デフォルト	0

STAIR_LANDING_NUMBER											階段全体の踊り場領域の数
2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	デフォルト	0

STAIR_STAIR_GRADIENT											階段の傾斜：蹴上/踏面の角度比率（ラジアン）
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	0

STAIR_RULE_LIMITS											2次元の長さ/角度の配列（[6][2]）。[階段のデフォルト設定]/[[ルールと基準]/[踏面]および[蹴上]パネルで設定された最小値と最大値の集合
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	[0]

[プロジェクト設定]でのこれらの値の可視度の設定は、変数には影響ありません。

- [1][1] - [1][2]: 蹴上（R）の最小および最大値
- [2][1] - [2][2]: 踏面（T）の最小および最大値
- [3][1] - [3][2]: 2蹴上 + 1踏面（2*R + G）の最小および最大値
- [4][1] - [4][2]: 蹴上 / 踏面比率（R / G）の最小および最大値
- [5][1] - [5][2]: 蹴上 + 踏面（R + G）の最小および最大値
- [6][1] - [6][2]: 階段の勾配の最小および最大値

STAIR_RULE_FLAGS											2次元のブール配列（[6][2]）。STAIR_RULE_LIMITSに従って、[階段のデフォルト設定]/[ルールと基準]/[踏面]および[蹴上]パネルで設定された制限のステータス集合を有効化/無効化します。
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	[0]

値インデックスはSTAIR_RULE_LIMITSと平行です。有効値：

- 0 - STAIR_RULE_LIMITSの同じインデックスの制限オプションは現在使用されていません
- 1 - STAIR_RULE_LIMITSの同じインデックスの制限オプションが現在使用されています

踏面の一般的な変数 - リストおよびラベルに使用可能

互換性：ARCHICAD 21で導入されました。

TREAD_STEP_INDEX											選択した（現在の）踏面のステップインデックス
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	0

TREAD_GOING				選択した（現在の）踏面の踏面長さ							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	0
TREAD_ELEVATION				選択した（現在の）踏面のGLからの高度							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	0
TREAD_AREA				選択した（現在の）踏面の投影2D領域							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	0
TREAD_FRONT_AREA				選択した（現在の）踏面の前面材質領域							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	0
TREAD_VOLUME				選択した（現在の）踏面の体積							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	0
TREAD_BMATS				1次元の配列（[n]）。選択した（現在の）踏面のビルディングマテリアル（n = ビルディングマテリアルの数）							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	0

蹴上の一般的な変数 - リストおよびラベルに使用可能

互換性：ARCHICAD 21で導入されました。

RISER_STEP_INDEX				選択した（現在の）蹴上のステップインデックス							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	0
RISER_WIDTH				選択した（現在の）蹴上のポリライン長さ							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	0
RISER_FRONT_AREA				選択した（現在の）蹴上の前面材質領域							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	0
RISER_VOLUME				選択した（現在の）蹴上の体積							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	0

RISER_BMATS				1次元の配列（[n]）。選択した（現在の）蹴上のビルディングマテリアル（n = ビルディングマテリアルの数）							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0

階段構造の変数 - リストおよびラベルに使用可能

互換性：ARCHICAD 21で導入されました。

STRUCTURE_WIDTH				選択した（現在の）構造構成要素の幅							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0
STRUCTURE_HEIGHT				選択した（現在の）構造構成要素の高さ（zの最小と最大の差）							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0
STRUCTURE_3DLENGTH				選択した（現在の）構造構成要素の3D全体の長さ							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0
STRUCTURE_VOLUME				選択した（現在の）構造構成要素の体積							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0
STRUCTURE_THICKNESS				選択した（現在の）構造構成要素の厚さ							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0

階段モデル表示オプションの変数

関連する設定は、[モデル表示オプション]/[階段オプション]および[手摺りオプション]ダイアログで使用できます。

互換性：ARCHICAD 21で導入されました。

GLOB_MVO_STAIR_FLOOR_PLAN_OPT				階段MVO平面図オプション：0 - 平面図、1 - 見上げ図							
2D	✓	3D	⛔	UI	✓	Parameter	⛔	Property	⛔	デ#フォルト	1

GLOB_MVO_STAIR_FLOOR_PLAN_COMP				階段MVO構成要素ビットセット							
2D	✓	3D	⛔	UI	✓	Parameter	⛔	Property	⛔	デ#フォルト	1

mask: 平面図に表示される階段の構成要素についての情報を返します
*mask = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9*, ここで、 各 jiフラグは0または1をとります.
j_1: 移動線
j_2: 番号付け
j_3: 上/下のテキスト
j_4: 説明
j_5: 踏面付属品
j_6: 構造 - 梁
j_7: 構造 - 階段ばり
j_8: 構造 - 片持ち
j_9: 構造 - 板形 互換性 : ARCHICAD 22で導入されました。

GLOB_MVO_RAILING_PLAN_COMP				手摺りMVO構成要素ビットセット							
2D	✓	3D	⛔	UI	✓	Parameter	⛔	Property	⛔	デ#フォルト	127

mask: 平面図に表示される階段の構成要素についての情報を返します
*mask = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7*, ここで、 各 jiフラグは0または1をとります.
j_1: 支柱
j_2: トップレール
j_3: ハンドレール
j_4: レール
j_5: 子柱
j_6: 手摺子
j_7: パネル

階段の2D変数 - 平面図の表示にのみ使用できます

互換性：ARCHICAD 21で導入されました。

階段の通り芯変数

STAIR2D_FULL_TPOLYGON_GEOM 2次元の配列（[n][3]）。踏面ポリゴンの節点のデータトリプレット

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0]

n = 踏面数 * 各踏面に5つの節点（通常）：

- [n][1] - 階段の原点から計測したポリゴンの節点の座標x
- [n][2] - 階段の原点から計測したポリゴンの節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

STAIR2D_FULL_TPOLYGON_FLAGS 2次元の配列（[n][4]）。STAIR2D_FULL_TPOLYGON_GEOMに応じた踏面ポリゴンの追加データ

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0]

n = 踏面数 * 各踏面に5つの節点（通常）：

- [n][1] - 踏面のインデックス
- [n][2] - 節点から開始する辺のタイプ（0-リーディング、1-トレーリング、2-左、3-右、4-破断線、-1-クロージング）
- [n][3] - 節点から開始する辺の可視度（1-可視、0-省略）
- [n][4] - 踏面のタイプ（0-階段、1-踊り場）

STAIR2D_FULL_RPOLYLINE_GEOM 2次元の配列（[n][3]）。蹴上ポリゴンの節点のデータトリプレット

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0]

n = 蹴上数 * 各踏面に2つの節点（通常）：

- [n][1] - 階段の原点から計測したポリラインの節点の座標x
- [n][2] - 階段の原点から計測したポリラインの節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

STAIR2D_FULL_RPOLYLINE_FLAGS 2次元の配列（[n][1]）。STAIR2D_FULL_RPOLYLINE_GEOMに応じた蹴上ポリラインの追加データ

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0]

n = 蹴上数 * 各踏面に2つの節点（通常）：

- [n][1] - 蹴上のインデックス

STAIR2D_FULL_BOUNDARY_GEOM										2次元の配列（[n][3]）。階段境界のポリゴンの節点のデータトリプレット	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

n = 境界節点の数（通常は5）：

- [n][1] - 階段の原点から計測したポリゴンの節点の座標x
- [n][2] - 階段の原点から計測したポリゴンの節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

次のグローバルを使用して、最初の破断線の下に示される階段の部分を定義します。

STAIR2D_LOWER_TPOLYGON_GEOM										2次元の配列（[n][3]）。下部の踏面ポリゴンの節点のデータトリプレット、STAIR2D_FULL_TPOLYGON_GEOMと同様	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

STAIR2D_LOWER_TPOLYGON_FLAGS										2次元の配列（[n][4]）。下部の踏面ポリゴンの節点の追加データ、STAIR2D_FULL_TPOLYGON_FLAGSと同様	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

STAIR2D_LOWER_RPOLYLINE_GEOM										2次元の配列（[n][3]）。下部の蹴上ポリラインの節点のデータトリプレット、STAIR2D_FULL_RPOLYLINE_GEOMと同様	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

STAIR2D_LOWER_RPOLYLINE_FLAGS										2次元の配列（[n][1]）。下部の蹴上ポリラインの節点の追加データ、STAIR2D_FULL_RPOLYLINE_FLAGSと同様	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

STAIR2D_LOWER_BOUNDARY_GEOM										2次元の配列（[n][3]）。下部の階段境界のポリゴンの節点のデータトリプレット、STAIR2D_FULL_BOUNDARY_GEOMと同様	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

次のグローバルを使用して、2つの破断線の間に示される階段の部分を定義します。

STAIR2D_MIDDLE_TPOLYGON_GEOM										2次元の配列（[n][3]）。中部の踏面ポリゴンの節点のデータトリプレット、STAIR2D_FULL_TPOLYGON_GEOMと同様	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

STAIR2D_MIDDLE_TPOLYGON_FLAGS										2次元の配列（[n][4]）。中部の踏面ポリゴンの節点の追加データ、STAIR2D_FULL_TPOLYGON_FLAGSと同様	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

STAIR2D_MIDDLE_RPOLYLINE_GEOM										2次元の配列（[n][3]）。中部の蹴上ポリラインの節点のデータトリプレット、STAIR2D_FULL_RPOLYLINE_GEOMと同様	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]
STAIR2D_MIDDLE_RPOLYLINE_FLAGS										2次元の配列（[n][1]）。中部の蹴上ポリラインの節点の追加データ、STAIR2D_FULL_RPOLYLINE_FLAGSと同様	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]
STAIR2D_MIDDLE_BOUNDARY_GEOM										2次元の配列（[n][3]）。中部の階段境界のポリゴンの節点のデータトリプレット、STAIR2D_FULL_BOUNDARY_GEOMと同様	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]
次のグローバルを使用して、最後の破断線の上に示される階段の部分を定義します。											
STAIR2D_UPPER_TPOLYGON_GEOM										2次元の配列（[n][3]）。上部の踏面ポリゴンの節点のデータトリプレット、STAIR2D_FULL_TPOLYGON_GEOMと同様	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]
STAIR2D_UPPER_TPOLYGON_FLAGS										2次元の配列（[n][4]）。上部の踏面ポリゴンの節点の追加データ、STAIR2D_FULL_TPOLYGON_FLAGSと同様	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]
STAIR2D_UPPER_RPOLYLINE_GEOM										2次元の配列（[n][3]）。上部の蹴上ポリラインの節点のデータトリプレット、STAIR2D_FULL_RPOLYLINE_GEOMと同様	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]
STAIR2D_UPPER_RPOLYLINE_FLAGS										2次元の配列（[n][1]）。上部の蹴上ポリラインの節点の追加データ、STAIR2D_FULL_RPOLYLINE_FLAGSと同様	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]
STAIR2D_UPPER_BOUNDARY_GEOM										2次元の配列（[n][3]）。上部の階段境界のポリゴンの節点のデータトリプレット、STAIR2D_FULL_BOUNDARY_GEOMと同様	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

階段の移動線記号の変数

STAIR2D_FULL_WALKLINE_GEOM 2次元の配列（[n][3]）。階段の移動線全体の長さの節点のデータトリプレット

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0]

n = 移動線全体の長さにおけるポリラインの節点の数

- [n][1] - 階段の原点から計測したポリラインの節点の座標x
- [n][2] - 階段の原点から計測したポリラインの節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

STAIR2D_FULL_WALKLINE_FLAGS 2次元の配列（[n][2]）。階段の移動線全体の長さの節点の追加データ

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0]

n = 移動線全体の長さにおけるポリラインの節点の数

mask: 節点の位置についての情報を返します。

mask = $j_1 + 2*j_2 + 4*j_3 + 8*j_4$, ここで、 各 jiフラグは0または1をとります.

j_1: 節点は最初の踏面のトレイル終端にあります
j_2: 節点は踊り場のリード終端にあります
j_3: 節点は踊り場のトレイル終端にあります
j_4: 節点は最後の踏面のリード終端にあります

- [n][1] - 節点の位置マスク
- [n][2] - 節点がトレイル終端または上にある踏面のインデックス

STAIR2D_LOWER_WALKLINE_GEOM 2次元の配列（[n][3]）。階段の移動線の節点のデータトリプレット、下部（階段のポリゴン切断と同様のロジック）

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0]

n = 移動線下部のポリラインの節点の数

- [n][1] - 階段の原点から計測したポリラインの節点の座標x
- [n][2] - 階段の原点から計測したポリラインの節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

STAIR2D_LOWER_WALKLINE_FLAGS　　2次元の整数配列（[n][2]）。階段の移動線の節点の追加データ、下部

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

n = 移動線下部のポリラインの節点の数

mask: 節点の位置についての情報を返します。
*mask = j_1 + 2*j_2 + 4*j_3 + 8*j_4*, ここで、 各 jiフラグは0または1をとります.
j_1: 節点は最初の踏面のトレイル終端にあります
j_2: 節点は踊り場のリード終端にあります
j_3: 節点は踊り場のトレイル終端にあります
j_4: 節点は最後の踏面のリード終端にあります

- [n][1] - 節点の位置マスク
- [n][2] - 節点がトレイル終端または上にある踏面のインデックス

STAIR2D_MIDDLE_WALKLINE_GEOM　　2次元の配列（[n][3]）。階段の移動線の節点のデータトリプレット、中部

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

n = 移動線中部のポリラインの節点の数

- [n][1] - 階段の原点から計測したポリラインの節点の座標x
- [n][2] - 階段の原点から計測したポリラインの節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

STAIR2D_MIDDLE_WALKLINE_FLAGS　　2次元の整数配列（[n][2]）。階段の移動線の節点の追加データ、中部

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

n = 移動線中部のポリラインの節点の数

mask: 節点の位置についての情報を返します。
*mask = j_1 + 2*j_2 + 4*j_3 + 8*j_4*, ここで、 各 jiフラグは0または1をとります.
j_1: 節点は最初の踏面のトレイル終端にあります
j_2: 節点は踊り場のリード終端にあります
j_3: 節点は踊り場のトレイル終端にあります
j_4: 節点は最後の踏面のリード終端にあります

- [n][1] - 節点の位置マスク
- [n][2] - 節点がトレイル終端または上にある踏面のインデックス

STAIR2D_UPPER_WALKLINE_GEOM										2次元の配列（[n][3]）。階段の移動線の節点のデータトリプレット、上部
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト [0]

n = 移動線上部のポリラインの節点の数

- [n][1] - 階段の原点から計測したポリラインの節点の座標x
- [n][2] - 階段の原点から計測したポリラインの節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

STAIR2D_UPPER_WALKLINE_FLAGS										2次元の配列（[n][2]）、階段の移動線の節点の追加データ、上部
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト [0]

n = 移動線上部のポリラインの節点の数

mask: 節点の位置についての情報を返します。

mask = $j_1 + 2*j_2 + 4*j_3 + 8*j_4$. ここで、 各 jiフラグは0または1をとります.

j_1: 節点は最初の踏面のトレイル終端にあります

j_2: 節点は踊り場のリード終端にあります

j_3: 節点は踊り場のトレイル終端にあります

j_4: 節点は最後の踏面のリード終端にあります

- [n][1] - 節点の位置マスク
- [n][2] - 節点がトレイル終端または上にある踏面のインデックス

階段破断線記号の変数

STAIR2D_BREAKMARK_GEOM										2次元の配列（[n][9]）。破断線ポリラインの節点のデータ
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト [0]

n = 破断線の数 * 各破断線に対し9節点のデータ（破断線の最大表示数は4）：

- [n][1] - 階段の原点から計測したポリラインの開始の座標x
- [n][2] - 階段の原点から計測したポリラインの開始の座標y
- [n][3] - 階段の原点から計測したポリラインの終了の座標x
- [n][4] - 階段の原点から計測したポリラインの終了の座標y
- [n][5] - 移動線の垂直位置から算出した破断線角度（単位：度）、編集により値を更新。STAIR2D_BREAKMARK_ANGLEも参照。
- [n][6] - 階段の原点から計測した開始の座標xの拡張 互換性：ARCHICAD 22で導入されました。
- [n][7] - 階段の原点から計測した開始の座標yの拡張 互換性：ARCHICAD 22で導入されました。
- [n][8] - 階段の原点から計測した終了の座標xの拡張 互換性：ARCHICAD 22で導入されました。
- [n][9] - 階段の原点から計測した終了の座標yの拡張 互換性：ARCHICAD 22で導入されました。

STAIR2D_BREAKMARK_FLAGS										2次元の整数配列（[n][1]）、破断線ポリライン可視度の追加データ
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト [0]

n = 現在の設定で表示される破断線の数（最大4）

Attribute Set Index:

1: 破断線を表示
2: 破断線を非表示

STAIR2D_BREAKMARK_ANGLE										[階段の設定]ダイアログで設定された破断線角度（単位：度、実数型の値）。破断線が編集されてもプリセットの値を保持します。
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト 0

蹴上と踏面の記述の変数

STAIR2D_DESCRIPTION_POSITION										2次元の配列（[n][4]）。説明テキストの位置と方向についての情報を含む
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト [0]

n: 位置の定義：

1: 最初のフライトの中央、
2: 最初の踊り場の中央、
3: 最後のフライトの中央、
4: 最後の踊り場の中央、
5: 階段の中央。

- [n][1] - 階段の原点から計測した記述位置の座標x
- [n][2] - 階段の原点から計測した記述位置の座標y
- [n][3] - 記述位置の移動線の法線ベクトルのx座標
- [n][4] - 記述位置の移動線の法線ベクトルのy座標

階段の排水溝の2D変数

次のグローバルの形状では、破断線での切断は表示されません。

STAIR2D_EXT_TPOLYGON_GEOM										2次元の配列（[n][3]）。拡張踏面ポリゴンの節点（排水溝を含む）のデータトリプレット。STAIR2D_FULL_TPOLYGON_GEOMと同様の構造。
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト [0]

n = 踏面数 * 各踏面に5つの節点（通常）：

- [n][1] - 階段の原点から計測した拡張踏面ポリゴンの節点の座標x
- [n][2] - 階段の原点から計測した拡張踏面ポリゴンの節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

STAIR2D_EXT_TPOLYGON_FLAGS

2次元の配列（[n][4]）。STAIR2D_EXT_TPOLYGON_GEOMに応じた踏面拡張ポリゴンの追加データ。STAIR2D_FULL_TPOLYGON_FLAGSと同様の構造。

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

n = 踏面数 * 各踏面に5つの節点（通常）：

- [n][1] - 踏面のインデックス
- [n][2] - 節点から開始する辺のタイプ（0-リーディング、1-トレーリング、2-左、3-右、-1-クロージング）
- [n][3] - 節点から開始する辺の可視度（1-可視、0-省略）
- [n][4] - 踏面のタイプ（0-階段、1-踊り場）

STAIR2D_EXT_RPOLYLINE_GEOM

2次元の配列（[n][3]）。拡張蹴上ポリラインの節点（排水溝を含む）のデータトリプレット。STAIR2D_FULL_RPOLYLINE_GEOMと同様の構造。

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

n = 蹴上数 * 各踏面に2つの節点（通常）：

- [n][1] - 階段の原点から計測したポリラインの節点の座標x
- [n][2] - 階段の原点から計測したポリラインの節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

STAIR2D_EXT_RPOLYLINE_FLAGS

2次元の配列（[n][1]）。STAIR2D_EXT_RPOLYLINE_GEOMに応じた蹴上ポリラインの追加データ。STAIR2D_FULL_RPOLYLINE_FLAGSと同様の構造。

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

n = 蹴上数 * 各踏面に2つの節点（通常）：

- [n][1] - 蹴上のインデックス

STAIR2D_DRAIN_TPOLYGON_GEOM

2次元の配列（[n][3]）。排水溝ポリゴンの節点のデータトリプレット。STAIR2D_FULL_TPOLYGON_GEOMと同様の構造。

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

n = 踏面数 * 各踏面に5つの節点（通常）：

- [n][1] - 階段の原点から計測した拡張踏面ポリゴンの節点の座標x
- [n][2] - 階段の原点から計測した拡張踏面ポリゴンの節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

STAIR2D_DRAIN_TPOLYGON_FLAGS

2次元の配列（[n][4]）、STAIR2D_DRAIN_TPOLYGON_GEOMに応じた排水溝ポリゴンの追加データ。STAIR2D_FULL_TPOLYGON_FLAGSと同様の構造。

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

n = 踏面数 * 各踏面に5つの節点（通常）：

- [n][1] - 踏面のインデックス
- [n][2] - 節点から開始する辺タイプ
  - 0 - リーディング
  - 1 - トレーリング
  - 2 - 左（+100 - 左側排水溝、+200 - 左側排水溝および「段」タイプ）
  - 3 - 右（+100 - 右側排水溝、+200 - 右側排水溝および「段」タイプ）
  - -1 - クロージング
- [n][3] - 節点から開始する辺の可視度（1-可視、0-省略）
- [n][4] - 踏面のタイプ（0-階段、1-踊り場）

階段構造の2D変数 - 梁の構造

STAIR2D_POLYLINES_GEOM

2次元の配列（[n][3]）。nは2Dでの現在の階段/踊り場の構造ポリラインの節点の数であり、階段の境界から派生した次のポリラインの節点の形状データを含む：左境界線、右境界線および中央線。

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [n][1] - 階段の原点から計測した構造ポリラインの節点の座標x
- [n][2] - 階段の原点から計測した構造ポリラインの節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

STAIR2D_POLYLINES_FLAGS

2次元の配列（[n][1]）。nは2Dでの現在の階段/踊り場の構造ポリラインの節点の数であり、ポリラインの節点のグループデータを含む。

2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [n][1] - ポイントnが次に属する位置フラグ：0 - 左境界、1 - 右境界、2 - 中央線

STAIR2D_STRUCT_ATTRIBUTES											2次元の配列（[2][7]）。構造の表示（[1][n]）および非表示（[2][n]）部分の属性設定を含む
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0, 0, 0, 0, 0, 0, 0]

- [n][1] - 境界線種インデックス
- [n][2] - 境界ペンインデックス
- [n][3] - シンボル塗りつぶし種類
- [n][4] - シンボル塗りつぶしペン
- [n][5] - シンボル塗りつぶし背景ペン
- [n][6] - シンボル塗りつぶしオン/オフのブール制御（カスタム表示のみ）（STAIR2D_CUSTOMDISPLAY = 1）
- [n][7] - シンボル部分有効化オン/オフのブール制御（カスタム表示のみ）（STAIR2D_CUSTOMDISPLAY = 1）

階段構造の2D変数 - 板形構造

互換性：ARCHICAD 22で導入されました。

これらのグローバルには、実際の平面図/見上げ図ビューに関する値が入力されます。

STAIR2D_FULL_SPOLYGON_GEOM											2次元の配列（[n][3]）。階段2D投影のサブポリゴンを含む。
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0]

- [n][1] - 階段の原点から計測した構造ポリゴンの節点の座標x
- [n][2] - 階段の原点から計測した構造ポリゴンの節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、＞0-反時計回りの曲線、＜0-時計回りの曲線）

STAIR2D_FULL_SPOLYGON_FLAGS 2次元の配列（[n][2]）。nはポリゴンノードの数。

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [n][1] - サブポリゴンのインデックス
- [n][2] - サブポリゴンのタイプ
  - 1: 階段境界 - 接続なし
  - 2: 階段境界 - 接続あり
  - 3: 踊り場境界 - 接続なし
  - 4: 踊り場境界 - 接続あり
  - 5: 接続
  - 6: 排水溝
- [n][3] - サブポリゴン辺のタイプ
  - -1: 終了点（次の点は新しいサブポリゴン）
  - 0: 非表示辺（バインド領域のみ）
  - 1: 表示板形辺（表示または非表示の属性はSTAIR2D_VISIBILITYで設定）
  - 2: 表示接続辺（表示または非表示の属性はSTAIR2D_VISIBILITYで設定）
  - 3: 表示接続詳細辺（表示または非表示の属性はSTAIR2D_VISIBILITYで設定）

  接続タイプのサブポリゴンに板形と接続の混合の辺を含めることができます。

STAIR2D_FULL_SPOLYLINE_GEOM 2次元の配列（[n][2]）。境界内の辺を含む。

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [2*i][1] - 階段の原点から計測した開始節点の座標x
- [2*i][2] - 階段の原点から計測した開始節点の座標y
- [2*i + 1][1] - 階段の原点から計測した終了節点の座標x
- [2*i + 1][2] - 階段の原点から計測した終了節点の座標y

STAIR2D_FULL_SPOLYLINE_FLAGS				2次元の配列（[n]）、nは、STAIR2D_FULL_SPOLYLINE_GEOMの節点の数							
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

- [n] - 辺のタイプ
  - 1: 踏面、排水溝で切断
  - 2: 踏面、全て
  - 3: 踏面の段鼻（蹴上（傾斜））、排水溝で切断
  - 4: 踏面の段鼻（蹴上（傾斜））、全て
  - 5: 踊り場線、排水溝で切断
  - 6: 踊り場線、全て
  - 7: 接続
  - 8: 接続の詳細
  - 9: 排水溝（段排水溝）
  - 10: 排水溝の段鼻（傾斜段のある排水溝）

STAIR2D_MONOLITH_ATTRIBUTES　　2次元の配列（[2][23]）。板形構造の表示（[1][n]）および非表示（[2][n]）部分の属性設定と可視度設定を含む。

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [n][1] - 構造可視度（ブール）
- [n][2] - 排水溝可視度（ブール）
- [n][3] - 輪郭の線種
- [n][4] - 輪郭線ペン
- [n][5] - 構造踏面可視度（ブール）
- [n][6] - 排水溝線可視度（ブール）
- [n][7] - 踏面の線種
- [n][8] - 踏面線ペン
- [n][9] - 構造踏面段鼻可視度（ブール）
- [n][10] - 踏面段鼻の線種
- [n][11] - 踏面段鼻線ペン
- [n][12] - 接続可視度（ブール）
- [n][13] - 接続の線種
- [n][14] - 接続線ペン
- [n][15] - 接続詳細の線種
- [n][16] - 接続詳細線ペン
- [n][17] - 排水溝の塗りつぶしおよび構造の塗りつぶしの可視度（ブール）
- [n][18] - 構造の塗りつぶし種類
- [n][19] - 構造の塗りつぶしペン
- [n][20] - 構造の塗りつぶし背景ペン
- [n][21] - 排水溝の塗りつぶし種類
- [n][22] - 排水溝の塗りつぶしペン
- [n][23] - 排水溝の塗りつぶし背景ペン

2D関連の一般的な変数

STAIR2D_CURRSTORY_LOCATION　　階段の現在のフロアの可視度についての情報

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	0

values:

1: 関連フロアの下、
2: 最初の関連フロア、
3: 関連フロア間、
4: 上部最後の関連フロア、
5: 関連フロアの上。

STAIR2D_LAYOUT_TYPES[5]　1次元の配列（[5]）。STAIR2D_CURRSTORY_LOCATIONに応じた階段の表示レイアウトタイプについての情報

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	[0]

- [1] - 関連フロアの下のレイアウト設定
- [2] - 最初の関連フロアのレイアウト設定
- [3] - 関連フロア間のレイアウト設定
- [4] - 上部最後の関連フロアのレイアウト設定
- [5] - 関連フロアの上のレイアウト設定

Layout setting values:

0: 無効
1: 破断線あり：表示 - 非表示
2: 破断線なし：表示
3: 破断線の下：表示
4: 破断線の上：表示
7: 破断線なし：非表示
8: 破断線の上：非表示
9: 破断線の下：非表示
13: 破断線あり：全て表示
14: 破断線あり：非表示 - 表示
5: 複数フロア2D：破断線間の表示
6: 複数フロア2D：非表示 - 表示 - 非表示
10: 複数フロア2D：全て表示
11: 複数フロア2D：非表示 - 表示 - なし
12: 複数フロア2D：なし - 表示 - 非表示

STAIR2D_VISIBILITY　現在の図面の有効な属性セットのタイプ

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0

通り芯および踏面の2Dシンボルと組み合わせてのみ使用可能

values:

1: 「表示」属性セットが有効
0: 「非表示」属性セットが有効

STAIR2D_CUSTOMDISPLAY										階段のモデルビュー設定についての情報を含む	
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0

values:

0: モデル表示オプション設定に従って階段を表示
1: カスタム設定で階段を表示

STAIR_START_WITH_RISER										階段が蹴上で開始するかどうか示すブール。	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	0

互換性：ARCHICAD 22で導入されました。

STAIR_END_WITH_RISER										階段が蹴上で終了するかどうか示すブール。	
2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	0

互換性：ARCHICAD 22で導入されました。

STAIR_TREAD_EXIST										階段に踏み面構成要素があるかどうか示すブール配列（[2]）。	
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0

互換性：ARCHICAD 22で導入されました。
編集モードでカスタマイズされた踏み面はこれらの値に影響を与えません。

- [1] - 階段
- [2] - 踊り場

STAIR_RISER_EXIST										階段に蹴上構成要素があるかどうか示すブール配列（[2]）。	
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0

互換性：ARCHICAD 22で導入されました。
編集モードでカスタマイズされた蹴上はこれらの値に影響を与えません。

- [1] - 階段
- [2] - 踊り場

STAIR_NOSING_EXIST										階段に踏面段鼻（長さ> 0）が設定されているかどうか示すブール配列（[2]）。	
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0

互換性：ARCHICAD 22で導入されました。
編集モードでカスタマイズされた踏み面はこれらの値に影響を与えません。

- [1] - 階段
- [2] - 踊り場

階段の3D変数 - 3D表示（および接続されるビューポイント）のみで使用可能

互換性：ARCHICAD 21で導入されました。

階段の蹴上の3D変数

RISER_HEIGHT — 選択した蹴上の3D高さの値

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	0

RISER_THICKNESS — 選択した蹴上の3D厚さの値

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	0

STAIR_RISER_GEOMETRY — 2次元の配列（[n][3]）。階段の蹴上のポリラインパスの節点のデータトリプレット

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

n = 蹴上のポリラインパスの節点の数：

- [n][1] - 階段の原点から計測したポリラインの節点の座標x
- [n][2] - 階段の原点から計測したポリラインの節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

RISER_CUT — 2次元の配列（[2][2]）。理想的なTUBE（3Dで蹴上をモデリング）の開始および終了ポイントのデータを含む

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

チューブの表示分割は、この変数の値には影響がありません。あらゆるケースで正確なモデルを実現するには、このグローバルを使用して、分割された円弧に続くチューブの実際の開始および終了ポイントを計算します。

- [n][1] - 階段の原点から計測した開始節点の座標x
- [n][1] - 階段の原点から計測した開始節点の座標y
- [n][2] - 階段の原点から計測した終了節点の座標x
- [n][2] - 階段の原点から計測した終了節点の座標y

RISER_SLANT_ANGLE — 選択した蹴上の傾斜角度

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	0

傾斜が0に設定されている場合、このグローバルの値は90度です。

階段の踏面の2D-3D変数

TREAD_THICKNESS — 選択した踏面の3D厚さの値

2D	✅	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	0

STAIR_TREAD_GEOMETRY				2次元の配列（[n][3]）。階段の踏面ポリゴンの節点のデータトリプレット							
2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0]

n = 踏面ポリゴンの節点の数

- [n][1] - 階段の原点から計測した節点の座標x
- [n][2] - 階段の原点から計測した節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

STAIR_TREAD_FLAGS				2次元の配列（[n][1]）。STAIR_TREAD_GEOMETRYに応じた踏面ポリゴンの辺（節点から開始）の追加データ							
2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0]

n = 踏面ポリゴンの節点の数

- [n][1] - ポリゴンのnth辺のフラグ

flags:

0: 踏面ポリゴンのリード辺
1: 踏面ポリゴンのトレイル辺
2: 踏面ポリゴンの左辺
3: 踏面ポリゴンの右辺
-1: 踏面ポリゴンの終了節点

TREAD_LOWER_RISER_THICKNESS				現在の踏面下の蹴上の厚さ（通り芯と構造の間を計測）							
2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	0

TREAD_LOWER_RISER_HEIGHT				現在の踏面下の蹴上の高さ（現在の踏面下部の平面と前の踏面上部の平面との間を計測）							
2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	0

TREAD_LOWER_RISER_SLANT_ANGLE				現在の踏面下の蹴上の傾斜角度（単位：度）（傾斜 = 0の場合、値は90度）							
2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	0

TREAD_UPPER_RISER_THICKNESS				現在の踏面上の蹴上の厚さ（通り芯と構造の間を計測）							
2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	0

TREAD_UPPER_RISER_HEIGHT				現在の踏面上の蹴上の高さ（現在の踏面上部の平面と次の踏面下部の平面との間を計測）							
2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	0

TREAD_UPPER_RISER_SLANT_ANGLE				現在の踏面上の蹴上の傾斜角度（単位：度）（傾斜 = 0の場合、値は90度）							
2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	0

TREAD_NOSING_METHOD										[階段の設定]ダイアログで設定された現在の踏面の段鼻方式についての情報	
2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0

values:

1: 値の長さによる段鼻
2: 傾斜の長さによる段鼻

TREAD_NOSING										TREAD_NOSING_METHOD = 1（値の長さによる段鼻）の場合、選択した踏面の踏み面 段鼻の奥行きの値を含む（水平オフセット、[階段の設定]ダイアログでの設定）	
2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0

TREAD_NOSING_BY_SLANTING										TREAD_NOSING_METHOD = 2（傾斜の長さによる段鼻）の場合、踏み面 段鼻の長さの値を含む（蹴上の交点を制御する垂直オフセット、[階段の設定]ダイアログでの設定）	
2D	✓	3D	✓	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0

階段構造の変数

STAIR_STRUCTURE_GEOMETRY											2次元の配列（[n][14]）。nは現在の階段/踊り場の構造ポリゴンの頂点数
2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	テ#フォルト	[0]

- [n][1] - 階段の原点から計測した構造ポリゴンの節点nの座標x
- [n][2] - 階段の原点から計測した構造ポリゴンの節点nの座標y
- [n][3] - 階段の原点から計測した構造ポリゴンの節点nの座標z
- [n][4] - 節点nから開始する辺の中心角（単位：度）（0-直線、＞0-反時計回りの曲線、＜0-時計回りの曲線）
- [n][5] - 節点nから開始する辺の蹴上がりの傾斜角度。リーディング辺の場合は、角度はポリゴン下の蹴上に属します。トレーリング辺の場合は、角度はポリゴン上の蹴上に属します。その他全ての辺カテゴリには0が含まれます。
- [n][6] - 構造ポリゴン下の蹴上の高さ（蹴上がない場合、またはポリゴンが接続ポリゴンの場合は0を返す）
- [n][7] - 構造ポリゴン上の蹴上の高さ（蹴上がない場合、またはポリゴンが接続ポリゴンの場合は0を返す）
- [n][8] - 構造ポリゴンの踏面の厚さ（ポリゴンが接続ポリゴンの場合は0を返す）
- [n][9] - 構造ポリゴンの踏面下の間隔の厚さ（ポリゴンが接続ポリゴンの場合は0を返す）
- [n][10-14] - 互換性：ARCHICAD 22で導入されました。
- [n][10] - この点の下の片持ち構造高さ（ac_stairStructureThicknessパラメータから読み取られる）。接続ポリゴンの場合は0。
- [n][11] - この点から開始される辺の踏み面段鼻。
  - 接続ポリゴンの場合は0
  - リーディング辺の上の踏み面の段鼻
  - トレーリング辺の上の次の踏み面の段鼻
  - その他の辺の場合は0。
- [n][12] - この点から開始される辺の蹴上厚さ。
  - 接続ポリゴンの場合は0
  - リーディング辺の下の蹴上厚さ
  - トレーリング辺の上の蹴上厚さ
  - その他の辺の場合は0。
- [n][13] - この点から開始される辺の蹴上間隔厚さ。
  - 接続ポリゴンの場合は0
  - リーディング辺の下の間隔蹴上厚さ
  - トレーリング辺の上の間隔蹴上厚さ
  - その他の辺の場合は0。
- [n][14] - この点から開始される辺の下持ち構造水平厚さ（ac_stairStructureHorizThickパラメータから読み取られる）。
  - 接続ポリゴンの場合は0
  - リーディング辺の下の下持ち構造の水平厚さ
  - トレーリング辺の上の下持ち構造の水平厚さ
  - その他の辺の場合は0。

STAIR_STRUCTURE_FLAGS											2次元の配列（[n][3]）。nは現在の階段/踊り場の構造ポリゴンの頂点数
2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	デ#フォルト	[0]

- [n][1] - 構造ポリゴンの節点nのポリゴンタイプ：躯体ポリゴンまたは接続ポリゴン
- [n][2] - 節点nが属する側を表示：階段の中央線で分割する場合の左側または右側構造ポリゴン：0 - 左、1 - 右
- [n][3] - 節点nから開始する辺のタイプ。次の値が有効：0 - リーディング辺、1 - トレーリング辺、2 - 左辺、3 - 右辺、4 - 中央線辺、-1 - 終了辺。

values: [n][1]で節点カテゴリを表示

-1: 接続なし（踏面全体）ポリゴン：踊り場の場合、全てのポイントがこのカテゴリにあります。階段ポリゴンの場合は、このような値はありません。

0: 接続なし（躯体）ポリゴン：踊り場の場合、開始-終了接続が形状に影響を及ぼします。階段の場合、ポリゴンは常に完成しています。

1: 接続開始ポリゴン：残存ポリゴン、または躯体と接続ポリゴンの間の閉じたポリゴンを示します。梁のパスを定義する、次の節点が必要です：境界ポリゴンポイントをもつ始点または終点。

2: 接続開始位置ポリゴン：接続の位置を示します。このポリゴンは常に包括ユニットポリゴンであり、梁のパスの一部ではありません。切断面と終端プレート位置にのみ必要です。

3: 接続開始区切りポリゴン：垂直または水平接続の場合に、ブレークポイントを含むポリゴンを示します。梁のパスに、次のポイントが必要です：境界ポリゴンポイントで完了する始点または終点。

4: 接続開始方向ポリゴン：2等分線接続の場合に2等分線切断を行う必要があるポリゴンを示します。このポリゴンは常に包括ユニットポリゴンであり、梁のパスの一部ではありません。切断面の位置を取得するためにのみ必要です。

5: 接続開始延長ポリゴン：切断部分を支持する梁の延長ポイントを示します。梁のパスに、次のポイントが必要です：境界ポリゴンポイントで完了する始点または終点。

6: 接続開始構造境界ポリゴン：梁の境界ポイントの始点を示します。このポリゴンは常に包括ユニットポリゴンです。梁のパスに、次のポイントが必要です：始点または終点。

101: 接続終了ポリゴン：残存ポリゴン、または躯体と接続ポリゴンの間の閉じたポリゴンを示します。梁のパスを定義する、次の節点が必要です：境界ポリゴンポイントをもつ始点または終点。

102: 接続終了位置ポリゴン：接続の位置を示します。このポリゴンは常に包括ユニットポリゴンであり、梁のパスの一部ではありません。切断面と終端プレート位置にのみ必要です。

103: 接続終了区切りポリゴン：垂直または水平接続の場合に、ブレークポイントを含むポリゴンを示します。梁のパスに、次のポイントが必要です：境界ポリゴンポイントで完了する始点または終点。

104: 接続終了方向ポリゴン：2等分線接続の場合に2等分線切断を行う必要があるポリゴンを示します。このポリゴンは常に包括ユニットポリゴンであり、梁のパスの一部ではありません。切断面の位置を取得するためにのみ必要です。

105: 接続終了延長ポリゴン：切断部分を支持する梁の延長ポイントを示します。梁のパスに、次のポイントが必要です：境界ポリゴンポイントで完了する始点または終点。

106: 接続終了構造境界ポリゴン：梁の境界ポイントの終点を示します。このポリゴンは常に包括ユニットポリゴンです。梁のパスに、次のポイントが必要です：始点または終点。

STAIR_STRUCTURE_CONN_OFFSETS										2次元の配列（[2][6]）。接続オフセットポイントのデータを含む	
2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	デ#フォルト	[0]

- [1][1] - 接続開始水平オフセット（dx）
- [1][2] - 接続開始垂直オフセット（dy）
- [1][3] - 接続開始水平オフセット 2（dx1）
- [1][4] - 接続開始垂直オフセット 2（dy1）
- [1][5] - 接続開始水平オフセット（cx）
- [1][6] - 接続開始垂直オフセット（cy）
- [2][1] - 接続終了水平オフセット（dx）
- [2][2] - 接続終了垂直オフセット（dy）
- [2][3] - 接続終了水平オフセット 2（dx1）
- [2][4] - 接続終了垂直オフセット 2（dy1）
- [2][5] - 接続終了水平オフセット（cx）
- [2][6] - 接続終了垂直オフセット (cy)

STAIR_STRUCTURE_CONN_FLAGS										2次元の配列（[2][3]）。構造接続の追加データを含む	
2D	✓	3D	✓	UI	✓	Parameter	⛔	Property	⛔	デ#フォルト	0

- [1][1] - 接続開始タイプ： 0 - 垂直切断、1 - 水平切断、2 - カットアウト、3 - 垂直および水平切断、4 - 水平接続、5 - 水平接続およびカットアウト、6 - 垂直接続、7 - 二等分線、8 - 自動。
- [1][2] - 接続開始役割：0 - 踏面および先端、1 - 踏面および踊り場、2 - 踊り場および踏面、3 - 踏面および踏面、4 - 踏面および終端、5 - 踊り場および踊り場。
- [1][3] - 踏面または蹴上の開始：0 - 蹴上の開始、1 - 踏面の開始。
- [2][1] - 接続終了タイプ： 0 - 垂直切断、1 - 水平切断、2 - カットアウト、3 - 垂直および水平切断、4 - 水平接続、5 - 水平接続およびカットアウト、6 - 垂直接続、7 - 二等分線、8 - 自動。
- [2][2] - 接続終了役割：0 - 踏面および先端、1 - 踏面および踊り場、2 - 踊り場および踏面、3 - 踏面および踏面、4 - 踏面および終端、5 - 踊り場および踊り場。
- [2][3] - 踏面または蹴上の終了：0 - 蹴上の終了、1 - 踏面の終了。

STAIR_STRINGER_PATH_OFFSET										[階段の設定]ダイアログの「踏み面からの高さ」に設定された値を含む	
2D	⛔	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	0

レール構成要素パラメータ

手摺りの一般的な変数 - リストおよびラベルに使用可能

互換性：ARCHICAD 21で導入されました。

RAILING_HEIGHT				レールセグメントの高さ（[手摺設定]ダイアログ /セグメント設定の指定）							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
RAILING_3DLENGTH				手摺り全体の3D長さ							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
RAILING_HORIZONTAL_LENGTH				手摺り全体の投影2D長さ							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
RAILING_VOLUME				手摺りの体積（全てのサブ要素を含む）							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
RAILING_NR_OF_SEGMENTS				手摺りの辺数							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
RAILING_NR_OF_POSTS				手摺りの支柱の数							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
RAILING_NR_OF_BALUSTERS				手摺りの手摺子の数							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
RAILING_NR_OF_PANELS				手摺りのパネル数							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0
RAILING_NR_OF_RAILS				手摺りのレール数							
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	テ#フォルト	0

手摺りの3D変数

互換性：ARCHICAD 21で導入されました。

RAILING_REFLINE_DISTANCE				1次元の配列（[2]）。手摺り基準線からのレール要素の水平オフセット							
2D	✓	3D	✓	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0]

- [1] - 手摺り基準線からの分節基準線のオフセット
- [2] - 分節基準線からのパネル要素のオフセット（子柱、支柱またはトップレールの場合は常に0）

RAIL_CONNECTING_POSTS_NUM — レールと交差する支柱および子柱の数

2D		3D		UI		Parameter		Property		デ#フォルト	0

RAIL_CONNECTING_POSTS — 2次元の配列（[n][2]）。nはレールに沿った支柱または子柱の数。これらの交差要素の位置データを含む

2D		3D		UI		Parameter		Property		デ#フォルト	[0]

- [n][1] - レールに沿った支柱および子柱の位置。位置の値はレールの長さに応じて比例的に計算されます（値は0～1の間）。
- [n][2] - 支柱または子柱のスキュー値（パネルの場合と同様）。コーナー要素の値は要素間の値と異なる場合があります。

RAIL_TYPE — 現在のライブラリ部品が選択されているサブ要素タイプを含む

2D		3D		UI		Parameter		Property		デ#フォルト	0

- 1 - トップレール（または トップレール終端）
- 2 - ハンドレール（またはハンドレール終端）
- 3 - レール（またはレール終端）

RAIL_POLYLINE_GEOMETRY — 2次元の配列（[n][5]）。nはレールノードの数（n > 2）。現在の手摺りの全ての分節の形状データを含む。

2D		3D		UI		Parameter		Property		デ#フォルト	[0]

- [n][1] - レールノードの座標x
- [n][2] - レールノードの座標y
- [n][3] - レールノードの座標z
- [n][4] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）
- [n][5] - TUBEコマンドの断面の回転。自動的に拡張されたツイストレール接続の回転は、傾斜値と異なる場合があります。

RAIL_SEGMENT_FLAGS — 1次元の配列（[n]）。nはレールノードの数（n > 2）。現在の手摺りの全ての分節の形状データを含む。

2D		3D		UI		Parameter		Property		デ#フォルト	[0]

RAIL_POLYLINE_GEOMETRYに従って、インデックスにより節点が属する分節を識別します。同じ分節の節点はまとめられます。

RAIL_CUTS 2次元の配列（[2][4]）。レールの終端材質の向きを定義します。

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

TUBEコマンドで、最初と最後のパス座標に戻り座標をそのまま使用することができます。[1][n]の項目は開始頂点に属し、[2][n]は終了頂点の座標を設定します。

- [1][1] - 座標x
- [1][2] - 座標y
- [1][3] - 座標z
- [1][4] - ブール、切断方法：0 - 連続する辺、1 - 留め切り

RAIL_DISCONNECTED_CUTS 2次元の配列（[2][4]）。切り離された接続の場合、終端材質の向きを定義します。

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

TUBEコマンドで、最初と最後のパス座標に戻り座標をそのまま使用することができます。[1][n]の項目は最初の分節の終了頂点に属し、[2][n]は2番目の分節の開始頂点の座標を設定します。

- [1][1] - 座標x
- [1][2] - 座標y
- [1][3] - 座標z
- [1][4] - 0 （追加の開発のために予約されています）

互換性：ARCHICAD 22で導入されました。

RAIL_COMPONENTS 1次元の配列（[3]）。レールについての追加情報を含む。

2D	✅	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

ブールタイプは取付け金具、キャップの必要性/存在を示します。

- [1] - 取付け金具が必要（接続または延長なし）
- [2] - 開口キャップが必要（レールの先端、またはその他のレールへの接続）
- [3] - 終端キャップが必要（レールの終端、またはその他のレールへの接続）

RAIL_SLANT_ANGLE 水平移動方向に対して垂直の方向を基準とするレールの傾斜角度（単位：度）

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	0

values: 使用可能な値（< 90度）

positive: 水平移動方向に対して左に傾斜

negative: 水平移動方向に対して右に傾斜

曲面の側面：角度は垂直の扇形平面で計測されます。

RAILINGPANEL_TYPE 平面の一般的な形状タイプ

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	0

- 1 - 平面
- 2 - 円柱（曲線 - 垂直または曲線 - スキュー）
- 3 - 円錐（曲線 - 水平 - 傾斜）
- 4 - ツイスト（曲線 - 傾斜 - 傾斜または曲線 - 傾斜 - スキュー）

RAILINGPANEL_UNCUT_GEOMETRY 2次元の配列（[n][4]）。nはパネルノードの数（n > 3）。生の形状が完全な（切断のない）現在の手摺りパネルの形状データを含む

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

- [n][1] - 手摺りのパネルノードの座標x
- [n][2] - 手摺りのパネルノードの座標y
- [n][3] - 手摺りのパネルノードの座標z
- [n][4] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

RAILINGPANEL_GEOMETRY 2次元の配列（[n][5]）。nはパネルノードの数（n > 3）。切断面が適用された現在の手摺りパネルの形状データを含む

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

- [n][1] - 手摺りのパネルノードの座標x
- [n][2] - 手摺りのパネルノードの座標y
- [n][3] - 手摺りのパネルノードの座標z
- [n][4] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）
- [n][5] - 切断辺の場合は、切断面の角度を含む（0 - パネルに対して垂直）。曲面パネルの場合、基準面は接線。

RAILINGPANEL_FLAGS 2次元の配列（[n][3]）。nはRAILINGPANEL_GEOMETRYに応じたパネルノードの数（n > 3）。切断面が適用された現在の手摺りパネル辺の形状データを含む

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

- [n][1] - 投影パネルの節点から開始する辺の分節数
- [n][2] - 投影パネルの節点から開始する辺の位置（RAILINGPANEL_SIDE_OFFSETSに応じたインデックス）
- [n][3] - 辺のステータスビット

mask: [n][3] 使用可能な値

mask = j_1 + 2*j_2. ここで、 各 jiフラグは0または1をとります.

j_1: 節点から開始しその他のパネルに接続する辺

j_2: 節点から開始する辺を切断

RAILINGPANEL_SIDE_OFFSETS

1次元の配列（[4]）。[手摺設定]ダイアログで設定されたパネルオフセットを含む。 垂直平面の隣接する要素軸の距離（傾斜とスキューは無視）

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [1] - 下部オフセット
- [2] - 終了頂点側面オフセット
- [3] - 上部オフセット
- [4] - 開始頂点側面オフセット

RAILINGPANEL_SLANT_ANGLE

傾斜角度（単位：度）。水平移動方向に対して垂直。垂直は0度で、正の値は水平移動方向に対して左を、負の値は右を意味します（角度< 90）。曲線の辺：辺の開始接平面に対して垂直な平面で計測（スキュー前）

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	0

RAILINGPANEL_SKEW_ANGLE

スキュー角度（単位：度）。水平移動方向に対して平行。垂直は0度で、正の値は水平移動方向に従って後方の、負の値は前方のスキューを意味します（角度< 90）。曲線の辺：辺の開始接平面で計測（傾斜前）

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	0

RAILINGPOST_TYPE

現在のライブラリ部品が選択されているサブ要素タイプを含む

2D	✅	3D	✅	UI	✅	Parameter	✅	Property	⛔	デフォルト	0

- 1 - 支柱
- 2 - 子柱
- 3 - 手摺子

RAILINGPOST_TOP_COORD

1次元の配列（[3]）。現在の支柱の上部の座標

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0, 0, 0]

- [1] - 座標x
- [2] - 座標y
- [3] - 座標z

RAILINGPOST_CUTS 2次元の配列（[2][3]）。支柱の終端材質の方向を定義

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0]

TUBEコマンドで、最初と最後のパス座標に戻り座標をそのまま使用することができます。[1][n]の項目は基部頂点に属し、[2][n]は上部頂点の座標を設定します。

- [1][1] - 座標x
- [1][2] - 座標y
- [1][3] - 座標z

RAILINGPOST_SEGMENT_CUTS 2次元の配列（[2][4]）。手摺りの傾斜時に辺の境界で子柱と手摺子を切断するために使用される2つの平面のベクトルパラメータを含む

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0]

平面の法線はベクトル（A; B; C）であり、Dは法線の向きで計測された原点からの距離です。モデルは法線の向きで切断されます。 2つの平面のいずれかで全て0の戻り値の場合、平面は存在しません（切断なし）。

- [1][1] - A
- [1][2] - B
- [1][3] - C
- [1][4] - D

RAILINGEND_DIRECTION_AND_ANGLE 2次元の配列（[n][5]）。接線方向の接続レールから離れたポイントのベクトルデータ。直線レールの2つの終端では、これらのベクトルの向きは反対です。

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0]

- [n][1] - ベクトルの座標x
- [n][2] - ベクトルの座標y
- [n][3] - ベクトルの座標z
- [n][4] - ベクトルの軸周囲での、傾斜の効果をもつ接続レールの回転。継ぎ目なく接続するには、レール終端を同じ回転でモデル化する必要があります。

手摺りの2D変数

互換性：ARCHICAD 21で導入されました。

RAIL2D_FULL_POLYLINE_GEOM 2次元の配列（[n][3]）。nはパネル/レール/レール終端軸ポリラインの頂点数。手摺りパネルの形状データを含む

2D	✅	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	テ#フォルト	[0]

- [n][1] - 手摺りの原点から計測した節点の座標x
- [n][2] - 手摺りの原点から計測した節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、＞0-反時計回りの曲線、＜0-時計回りの曲線）

RAIL2D_FULL_POLYLINE_FLAGS　2次元の配列（[n][1]）。nはパネル/レール/レール終端軸ポリラインの頂点数。辺の表示データを含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [n][1] - 節点nで開始する辺の表示フラグ（0 - 非表示、1 - 表示）

RAIL2D_FULL_POLYGON_GEOM　2次元の配列（[n][3]）。nはパネル/レール/レール終端シンボルポリゴンの頂点数。手摺りパネルの形状データを含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [n][1] - 手摺りの原点から計測した節点の座標x
- [n][2] - 手摺りの原点から計測した節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、＞0-反時計回りの曲線、＜0-時計回りの曲線）

RAIL2D_FULL_POLYGON_FLAGS　2次元の配列（[n][1]）。nはパネル/レール/レール終端シンボルポリゴンの頂点数。辺の表示データを含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [n][1] - 節点nで開始する辺の表示フラグ（0 - 非表示、1 - 表示）

RAIL2D_FULL_VISIBILITY　現在の手摺りの図面の有効な属性セットのタイプ

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	0

values:

1: 「表示」属性セットが有効
0: 「非表示」属性セットが有効

RAIL2D_CUSTOMDISPLAY　手摺りのモデルビュー設定についての情報を含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	0

values:

0: モデル表示オプション設定に従って手摺りを表示
1: カスタム設定で手摺りを表示

次のグローバルを使用して、最初の破断線の下に示される手摺りの部分を定義します。

RAIL2D_LOWER_POLYLINE_GEOM 2次元の配列（[n][3]）。nは最初の破断線下のパネル/レール/レール終端軸ポリラインの頂点数。手摺りパネルポリラインの形状データを含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [n][1] - 手摺りの原点から計測した節点の座標x
- [n][2] - 手摺りの原点から計測した節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

RAIL2D_LOWER_POLYLINE_FLAGS 2次元の配列（[n][1]）。nは最初の破断線下のパネル/レール/レール終端軸ポリラインの頂点数。辺の表示データを含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [n][1] - 節点nで開始する辺の表示フラグ（0 - 非表示、1 - 表示）

RAIL2D_LOWER_POLYGON_GEOM 2次元の配列（[n][3]）。nは最初の破断線下のパネル/レール/レール終端シンボルポリゴンの頂点数。手摺りパネルポリゴンの形状データを含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [n][1] - 手摺りの原点から計測した節点の座標x
- [n][2] - 手摺りの原点から計測した節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、> 0-反時計回りの曲線、< 0-時計回りの曲線）

RAIL2D_LOWER_POLYGON_FLAGS 2次元の配列（[n][1]）。nは最初の破断線下のパネル/レール/レール終端シンボルポリゴンの頂点数。ポリゴン辺の表示データを含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [n][1] - 節点nで開始する辺の表示フラグ（0 - 非表示、1 - 表示）

RAIL2D_LOWER_VISIBILITY 最初の破断線下の現在の手摺りの図面の有効な属性セットのタイプ

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	0

values:

1: 「表示」属性セットが有効
0: 「非表示」属性セットが有効

次のグローバルを使用して、2つの破断線の間に示される手摺りの部分を定義します。

RAIL2D_MIDDLE_POLYLINE_GEOM 2次元の配列（[n][3]）。nは破断線間のパネル/レール/レール終端軸ポリラインの頂点数。手摺りパネルポリラインの形状データを含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

- [n][1] - 手摺りの原点から計測した節点の座標x
- [n][2] - 手摺りの原点から計測した節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、＞0-反時計回りの曲線、＜0-時計回りの曲線）

RAIL2D_MIDDLE_POLYLINE_FLAGS 2次元の配列（[n][1]）。nは破断線間のパネル/レール/レール終端軸ポリラインの頂点数。辺の表示データを含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

- [n][1] - 節点nで開始する辺の表示フラグ（0 - 非表示、1 - 表示）

RAIL2D_MIDDLE_POLYGON_GEOM 2次元の配列（[n][3]）。nは破断線間のパネル/レール/レール終端シンボルポリゴンの頂点数。手摺りパネルポリゴンの形状データを含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

- [n][1] - 手摺りの原点から計測した節点の座標x
- [n][2] - 手摺りの原点から計測した節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、＞0-反時計回りの曲線、＜0-時計回りの曲線）

RAIL2D_MIDDLE_POLYGON_FLAGS 2次元の配列（[n][1]）。nは破断線間のパネル/レール/レール終端シンボルポリゴンの頂点数。ポリゴン辺の表示データを含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0]

- [n][1] - 節点nで開始する辺の表示フラグ（0 - 非表示、1 - 表示）

RAIL2D_MIDDLE_VISIBILITY 破断線間の現在の手摺りの図面の有効な属性セットのタイプ

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0

values:

1: 「表示」属性セットが有効
0: 「非表示」属性セットが有効

次のグローバルを使用して、最後の破断線の上に示される手摺りの部分を定義します。

RAIL2D_UPPER_POLYLINE_GEOM — 2次元の配列（[n][3]）。nは最後の破断線上のパネル/レール/レール終端軸ポリラインの頂点数。手摺りパネルポリラインの形状データを含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [n][1] - 手摺りの原点から計測した節点の座標x
- [n][2] - 手摺りの原点から計測した節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、＞0-反時計回りの曲線、＜0-時計回りの曲線）

RAIL2D_UPPER_POLYLINE_FLAGS — 2次元の配列（[n][1]）。nは最後の破断線上のパネル/レール/レール終端軸ポリラインの頂点数。辺の表示データを含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [n][1] - 節点nで開始する辺の表示フラグ（0 - 非表示、1 - 表示）

RAIL2D_UPPER_POLYGON_GEOM — 2次元の配列（[n][3]）。nは最後の破断線上のパネル/レール/レール終端シンボルポリゴンの頂点数。手摺りパネルポリゴンの形状データを含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [n][1] - 手摺りの原点から計測した節点の座標x
- [n][2] - 手摺りの原点から計測した節点の座標y
- [n][3] - 節点から開始する辺の中心角（0-直線、＞0-反時計回りの曲線、＜0-時計回りの曲線）

RAIL2D_UPPER_POLYGON_FLAGS — 2次元の配列（[n][1]）。nは最後の破断線上のパネル/レール/レール終端シンボルポリゴンの頂点数。ポリゴン辺の表示データを含む

2D	✓	3D	⛔	UI	⛔	Parameter	⛔	Property	⛔	デフォルト	[0]

- [n][1] - 節点nで開始する辺の表示フラグ（0 - 非表示、1 - 表示）

RAIL2D_UPPER_VISIBILITY — 最後の破断線上の現在の手摺りの図面の有効な属性セットのタイプ

2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デフォルト	0

values:

1: 「表示」属性セットが有効
0: 「非表示」属性セットが有効

次のグローバルは、手摺り支柱の2D表示に属します。

RAILPOST2D_VISIBILITY										現在の支柱の図面の有効な属性セットのタイプ
2D	✓	3D	⛔	UI	✓	Parameter	✓	Property	⛔	デ#フォルト	0

values:

1: 「表示」属性セットが有効
0: 「非表示」属性セットが有効

屋根パラメータ - 天窓、リストおよびラベルに使用可能

ROOF_THICKNESS	屋根の厚さ
ROOF_ANGLE	屋根勾配
ROOF_MAT_TOP	屋根上部の材質の属性インデックス
ROOF_MAT_EDGE	屋根の辺の材質の属性インデックス
ROOF_MAT_BOTT	屋根下部の材質の属性インデックス
ROOF_LINETYPE 平面図ウィンドウでのみ輪郭に適用される	屋根の線種
ROOF_FILL 塗りつぶしインデックス-複合構造の場合の値は負	屋根の塗りつぶし
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	屋根の複合構造の名前
ROOF_BMAT_NAME	屋根のビルディングマテリアル名、複合構造スラブの場合は空白
ROOF_BMAT 互換性：ARCHICAD 21で導入されました。	屋根のビルディングマテリアルのインデックス、複合構造屋根の場合は0
ROOF_SKINS_NUMBER 1～8の範囲、単一塗りつぶしの場合は0	複合屋根の層の数

ROOF_SKINS_PARAMS	複合屋根の層のパラメータ

任意の数の行を含む18列の配列：

- [1] 塗りつぶし
- [2] 厚さ
- [3] （古い輪郭ペン）
- [4] 塗りつぶしのペン
- [5] 塗りつぶし背景のペン
- [6] 躯体の状態
- [7] 上部線ペン
- [8] 上部線種
- [9] 下部線ペン
- [10] 下部線種
- [11] 終端面ペン
- [12] 塗りつぶし向き
- [13] 面の種類
- [14] 終端面線種
- [15] 仕上げ塗りつぶし状態
- [16] 与えられた方向の塗りつぶし状態
- [17] 躯体の状態（躯体が存在しない場合は、最も厚い塗りつぶし）
- [18] ビルディングマテリアルのインデックス

躯体の状態：0 - 部分ではない、1 - 部分、3 - 躯体の最終層、塗りつぶし向き：0 - グローバル、1 - ローカル、 面の種類：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_SKINS_BMAT_NAMES	複合屋根の層のビルディングマテリアル名

列が1の配列：任意の数の行を有する層のビルディングマテリアル名

ROOF_SECT_PEN	屋根切断面の輪郭ペン

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）に適用される

屋根パラメータ-リストとラベル用のみ

ROOF_BOTTOM_SURF	屋根の下部の表面積

指定された値より大きい開口表面積を減算しない

ROOF_GROSS_BOTTOM_SURF	屋根下部総表面積

開口表面積を減算

ROOF_BOTTOM_SURF_CON 指定された値より大きい穴表面積を減算	屋根の下部の表面積
ROOF_TOP_SURF 指定された値より大きい開口表面積を減算しない	屋根の上部の表面積
ROOF_GROSS_TOP_SURF 開口表面積を減算	屋根上部総表面積
ROOF_TOP_SURF_CON 指定された値より大きい穴表面積を減算	屋根の条件付き表面積
ROOF_EDGE_SURF 開口表面積を減算しない	屋根辺表面積
ROOF_GROSS_EDGE_SURF 開口表面積を減算	屋根辺総表面積
ROOF_CONTOUR_AREA	屋根に覆われた面積
ROOF_PERIMETER	屋根周囲長さ
ROOF_VOLUME 穴体積を減算しない	屋根の体積
ROOF_GROSS_VOLUME 穴体積を減算	屋根総体積
ROOF_VOLUME_CON 指定された値より大きい穴体積を減算	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_BMAT_NAME	塗りつぶしのビルディングマテリアル名
FILL_BMAT 互換性：ARCHICAD 21で導入されました。	塗りつぶしのビルディングマテリアルのインデックス
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	塗りつぶし周囲長さ
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
FILL_FILL_CATEGORY 0 - 図面、1 - 切断、2 - 表面	塗りつぶしの塗りつぶしカテゴリ

メッシュパラメータ-リストとラベル用のみ

MESH_TYPE	メッシュタイプ
1-閉じているボディ、2-上面と辺、3-上面のみ	
MESH_BASE_OFFSET	基準レベルに対する下面のオフセット
MESH_USEREDGE_PEN	メッシュユーザー定義尾根ペン
MESH_TRIEDGE_PEN	メッシュ三角辺ペン
MESH_SECT_PEN	断面のメッシュの輪郭ペン
平面図ウィンドウと断面/立面ウィンドウの両方で、壁の断面の輪郭に適用される	
MESH_VIEW_PEN	ビュー上の輪郭ペン
3Dウィンドウ内の全ての辺と断面/立面ウィンドウ内のビューの辺に適用される	
MESH_MAT_TOP	メッシュ上部の材質の属性インデックス
MESH_MAT_EDGE	メッシュの辺の材質の属性インデックス
MESH_MAT_BOTT	メッシュ下部の材質の属性インデックス
MESH_LINETYPE	メッシュの線種
平面図ウィンドウでのみ輪郭に適用される	
MESH_FILL	メッシュの塗りつぶし種類
MESH_BMAT_NAME	メッシュのビルディングマテリアル名
MESH_BMAT	メッシュのビルディングマテリアルのインデックス
互換性：ARCHICAD 21で導入されました。	
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	メッシュの下部の表面積
MESH_TOP_SURF	メッシュの上部の表面積
MESH_EDGE_SURF	メッシュの辺の表面積
MESH_PERIMETER	メッシュ周囲長さ
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

カーテンウォール構成要素パラメータ

CW_BOUNDARY_PLACEMENT　　カーテンウォールフレームの配置

2D	✓	3D	✓	UI	✓	Parameter	✓	Property	✓	デフォルト	0

互換性：ARCHICAD 22で導入されました。
カーテンウォールシステム/部材配置で定義された境界フレームの配置設定を含みます。

values:
0: 境界の中心、つまり実際のフレームは境界フレームではありません。
-1: 内側境界
1: 外側境界

GLOB_MVO_CWFRAME_DETLEVEL　　詳細レベルのカーテンウォールフレームは、[モデル表示オプション]/[カーテンウォールオプション]で設定されます。

2D	✓	3D	✓	UI	⚠	Parameter	⛔	Property	⛔	デフォルト	4

互換性：ARCHICAD 22で導入されました。

values:
1: 軸のみ
2: スキーム
3: 簡略
4: 詳細

GLOB_MVO_CWPANEL_DETLEVEL　　詳細レベルのカーテンウォールパネルは、[モデル表示オプション]/[カーテンウォールオプション]で設定されます。

2D	✓	3D	✓	UI	⚠	Parameter	⛔	Property	⛔	デフォルト	4

互換性：ARCHICAD 22で導入されました。

values:
2: スキーム
3: 簡略
4: 詳細

GLOB_MVO_CWJUNCT_DETLEVEL	詳細レベルのカーテンウォール接続部は、[モデル表示オプション]/[カーテンウォールオプション]で設定されます。

2D	✓	3D	✓	UI	⚠	Parameter	⛔	Property	⛔	デフォルト	4

互換性：ARCHICAD 22で導入されました。

values:

2: スキーム
3: 簡略
4: 詳細

GLOB_MVO_CWACC_DETLEVEL	詳細レベルのカーテンウォール付属品は、[モデル表示オプション]/[カーテンウォールオプション]で設定されます。

2D	✓	3D	✓	UI	⚠	Parameter	⛔	Property	⛔	デフォルト	4

互換性：ARCHICAD 22で導入されました。

values:

2: スキーム
3: 簡略
4: 詳細

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	カーテンウォール内境界フレーム長さ
CWALL_MAINAXIS_FRAMES_LENGTH	カーテンウォール内主グリッドラインフレーム長さ
CWALL_SECAXIS_FRAMES_LENGTH	カーテンウォール内副グリッドラインフレーム長さ
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	カーテンウォール内パネル表面積
CWALL_PANELS_SURF_N	カーテンウォール内北パネル表面積
CWALL_PANELS_SURF_S	カーテンウォール内南パネル表面積
CWALL_PANELS_SURF_E	カーテンウォール内東パネル表面積
CWALL_PANELS_SURF_W	カーテンウォール内西パネル表面積
CWALL_PANELS_SURF_NE	カーテンウォール内北東パネル表面積

CWALL_PANELS_SURF_NW	カーテンウォール内北西パネル表面積
CWALL_PANELS_SURF_SE	カーテンウォール内南東パネル表面積
CWALL_PANELS_SURF_SW	カーテンウォール内南西パネル表面積
CWALL_SURF	カーテンウォール表面積
CWALL_SURF_BOUNDARY	境界フレームによるカーテンウォール表面積境界
CWALL_LENGTH	カーテンウォール長さ
CWALL_HEIGHT	カーテンウォール高さ
CWALL_SLANT_ANGLE	カーテンウォール傾斜角度
CWALL_THICKNESS	カーテンウォール厚さ
CWALL_PANELS_NR	カーテンウォールパネルの数
CWALL_PATTERN_ANGLE	カーテンウォールパターン角度

カーテンウォールフレームパラメータ

一般的なカーテンウォールフレーム変数 - リストとラベルでのみ使用可能

CWFRAME_TYPE	フレームタイプ
「不可視」またはGDLオブジェクトの名前 互換性：ARCHICAD 21以前は、「一般」、「ガラス縁」、「不可視」またはGDLオブジェクトの名前	
CWFRAME_POSITION	フレーム位置 ARCHICAD 22以降は、フレーム位置の定義はフレーム相対座標の値に基づきます。
0 - 垂直、1 - 水平、2 - 輪郭、3 - 対角線 互換性：ARCHICAD 21以前は、フレームの位置の定義はグリッドラインに基づきます。値は下記を参照します。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	
CWFRAME_DIRECTION	フレームの傾斜角度
0から90までの角度	
CWFRAME_WIDTH	フレーム幅
CWFRAME_DEPTH	フレーム奥行き
CWFRAME_LENGTH	フレーム長さ
CWFRAME_MAT	枠の材質属性インデックス

カーテンウォールフレーム3D変数

CWFRAME_TOP_CUTTYPE — カーテンウォールフレームの上部の接続で切断方法を定義します。

2D		3D		UI		Parameter		Property		デ#フォルト	0

互換性：ARCHICAD 22で導入されました。

values:

0: 平面 - フレームをCWFRAME_TOP_CUTPLANEでグローバルに定義された平面で切断する必要があります
1: ポリライン - フレームをCWFRAME_TOP_CUTPOLYLINEでグローバルに定義された平面で切断する必要があります

CWFRAME_TOP_CUTPLANE — 2次元の配列（[4]）。ライブラリ部品のローカル座標系に定義された、フレームの上部切断面の配置を含む

2D		3D		UI		Parameter		Property		デ#フォルト	[0, 0, 0, 0]

互換性：ARCHICAD 22で導入されました。

values:

[1]: 上部切断面の法線ベクトルのXコンポーネント
[2]: 上部切断面の法線ベクトルのYコンポーネント
[3]: 上部切断面の法線ベクトルのZコンポーネント
[4]: フレームの原点からの上部切断面の距離

CWFRAME_TOP_CUTPOLYLINE — 2次元の配列（[n][2]）。nは切断ポリラインの節点の数。ライブラリ部品のローカル座標系のY-Z平面に定義される、フレームの上部接続の切断ポリライン座標を含みます。ポリラインの最初の節点はY座標の最小値で、最後の節点はカーテンウォールフレームのY座標の最大値です。

2D		3D		UI		Parameter		Property		デ#フォルト	[0, 0]

互換性：ARCHICAD 22で導入されました。

values:

[n][1]: カーテンウォールフレームの原点から計測した節点の座標X
[n][2]: カーテンウォールフレームの原点から計測した節点の座標Y

CWFRAME_BOTTOM_CUTTYPE — カーテンウォールフレームの下部の接続で切断方法を定義します。

2D		3D		UI		Parameter		Property		デ#フォルト	0

互換性：ARCHICAD 22で導入されました。

values:

0: 平面 - フレームをCWFRAME_BOTTOM_CUTPLANEでグローバルに定義された平面で切断する必要があります
1: ポリライン - フレームをCWFRAME_BOTTOM_CUTPOLYLINEでグローバルに定義された平面で切断する必要があります

CWFRAME_BOTTOM_CUTPLANE　2次元の配列（[4]）。ライブラリ部品のローカル座標系に定義された、フレームの下部切断面の配置を含む

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0, 0, 0, 0]

互換性：ARCHICAD 22で導入されました。

values:

[1]: 下部切断面の法線ベクトルのXコンポーネント
[2]: 下部切断面の法線ベクトルのYコンポーネント
[3]: 下部切断面の法線ベクトルのZコンポーネント
[4]: フレームの原点からの下部切断面の距離

CWFRAME_BOTTOM_CUTPOLYLINE　2次元の配列（[n][2]）。nは切断ポリラインの節点の数。ライブラリ部品のローカル座標系のY-Z平面に定義される、フレームの下部接続の切断ポリライン座標を含みます。ポリラインの最初の節点はY座標の最小値で、最後の節点はカーテンウォールフレームのY座標の最大値です。

2D	⛔	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[0, 0]

互換性：ARCHICAD 22で導入されました。

values:

[n][1]: カーテンウォールフレームの原点から計測した節点の座標X
[n][2]: カーテンウォールフレームの原点から計測した節点の座標Y

カーテンウォールパネル変数

CWPANEL_HORIZONTAL_DIRECTION　パネルの外壁側表面のプロジェクトの北からの角度

2D	✅	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[180]

互換性：ARCHICAD 22で2Dおよび3Dスクリプトに導入されました。以前のバージョンでは、ラベルとリストでのみ使用できます。
値：-180から180までの角度

CWPANEL_VERTICAL_DIRECTION　パネルの外壁側表面の傾斜角度

2D	✅	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	[90]

互換性：ARCHICAD 22で2Dおよび3Dスクリプトに導入されました。以前のバージョンでは、ラベルとリストでのみ使用できます。
値：-90から90までの角度

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	パネルタイプ
CWPANEL_CLASS 0 - メイン、1 - 個別、2 - カスタム	パネルクラス
CWPANEL_WIDTH	パネル幅
CWPANEL_NOMINAL_WIDTH	パネル有効幅
CWPANEL_HEIGHT	パネル高さ
CWPANEL_NOMINAL_HEIGHT	パネル有効高さ
CWPANEL_THICKNESS	パネルの厚さ
CWPANEL_SURF	パネル表面積
CWPANEL_GROSS_SURF	パネル総表面積
CWPANEL_NOMINAL_SURF	パネル公称表面積
CWPANEL_PERIMETER	パネル円周
CWPANEL_MAT_OUTER	パネルの外壁側表面の材質属性インデックス
CWPANEL_MAT_INNER	パネルの内壁側表面の材質属性インデックス
CWPANEL_MAT_CUT	パネルの辺の材質属性インデックス
CWPANEL_FUNCTION 0 - 固定、1 - ドア、2 - 窓	パネルの機能
CWPANEL_ORIENTATION 左/右	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

移行パラメータ - 移行スクリプト用のみ

FROM_GUID	移行元のライブラリ部品のメインGUID
TO_GUID	移行先のライブラリ部品のメインGUID

天窓パラメータ - リストとラベル用のみ

SKYL_MARKER_TXT	天窓マーカーテキスト
SKYL_OPENING_SURF	天窓開口部面積
SKYL_OPENING_VOLUME	天窓開口部切断の体積
SKYL_OPENING_HEIGHT	天窓開口部高さ
SKYL_OPENING_WIDTH	天窓開口部幅
SKYL_HEADER_HEIGHT	天窓上端高さ
SKYL_SILL_HEIGHT	天窓下端高さ

シェルと屋根の共通パラメータ - リストとラベル用のみ

SHELLBASE_THICKNESS 屋根のROOF_THICKNESSと同じ	シェル/屋根/スラブの厚さ
SHELLBASE_MAT_REFERENCE 屋根のROOF_MAT_BOTTと同じ	シェル/屋根下部の材質の属性インデックス
SHELLBASE_MAT_EDGE 屋根のROOF_MAT_EDGEと同じ	シェル/屋根の辺の材質の属性インデックス
SHELLBASE_MAT_OPPOSITE 屋根のROOF_MAT_TOPと同じ	シェル/屋根上部の材質の属性インデックス
SHELLBASE_LINETYPE 平面図ウィンドウでのみ輪郭に適用される、屋根のROOF_LINETYPEと同じ	シェル/屋根の線種

SHELLBASE_FILL	シェル/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、屋根のROOF_FILLと同じ	
SHELLBASE_FILL_PEN	シェル/屋根の塗りつぶしペン
屋根のROOF_FILL_PENと同じ	
SHELLBASE_FBGD_PEN	シェル/屋根の塗りつぶし背景ペン
屋根のROOF_FBGD_PENと同じ	
SHELLBASE_COMPS_NAME	シェル/屋根の複合構造の名前
屋根のROOF_COMPS_NAMEと同じ	
SHELLBASE_BMAT_NAME	シェル/屋根のビルディングマテリアル名
屋根のROOF_BMAT_NAMEと同じ	
SHELLBASE_BMAT	シェル/屋根のビルディングマテリアルインデックス
互換性：ARCHICAD 21で導入されました。 屋根のROOF_BMATと同じ	
SHELLBASE_SKINS_NUMBER	シェル/屋根の複合構造の層数
1～8の範囲、単一塗りつぶしの場合は0、屋根のROOF_SKINS_NRと同じ	

SHELLBASE_SKINS_PARAMS シェル/屋根の複合構造の層のパラメータ

任意の数の行を含む18列の配列：

- [1] 塗りつぶし
- [2] 厚さ
- [3] （古い輪郭ペン）
- [4] 塗りつぶしのペン
- [5] 塗りつぶし背景のペン
- [6] 躯体の状態
- [7] 上部線ペン
- [8] 上部線種
- [9] 下部線ペン
- [10] 下部線種
- [11] 終端面ペン
- [12] 塗りつぶし向き
- [13] 面の種類
- [14] 終端面線種
- [15] 仕上げ塗りつぶし状態
- [16] 与えられた方向の塗りつぶし状態
- [17] 躯体の状態（躯体が存在しない場合は、最も厚い塗りつぶし）
- [18] ビルディングマテリアルのインデックス

躯体の状態：0 - 部分ではない、1 - 部分、3 - 躯体の最終層、塗りつぶし向き：0 - グローバル、1 - ローカル、 面の種類：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

屋根のROOF_SKINS_PARAMSと同じ

SHELLBASE_SKINS_BMAT_NAMES シェル/屋根の複合屋根のビルディングマテリアル名

列が1の配列：任意の数の行を有する層のビルディングマテリアル名

屋根のROOF_SKINS_BMAT_NAMESと同じ

SHELLBASE_SECT_PEN シェル/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断面の輪郭に適用される、屋根のROOF_SECT_PENと同じ

SHELLBASE_VIEW_PEN ビュー上のシェル/屋根のペン

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、 屋根のROOF_VIEW_PENと同じ

SHELLBASE_REFERENCE_SURF シェル/屋根の基準側面積

穴面積を減算しない、屋根のROOF_BOTTOM_SURFと同じ

SHELLBASE_COND_REFERENCE_SURF	シェル/屋根の条件付き基準側面積
屋根のROOF_BOTTOM_SURF_CONと同じ	
SHELLBASE_GROSS_REFERENCE_SURF	シェル/屋根基準側の総面積
穴面積を減算、屋根のROOF_GROSS_BOTTOM_SURFと同じ	
SHELLBASE_OPPOSITE_SURF	シェル/屋根基準側の反対側の面積
穴面積を減算しない、屋根のROOF_TOP_SURFと同じ	
SHELLBASE_COND_OPPOSITE_SURF	シェル/屋根基準側の反対側の条件付き面積
指定された値より大きい穴面積を減算、 屋根のROOF_TOP_SURF_CONと同じ	
SHELLBASE_GROSS_OPPOSITE_SURF	シェル/屋根基準側の反対側の総面積
穴面積を減算、屋根のROOF_GROSS_TOP_SURFと同じ	
SHELLBASE_EDGE_SURF	シェル/屋根辺の面積
穴面積を減算しない、屋根のROOF_EDGE_SURFと同じ	
SHELLBASE_GROSS_EDGE_SURF	シェル/屋根辺の総面積
穴面積を減算しない、屋根のROOF_GROSS_EDGE_SURFと同じ	
SHELLBASE_PERIMETER	シェル/屋根の外周
屋根のROOF_PERIMETERと同じ	
SHELLBASE_VOLUME	シェル/屋根の体積
穴体積を減算しない、屋根のROOF_VOLUMEと同じ	
SHELLBASE_COND_VOLUME	シェル/屋根の条件付き体積
指定された値より大きい穴体積を減算 屋根のROOF_VOLUME_CONと同じ	
SHELLBASE_GROSS_VOLUME	シェル/屋根の総体積
穴体積を減算しない、屋根のROOF_GROSS_VOLUMEと同じ	
SHELLBASE_HOLES_NR	シェル/屋根の穴数
屋根のROOF_HOLES_NRと同じ	
SHELLBASE_HOLES_SURF	シェル/屋根の穴面積
屋根のROOF_HOLES_AREAと同じ	

SHELLBASE_HOLES_PRM 屋根のROOF_HOLES_PRMと同じ	シェルの穴外周
SHELLBASE_OPENINGS_NR	シェルの開口部の数
SHELLBASE_OPENINGS_SURF	シェルの開口部面積
SHELLBASE_INSU_THICKNESS 屋根のROOF_INSU_THICKNESSと同じ	シェル/屋根の壁断熱材厚
SHELLBASE_RIDGE 屋根のROOF_RIDGEと同じ	シェル/屋根棟長さ
SHELLBASE_VALLEY 屋根のROOF_VALLEYと同じ	シェル/屋根谷長さ
SHELLBASE_GABLE 屋根のROOF_GABLEと同じ	シェル/屋根破風長さ
SHELLBASE_HIP 屋根のROOF_HIPと同じ	シェル/屋根隅棟長さ
SHELLBASE_EAVES 屋根のROOF_EAVESと同じ	シェル/屋根軒長さ
SHELLBASE_PEAK 屋根のROOF_PEAKと同じ	シェル/屋根頂点長さ
SHELLBASE_SIDE_WALL 屋根のROOF_SIDE_WALLと同じ	シェル/屋根壁側面接続長さ
SHELLBASE_END_WALL 屋根のROOF_END_WALLと同じ	シェル/屋根端部壁接続長さ
SHELLBASE_TRANSITION_DOME 屋根のROOF_TRANSITION_DOMEと同じ	シェル/ドーム屋根接続長さ
SHELLBASE_TRANSITION_HOLLOW 屋根のROOF_TRANSITION_HOLLOWと同じ	シェル/ボールド屋根接続長さ

モルフのパラメータ - リストとラベル用のみ

MORPH_LINETYPE	ビュー上のモルフの線種
MORPH_FILL	モルフ切断面の塗りつぶし
MORPH_BMAT_NAME	モルフ切断面のビルディングマテリアル名
MORPH_BMAT 互換性：ARCHICAD 21で導入されました。	モルフ切断面のビルディングマテリアルインデックス
MORPH_FILL_PEN	モルフ切断面のペン
MORPH_FBGD_PEN	モルフ切断面の塗りつぶし背景ペン
MORPH_SECT_LINETYPE	モルフ切断面の輪郭線種
MORPH_SECT_PEN	モルフ切断面の輪郭ペン
MORPH_VIEW_PEN	ビュー上のモルフの輪郭ペン
MORPH_SOLID	モルフボディソリッド（オンまたはオフ）
MORPH_MAT_DEFAULT	モルフデフォルト材質の属性インデックス
MORPH_CASTS_SHADOW	影を投影（オン/オフ）
MORPH_RECEIVES_SHADOW	影を受影（オン/オフ）
MORPH_SURFACE	モルフ総面積
MORPH_VOLUME	モルフ体積
MORPH_FLOOR_PERIMETER	平面図上のモルフ外周

自由なユーザーグローバル変数

GLOB_USER_1
GLOB_USER_2
GLOB_USER_3
GLOB_USER_4
GLOB_USER_5
GLOB_USER_6
GLOB_USER_7

GLOB_USER_8	
GLOB_USER_9	
GLOB_USER_10	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

グローバル変数の使用例

例: グローバル変数GLOB_WORLD_ORIGO_...の使用例：

```
ADD2 -GLOB_WORLD_ORIGO_OFFSET_X-SYMB_POS_X, -GLOB_WORLD_ORIGO_OFFSET_X-SYMB_POS_Y
LINE2 -0.1, 0.0, 0.1, 0.0
LINE2 0.0, -0.1, 0.0, 0.1
HOTSPOT2 0.0, 0.0, 1
TEXT2 0, 0, "( 0.00 ; 0.00 )"
TEXT2 0, 0.5, "World Origin"
DEL TOP
if ABS(GLOB_WORLD_ORIGO_OFFSET_X) > 0.01 OR¥
        ABS(GLOB_WORLD_ORIGO_OFFSET_Y) > 0.01 THEN
   ADD2 - SYMB_POS_X, - SYMB_POS_Y
   LINE2 -0.1, 0.0, 0.1, 0.0
   LINE2 0.0, -0.1, 0.0, 0.1
   HOTSPOT2 0.0, 0.0, 2
   TEXT2 0, 0, "(" +
         STR (GLOB_WORLD_ORIGO_OFFSET_X, 9, 4) + "; " +
         STR (GLOB_WORLD_ORIGO_OFFSET_Y, 9, 4) + " )"
   TEXT2 0, 0.5, "Virtual Origin"
   DEL TOP
ENDIF
if ABS(GLOB_WORLD_ORIGO_OFFSET_X + SYMB_POS_X) > 0.01 OR¥
        ABS(GLOB_WORLD_ORIGO_OFFSET_Y + SYMB_POS_Y) > 0.01 THEN
   LINE2 -0.1, 0.0, 0.1, 0.0
   LINE2 0.0, -0.1, 0.0, 0.1
   HOTSPOT2 0.0, 0.0, 3
   TEXT2 0, 0, "(" +
         STR (GLOB_WORLD_ORIGO_OFFSET_X + SYMB_POS_X, 9, 4) + "; " +
         STR (GLOB_WORLD_ORIGO_OFFSET_Y + SYMB_POS_Y, 9, 4) + " )"
   TEXT2 0, 0.5, "Object Placement"
ENDIF
```

廃止されたグローバル変数

これらのグローバルは引き続きARCHICADの環境で動作しますが、今後オブジェクトの作成には使用しないことをお勧めします。

GLOB_CONTEXT											表現のコンテキスト（ビュー依存、パラメータ/プロパティスクリプトで使用しない）。
2D	✅	3D	✅	UI	⛔	Parameter	⛔	Property	⛔	デ#フォルト	2

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て生成、46-リストからオペレータとして生成 詳細は、「GDLの実行コンテキスト」を参照してください。

梁/柱のグローバル変数 - リストとラベル用のみは廃止されました

ARCHICAD 23以降では、これらの値はBEAM_SEGMENT_INFOおよびCOLU_SEGMENT_INFOグローバル変数で統一された値参照により使用できます。「「廃止された梁/柱パラメータ - リストとラベル用のみ」」も参照してください。互換性のために、これらは均一な直線梁または垂直湾曲梁および均一な柱（GLOB_ELEM_TYPE = 12または6）では引き続き使用できます。

COLU_CROSSSECTION_TYPE		柱の断面タイプ
⛔柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

0 - 断面形状、1 - 矩形、4 - 円形

BEAM_CROSSSECTION_TYPE		梁の断面タイプ
⛔ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔複数セグメントの梁（12）

0 - 断面形状、1 - 矩形

廃止されたラベルグローバル変数

これらのグローバルは、ARCHICAD 22以降、固定名オプションパラメータで置き換えられています（断面を参照）。互換性上の理由により、グローバル変数は保持されるため、旧式のオブジェクトの動作は影響しません。

グローバルのグループを処理するラベルテキスト：

LABEL_ALWAYS_READABLE	ラベルテキストが常に読める
1-常に読める場合、0-その他の場合	

LABEL_TEXT_WRAP	ラベルテキストを折り返す
1-テキストを折り返す場合、0-その他の場合	

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
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_TEXT_BG_PEN	テキストボックスの背景ペン
不透過がオフの場合は0、その他の場合は背景ペンインデックス	
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-斜体、4-下線	
LABEL_FONT_STYLE2	[設定]ダイアログボックスのフォントスタイル
0 - 標準、その他の場合 - j1 + 2*j2 + 4*j3 + 32*j6 + 64*j7 + 128*j8、j1 - 太字、j2 - 斜体、j3 - 下線、j6 - 上付き文字、j7 - 下付き文字、j8 - 取り消し線	
グローバルのグループを処理するラベルポインタ/フレーム：	
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_LINETYPE	矢印の線の線種
LABEL_ARROW_PEN	[設定]ダイアログボックスでの矢印線のペン
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_FRAME_OFFSET	枠のオフセット
LABEL_ANCHOR_POS	ラベル配置位置
0-中央、1-上端、2 -下端、3 - 右下	

廃止されたカーテンウォールフレームグローバル変数 - リストとラベルでのみ使用可能

ARCHICAD 22以降、事前定義されたクラスの代わりに、ユーザー定義のカーテンウォールフレームクラスが導入されました。互換性上の理由から、CWFRAME_CLASSグローバル変数は維持されています。

CWFRAME_CLASS	フレームクラス
2 - 境界、3 - その他 互換性：ARCHICAD 21以前までは、値を0 - マリオン、1 - トランザム、2 -境界、3 - カスタムにできます	

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
E_ WALL_SECT_PEN
F_ WALL_FILL_PEN
G_ WALL_MAT_A
H_ WALL_MAT_B
I_ WALL_MAT_EDGE
J_ GLOB_ELEVATION
K_ WIDO_SILL
L_ SYMB_VIEW_PEN
M_ SYMB_MAT
N_ GLOB_FRAME_NR
O_ GLOB_FIRST_FRAME
P_ GLOB_LAST_FRAME
Q_ GLOB_HSTORY_HEIGHT
R_ WIDO_ORIG_DIST
S_ GLOB_USER_1
T_ GLOB_USER_2
U_ GLOB_USER_3
V_ GLOB_USER_4
W_ GLOB_USER_5
X_ GLOB_USER_6
Y_ GLOB_USER_7
Z_ GLOB_USER_8

A~ WALL_FILL
B~ WIDO_RIGHT_JAMB
C~ WIDO_THRES_DEPTH
D~ WIDO_HEAD_DEPTH
E~ WIDO_REVEAL_SIDE
F~ WIDO_FRAME_THICKNESS
G~ GLOB_USER_9
H~ WIDO_POSITION
I~ GLOB_USER_10
J~ WALL_RESOL
K~ GLOB_EYEPOS_X
L~ GLOB_EYEPOS_Y
M~ GLOB_EYEPOS_Z
N~ GLOB_TARGPOS_X
O~ GLOB_TARGPOS_Y
P~ GLOB_TARGPOS_Z
Q~ GLOB_CSTORY_ELEV
R~ GLOB_CSTORY_HEIGHT
S~ GLOB_CH_STORY_DIST
T~ GLOB_SCRIPT_TYPE
U~ GLOB_NORTH_DIR
V~ SYMB_MIRRORED
W~ SYMB_ROTANGLE
X~ SYMB_POS_X
Y~ SYMB_POS_Y
Z~ SYMB_POS_Z

固定名パラメータ

ARCHICADに設定されたパラメータ

情報を提供するためのARCHICADの新しい方法は、固定名のオプションパラメータの方法があります。 指定されたライブラリに、オプションのパラメータの名前とタイプが一致する固定名持つパラメータがある場合、ARCHICADはその機能に応じてその値を設定します。

ドア/窓属性のパラメータ（ドア、窓、ラベル、一覧表で使用可能）

フロアプラン表示

ac_hole_cut_linetype	線種
切断線ペン [平面図と切断図]	

ac_hole_overhead_pen	ペン
ビューのエッジの上のペン（上部線） [平面図のみ]	
ac_hole_overhead_linetype	線種
ビューのエッジの上の線（上部線） [平面図のみ]	
ac_hole_uncut_pen	ペン
ビューのエッジの下のペン（上部線） [平面図のみ]	
ac_hole_uncut_linetype	線種
ビューのエッジの下の線種（上部線） [平面図のみ]	
ac_hole_display_option	整数
フロアプラン表示 オプション：1 - 投影線、2 - 投影線と上部線、3 - シンボル表示、5 - 上部線表示	

方向

ac_hole_direction_type	整数
開口面方向：1 - 壁に関連、2 - 垂直	
ac_wido_rotation	角度
ドア/窓 水平切断面の回転角度	
ac_openingside	文字列
開口設定（自動、カスタム）による、リストのためのドア/窓の向きパラメータ（L - 左、R - 右、カスタム）。 このパラメータが存在している場合、ARCHICADは互換オプション/向きの表示の設定を無視します。	

ポリゴン壁データ

ac_walltype	整数
窓がポリゴン壁に配置されているか確認します。 1 - ポリゴン壁にはない、2 - ポリゴン壁にある。	
ac_wallContourPolygon[][3]	長さ
2Dの点における壁のポリゴンと円弧セクションのための余分な角度値。 [**ac_walltype** が2に等しい場合に設定]	
ac_windowInWallContour[4]	整数
窓のコーナーポイントとして壁輪郭線一部の **ac_wallContourPolygon** ポリゴンの4つの頂点のインデックス。 [**ac_walltype** が2に等しい場合に設定]	

穴の位置

ac_hole_position_angle	角度
曲線壁の場合、その開口部の軸と壁の始点における法線ベクトル間の角度を与える。	

固定位置データ

ac_vertAnchorPos	整数
ドア/窓の直固定位置：1 - 下端、2 - 上端	
ac_revealAnchorPos	整数
ドア/窓の抱き位置：1 - 面、2 - 躯体	
ac_revealToWallCore	長さ
壁の躯体の外壁側から計測された抱きの深さ。	

壁属性のパラメータ（ドア、窓、ラベル、一覧表で使用可能）

フロアプラン表示

ac_wall_overhead_pen	ペン
壁のビューのエッジの上のペン（上部線）[平面図のみ]	
ac_wall_overhead_linetype	線種
壁のビューのエッジの上の線種（上部線）[平面図のみ]	
ac_wall_uncut_linetype	線種
壁のビューのエッジの下の線種（上部線）[平面図のみ]	
ac_wall_display_option	整数
壁のフロアプラン表示 オプション： 1 - 投影線、2 - 投影線と上部線、3 - シンボル、4 - 輪郭線表示、5 - 上部線表示	
ac_wall_show_projection_to	整数
壁の垂直ビューの深さ制限： 1 - 相対範囲、2 - 絶対限度、3 - 要素全体	

幾何学的データ

ac_wall_elevation	長さ
壁の配置フロアに対して、壁の下端の高度	

ac_wall_crosssection_type	**整数**
壁の断面タイプ： 1 - 単純、2 - 断面形状、3 - 傾斜、4 - 台形	
ac_wall_profile_name	**文字列**
壁が断面形状属性の複雑な壁の場合はプロファイル名、 "Custom_Profile_i" カスタム断面形状の場合 （**i** 配置された壁のID）または 単純壁、傾斜壁、台形壁の場合は"n/a"	
ac_wall_slant_angle1	**角度**
水平を基準に壁の第1の傾斜角度（壁が垂直の場合は90度）	
ac_wall_slant_angle2	**角度**
水平を基準に壁の第2の傾斜角度（壁が垂直の場合は90度）	
ac_wall_direction_type	**整数**
壁方向タイプ; 壁の組み立て法、 壁体と基準線の調整を意味する： 0 - 右、1 - 左、2 - 中央（右）、3 - 中央（左）。 中心値は壁がユーザーインターフェイスの「中央」に設定されていることを意味しますが、サイド表記は壁が内部で動作する方法を示します。	

柱属性のパラメータ（ラベル、リストで使用可能）

各パラメータの使用可能性（意味のあるデータを含むかどうか）が表にアイコンで表示され、GLOB_ELEM_TYPEグローバル変数の値が括弧内に表示されます。

フロアプラン表示

ac_colu_overhead_pen	**ペン**
柱のビューのエッジの上のペン（上部線） [平面図のみ]	

✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）

ac_colu_overhead_linetype	**線種**
柱のビューのエッジの上の線種（上部線） [平面図のみ]	

✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）

ac_colu_uncut_linetype	**線種**
柱のビューのエッジの下の線種（上部線） [平面図のみ]	

✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）

ac_colu_display_option 整数

柱のフロアプラン表示 オプション： 1 - 投影線、2 - 投影線と上部線、3 - シンボル、4 - 輪郭線表示、5 - 上部線表示

✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）

ac_colu_show_projection_to 整数

柱の垂直ビューの深さ制限： 1 - 相対範囲、2 - 絶対限度、3 - 要素全体

✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）

幾何学的データ

ac_colu_profile_name 文字列

柱が断面形状属性の場合はプロファイル名、 "Custom_Profile_i" カスタム断面形状の場合 （**i** 配置された柱のID） または 柱が矩形、円形の場合は"n/a"

✅柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

ac_colu_inclination 角度

水平線に対して柱の傾斜角度

✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）

ac_colu_twist_angle 角度

切断面の回転角度

✅柱セグメント（27）	✅単一セグメントの柱（6）	✅複数セグメントの柱（6）

梁属性のパラメータ（ラベル、リストで使用可能）

各パラメータの使用可能性（意味のあるデータを含むかどうか）が表にアイコンで表示され、GLOB_ELEM_TYPEグローバル変数の値が括弧内に表示されます。

平面図表示

ac_beam_overhead_pen ペン

梁のビューのエッジの上のペン（上部線） [平面図のみ]

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅複数セグメントの梁（12）

ac_beam_cut_linetype 線種

梁の切断辺の線種（断面形状では無効）

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅複数セグメントの梁（12）

ac_beam_uncut_pen ペン

梁のビューのエッジの下のペン（上部線） [平面図のみ]

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅複数セグメントの梁（12）

ac_beam_uncut_linetype 線種

梁のビューのエッジの下の線種（上部線） [平面図のみ]

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅複数セグメントの梁（12）

ac_beam_display_option 整数

梁のフロアプラン表示 オプション： 1 - 投影線、2 - 投影線と上部線、3 - シンボル、4 - 輪郭線表示、5 - 上部線表示

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅複数セグメントの梁（12）

ac_beam_show_projection_to 整数

梁の垂直ビューの深さ制限： 1 - 相対範囲、2 - 絶対限度、3 - 要素全体

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅複数セグメントの梁（12）

幾何学的データ

ac_beam_profile_name 文字列

断面形状属性の複雑な場合はプロファイル名、 "Custom_Profile_i" カスタム断面形状の場合 （**i** 配置された梁のID） または 梁が矩形の場合は"n/a"

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔ 複数セグメントの梁（12）

ac_beam_inclination 角度

水平線に対して梁の傾斜角度

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅ 複数セグメントの梁（12）

ac_beam_twist_angle 角度

切断面の回転角度（単純梁の場合0.0）

✅ 梁セグメント（28）	✅ 単一セグメントの梁（12）	✅ 複数セグメントの梁（12）

屋根属性のパラメータ（ラベル、リストで使用可能）

フロアプラン表示

ac_roof_overhead_pen ペン

屋根のビューのエッジの上のペン（上部線） [平面図のみ]

ac_roof_overhead_linetype 線種

屋根のビューのエッジの上の線種（上部線） [平面図のみ]

ac_roof_display_option 整数

屋根のフロアプラン表示 オプション： 1 - 投影線、2 - 投影線と上部線、3 - シンボル、4 - 輪郭線表示、5 - 上部線表示

ac_roof_show_projection_to 整数

屋根の垂直ビューの深さ制限： 1 - 相対範囲、2 - 絶対限度、3 - 要素全体

ドア/窓 マーカー属性

ac_wido_id 文字列

開口のID

ac_wido_a_size 開口幅	長さ
ac_wido_b_size 開口高さ	長さ
ac_wido_z_size 開口 深さ/厚さ	長さ
ac_glob_elevation 開口の下端の高度	長さ
ac_wido_subfl_thickness サブフロアの壁部分の高さ	長さ
ac_wido_reveal_side 旧式の開口外壁側の値には ac_wido_reveal_side_2 を使用してください	ブール
ac_wido_reveal_side_2 外壁側、 開口部に **WIDO_REVEAL_SIDE** グローバル変数の値を設定	ブール
ac_wido_mirrored 開口部の反転ステータス	ブール
ac_wall_thickness 開口部の原点における壁の厚さ	長さ
ac_wido_oversize_l 左開口部のオーバーサイズ	長さ
ac_wido_oversize_r 右開口部のオーバーサイズ	長さ
ac_wido_oversize_t 上部開口部のオーバーサイズ	長さ
ac_wido_oversize_b 下部開口部のオーバーサイズ	長さ

ac_wido_orientation	文字列
マーカーの位置："L" - 左、 "R" - 右、 現在の反転ステータスに応じてライブラリ部品エディタの[詳細]ウィンドウで設定されたカスタム値	
ac_wido_type	整数
1 - ドア、2 - 窓	
ac_symb_rotangle	角度
壁の開口部の回転	
ac_sill_to_curr_story	長さ
窓下端にリンクされたフロアの始まりから計測した開口部の下端の高さ;	
ac_sill_to_anchor_level	長さ
配置基準点の高度から計測した開口部の下端の高さ; 配置基準点は、壁の下端または選択されたフロアです。	

詳細図/ワークシート マーカー属性

ac_showboundary	ブール
マーカー境界 ポリゴンのステータス。0 - 境界非表示、1 - 境界表示。	

図面タイトル属性

ac_drawingName	文字列
図面の名前。	
ac_drawingNumber	文字列
図面のID。	
ac_sourceFileName	文字列
図面のソースファイル名（図面が外部ファイルからのものである場合）。	
ac_sourceFilePath	文字列
図面のソースファイルパス（図面が外部ファイルからのものである場合）。	
ac_drawingScale	文字列
図面の図面スケールセット。	
ac_magnification	実数
図面の拡大率パーセント。	

ac_originalDrawingScale	文字列
元のビューの図面スケール。	
ac_enableBackReference	ブール
図面の逆参照が有効。	
ac_backReferenceGUIDList	文字配列
参照されたレイアウトGUIDのリスト。それらは自動出力で使用できます。	
ac_showDrawingReferences	ブール
逆参照を表示。	

一般実行中コンテキスト

ac_programVersion	整数
このパラメータは、ライブラリの部品のスクリプトの実行のARCHICADのバージョンが含まれています。	

部屋パラメータ（ゾーンスタンプで使用可能）

名前	タイプ	デフォルト	説明
ROOM_NAME	文字列	""	ゾーン名
ROOM_NUMBER	文字列	""	ゾーン番号
ROOM_AREA	実数	0.0	総体/正味ポリゴンの面積
ROOM_PERIM	長さ	0.0	総体/正味ポリゴンの外周
ROOM_HOLES_PRM	長さ	0.0	総体/正味ポリゴンの穴
ROOM_WALLS_PRM	長さ	0.0	正味ポリゴンの外周（穴あり）壁に囲まれている場合のみ
ROOM_CORNERS	整数	0	正味ポリゴンの角（穴あり）
ROOM_CONCAVES	整数	0	正味ポリゴンのへこみ角（穴あり）
ROOM_WALLS_SURF	実数	0.0	壁の材質（境界部分）
ROOM_DOORS_WID	実数	0.0	ボーダーでのドアの長さ
ROOM_DOORS_SURF	実数	0.0	ボーダーでのドアの表面
ROOM_WINDS_WID	長さ	0.0	ボーダーでの窓の長さ
ROOM_WINDS_SURF	実数	0.0	ボーダーでの窓の表面
ROOM_BASELEV	長さ	0.0	ゾーンレベル
ROOM_FL_THICK	長さ	0.0	ゾーンサブフロア厚さ
ROOM_HEIGHT	長さ	0.0	ゾーン高さ
ROOM_NET_AREA	実数	0.0	正味ポリゴンの面積（穴あり）
ROOM_NET_PERIMETER	長さ	0.0	総体/正味ポリゴン（穴あり）
ROOM_WALL_EXTR_AREA	実数	0.0	壁の内側まで面積を縮小（ゾーン境界タイプではなく）

名前	タイプ	デフォルト	説明
ROOM_COLUMN_EXTR_AREA	実数	0.0	柱の内側まで面積を縮小（ゾーン境界タイプではなく）
ROOM_FILL_EXTR_AREA	実数	0.0	ゾーンの内側のハッチまで面積を縮小（パーセンテージを考慮）
ROOM_LOW_EXTR_AREA	実数	0.0	低い部分により面積の縮小（切り取り）（パーセンテージを考慮）
ROOM_TOTAL_EXTR_AREA	実数	0.0	前回の値の合計（総抽出）
ROOM_REDUCED_AREA	実数	0.0	ROOM_NET_AREA - ROOM_TOTAL_EXTR_AREA
ROOM_AREA_FACTOR	実数	0.0	1 - Reduced_by_in_dialog / 100
ROOM_CALC_AREA	実数	0.0	ROOM_REDUCED_AREA * ROOM_AREA_FACTOR
ROOM_VOLUME	実数	0.0	ネットポリゴン時にトリミングされた部屋から計算
ROOM_BOUNDARY_SURF	実数	0.0	境界再度のページの表面
ROOM_TOP_SURFACE	実数	0.0	ゾーン上部の表面
ROOM_BOT_SURFACE	実数	0.0	ゾーン下部の表面
ROOM_ROOF_TOP_SURF	実数	0.0	屋根で切り取られたゾーン上部の表面
ROOM_ROOF_BOT_SURF	実数	0.0	屋根で切り取られたゾーン下部の表面
ROOM_SLAB_TOP_SURF	実数	0.0	スラブで切り取られたゾーン上部の表面
ROOM_SLAB_BOT_SURF	実数	0.0	スラブで切り取られたゾーン下部の表面
ROOM_BEAM_TOP_SURF	実数	0.0	梁で切り取られたゾーン上部の表面
ROOM_BEAM_BOT_SURF	実数	0.0	梁で切り取られたゾーン下部の表面
ROOM_WALL_IN_TOP_SURF	実数	0.0	インセット上面の合計（またはリセスかニッチ）
ROOM_WALL_IN_BACK_SURF	実数	0.0	インセットの後部の表面の合計（窓に面して）
ROOM_WALL_IN_SIDE_SURF	実数	0.0	インセット側面表面の合計
ROOM_POLY_STATUS	整数	0	0：手動、1：自動、2：自動定義

階段関連のパラメータ

階段/踊り場サイドサポートサブタイプ

ac_beamPlacement　整数

このパラメータは、階段のどちら側に階段梁が配置されているかを反映します。互換性：ARCHICAD 21で導入されました。

- 0 - 左側（階段を上るように見る場合）
- 1 - 右側

蹴上構成要素サブタイプ

ac_RiserPosition　整数

このパラメータは、蹴上と踏面の接続の現在の設定を表します。

- 0 - [蹴上を踏面上に配置]が無効
- 1 - [蹴上を踏面上に配置]が有効

階段2D構成要素サブタイプ

ac_treadClassifications　整数 - 配列

各踏面のタイプ（ボトムアップ式のインデックス付き）：

- 0 - 踏面
- 1 - 踊り場

ARCHICADに設定された/読み取られたパラメータ

ARCHICADでは、事前に定義された名前と機能を持つパラメータによって値をライブラリ部品と同期することができます。 そのようなパラメータの一覧が以下になります。

階段関連のパラメータ

構造サブタイプ

ac_stairStructureWidth　長さ

[階段の設定]で設定される構造階段ばりの幅。梁構造タイプの場合、値は0です。

ac_stairStructureThickness　長さ

[階段の設定]で設定される梁または下持ち構造の厚さ/高さ。

ARCHICADに読み込まれたパラメータ

ARCHICADは事前に定義された名前と機能を持つパラメータによって部品ライブラリから値を取得することができます。 そのようなパラメータの一覧が以下になります。

平面図上のオブジェクト

平面図の平面の切断要素（例： 天窓オブジェクト、屋根付属オブジェクト）

ac_special_2d_symbol	ブール
このパラメータはARCHICADの平面図での2D切断の仕組みを可能にします。 パラメータが1に設定してある場合、ARCHICADは2Dモデルをパラメータ値に応じて切断します（ライブラリ部品の2Dスクリプトから生成）：ac_symb_display_option, ac_symb_show_projection_to and ac_plane_definition. この2Dベース切断は同じ設定の単純屋根と同様の表示をします。 当然のことながら、この方法では平面状の要素のみで正しい出力が得られます- 天窓と屋根付属品など。 平面状のオブジェクト - ac_plane_definition パラメータによって定義されています。 天窓と屋根の付属品要素 - もし ac_special_2d_symbol が1の場合、上記のパラメータは自動的にアドオンによって設定されます。 他の要素の場合には、それらはライブラリ開発者によって記入されるべきです。	
ac_plane_definition	長さ
平面の定義：([1],[2],[3]): 平面上のポイント, ([4],[5],[6]): 平面の標準ベクトル。	
ac_symb_display_option	整数
1 - 投影線、2 - 投影線と上部線、3 - シンボル、4 - 輪郭線表示、5 - 上部線表示	
ac_symb_show_projection_to	整数
1 - 相対範囲、2 - 絶対限度、3 - 要素全体	
ac_bottomlevel	長さ
このパラメータは、オブジェクトの最下点を示しています。表示フロアは、関連フロア全てに設定されている場合、 この最下点（オブジェクトの配置フロア設定から計算）はフロアに含み、垂直の拡張、オブジェクトがフロアに表示されている。上部レベルは下部レベルの上にくる必要があります。 **ac_bottomlevel** はオブジェクトのオブジェクト原点から始まります。	
ac_toplevel	長さ
表示フロアは、関連フロア全てに設定されている場合、このパラメータは、オブジェクトの上部に指示します。 フロアの高さが下部レベルと上部レベルの間の場合、オブジェクトはフロア上に表示されます。 上部レベルは下部レベルの上にくる必要があります。 **ac_toplevel** はオブジェクトのオブジェクト原点から始まります。	

ドア/窓 オブジェクト

ac_wido_sill　　長さ

このパラメータは、開口オブジェクトの下端奥行きのフルアクセスを提供します。 このパラメータは値リストを取得でき、ロック、非表示することができます。その値はパラメータスクリプトで設定ができます。 その現在の値は、古いスクリプトとの互換性のため WIDO_SILL global 変数に割り当てられます。

ac_wido_hide_options　　整数

このビットフィールドのパラメータを使用して、窓/ドアの設定ダイアログからオプションを無効にすることができます。
ac_wido_hide_options = j1 + 2*j2. j1 が設定されてる場合、デフォルトのARCHICAD 設定ダイアログに下端奥行きが非表示になります。
j2 が設定されてる場合、設定ダイアログの抱き設定が無効化されます。

ac_wido_flip_once　　ブール

パラメータが存在し、値が **true**の場合、移行スクリプト処理後に窓とドアを反転する。

ac_wido_flip_disable　　整数

このパラメータはユーザーインターフェースで「反転」ボタンを無効化することができます。デフォルト値は、オブジェクトの配置のみに影響します。

-1: 反転が有効です。

0: 反転が無効です。デフォルトは反転してません。

1: 反転が無効です。デフォルトは反転してます。

ac_wido_mirror_once　　ブール

パラメータが存在し、値が **true**の場合、移行スクリプト処理後に窓とドアをミラーする。

ac_hole_hotspot_control　　整数

開口部に動ホットスポットがあるかコントロールします。 0 - 自動ホットスポットなし、1 - 2Dにのみあり、2 - 3Dにみあり、3 - 全てにあり

ac_holeSideMaterial　　ブール

壁開口の継承表面を制御します。 壁開口がGDLスクリプトから定義されている場合、このパラメータは無効です。

1: 壁開口は壁の辺と同じ材質を持ちます。

0: 壁開口の表面は、内部/外部壁表面を使用して、下端の奥行き線によって分割されます。

ac_holeMaterialCurved　　ブール

曲線壁で壁開口の内部/外部壁表面における分割線の形状を制御します。 壁開口がGDLスクリプトから定義されている場合、このパラメータは無効です。

1: 表面は曲線で接続されます。

0: 表面は直線で接続されます。

カスタムコンポーネントテンプレート

ac_custom_component_type_name 文字列

このパラメータは、"コンポーネントを保存..."メニューに表示されているカスタムコンポーネントテンプレートの名前が含まれています。 オブジェクト名と異なる場合があります。

階段関連のパラメータ

構造サブタイプ

ac_beamProfileID 整数

構造梁で使用するために選択した断面形状のインデックス。ARCHICADでは、要求のオプションによって断面形状の境界線サイズを使用することができます。

階段/踊り場アンダーサポートサブタイプ

ac_stairStructureHorizThick 長さ

下持ち構造（つまり、蹴上の後ろの部分）の水平厚さ 互換性：ARCHICAD 22で導入されました。

ac_stairStructureBoundsRiser ブール

これは、蹴上を階段の境界より狭くする必要があることを示すために使用できます。下記の4つのパラメータと共に使用します。互換性：ARCHICAD 22で導入されました。

ac_stairRiserLeftBoundaryFrom 整数

下記に対する蹴上のストレッチの左辺：互換性：ARCHICAD 22で導入されました。

- -1 - 階段の左境界
- 0 - 階段の中央線
- 1 - 階段の右境界
- これらは、必ずしも相互に平行である必要はありません。

ac_stairRiserLeftBoundary 長さ

ac_stairRiserLeftBoundaryFromで設定されたアンカー線から測定される蹴上の左辺の距離。右側が正の方向。互換性：ARCHICAD 22で導入されました。

ac_stairRiserRightBoundaryFrom 整数

下記に対する蹴上のストレッチの右垂直辺：互換性：ARCHICAD 22で導入されました。

- -1 - 階段の左境界
- 0 - 階段の中央線
- 1 - 階段の右境界
- これらは、必ずしも相互に平行である必要はありません。

ac_stairRiserRightBoundary　　長さ

ac_stairRiserRightBoundaryFromで設定されたアンカー線から測定される蹴上の右辺の距離。右側が正の方向。互換性：ARCHICAD 22で導入されました。

階段アンダーサポート下持ち/階段下持ちサポートサブタイプ

ac_stairWallFixingWidthLeft　　長さ

（階段の左側の） 壁の下持ちの奥行き。下持ちの2Dシンボルを描画する場合に使用：これを超える描画はクロップされます。互換性：ARCHICAD 21で導入されました。

ac_stairWallFixingWidthRight　　長さ

（階段の右側の） 壁の下持ちの奥行き。下持ちの2Dシンボルを描画する場合に使用：これを超える描画はクロップされます。互換性：ARCHICAD 21で導入されました。

手摺り関連のパラメータ

手摺りパネル構成要素サブタイプ

ac_panelThickness　　長さ

手摺りのパネル断面の厚さ。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。

手摺りレール構成要素サブタイプ

ac_railWidth　　長さ

レール軸に対して垂直な平面で計測したレール断面の絶対幅。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。

ac_railHeight　　長さ

レール軸に対して垂直な平面で計測したレール断面の絶対高さ。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。

ac_rail_boundingbox_left　　長さ

レール軸に対して垂直な平面でレールの方向に向かってレール断面軸（RAIL_POLYLINE_GEOMETRY）から計測した、レール断面境界線左側の相対距離。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。構成要素軸の断面境界線オフセットに使用されます。

ac_rail_boundingbox_right　長さ

レール軸に対して垂直な平面でレールの方向に向かってレール断面軸（RAIL_POLYLINE_GEOMETRY）から計測した、レール断面境界線右側の相対距離。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。構成要素軸の断面境界線オフセットに使用されます。

ac_rail_boundingbox_top　長さ

レール軸に対して垂直な平面でレールの方向に向かってレール断面軸（RAIL_POLYLINE_GEOMETRY）から計測した、レール断面境界線上部の相対距離。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。構成要素軸の断面境界線オフセットに使用されます。

ac_rail_boundingbox_bottom　長さ

レール軸に対して垂直な平面でレールの方向に向かってレール断面軸（RAIL_POLYLINE_GEOMETRY）から計測した、レール断面境界線下部の相対距離。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。構成要素軸の断面境界線オフセットに使用されます。

ac_railProfileID　整数

手摺りで使用するために選択した断面形状のインデックス。ARCHICADでは、要求のオプションによって断面形状の境界線サイズを使用することができます。

レール支柱構成要素サブタイプ

ac_postWidth　長さ

支柱軸に対して垂直な平面で計測した支柱断面の絶対幅。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。

ac_postHeight　長さ

支柱軸に対して垂直な平面で計測した支柱断面の絶対高さ。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。

ac_post_boundingbox_left　長さ

支柱軸に対して垂直な平面で支柱断面軸から計測した、支柱断面境界線左側の相対距離（「左」は、内側から見て基準線の開始方向）。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。構成要素軸の断面境界線オフセットに使用されます。

ac_post_boundingbox_right　長さ

支柱軸に対して垂直な平面で支柱断面軸から計測した、支柱断面境界線右側の相対距離（「右」は、内側から見て基準線の終了方向）。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。構成要素軸の断面境界線オフセットに使用されます。

ac_post_boundingbox_inside　長さ

支柱軸に対して垂直な平面で支柱断面軸から水平に計測した、支柱断面境界線の相対距離（「内側」は、右側のハンドレールが配置される側）。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。構成要素軸の断面境界線オフセットに使用されます。

ac_post_boundingbox_outside	長さ
支柱軸に対して垂直な平面で支柱断面軸から水平に計測した、支柱断面境界線の相対距離（「外側」は、右側のハンドレールが配置される反対側）。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。構成要素軸の断面境界線オフセットに使用されます。	
ac_postProfileID	整数
支柱で使用するために選択した断面形状のインデックス。ARCHICADでは、要求のオプションによって断面形状の境界線サイズを使用することができます。	

レール終端構成要素サブタイプ

ac_railendWidth	長さ
レール終端軸に対して垂直な平面で計測したレール終端断面の絶対幅。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。	
ac_railendHeight	長さ
レール終端軸に対して垂直な平面で計測したレール終端断面の絶対高さ。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。	
ac_railend_boundingbox_left	長さ
レール終端軸に対して垂直な平面でレール終端の方向に向かってレール終端断面軸から計測した、レール断面境界線左側の相対距離。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。構成要素軸の断面境界線オフセットに使用されます。	
ac_railend_boundingbox_right	長さ
レール終端軸に対して垂直な平面でレール終端の方向に向かってレール終端断面軸から計測した、レール終端断面境界線右側の相対距離。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。構成要素軸の断面境界線オフセットに使用されます。	
ac_railend_boundingbox_top	長さ
レール終端軸に対して垂直な平面でレール終端の方向に向かってレール終端断面軸から計測した、レール終端断面境界線上部の相対距離。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。構成要素軸の断面境界線オフセットに使用されます。	
ac_railend_boundingbox_bottom	長さ
レール終端軸に対して垂直な平面でレール終端の方向に向かってレール終端断面軸から計測した、レール終端断面境界線下部の相対距離。パラメータスクリプトによって、モデルサイズのパラメータ値は整合性を保持します。構成要素軸の断面境界線オフセットに使用されます。	
ac_railendProfileID	整数
レール終端で使用するために選択した断面形状のインデックス。ARCHICADでは、要求のオプションによって断面形状の境界線サイズを使用することができます。	

カーテンウォールのパラメータ

ライブラリ部品は、事前に定義された名前と機能を持つパラメータの値によってARCHICAとコラボレーションすることができます。
カーテンウォールツールに関連するこのようなパラメータの一覧を次に示します。

ARCHICADに設定および読み込まれたカーテンウォールパラメータ

カーテンウォールフレームパラメータ

ac_frameWidth	長さ
カーテンウォールフレームの幅。 互換性：ARCHICAD 22で導入されました。	
ac_frameDepth	長さ
カーテンウォールフレームの奥行き。 互換性：ARCHICAD 22で導入されました。	
ac_frameBackDepth	長さ
パネル中心線からの中心線フレームの背面のオフセット。 互換性：ARCHICAD 22で導入されました。	
ac_clampWidth	長さ
カーテンウォールパネルの挿入間隔の幅。 互換性：ARCHICAD 22より前は、このパラメータはARCHICADでのみ設定できました。	
ac_clampDepth	長さ
カーテンウォールパネルの挿入間隔の奥行き。 互換性：ARCHICAD 22より前は、このパラメータはARCHICADでのみ設定できました。	

ARCHICADに設定されたカーテンウォールパラメータ

カーテンウォールフレームパラメータ

ac_nConnectingPanels	長さ
フレームに接続しているパネルの数。境界フレームの場合はこの値は1、それ以外の場合は2。	

ac_clampVector[2][2] 長さ、列

接続しているカーテンウォールパネルの方向ベクトル。（ac_clampVector[1][1]、 ac_clampVector[1][2]）ベクトルには、ローカル軸Yから測定されたより小さい回転角を持つパネルの（X、 Y）ベクトルが含まれます。境界フレームの場合、（ac_clampVector[2][1]、ac_clampVector[2][2]）ベクトルには、接続しているパネルの（X、 Y）方向が含まれ、それ以外の場合はベクトル値は0です。

カーテンウォールのパネルのパラメータ

ac_panelCoords[][2] 長さ、列

ローカル座標系の接続しているフレームの挿入間隔で測定されるカーテンウォールパネルポリゴンのX座標とY座標。このようなポリゴンが存在しない場合はこのパラメータの第1寸法は1に設定され、含まれている値が0の場合は0に設定されます。

互換性：1つの節点への無効なポリゴン形状のフォールバックはARCHICAD 22で導入されました。

ac_clampFreeRegion[][2] 長さ、列

ローカル座標系の接続しているフレームの側面で測定されるカーテンウォールパネルポリゴンのX座標とY座標。このようなポリゴンが存在しない場合はこのパラメータの第1寸法は1に設定され、含まれている値が0の場合は0に設定されます。

互換性：1つの節点への無効なポリゴン形状のフォールバックはARCHICAD 22で導入されました。

ac_frameAxisCoords[][2] 長さ、列

互換性：ARCHICAD 22で導入されました。

ローカル座標系の接続しているフレームの軸で測定されるカーテンウォールパネルポリゴンのX座標とY座標。

カーテンウォール接続部パラメータ

ac_frameDirs[][3] 長さ、列

接続しているカーテンウォールフレーム軸の終了部分の座標。

ac_panelOffsets[] 長さ、列

接続しているカーテンウォールパネルのクランプの厚さ。

注記：組み込まれたパネルのパネル全体の厚さ。

ac_panelPresences[] ブール、配列

接続しているカーテンウォールパネルの存在。

カーテンウォール付属品パラメータ

ac_frameWidthLeft 長さ

参照枠輪郭の幅 1（通常は A/2）

ac_frameWidthRight	長さ
参照枠輪郭の幅 2（通常は A/2）	
ac_frameWidthFront	長さ
参照枠輪郭の長さ 1（通常は A/2）	
ac_frameWidthBack	長さ
参照枠輪郭の長さ 2（通常は A/2）	
ac_accessoryFlipped	ブール
付属品反転のステータス。0 - 反転されていない、1 - 反転された	
ac_globalZDir[1][3]	長さ、列
グローバル座標系におけるローカルZ軸のベクトル	
ac_cellAngle1	角度
付属品の枠は最大2つセルに接続できます。これらのセルは cell1 と cell2です。 セル1は、ローカルY方向から測定されるより小さな回転角度を有するセルです（左回り、ローカル軸Xの正の方向を考慮）。 Parameter ac_cellAngle1 はセル1とローカル軸Yの間の角度です。	
ac_cellAngle2	角度
付属品の枠は最大2つセルに接続できます。これらのセルは cell1 と cell2です。 セル2は、ローカルY方向から測定されるより大きな回転角度を有するセルです（左回り、ローカル軸Xの正の方向を考慮）。 Parameter ac_cellAngle2 はセル2とローカル軸Yの間の角度です。	
ac_validCellAngle1	ブール
セル1が有無を定義	
ac_validCellAngle2	ブール
セル2が有無を定義	

カーテンウォールフレームの廃止されたパラメータ

これらのパラメータは引き続きARCHICADの環境で動作しますが、今後オブジェクトの作成には使用しないことをお勧めします。

gs_rightOffset	長さ
境界からの、カーテンウォールのフレームの内側の距離。外側境界の場合はこの値はフレーム幅で、内側境界の場合はこの値は0です。	
gs_originOffsetX	長さ
境界からの、カーテンウォールのフレームの外側の距離。外側境界の場合はこの値は0で、内側境界の場合はこの値はフレーム幅です。	
gs_frontOffset	長さ
パネルの中心線からの、カーテンウォールのフレームの前側の距離。	

gs_originOffsetY 長さ

パネルの中心線からの、カーテンウォールのフレームの後ろ側の距離。

ac_topConnPlane[4] 長さ、列

ライブラリ部品のローカル座標系に定義された、フレームの上部切断面の配置。
ac_topConnPlane[1]: 上部切断面の法線ベクトルのXコンポーネント
ac_topConnPlane[2]: 上部切断面の法線ベクトルのYコンポーネント
ac_topConnPlane[3]: 上部切断面の法線ベクトルのZコンポーネント
ac_topConnPlane[4]: フレームの原点からの上部切断面の距離

ac_bottomConnplane[4] 長さ、列

ライブラリ部品のローカル座標系に定義された、フレームの下部切断面の配置。
ac_bottomConnplane[1]: 下部切断面の法線ベクトルのXコンポーネント
ac_bottomConnplane[2]: 下部切断面の法線ベクトルのYコンポーネント
ac_bottomConnplane[3]: 下部切断面の法線ベクトルのZコンポーネント
ac_bottomConnplane[4]: フレームの原点からの下部切断面の距離

ARCHICADに読み込まれたカーテンウォールパラメータ

カーテンウォールパネルおよびフレームのパラメータ

AC_AutoSchematicModel ブール

[モデル表示オプション]/[カーテンウォールオプション]で設定されたカーテンウォールパネルまたはフレームの配置図をARCHICADで操作するかどうかを制御します。
互換性：ARCHICAD 22で導入されました。

カーテンウォールフレームパラメータ

ac_iCWFrameType 整数

フレームのタイプを指定します
- 0 - 標準フレーム
- 1 - 対角線コーナー
- 2 - 標準ダブルコーナー
- 3 - 標準ブロックコーナー
- 4 - 断面形状フレーム

このパラメータの値は、フレームタイプおよび形状ダイアログに影響を及ぼし、標準コーナータイプの場合はフレームのシンボル表示。
互換性：ARCHICAD 22で導入されました。

ac_bButtGlazedFrame	ブール

フレームのタイプを指定します
- 0 - キャップ付フレーム
- 1 - バックマリオンフレーム

このパラメータの値は、フレームタイプおよび形状ダイアログに影響を及ぼします。
互換性：ARCHICAD 22で導入されました。

ac_capProfileID	断面形状

カーテンウォールフレームキャップで使用するために選択した断面形状のインデックス。 このパラメータとac_beamProfileIDパラメータがカーテンウォールライブラリ部品に存在する場合、フレームとキャップのサイズの計算はARCHICADで処理されます。
注記：このパラメータは、ac_beamProfileIDパラメータ（「階段関連のパラメータ」を参照）が存在する場合はARCHICADにのみ影響を及ぼします。
互換性：ARCHICAD 22で導入されました。

ac_frameOffsetSide	長さ

パネルの中心線で測定される、フレームの原点からのフレームの側面の距離。クランプの奥行きは、パネルの中心線のこの距離から測定されます。
互換性：ARCHICAD 22で導入されました。

ac_CWFrameBuildMat	ビルディングマテリアル

フレームのビルディングマテリアルのインデックス。フレームのシンボル表示は、このビルディングマテリアルの塗りつぶし属性で描画されます。
互換性：ARCHICAD 22で導入されました。

ac_CWFrameCutLinePen	ペン

フレーム切断線ペンのインデックス。フレームのシンボル表示の輪郭線は、設定されたペンインデックスで描画されます。
互換性：ARCHICAD 22で導入されました。

ac_CWFrameCutLineType	線種

フレーム切断線タイプのインデックス。フレームのシンボル表示の輪郭線は、設定された線タイプインデックスで描画されます。
互換性：ARCHICAD 22で導入されました。

カーテンウォールのパネルのパラメータ

ac_panel_type	整数

リストするパネルのタイプを指定します。 0 - 固定、1 - ドア、2 - 窓

ac_openingDir	整数

リストするドアと窓の開口方向を指定します。0 - 固定、1 - 内側、2 - 外側

ac_width	長さ
リストするパネルの幅。	
ac_nominalWidth	長さ
リストするパネルの有効幅。	
ac_height	長さ
リストするパネルの高さ。	
ac_nominalHeight	長さ
リストするパネルの有効高さ。	
ac_thickness	長さ
リストするパネルの厚さ。	
ac_originIsFrameCenter	ブール
パラメータが存在し、値が **true**の場合、 パネルの原点は、開始（左）フレームの中心点です。 そうでない場合は、原点は左のクランプの開始です。	
ac_aSizeIsWithClamp	ブール
パラメータが存在し、値が **true** の場合、 ARCHICADはフレームとクランプサイズを足した距離を **A** として設定します。 そうでない場合は、**A** はフレームの間のサイズです。	

アドオンのパラメータ

アドオンは事前に定義された名前と機能を持つパラメータによって部品ライブラリから値を取得することができます。 ARCHICADのパッケージのアドオンに関連するようなパラメータの一覧を以下になります。

天窓アドオンのパラメータ

穴のエッジ切断の操作

ac_edge_lower_type	整数
下部エッジの切断タイプ：0 - 垂直、1 - 垂直、2 - 水平、3 - カスタム	
ac_edge_lower_angle	角度
もし ac_edge_lower_type が3の場合、下部エッジの切断角度。 値の範囲は [1-179] 度です、90 は直角の場合です。	
ac_edge_upper_type	整数
上部エッジの切断タイプ：0 - 垂直、1 - 垂直、2 - 水平、3 - カスタム	

ac_edge_upper_angle	角度
もし ac_edge_upper_type が3の場合、上部エッジの切断角度。 値の範囲は [1-179] 度です、90 は直角の場合です。	

コーナー窓アドオンのパラメータ

コーナー窓オブジェクトの一般パラメータ

ac_cw_function	ブール
アドオンでコントロールされている窓配置モード。0 - 窓、1 - コーナー窓	
ac_corner_window	ブール
オブジェクトでコントロールされているコーナー窓モード選択。0 - コーナー窓モードの無効化、1 - コーナー窓モードの有効化	
ac_corner_angle	角度
連続壁の角度。	
ac_diff_con_wall_thk	ブール
常にtrue (1)。 これは連結壁を含む壁とは異なる厚さを有しているか否かを示す機能です。	
ac_con_wall_thk	長さ
連続壁の厚さ	
ac_cw_debug	ブール
内部の使用のみ。GDLプログラマにとっては、特に関係はありません。	

コーナー窓オブジェクトの壁の層データパラメータ（ARCHICAD 12から使用可能）

ac_con_wall_skins_number	整数
連続壁の層の数 単一壁の場合はゼロです。	
ac_con_wall_skins_params	長さ
連続複合構造壁のパラメータ 所有壁の WALL_SKINS_PARAMS GDL グローバルパラメータと同じです。	
ac_con_wall_direction_type	整数
連続壁の反転の状態; 壁の反転状態、 壁体と基準線の調整を意味する： 0 - 反転されていない、1 - 反転された （以前の意味：0 - 右、1 - 左、2 - 中央（右）、3 - 中央（左））	

IFCアドオンのパラメータ

ドア/窓オブジェクトの共通一般パラメータ

ifc_LiningDepth 窓/ドア枠の厚さ	長さ
ifc_LiningThickness 窓/ドア枠の幅	長さ
ifc_TransomThickness トランザム幅	長さ
IFC2x_ConstEnum	整数/文字列

このパラメータはドア/窓の基本タイプの構造を定義します。

ifc_ConstEnum (integer) パラメータ値	**IFC2x_ConstEnum (string) パラメータ値**	**IfcDoorStyleConstructionEnum カテゴリ IfcWindowStyleConstructionEnum カテゴリ**
0	未定義	NOTDEFINED
1	アルミニウム	ALUMINIUM
2	ハイグレードスチール	HIGH_GRADE_STEEL
3	スチール	STEEL
4	木材	WOOD
5	アルミニウム木材	ALUMINIUM_WOOD
6	アルミニウムプラスチック	ALUMINIUM_PLASTIC
7	プラスチック	PLASTIC
8	ユーザー定義	USERDEFINED

ドアオブジェクトの基本パラメータ

ifc_optype - Doors 整数/文字列

ドアの開口部のタイプ、IFC_optype_door.gsm macro でコントロールされます。

ifc_optype (integer) パラメータ値	ifc_optypestr (string) パラメータ値	IfcDoorStyleOperationEnum カテゴリ
0	未定義	NOTDEFINED
1	ドア 片開き	SINGLE_SWING_LEFT SINGLE_SWING_RIGHT
2	両開きドア 片開き	DOUBLE_DOOR_SINGLE_SWING
3	ドア 両開き	DOUBLE_SWING_LEFT DOUBLE_SWING_RIGHT
4	両開きドア 両開き	DOUBLE_DOOR_DOUBLE_SWING
5	両開きドア 片開き 反対	DOUBLE_DOOR_SINGLE_SWING_OPPOSITE_LEFT DOUBLE_DOOR_SINGLE_SWING_OPPOSITE_RIGHT
6	引き戸	SLIDING_TO_LEFT SLIDING_TO_RIGHT
7	両開き引き戸	DOUBLE_DOOR_SLIDING
8	折りたたみドア	FOLDING_TO_LEFT FOLDING_TO_RIGHT
9	折りたたみドア マルチ	DOUBLE_DOOR_FOLDING
10	回転ドア	REVOLVING
11	ロールアップ	ROLLINGUP
12	その他	USERDEFINED
13	開き固定	SWING_FIXED 互換性：ARCHICAD 23で導入されました。

ifc_LiningOffset 長さ

ドア枠オフセット。

ifc_CasingDepth 長さ

ドア額縁の厚さ。

ifc_CasingThickness ドア額縁の幅。	長さ
ifc_ThresholdDepth ドア沓摺の深さ。	長さ
ifc_ThresholdThickness ドア沓摺の厚さ。	長さ
ifc_ThresholdOffset 沓摺のオフセット。	長さ
ifc_TransomOffset トランザムオフセット。	長さ
ifc_DoorPanel ifc_DoorPanel[x][1] - ドアサッシの厚さ。 ifc_DoorPanel[x][2] - ドアサッシの幅。	長さ - 配列

ifc_DoorPanel[x][3] パラメータ値	**IfcDoorPanelOperationEnum カテゴリ**
0	NOTDEFINED
1	SWINGING
2	DOUBLE_ACTING
3	SLIDING
4	FOLDING
5	REVOLVING
6	ROLLINGUP
7	USERDEFINED

ifc_DoorPanel[x][4] パラメータ値	**IfcDoorPanelPositionEnum カテゴリ**
0	NOTDEFINED
1	LEFT
2	MIDDLE
3	RIGHT

窓オブジェクトの基本パラメータ

ifc_optype - Windows 整数/文字列

窓の開口部のタイプ、IFC_optype_window.gsm macro でコントロールされます。

ifc_optype (integer) パラメータ値	ifc_optypestr (string) パラメータ値	IfcWindowStyleOperationEnum カテゴリ
0	未定義	NOTDEFINED
1	単一	SINGLE_PANEL
2	両開き 垂直	DOUBLE_PANEL_VERTICAL
3	両開き 水平	DOUBLE_PANEL_HORIZONTAL
4	3枚 垂直	TRIPLE_PANEL_VERTICAL
5	3枚 水平	TRIPLE_PANEL_HORIZONTAL
6	3枚 下	TRIPLE_PANEL_BOTTOM
7	3枚 上	TRIPLE_PANEL_TOP
8	3枚 左	TRIPLE_PANEL_LEFT TRIPLE_PANEL_RIGHT
9	3枚 右	TRIPLE_PANEL_RIGHT TRIPLE_PANEL_RIGHT
10	その他	USERDEFINED

ifc_MullionThickness ifc_MullionThickness - 長さ

マリオン幅。

ifc_FirstMullionOffset ifc_FirstMullionOffset - 長さ

マリオン中央線のオフセット。

ifc_SecondMullionOffset ifc_SecondMullionOffset - 長さ

第2マリオンのマリオン中央線のオフセット。

ifc_FirstTransomOffset ifc_FirstTransomOffset - 長さ

トランザム中央線オフセット。

ifc_SecondTransomOffset ifc_SecondTransomOffset - 長さ

第2トランサムに対してのマリオン中央線のオフセット。

ifc_WindowPanel	長さ - 配列

ifc_WindowPanel[x][1] - 窓サッシの厚さ。
ifc_WindowPanel[x][2] - 窓サッシの幅。

ifc_WindowPanel[x][3] パラメータ値	**IfcWindowPanelOperationEnum カテゴリ**
0	NOTDEFINED
1	SIDEHUNGRIGHTHAND
2	SIDEHUNGLEFTHAND
3	TILTANDTURNRIGHTHAND
4	TILTANDTURNLEFTHAND
5	TOPHUNG
6	BOTTOMHUNG
7	PIVOTHORIZONTAL
8	PIVOTVERTICAL
9	SLIDINGHORIZONTAL
10	SLIDINGVERTICAL
11	REMOVABLECASEMENT
12	FIXEDCASEMENT
13	OTHEROPERATION

ifc_WindowPanel[x][4] パラメータ値	**IfcWindowPanelPositionEnum カテゴリ**
0	NOTDEFINED
1	LEFT
2	MIDDLE
3	RIGHT
4	BOTTOM
5	TOP

移送要素の基本パラメータ

ifc_optype - Transport Elements 整数

移送要素のタイプの選択。互換性：ARCHICAD 23以降廃止されました。

ifc_optype (integer) パラメータ値	IfcTransportElementTypeEnum カテゴリ
0	NOTDEFINED
1	ELEVATOR
2	ESCALATOR
3	MOVINGWALKWAY
4	USERDEFINED

上下移動オブジェクトの基本パラメータ

ifc_CapacityByWeight realnum

輸送要素の容量を重量で測定。互換性：ARCHICAD 23以降廃止されました。

ifc_CapacityByNumber 整数

輸送要素の容量を人数で測定。互換性：ARCHICAD 23以降廃止されました。

階段オブジェクトの基本パラメータ

ifc_StairType 整数

階段の一続きの数と踊り場の数による階段タイプの基本設定、組み込まれた階段のStairMakerアドオンで制御されています。

0 未定義
1 StraightRunStair
2 TwoStraightRunStair
3 QuarterWindingStair
4 QuarterTurnStair
5 HalfWindingStair
6 HalfTurnStair
7 TwoQuarterWindingStair
8 TwoQuarterTurnStair
9 ThreeQuarterWindingStair
10 ThreeQuarterTurnStair
11 SpiralStair
12 DoubleReturnStair
13 CurvedRunStair
14 TwoCurvedRunStair
15 OtherOperation

ifc_NumberOfRiser 整数

階段の蹴上の総数。

ifc_NumberOfTreads 整数

階段の踏み板の総数。

ifc_RiserHeight 長さ

踏面から踏面の垂直距離。蹴上高さは階段または階段部において、同じ高さです。

ifc_TreadLength 長さ

踏み面の前か次の踏み面までの水平距離。 踏み面の距離は階段または歩行線の階段部において、同じ長さです。

MEP要素の一般パラメータ

ifc_subtype		整数			
1	IfcAirTerminalBoxType	25	IfcPumpType	49	IfcLightFixtureType
2	IfcAirTerminalType	26	IfcSpaceHeaterType	50	IfcMotorConnectionType
3	IfcAirToAirHeatRecoveryType	27	IfcTankType	51	IfcOutletType
4	IfcBoilerType	28	IfcTubeBundleType	52	IfcProtectiveDeviceType
5	IfcChillerType	29	IfcUnitaryEquipmentType	53	IfcSwitchingDeviceType
6	IfcCoilType	30	IfcValveType	54	IfcTransformerType
7	IfcCompressorType	31	IfcVibrationIsolatorType	55	IfcActuatorType
8	IfcCondenserType	32	IfcFireSuppressionTerminalType	56	IfcAlarmType
9	IfcCooledBeamType	33	IfcSanitaryTerminalType	57	IfcControllerType
10	IfcCoolingTowerType	34	IfcStackTerminalType	58	IfcFlowInstrumentType
11	IfcDamperType	35	IfcWasteTerminalType	59	IfcSensorType
12	IfcDuctFittingType	36	IfcCableCarrierFittingType	60	IfcProtectiveDeviceTrippingUnitType
13	IfcDuctSegmentType	37	IfcCableCarrierSegmentType	61	IfcUnitaryControlElementType
14	IfcDuctSilencerType	38	IfcCableSegmentType	62	IfcBurnerType
15	IfcEvaporativeCoolerType	39	IfcElectricApplianceType	63	IfcEngineType
16	IfcEvaporatorType	40	この値はスキップします	64	IfcSolarDeviceType
17	IfcFanType	41	IfcElectricFlowStorageDeviceType	65	IfcElectricDistributionBoardType
18	IfcFilterType	42	IfcElectricGeneratorType	66	IfcCableFittingType
19	IfcFlowMeterType	43	IfcSpaceHeaterType	67	IfcAudioVisualApplianceType
20	IfcSpaceHeaterType	44	IfcElectricMotorType	68	IfcCommunicationsApplianceType
21	IfcHeatExchangerType	45	IfcElectricTimeControlType	69	IfcMedicalDeviceType
22	IfcHumidifierType	46	この値はスキップします	70	IfcInterceptorType
23	IfcPipeFittingType	47	IfcJunctionBoxType		
24	IfcPipeSegmentType	48	IfcLampType		

互換性：ARCHICAD 23での変更点
- 新規タイプ60～70
- 20 IfcGasTerminalTypeがIfcSpaceHeaterTypeに名称変更
- 43 IfcElectricHeaterTypeがIfcSpaceHeaterTypeに名称変更

テキスト処理用パラメータ

ARCHICAD 22より前は、選択したテキストスタイルのコントロールを各種設定ダイアログパネルで展開された注釈ツール（そのためライブラリ部品）で使用できました。 ARCHICAD 22以降は、標準のテキスト処理コントロールセットが専用の「テキストスタイル」パネル形式で各注釈ダイアログに含まれるようになりました。

このようなツールに属するライブラリ部品は、次の固定名オプションパラメータを使用してこれらの機能にアクセスできます。

ARCHICAD 22以降、対応するARCHICADインターフェースコントロール（存在する場合）のロック/非表示は、[特定の固定名オプションパラメータの非表示/ロックを有効化]設定 （ライブラリ部品エディタでオブジェクトの[詳細/互換性オプション]ダイアログを参照）と組み合わせて、ライブラリ部品のパラメータスクリプトでLOCKおよびHIDEPARAMETERコマンドを使用するオプションです。

ツール/バージョン履歴については、各パラメータの互換性の注記を参照してください。

デフォルトでは、ARCHICAD 22より前は以下のパラメータが使用可能かつ互換性があると見なされ、例外は別途記載されます。

AC_TextFont_1 文字列

現在選択されている/保存されているフォントタイプの名前。

互換性：ラベルオブジェクトはARCHICAD 22で導入されました。類似するグローバル変数"LABEL_FONT_NAME"は、それ以降は廃止される見込みです。

AC_TextSize_1 実数

現在設定/保存されているフォントサイズはmm単位です。

互換性：ラベルオブジェクトとゾーンオブジェクトはARCHICAD 22で導入されました。 類似するグローバル変数"LABEL_TEXT_SIZE"は、ラベルで廃止される見込みです。 ゾーンオブジェクトの"ROOM_LSIZE"固定名オプションパラメータもそれ以降は廃止される見込みです。

AC_TextStyle_1 整数

フォントスタイル値マスク。直接入力パラメータとして*DEFINE STYLE{2}*または*UI_TEXTSTYLE_INFIELD*に使用できます。

values:

values = $j_1 + 2*j_2 + 4*j_3 + 128*j_8$, ここで、 各 jiフラグは0または1をとります.

j_1: 太字

j_2: 斜体

j_3: 下線

j_8: 取り消し線

AC_TextStyle_1 = 0の場合、スタイルは標準です。

互換性：ゾーンオブジェクトとラベルオブジェクトはARCHICAD 22で導入されました。類似するグローバル変数"LABEL_FONT_STYLE2"は、それ以降は廃止される見込みです。

AC_TextPen_1 整数

現在選択/保存されているテキストペンのインデックス。

互換性：ゾーンオブジェクトとラベルオブジェクトはARCHICAD 22で導入されました。類似するグローバル変数"LABEL_TEXT_PEN"は、それ以降はラベルで廃止される見込みです。

次のテキスト処理パラメータセットは、全ての注釈タイプのライブラリで使用できます。ラベルの対応するグローバル変数は廃止される見込みです。 廃止されたラベルグローバルの全リストについては、「廃止されたラベルグローバル変数」を参照してください。

AC_TextAlignment_1	整数
テキストの配置。 1 - 左揃え、2 - 中央揃え、3 - 右揃え、4 - 両端揃え 互換性：ARCHICAD 22で導入されました。ラベルの類似するグローバル変数"LABEL_TEXT_ALIGN"は、ARCHICAD 22以降は廃止される見込みです。	
AC_TextLeading_1	実数
テキストの線間隔係数。 互換性：ARCHICAD 22で導入されました。ラベルの類似するグローバル変数"LABEL_TEXT_LEADING"は、ARCHICAD 22以降は廃止される見込みです。	
AC_TextCharWidthFactor_1	実数
テキストの文字幅係数。 互換性：ARCHICAD 22で導入されました。ラベルの類似するグローバル変数"LABEL_TEXT_WIDTH_FACT"は、ARCHICAD 22以降は廃止される見込みです。	
AC_TextCharSpaceFactor_1	実数
テキストの文字間隔係数。 互換性：ARCHICAD 22で導入されました。ラベルの類似するグローバル変数"LABEL_TEXT_CHARSPACE_FACT"は、ARCHICAD 22以降は廃止される見込みです。	

ラベルのパラメータ

ARCHICADに設定されたまたは読み込まれたパラメータ

次のラベルパラメータセットは全てARCHICAD 22で導入されたため、同等のグローバル変数は廃止と見なされます。 廃止されたラベルグローバルの全リストについては、「廃止されたラベルグローバル変数」を参照してください。

ARCHICAD 22以降、対応するARCHICADインターフェースコントロール（存在する場合）のロック/非表示は、[特定の固定名オプションパラメータの非表示/ロックを有効化]設定 （ライブラリ部品エディタでオブジェクトの[詳細/互換性オプション]ダイアログを参照）と組み合わせて、ライブラリ部品のパラメータスクリプトでLOCKおよびHIDEPARAMETERコマンドを使用するオプションです。

AC_bLabelAlwaysReadable	ブール
1 - [ラベル設定/テキストスタイル]パネルで[常に読み取り可能]にチェックされている場合、0 - チェックされていない場合 互換性：ARCHICAD 22で導入されました。類似するグローバル変数"LABEL_ALWAYS_READABLE"は、ARCHICAD 22以降は廃止される見込みです。	

AC_bLabelTextWrap	ブール
1 - [ラベル設定/テキストスタイル]パネルで[テキストを折り返し]にチェックされている場合、0 - チェックされていない場合 互換性：ARCHICAD 22で導入されました。類似するグローバル変数"LABEL_TEXT_WRAP"は、ARCHICAD 22以降は廃止される見込みです。	
AC_bLabelOpaqueFill	ブール
1 - [ラベル設定/テキストスタイル]パネルで[不透過]にチェックされている場合、0 - チェックされていない場合 互換性：ARCHICAD 22で導入されました。類似するグローバル変数"LABEL_TEXT_BG_PEN"は、ARCHICAD 22以降は廃止される見込みです。	
AC_LabelTextBgrPen	整数
選択したテキスト塗りつぶしペンのインデックスが含まれます。 互換性：ARCHICAD 22で導入されました。類似するグローバル変数"LABEL_TEXT_BG_PEN"は、ARCHICAD 22以降は廃止される見込みです。	
AC_bLabelFrame	ブール
1 - [フレーム]にチェックされている場合、0 - [フレーム]にチェックされていない場合 互換性：ARCHICAD 22で導入されました。類似するグローバル変数"LABEL_FRAME_ON"は、ARCHICAD 22以降は廃止される見込みです。	
AC_LabelFrameOffset	長さ
[ラベル設定/テキストスタイル]パネルに従うフレームオフセット値。 互換性：ARCHICAD 22で導入されました。類似するグローバル変数"LABEL_FRAME_OFFSET"は、ARCHICAD 22以降は廃止される見込みです。	
AC_LabelPointerPen	ペン
選択した参照線ペンのインデックスが含まれます。 互換性：ARCHICAD 22で導入されました。類似するグローバル変数"LABEL_ARROW_PEN"は、ARCHICAD 22以降は廃止される見込みです。	
AC_LabelPointerLineType	線種
選択した参照線タイプのインデックスが含まれます。 互換性：ARCHICAD 22で導入されました。類似するグローバル変数"LABEL_ARROW_LINETYPE"は、ARCHICAD 22以降は廃止される見込みです。	

AC_LabelPointerConnection　　整数

参照線の接続位置。

0-中央、1-上端、2 -下端、3 - 右下

互換性：ARCHICAD 22で導入されました。類似するグローバル変数"LABEL_ANCHOR_POS"は、ARCHICAD 22以降は廃止される見込みです。

AC_LabelOrientation　　整数

[シンボルラベル]パネルの[ラベル向き]設定のラベル向きタイプ値。"LABEL_ROTANGLE"値は、方向の設定に従って変わることがあります。

1 - 平行、2 - 直角、3 - 垂直、4 - 水平、5 - カスタム角度

互換性：ARCHICAD 22で導入されました。

拡張されたLOCK/HIDEPARAMETER機能はこのパラメータには使用できません。対応するインターフェースコントロール値のマスキングは"AC_DisableLabelOrientationVal"で設定します。 このパラメータでライブラリ部品のデフォルト値を定義します。

ARCHICADに読み込まれたパラメータ

AC_DisableLabelOrientationVal　　整数

ラベル向き（"AC_LabelOrientation）タイプ値マスク。

bitset: [シンボルラベル]ダイアログパネルで[ラベル向き]の以下のオプションを無効にします。

bitset = $j_1 + 2*j_2 + 4*j_3 + 8*j_4$, ここで、 各 jiフラグは0または1をとります.

j_1: "平行"

j_2: "直角"

j_3: "垂直"

j_4: "水平"

"カスタム"タイプは無効にできません。

互換性：ARCHICAD 22で導入されました。

ac_bDisableLabelFrameDisplay　　ブール

互換性：ARCHICAD 20で導入されました。

組み込みのポインタと枠が設定されている場合にラベルシンボルの周囲に表示される組み込みの長方形の枠描画を非表示にして、ユーザーがカスタム形状の枠をスクリプト作成できるようにします。

ac_bCustomPointerConnection　　ブール

互換性：ARCHICAD 20で導入されました。

組み込みのポインタが設定されている場合に、ラベルシンボルの自動ポインタ接続を制御します。このパラメータをオンに設定した場合、 組み込みのタイプに応じて、カスタムのポインタ接続のために2Dスクリプトで6つのホットスポットを定義できます。 これらのホットスポットは1~6の固定IDを持つ必要があります。IDは次の接続位置を示します。

ポインタがラベルシンボルの左側にある場合：

- 1: 左上接続
- 3: 左中接続
- 5: 左下接続
- 6: 右下接続

ポインタがラベルシンボルの右側にある場合：

- 2: 右上接続
- 4: 右中接続
- 6: 右下接続
- 5: 左下接続

廃止されたパラメータ

廃止された梁/柱パラメータ - リストとラベル用のみ

ARCHICAD 23以降では、これらの値はBEAM_SEGMENT_INFOおよびCOLU_SEGMENT_INFOグローバル変数で統一された値参照により使用できます。「「梁/柱のグローバル変数 - リストとラベル用のみは廃止されました」」も参照してください。互換性のために、これらは均一な直線梁または垂直湾曲梁および均一な柱（GLOB_ELEM_TYPE = 12または6）では引き続き使用できます。

ac_colu_crosssection_type　　柱の断面タイプ（整数）

⛔柱セグメント（27）	✅単一セグメントの柱（6）	⛔複数セグメントの柱（6）

1 - 矩形、2 - 円形、3 - 断面形状

ac_beam_crosssection_type　　梁の断面タイプ（整数）

⛔ 梁セグメント（28）	✅ 単一セグメントの梁（12）	⛔複数セグメントの梁（12）

1 - 矩形、2 - 断面形状

廃止されたゾーンスタンプパラメータ

ROOM_LSIZE　　実数

テキストのフォントサイズ（単位：mm）

互換性：ARCHICAD 22以降廃止されました。代わりに*AC_TextSize_1*を使用します。

ac_disable_controls　　整数

このパラメータは、[ゾーンスタンプ]設定ダイアログの[フォントサイズ]（"ROOM_LSIZE"）入力領域の表示の有無を制御できます。0またはオブジェクトがパラメータを持たない - フォントサイズを表示、1 - フォントサイズを非表示（そのためパラメータリストに余分なスペースを確保できます）

互換性：ARCHICAD 22以降は無効。[フォントサイズ]入力コントロールが[テキストスタイル]パネルに移動され、その可視性はLOCK/HIDEPARAMETERコマンドで制御します。

REQUESTオプション

n = REQUEST (question_name, name_or_index, variable1 [, variable2, ...])
n = REQUEST{2} (question_name, name_or_index, name, variable1 [, variable2, ...])
n = REQUEST{3} (question_name, name, name_or_index_array, variable1 [, variable2, ...])
n = REQUEST{4} (question_name, name_or_index, index, name, variable1 [, variable2, ...])

最初のパラメータは質問文字列を示し、2番目（以降）は質問のオブジェクト（存在する場合）を示します。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数です（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）。

ARCHICADは、REQUESTコマンドのバージョンまたは要求オプションの正確な名前（定数文字列として）のいずれかによって、入力パラメータの順序と数値を識別します。つまり、次のオプションの前者または両方を使用するのが最も確実です。

- 要求名（常に定数文字列）
- コマンドにバージョンを追加

Requestパラメータスクリプトの互換性

パラメータスクリプト（またはマスタースクリプトをパラメータスクリプトとして起動）で使用されるほとんどのクエリは、結果的に不安定な戻り値になる可能性があり、その利用をお勧めできません。

ARCHICAD 19以前の互換性：パラメータスクリプト（またはマスタースクリプトをパラメータスクリプトとして起動）で使用されるほとんどのクエリは、結果的に不安定な戻り値になる可能性があります。

ARCHICAD 20以降の互換性：パラメータスクリプトの場合、以下が適用されます。

- 要求式は、成功の戻り値として常に0を持ちます
- 要求された値は、型に一致したデフォルト値のみ（空白の文字列または0）を含みます

ARCHICAD 19以降、パラメータスクリプトで制限された要求を使用した場合もGDL警告が生成されます。

パラメータスクリプトの互換性を確認するには、以下の表を参照してください。：

凡例：

✅	制限なしに機能します
⚠	機能します（追加の警告あり）
⛔	機能しません：式は0を返し、返された変数に型が一致するダミーのデフォルト値（空白の文字列または0）を含み、追加の警告が生成されます。

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
ANCESTRY_INFO	✅	⚠	⛔
ANGULAR_DIMENSION	✅	⚠	⛔
ANGULAR_LENGTH_DIMENSION	✅	⚠	⛔
AREA_DIMENSION	✅	✅	✅
ASSOCEL_PROPERTIES	✅	⚠	⛔
ASSOCLP_NAME	✅	⚠	⛔
ASSOCLP_PARVALUE	✅	✅	✅
ASSOCLP_PARVALUE_WITH_DESCRIPTION	✅	✅	✅
AUTOTEXT_LIST	-	-	⛔

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
BUILDING_MATERIAL_INFO	✅	⚠	⛔

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
CALC_ANGLE_UNIT	✅	⚠	⛔
CALC_AREA_UNIT	✅	⚠	⛔
CALC_LENGTH_UNIT	✅	✅	✅
CALC_VOLUME_UNIT	✅	✅	✅
CLASS_OF_FILL	✅	✅	✅
CLEAN_INTERSECTIONS	✅	⚠	⛔
COMPONENT_PROJECTED_AREA	✅	⚠	⛔
COMPONENT_VOLUME	✅	⚠	⛔
CONFIGURATION_NUMBER	-	-	⛔
CONSTR_FILLS_DISPLAY	✅	⚠	⛔
CUSTOM_AUTO_LABEL	✅	⚠	⛔

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
DATETIME	✅	⚠	⚠
DOOR_SHOW_DIM	✅	⚠	⛔

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
ELEVATION_DIMENSION	✅	⚠	⛔

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
FLOOR_PLAN_OPTION	✓	⚠	⛔
FONTNAMES_LIST	✓	✓	✓
FULL_ID_OF_PARENT	-	⚠	⛔

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
HEIGHT_OF_STYLE	✓	✓	✓
HOME_STORY	✓	✓	✓
HOME_STORY_OF_OPENING	✓	✓	✓
HOMEDB_INFO	✓	✓	✓

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
ID_OF_MAIN	✓	⚠	⛔
INTERNAL_ID	✓	✓	✓

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
LAYOUT_LENGTH_UNIT	✓	⚠	⚠
LAYOUT_TEXT_SIZE_UNIT	✓	✓	✓
LEVEL_DIMENSION	✓	⚠	⛔
LINEAR_DIMENSION	✓	✓	✓

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
MATCHING_PROPERTIES	✓	⚠	⛔
MATERIAL_INFO	✓	⚠	⛔
MODEL_LENGTH_UNIT	✓	⚠	⚠
MODEL_TEXT_SIZE_UNIT	✓	⚠	⛔

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
NAME_OF_FILL	✓	⚠	⛔
NAME_OF_LINE_TYPE	✓	⚠	⛔
NAME_OF_LISTED	✓	⚠	⛔
NAME_OF_MACRO	✓	✓	✓
NAME_OF_MAIN	✓	✓	✓
NAME_OF_MATERIAL	✓	⚠	⛔
NAME_OF_PLAN	✓	⚠	⛔
NAME_OF_PROGRAM	✓	⚠	⛔
NAME_OF_STYLE	✓	⚠	⛔

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
PEN_OF_RGB	✓	⚠	⛔
PROGRAM_INFO	✓	✓	✓
PROPERTIES_OF_PARENT	-	-	⛔
PROPERTY_NAME	-	-	⛔
PROPERTY_VALUE_OF_PARENT	-	-	⛔

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
RADIAL_DIMENSION	✓	⚠	⛔
REFERENCE_LEVEL_DATA	✓	✓	✓
RGB_OF_MATERIAL	✓	⚠	⛔
RGB_OF_PEN	✓	⚠	⛔

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
SILL_HEIGHT_DIMENSION	✓	✓	✓
STORY	✓	⚠	⛔
STORY_INFO	✓	✓	✓
STYLE_INFO	✓	⚠	⚠
SUM_WITH_ROUNDING	-	-	✓

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
TEXTBLOCK_INFO	✓	⚠	⛔

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
VIEW_ROTANGLE	✓	✓	✓
WINDOW_DOOR_DIMENSION	✓	✓	✓
WINDOW_DOOR_SHOW_DIM	✓	⚠	⛔
WINDOW_DOOR_ZONE_RELEV	✓	⚠	⚠
WINDOW_DOOR_ZONE_RELEV_OF_OWNER	✓	⚠	⚠
WINDOW_SHOW_DIM	✓	⚠	⛔
WORKING_ANGLE_UNIT	✓	⚠	⚠
WORKING_LENGTH_UNIT	✓	✓	✓

パラメータスクリプトの互換性	ARCHICAD 18	ARCHICAD 19	ARCHICAD 20
ZONE_CATEGORY	✓	✓	✓
ZONE_COLUS_AREA	✓	⚠	⛔
ZONE_RELATIONS	✓	✓	✓
ZONE_RELATIONS_OF_OWNER	✓	⚠	⚠

要求の詳細

n = 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。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

例 1: 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
```

n = REQUEST (**"Name_of_macro"**, "", my_name)
n = 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）。

n = 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）。 注釈要素には対応していません（例： ラベル、ドア/窓マーカー、ゾーンスタンプ）。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = REQUEST (**"Full_ID_of_parent"**, "", id_string)

平面図にリンクまたはホットリンクされた注釈要素の場合、ツールの設定ダイアログボックスで設定されている、リンクされているモジュールおよび、親ライブラリ部品の全ての識別子（Master ID）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）。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = 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。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = 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。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = 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。

n = REQUEST (**"Home_story_of_opening"**, "", 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。 ホームフロアは1階で、ここに開口部が表示されます。ドア、窓、壁の終端、コーナー窓、天窓のスクリプト、およびそのラベルとマーカーのスクリプトに使用できます。 パラメータスクリプトで使用された場合、警告させます。

n = 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 2:

```
DIM t[]
n = REQUEST ("STORY_INFO", "", nr, t)
FOR i = 1 TO nr
   nr = STR ("%.0m", t [4 * (i - 1) + 1])
   name = t [4 * (i - 1) + 2]
   elevation = STR ("%m", t [4 * (i - 1) + 3])
   height = STR ("%m", t [4 * (i - 1) + 4])
   TEXT2 0, -i, nr + "," + name + "," + elevation + "," + height
NEXT i
```

これらの要求で、ARCHICADの[オプション]→[プロジェクト設定]→[寸法]および[計算単位/規則]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。

n = REQUEST (**"Linear_dimension"**, "", format_string)
n = REQUEST (**"Angular_dimension"**, "", format_string)
パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = REQUEST (**"Angular_length_dimension"**, "", format_string)
パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = REQUEST (**"Radial_dimension"**, "", format_string)
パラメータスクリプトで使用された場合、警告させます。

n = REQUEST (**"Level_dimension"**, "", format_string)
パラメータスクリプトで使用された場合、警告させます。

n = REQUEST (**"Elevation_dimension"**, "", format_string)
パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = REQUEST (**"Window_door_dimension"**, "", format_string)
n = REQUEST (**"Sill_height_dimension"**, "", format_string)
n = REQUEST (**"Area_dimension"**, "", format_string)
n = REQUEST (**"Calc_length_unit"**, "", format_string)
n = REQUEST (**"Calc_area_unit"**, "", format_string)
パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = REQUEST (**"Calc_volume_unit"**, "", format_string)
n = REQUEST (**"Calc_angle_unit"**, "", format_string)

パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

例 3:

```
format = "" num = 60.55
n = REQUEST ("Angular_dimension", "",format)!"%.2dd"
TEXT2 0, 0, STR (format, num)!60.55
```

n = 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） パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = 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。

n = 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）。

n = 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付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）。 パラメータスクリプトで使用された場合、警告させます。

n = 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。 互換性確保のためにのみ使用できます。ゾーンスタンプ固定パラメータで設定された数量を使用することをお勧めします。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = 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。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = REQUEST (**"Rgb_of_material"**, name, r, g, b)

n = REQUEST (**"Rgb_of_pen"**, penindex, r, g, b)
n = REQUEST (**"Pen_of_RGB"**, "r g b", penindex)
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。パラメータスクリプトで使用された場合、3つの式は全て0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。
n = 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。
n = 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「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。 パラメータスクリプトで使用された場合、警告させます。
n = 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。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。
n = REQUEST (**"Name_of_building_material"**, index, name)
与えられた変数に、インデックスで識別されたビルディングマテリアル名を返します。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。
n = 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。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。
n = 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。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。
n = 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。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。
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）。
n = 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、ARCHICAD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。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

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。

n = 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。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = 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。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = 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。 パラメータスクリプトで使用された場合、警告させます。

n = 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。 ドア/窓の開口方向が「zone_relations」要求で識別された第1の部屋の方向である場合は、変数out_directionに1を返し、開口方向が第2の部屋の方向である場合は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。 パラメータスクリプトで使用された場合、警告させます。

n = 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。 パラメータスクリプトで使用された場合、警告させます。

n = 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。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = REQUEST (**"Working_length_unit"**, "", format_string)
n = 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。 ユーザーインターフェイススクリプトを解釈では両方の要求が動作しますが、"Working_angle_unit"がパラメータスクリプトで使用された場合、警告させます。

n = REQUEST (**"Model_length_unit"**, "", format_string)
n = 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。 パラメータスクリプトで使用された場合、2つの式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。どちらもユーザーインターフェイススクリプトでのみ機能します。

n = REQUEST (**"Model_text_size_unit"**, "", format_string)
n = 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。 要求がパラメータスクリプトで使用された場合、警告が生じます。

n = REQUEST (**"Properties_Of_Parent"**, propertyType, parentProperties)

親オブジェクトのプロパティを返します。全てのプロパティが次の形式の1つの配列で返されます：ID、タイプ、グループ、名前。 ラベルでのみ使用できます。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

```
躯体のプロパティ：      [id,   "",    "",            PropertyName]
ACプロパティ：        [guid, "",    "GroupName",    PropertyName]
IFCプロパティ：       [id,   "IFC",          "GroupName", PropertyName]
分類：     [guid, "Classification", "",          ClassificationSystemName]
断面形状パラメータ：[guid, "",       "Profile Parameters", ParameterName]
```

propertyType: 要求されたプロパティのタイプを定義するキーワード。空白の文字列は、全タイプのプロパティを返します。有効値：

- "COREPROPERTY"
- "ACPROPERTY"
- "IFCPROPERTY"
- "CLASSIFICATION"
- "PROFILEPARAMETER"

互換性：この要求はARCHICAD 20で導入されました。プロパティタイプオプションと分類プロパティタイプはARCHICAD 21で導入され、プロファイルパラメータプロパティタイプは22で導入されました。

例 4:

```
DIM parentProperties[]
n = REQUEST ("Properties_Of_Parent", "", parentProperties)
! parentProperties =   [Id1, TypeName1, GroupName1, PropertyName1,
                       Id2, TypeName2, GroupName2, PropertyName2,
                       ...
                       Idn, TypeNamen, GroupNamen, PropertyNamen]
```

n = REQUEST (**"Property_Value_Of_Parent"**, "id", type, dim1, dim2, propertyValues)

選択したプロパティの値配列を返します。 ラベルでのみ使用できます。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

互換性：ARCHICAD 20で導入されました。

id: 選択されたプロパティのID（文字列）

type: 選択されたプロパティ値の型
- 1: ブール
- 2: 整数
- 3: 実数
- 4: 文字列
- 5: 長さ
- 6: 面積
- 7: 体積
- 8: 角度

互換性：長さ、面積、体積、角度タイプがARCHICAD 22で導入されました。

dim1, dim2: propertyValues配列の次元
- dim1 = 0, dim2 = 0: 単純なスカラー値
- dim1 > 0, dim2 > 0: 値のリスト

例 5:

```
DIM propertyValues[]
n = REQUEST ("Property_Value_Of_Parent", "ExampleId", type, dim1, dim2, propertyValues)
```

n = REQUEST (**"Property_Values_Of_Parent"**, propInputIds, propOutputVals)

指定されたプロパティID辞書の値辞書を返します。 ラベルでのみ使用できます。パラメータスクリプトで使用された場合、または不明な入力辞書キーと一緒に使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

互換性：ARCHICAD 23で導入されました。

propInputIds: （辞書）選択したプロパティIDを定義します。

propInputIds.propertyIds[n]: （配列）選択した各プロパティIDの辞書を含みます。

propInputIds.propertyIds[n].id: （文字列）選択したプロパティのID。

propOutputVals: （辞書）選択したプロパティIDのプロパティ値データ。

propOutputVals.propertyValues[n]: （配列）各プロパティ値の辞書を含みます。

propOutputVals.propertyValues[n].value_status: 選択したプロパティ値のステータス
1: 選択したプロパティが使用可能で、値があります
2: 選択したプロパティが使用可能ですが、これに対して値が定義されていません
3: プロパティが使用できないか、選択したIDが無効です
4: プロパティ値を評価できません

propOutputVals.propertyValues[n].type: 選択されたプロパティ値の型 このキーは、propOutputVals.propertyValues[n].value_statusが1または2の場合のみ存在します。
1: ブール
2: 整数
3: 実数
4: 文字列
5: 長さ
6: 面積
7: 体積
8: 角度

propOutputVals.propertyValues[n].value[]: （配列）選択したプロパティ値のリストを含みます。

例 6:

```
dict propInputIds
    propInputIds.propertyIds[1].id = "ExampleId1"
    propInputIds.propertyIds[2].id = "ExampleId2"
    ...
    propInputIds.propertyIds[n].id = "ExampleIdn"
dict propOutputVals
n = REQUEST ("Property_Values_Of_Parent", propInputIds, propOutputVals)
! propOutputVals
    .propertyValues[1].value_status
    .propertyValues[1].type
    .propertyValues[1].value[]
    .propertyValues[2].value_status
    .propertyValues[2].type
    .propertyValues[2].value[]
    ...
    .propertyValues[n].value_status
    .propertyValues[n].type
    .propertyValues[n].value[]
```

n = REQUEST (**"Component_Properties_Of_Parent"**, propertyType, parentComponentProperties)

構成要素のプロパティを返します。これは親オブジェクトのビルディングマテリアル構成要素の少なくとも1つに使用できます。全てのプロパティが次の形式の1つの配列で返されます：ID、タイプ、グループ、名前。 ラベルでのみ使用できます。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。
ACプロパティ：　[guid,　"",　　"GroupName", PropertyName]
分類：[guid,　"Classification", "",　　ClassificationSystemName]

propertyType: 要求されたプロパティのタイプを定義するキーワード。空白の文字列は、使用可能な全タイプのプロパティを返します。有効値：
"ACPROPERTY"
"CLASSIFICATION"
互換性：この要求はARCHICAD 23で導入されました。

例 7:

```
DIM parentComponentProperties[]
n = REQUEST ("Component_Properties_Of_Parent", "", parentComponentProperties)
! parentComponentProperties =   [Id1, TypeName1, GroupName1, PropertyName1,
                                 Id2, TypeName2, GroupName2, PropertyName2,
                                 ...
                                 Idn, TypeNamen, GroupNamen, PropertyNamen]
```

n = REQUEST (**"Component_IDs_Of_Parent"**, collectComponents, outputCompIds)
親オブジェクトのビルディングマテリアル構成要素IDを辞書形式で返します。 ラベルでのみ使用できます。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。
互換性：ARCHICAD 23で導入されました。

collectComponents: （辞書）親オブジェクトのビルディングマテリアル構成要素を収集する方法を定義します。

collectComponents.collectMode: （整数）ビルディングマテリアル構成要素を収集する方法。このキーは省略可能で、存在しない場合、要求は1のデフォルトのcollectModeを使用します。
1: （デフォルト）親の全てのビルディングマテリアル構成要素IDを返します。
2: 親の要素タイプに応じて、同じビルディングマテリアル構成要素IDを
WALL_SKINS_PARAMS、SHELLBASE_SKINS_PARAMS、SLAB_SKINS_PARAMS、またはROOF_SKINS_PARAMSと同じ順序で返します。

outputCompIds: （辞書）親オブジェクトのビルディングマテリアル構成要素ID。

outputCompIds.componentIds[n]: （配列）各ビルディングマテリアル構成要素IDの辞書を含みます。

outputCompIds.componentIds[n].id: （整数）親要素のビルディングマテリアル構成要素ID。

例 8:

```
dict collectComponents
      collectComponents.collectMode = 1
dict outputCompIds
n = REQUEST ("Component_IDs_Of_Parent", collectComponents, outputCompIds)
! outputCompIds
      .componentIds[1].id
      .componentIds[2].id
      ...
      .componentIds[n].id
```

n = REQUEST (**"Component_Property_Values_Of_Parent"**, compPropInput, compPropVals)
指定された構成要素IDとプロパティIDの値辞書を返します。 ラベルでのみ使用できます。パラメータスクリプトで使用された場合、または不明な入力辞書キーと一緒に使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

互換性：ARCHICAD 23で導入されました。

compPropInput: （辞書）選択したビルディングマテリアル構成要素とプロパティIDを定義します。

compPropInput.componentId: （辞書）選択したビルディングマテリアル構成要素IDの辞書を含みます。

compPropInput.componentId.id: （整数）選択したビルディングマテリアル構成要素ID、"Component_IDs_Of_Parent"要求により使用できます。

compPropInput.propertyIds[n]: （配列）選択した各プロパティIDの辞書を含みます。

compPropInput.propertyIds[n].id: （文字列）選択したプロパティのID。

compPropVals: （辞書）ビルディングマテリアル構成要素の選択したプロパティIDのプロパティ値データ。

compPropVals.propertyValues[n]: （配列）各プロパティ値の辞書を含みます。

compPropVals.propertyValues[n].value_status: 選択したプロパティ値のステータス
1: 選択したプロパティが使用可能で、値があります
2: 選択したプロパティが使用可能ですが、これに対して値が定義されていません
3: プロパティが使用できないか、選択したIDが無効です
4: プロパティ値を評価できません

compPropVals.propertyValues[n].type: 選択されたプロパティ値の型 このキーは、compPropVals.propertyValues[n].value_statusが1または2の場合のみ存在します。
1: ブール
2: 整数
3: 実数
4: 文字列

5: 長さ
6: 面積
7: 体積
8: 角度

compPropVals.propertyValues[n].value[]: （配列）選択したプロパティ値のリストを含みます。

例 9:

```
dict compPropInput
    compPropInput.componentId.id = iActualID ! "Component_IDs_Of_Parent"要求から
    compPropInput.propertyIds[1].id = "ExampleId1"
    compPropInput.propertyIds[2].id = "ExampleId2"
    ...
    compPropInput.propertyIds[n].id = "ExampleIdn"
dict compPropVals
n = REQUEST ("Component_Property_Values_Of_Parent", compPropInput, compPropVals)
! compPropVals =  propertyValues[1].value_status
   .propertyValues[1].value_status
   .propertyValues[1].type
   .propertyValues[1].value[]
   .propertyValues[2].value_status
   .propertyValues[2].type
   .propertyValues[2].value[]
   ...
   .propertyValues[n].value_status
   .propertyValues[n].type
   .propertyValues[n].value[]
```

n = REQUEST (**"Property_Name"**, "id", typeName, groupName, propertyName)
選択したプロパティのタイプ、グループ、名前を返します。 ラベルでのみ使用できます。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。
互換性：ARCHICAD 20で導入されました。

id: 選択されたプロパティのID（文字列）

typeName: 選択されたプロパティの型（文字列）
"IFC": IFCプロパティの場合
"": その他のプロパティ

groupName: 選択したプロパティのグループ（文字列）
躯体プロパティの空白の文字列（""）

propertyName: 選択したプロパティの名前（文字列）

n = REQUEST (**"AUTOTEXT_LIST"**, "", autoTextListArray)
プロジェクトで使用される自動テキストの1つのAUTOTEXT配列を、次の3項目[“ID”, “カテゴリ”,"名前”]で返します。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。スクリプトでのみ使用できます。 IDはUI_CUSTOM_POPUP...コマンドを使用してパラメータに保存されます。
プロジェクト情報と自動テキストダイアログ（[テキストツール] - [自動テキストを挿入]）の全ての自動テキストを含みます。
互換性：ARCHICAD 20で導入されました。

例 10:

```
DIM autoTextListArray[]
n = REQUEST ("AUTOTEXT_LIST", "", autoTextListArray)
! autoTextListArray =   [ID1, CategoryName1, TextName1,
                         ID2, CategoryName2, TextName2,
                         ...
                         IDn, CategoryNamen, TextNamen]
```

n = REQUEST{3} (**"Sum_with_rounding"**, req_name, addends_array, result)
addends_array内の数の合計を、プロジェクト設定の[合計の計算]に従って四捨五入して返します。 この設定は、[オプション]→[プロジェクト設定]→[計算単位/規則]にあります。
プロジェクト設定の可能な設定値：

- [表示値]：要求は最初にreq_nameに従って加数を四捨五入してから、それらを合計します。
- [正確な値]：要求は単純に加数を合計します。

パラメータスクリプトで使用された場合、警告させます。
互換性：ARCHICAD 20で導入されました。
戻り値：

- req_nameが無効な場合は0
- 呼び出しが成功した場合は1

req_name: [合計の計算]を[表示値]に設定した場合に、加数の四捨五入方法を指定する形式要求の名前。
例えば、req_name = "Area_dimension"で、[プロジェクト設定]→[寸法]→[面積計算]が小数点以下3桁の[平方センチメートル]に設定され、0.025に四捨五入される場合、加数は0.025 cm²、つまり0.0000025 m²に四捨五入されます。
有効な要求名：
Linear_dimension、Angular_dimension、Radial_dimension、Level_dimension、Elevation_dimension
Window_door_dimension、Sill_height_dimension、Area_dimension、Calc_length_unit、Calc_area_unit
Calc_volume_unit、Calc_angle_unit
addends_array: 加算される数値の配列。m、m²、m³、度など、どの単位で処理する必要があるかは、req_nameで定義されます。

result: 数値、返されるときに、[合計の計算]の設定に従って加数の合計が設定されます。**結果**は加数と同じ単位であることに注意してください。**req_name**で指定されたターゲット単位には変換されません。

n = REQUEST (**"ASSOCLP_PARVALUE"**, expr, name_or_index, type, flags, dim1, dim2, p_values)

n = REQUEST (**"ASSOCLP_PARVALUE_WITH_DESCRIPTION"**, expr, name_or_index, type, flags, dim1, dim2, p_values_and_descriptions)

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。

expr: 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。

name_or_index: 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）。

type: 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
16: ビルディングマテリアル
17: 断面形状 互換性：ARCHICAD 23で導入されました。

flags:

flags = j_1 + 2*j_2 + 64*j_7 + 128*j_8, ここで、 各 jiフラグは0または1をとります.

j_1: パラメータリスト内の子/インデント
j_2: パラメータリスト内の太字テキスト
j_7: 無効（全ての状況でロック）
j_8: パラメータリスト内で非表示

dim1, dim2: dim1は行数、dim2は列数
dim1 = 0, dim2 = 0: 単純なスカラー値
dim1 > 0, dim2 = 0: 1次元配列
dim1 > 0, dim2 > 0: 2次元配列
If dim2 > 0, then dim1 > 0.

p_values: ASSOCLP_PARVALUE 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）。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。

p_values_and_descriptions: ASSOCLP_PARVALUE_WITH_DESCRIPTION はパラメータ値を返し、次にパラメータ説明文字列がきます。 （「VALUES」 コマンドで指定されています） または、これらのペアの配列。 文字列タイプパラメータの説明文字列は常に空です。 配列要素、配列要素説明文字列は、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）。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。

n = 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。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = 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。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

parameter_string: 特性データレコードの要求されたフィールドを表すコ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 11:

```
DIM DATA []
n = REQUEST ("ASSOCEL_PROPERTIES", "iscomp, code, name", nr, data)
IF nr = 0 THEN
   TEXT2 0, 0, "No properties"
ELSE
   j = 0
   FOR i = 1 TO nr
      IF i MOD 3 = 0 THEN
         TEXT2 0, -j, DATA [i] ! name
         j = j + 1
      ENDIF
   NEXT i
ENDIF
```

n = REQUEST (**"REFERENCE_LEVEL_DATA"**, "", name1, elev1, name2, elev2,
name3, elev3, name4, elev4)

与えられた変数に、「オプション」→「プロジェクト設定」→「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。

n = 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 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

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 12:

```
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
```

n = 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）。幅が0の場合、要求は計算済みの幅および高さを返します。幅が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。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = REQUEST{2} (**"Material_info"**, name_or_index, 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。「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。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。 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。

param_name:
"gs_mat_surface_rgb": 面積 R, G, B [0.0..1.0]
"gs_mat_surface_r": 面積 R [0.0..1.0]
"gs_mat_surface_g": 面積 G [0.0..1.0]
"gs_mat_surface_b": 面積 B [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_rgb": 鏡面反射カラー R, G, B [0.0..1.0]
"gs_mat_specular_r": 鏡面反射カラー R [0.0..1.0]
"gs_mat_specular_g": 鏡面反射カラー G [0.0..1.0]
"gs_mat_specular_b": 鏡面反射カラー B [0.0..1.0]
"gs_mat_emission_rgb": 放射カラー R, G, B [0.0..1.0]
"gs_mat_emission_r": 放射カラー R [0.0..1.0]
"gs_mat_emission_g": 放射カラー G [0.0..1.0]
"gs_mat_emission_b": 放射カラー B [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": テクスチャインデックス

例 13:

```
n = REQUEST{2} ("Material_info", "Brick-Face", "gs_mat_ambient", a)
n = REQUEST{2} ("Material_info", 1, "gs_mat_surface_rgb", r, g, b)
n = REQUEST{2} ("Material_info", "Brick-Face", "gs_mat_texture",
          file_name, w, h, mask, alpha)
n = REQUEST{2} ("Material_info", "My-Material", "my_extra_parameter", e)
```

n = REQUEST{2} (**"Building_Material_info"**, name_or_index, param_name, value_or_values)
与えられた変数に、指定された材質ビルディングマテリアルのパラメータに関する情報を返します。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。 **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：**

param_name:
"gs_bmat_id": ビルディングマテリアルID
"gs_bmat_surface": ビルディングマテリアル 表面インデックス
"gs_bmat_description": ビルディングマテリアル 説明
"gs_bmat_manufacturer": ビルディングマテリアル 製造業者
"gs_bmat_collisiondetection": ビルディングマテリアル 干渉に参加（0 または 1）
"gs_bmat_intersectionpriority": ビルディングマテリアル 交差の優先度
"gs_bmat_cutFill_properties": ビルディングマテリアル 切断塗りつぶし プロパティ（切断塗りつぶし 索引番号、切断塗りつぶし前景ペン 索引番号、切断塗りつぶし背景ペン 索引番号）
"gs_bmat_physical_properties": ビルディングマテリアル 物理的特性（熱伝導率、密度、 比熱、内包エネルギー、内包炭素）

例 14:

```
n = REQUEST{2} ("Building_Material_info", "Brick", "gs_bmat_id", id)
n = REQUEST{2} ("Building_Material_info", "Brick", "gs_bmat_surface", index)
n = REQUEST{2} ("Building_Material_info", "Brick", "gs_bmat_physical_properties",
          thermalConductivity, density, heatCapacity, embodiedEnergy, embodiedCarbon)
```

n = REQUEST (**"FONTNAMES_LIST"**, "", fontnames)
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（文字コードで）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 15:

```
dim fontnames[]
n = 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（文字コードを含めて）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）。 この機能をARCHICAD_Library_Masterファイルに含むことをお勧めします。

n = 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付けられている場所）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、戻り値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。 パラメータスクリプトで使用された場合、警告させます。

n = 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。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = 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。パラメータスクリプトで使用された場合、警告させます。

class: 有効値：
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

n = REQUEST (**"view_rotangle"**, "", angleViewRotation)
現在のビューの回転角を返します。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n = REQUEST (**"program_info"**, "", name[, version[, keySerialNumber[, isCommercial]]])
現在実行中のプログラムの情報を返します。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

name: プログラム名

version: プログラムのバージョン番号

keySerialNumber: キープラグのシリアル番号

isCommercial: フルバージョン（商用版）のプログラムを実行している場合は、trueを返します

n = REQUEST (**"Configuration_number"**, "", stConfigurationNumber)
ソフトライセンスまたはハードウェアキーの場合、現在のARCHICADライセンスに割り当てられた設定番号を返します（文字列式として）。教育版、体験版またはデモ版ライセンスの場合、空白の文字列を返します。各設定番号は一意で、変更されません。
パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。
互換性：ARCHICAD 20で導入されました。

n = 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。

n = REQUEST (**"COMPONENT_PROJECTED_AREA"**, idxSkin, projectedArea)
インデックス付きの層の投影面積を返します。特性スクリプトでのみ使用可能（他のスクリプトは0を返します）。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

idxSkin: 有効値：
0: 標準要素のため

1-：複合構造の層のインデックス
1-：断面形状のコンポーネントのインデックス

例 16:

```
n = request ("COMPONENT_PROJECTED_AREA", 0, a)
COMPONENT "Projected Area", a, "m2"
```

特性スクリプトで使用され、初めに層の面積を要求、次に戻り値を使ってコンポーネントを作成します。

n = REQUEST (**"COMPONENT_VOLUME"**, idxSkin, skinVolume)

インデックス付きの層/コンポーネントの体積を返します。特性スクリプトでのみ使用可能（他のスクリプトは0を返します）。 パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

idxSkin: 有効値：
0: 標準要素のため
1-：複合構造の層のインデックス
1-：断面形状のコンポーネントのインデックス

例 17:

```
n = request ("COMPONENT_VOLUME", 0, v)
COMPONENT "Volume", v, "m3"
```

特性スクリプトで使用され、初めに層の体積を要求、次に戻り値を使ってコンポーネントを作成します。

n = REQUEST (**"DateTime"**, format_string, datetimestring)

datetimestringの形式文字列で現在の日付と時間を返します。DateTimeアドオンを使用すると、必要なチャンネルが開閉します。

format_string: 「チャネルを開く」のparamStringパラメータに記述される形式文字列。

datetimestring: 形式文字列はこの変数で返されます。

要求がパラメータスクリプトで使用された場合、警告が生じます。

断面形状要求

n = REQUEST (**"Name_of_Profile"**, index, name)

変数nameに、インデックスで識別された断面形状名を返します。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

互換性：ARCHICAD 21で導入されました。

n = REQUEST (**"Profile_components"**, name_or_index, nComponents,
compType1, compType2, ..., compTypen)

名前またはインデックスで識別された断面形状の総数（nComponents）および構成要素タイプ（compTypen）を返します。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

compTypei: 断面形状構成要素タイプに使用可能な値：
0: 躯体
1: 仕上げ
2: その他

互換性：ARCHICAD 21で導入されました。

例 1:

```
_nComponents = 0
dim _componentTypes[]
n = REQUEST ("Profile_components", myProfileIdx, _nComponents, _componentTypes
```

n = REQUEST (**"Profile_default_boundingbox"**, name_or_index, xmin, ymin, xmax, ymax)
名前またはインデックスで識別された断面形状の原形に関連する、元の連結した長方形の2つの定義座標点を返します。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

互換性：ARCHICAD 21で導入されました。

n = REQUEST (**"Profile_default_geometry"**, name_or_index, n1, n2, ..., nm,
x11, y11, edgeVisible11, vertEdgeVisible11, additionalStatus11, ...,
x1n1, y1n1, edgeVisible1n1, vertEdgeVisible1n1, additionalStatus1n1,
x21, y21, edgeVisible21, vertEdgeVisible21, additionalStatus21, ...,
x2n2, y2n2, edgeVisible2n2, vertEdgeVisible2n2, additionalStatus2n2, ...,
xm1, ym1, edgeVisiblem1, vertEdgeVisiblem1, additionalStatusm1, ...,
xmnm, ymnm, edgeVisiblemnm, vertEdgeVisiblemnm, additionalStatusmnm)

名前またはインデックスで識別された断面形状の元の形状データを返します。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

n1...ni: その後、各断面形状構成要素の輪郭節点の総数を返します。断面形状構成要素の総数（m）は、"Profile_components"要求によって返されます。

edgeVisiblei: i節点から開始する輪郭が表示されます。

vertEdgeVisiblei: i節点から開始する垂直辺が表示されます。3Dで使用することができます（分節ポリゴンの場合は0）。

additionalStatusi: ポリラインの辺および円弧に使用されます（中心線の設定 = 900、中心点と角度を使用する円弧 = 4000など）。または、円弧の終端制御点に印をつけます（-1、この場合vertEdgeVisibleiおよびedgeVisibleiは自動的に0に設定されます）。

この構造で返されるステータスパラメータは、poly2、cprism、tubeの異なるステータスタイプ定義をサポートします。各形式は次の方法で計算することができます：

例 2:

```
poly2Status = edgeVisible + additionalStatus
prismStatus = additionalStatus
tubeStatus = additionalStatus
if additionalStatus >= 0 then          ! not contour end
   if edgeVisible then
      prismStatus = prismStatus + 15    ! j1, j2, j3, j4
   endif
   if verticalEdgeVisible = 0 then
      prismStatus = prismStatus+ 64     ! j7
      ! tubeで、節点から延びる側面の辺を輪郭の表示に使用
      tubeStatus = tubeStatus + 1
   endif
endif
```

互換性：ARCHICAD 21で導入されました。

n = REQUEST{4} (**"Profile_component_info"**, name_or_index, component_ind, param_name, value)

名前またはインデックスで識別された断面形状の専用の構成要素（component_indによる）の要求された属性値を返します。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

component_indは、（"Profile_components"要求で定義された）nComponentsの有効な範囲である必要があります。

param_name: 断面形状マネージャのアドレス属性設定。valueで返されます。

"gs_profile_bmat": 構成要素のビルディングマテリアルインデックス

"gs_profile_surface": 構成要素の材質上書きインデックス（上書き設定が有効な場合。そうでない場合、ビルディングマテリアルの材質を返します）

"gs_profile_showoutline": 構成要素の「輪郭線を表示」設定

"gs_profile_outlinetype": 構成要素の「輪郭タイプ」設定

"gs_profile_outlinepen": 構成要素の「輪郭ペン」設定

POLY2_B{6}などのコマンドに関連する属性で使用される属性値を返します。ポリゴンの輪郭部は、個別にカスタマイズ可能です。

互換性：ARCHICAD 21で導入されました。

n = REQUEST{4} (**"Profile_component_info"**, name_or_index, component_ind, param_name, value1, value2, ..., valuen)

名前またはインデックスで識別された断面形状の専用の構成要素（component_indによる）の全ての辺の要求された属性を返します。パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。

component_indは、（"Profile_components"要求で定義された）nComponentsの有効な範囲である必要があります。

param_name: 断面形状マネージャのアドレス属性設定。valueで返されます。

"gs_profile_comp_surfaces": 構成要素の辺の個別の材質インデックス

"gs_profile_comp_pens": 構成要素の辺の個別のペンインデックス

"gs_profile_comp_linetypes": 構成要素の線種の個別のペンインデックス
POLY2_B{6}などのコマンドに関連する属性で使用される属性値を返します。ポリゴンの輪郭部は、個別にカスタマイズ可能です。
互換性：ARCHICAD 21で導入されました。

廃止された要求

n = REQUEST (**"Constr_Fills_display"**, "", optionVal)
パラメータスクリプトで使用された場合、式は0を返し、ダミーの戻り値（空白の文字列または0）を含み、追加の警告が生成されます。
ARCHICAD 19以前の互換性：指定された変数に[ドキュメント]→[モデル表示を設定]→[モデル表示オプション]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 （以前の組み立て要素塗りつぶし）。
ARCHICAD 20以降の互換性：戻り値はデフォルトでは常に6です（切断塗りつぶしパターン：設定による）。

optionVal: 切断塗りつぶし表示コード
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）

n = REQUEST (**"internal_ID"**, "", id)
常に1を返します。代わりにGLOB_INTGUIDグローバル変数を使用してください。

アプリケーションクエリオプション

n = **APPLICATION_QUERY** (extension_name, parameter_string, variable1, variable2, ...)
以下はAPPLICATION_QUERYコマンドで使用できる、ARCHICADが提供する要求機能です。 これらの要求のオプションはコマンドの **extension_name** と **parameter_string** パラメータで利用できます。 APPLICATION_QUERYのクエリオプションと戻り値は、実行コンテキストに応じて変化するのでご注意ください。
次のアプリケーションクエリタイプの使用はパラメータスクリプトではサポートされていません。 これらのクエリはARCHICAD 19からGDL警告が生成されるようになり、次のARCHICADのバージョンでは0または空の文字列が返されます。 制限は以下に適用されます：

- "document_feature"

ドキュメント機能

このコマンドは、現在のドキュメント/ビューの機能を返すことができます。 現在は、一つの機能があります - ドキュメントのビュー方向を返します。 これらのタイプのクエリはパラメータスクリプトから制限され、GDL警告が生成されます。

ビュー方向

n = **APPLICATION_QUERY** (**"document_feature"**, "view_direction", type)

このコマンドは、オブジェクトが可視化されている現在のドキュメントタイプのビュー方向を返します。 このコマンドには、追加のパラメータはありません。

type: 戻りタイプ値：
"vertical_only": 平面図の場合
"horizontal_only": 3Dからの断面図と立面図の場合（オブジェクトが断面図/立面図に配置されない場合）
"free": 3Dと3Dドキュメントの場合
"none"
"unset"

MEPシステム

このコマンドは、MEPシステムに関するMEPシステムタイプと情報を返します。 parameter_string パラメータで対処できるより多くの機能があります：

MEPシステムを取得

DIM d[2][]
n = **APPLICATION_QUERY** (**"MEPSYSTEM"**, "GetMEPSystems(domain)", d)

domain: MEP クラスインデックス（ダクト – 1、パイプ – 2）（GDLはコネクタークラスに基づいてMEP分類を定義します）

d: 配列の値：
[2*k-1]: MEPシステム索引
[2*k]: MEPシステム名

n: MEPシステムを2倍にかけた数。

ドメインの取得

n = **APPLICATION_QUERY** (**"MEPSYSTEM"**, "GetDomain(idx)", d)

idx: MEPシステム索引

d: ドメイン（整数）
1: ダクト
2: パイプ
3: ダクトとパイプ
4: ケーブル
5: ダクトとケーブル
6: パイプとケーブル
7: ダクト、パイプ、ケーブル

n: 成功した場合は1、それ以外は0

輪郭ペンの取得
n = **APPLICATION_QUERY** (**"MEPSYSTEM"**, "GetContourPen(idx)", pen)
idx: MEPシステム索引
pen: 輪郭ペン索引（整数）
n: 成功した場合は1、それ以外は0

塗りつぶしペンの取得
n = **APPLICATION_QUERY** (**"MEPSYSTEM"**, "GetFillPen(idx)", pen)
idx: MEPシステム索引
pen: 塗りつぶしペン索引（整数）
n: 成功した場合は1、それ以外は0

背景ペンの取得
n = **APPLICATION_QUERY** (**"MEPSYSTEM"**, "GetBgPen(idx)", pen)
idx: MEPシステム索引
pen: 背景ペン索引（整数）
n: 成功した場合は1、それ以外は0

塗りつぶし種類の取得
n = **APPLICATION_QUERY** (**"MEPSYSTEM"**, "GetFillType(idx)", filltype)
idx: MEPシステム索引
filltype: 塗りつぶし種類索引（整数）
n: 成功した場合は1、それ以外は0

中心線線種の取得
n = **APPLICATION_QUERY** (**"MEPSYSTEM"**, "GetCenterLineType(idx)", line)
idx: MEPシステム索引
line: 中心線線種索引（整数）
n: 成功した場合は1、それ以外は0

中心線ペンの取得
n = **APPLICATION_QUERY** (**"MEPSYSTEM"**, "GetCenterLinePen(idx)", pen)
idx: MEPシステム索引

pen: 中心線ペン索引（整数）
n: 成功した場合は1、それ以外は0

システム材質の取得
n = **APPLICATION_QUERY** (**"MEPSYSTEM"**, "GetSystemMaterial(idx)", material)
idx: MEPシステム索引
material: システム材質索引（整数）
n: 成功した場合は1、それ以外は0

断熱材質の取得
n = **APPLICATION_QUERY** (**"MEPSYSTEM"**, "GetInsulationMaterial(idx)", material)
idx: MEPシステム索引
material: 断熱材質索引（整数）
n: 成功した場合は1、それ以外は0

MEPシステム
このコマンドは、MEPモデラーがアクティブであるかどうかを返します。 parameter_string パラメータで対処できる一つの機能があります：

使用可能か
n = **APPLICATION_QUERY** (**"MEPMODELER"**, "IsAvailable()", isavailable)
isavailable: MEPモデラーが存在している（整数）
n: 成功した場合は1、それ以外は0

MEP 接続タイプ
このコマンドは、接続タイプと接続タイプのスタイルを返します。 parameter_string パラメータで対処できる２つの機能があります：

接続タイプの取得
DIM d[2][]
n = **APPLICATION_QUERY** (**"MEPCONNECTIONTYPE"**, "GetConnectionTypes(connectorClass)", d)
connectorClass: 接続クラス（ダクト - 1、パイプ - 2、ケーブルキャリア - 3）
d: 配列の値：

[2*k-1]: 接続タイプ GUID
[2*k]: 接続タイプ名

n: 接続タイプを2倍にかけた数。

接続タイプスタイルの取得

DIM d[]
n = **APPLICATION_QUERY** (**"MEPCONNECTIONTYPE"**, "GetConnectionTypeStyle(connectorClass)", d)

connectorClass: 接続クラス（ダクト - 1、パイプ - 2、ケーブルキャリア - 3）

d: 配列の値：
[]: 接続タイプスタイル

n: 接続タイプの数。

MEP フレキシブルセグメント

このコマンドは、フレキシブルセグメントの形状を返します。 parameter_string パラメータで対処できる4つの機能があります：

断面の開始

n = **APPLICATION_QUERY** (**"MEPFLEXIBLESEGMENT"**, "StartSectioning()", r)
断面の開始を示します。

r: 未使用

n: 成功した場合は1、それ以外は0

コントロールポイントの追加

n = **APPLICATION_QUERY** (**"MEPFLEXIBLESEGMENT"**, "AddControlPoint(x; y; z)", r)
アドオンにコントロールポイントを与えます。

AddControlPoint:
x: コントロールポイントのX座標
y: コントロールポイントのY座標
z: コントロールポイントのZ座標

r: 未使用

n: 成功した場合は1、それ以外は0

方向と幅ベクトルの追加

n = **APPLICATION_QUERY** (**"MEPFLEXIBLESEGMENT"**,
"AddDirectionAndWidthVector(i; dx; dy; dz; wx; wy; wz)", r)

アドオンに方向とスプライン終端の側面ベクトルを提供します。2度呼ばれます。

AddDirectionAndWidthVector:
i: ポートのID（1: 0. ポート, 2: 1. ポート その他）
dx: ポートの方向ベクトルのXコンポーネント
dy: ポートの方向ベクトルのYコンポーネント
dz: ポートの方向ベクトルのZコンポーネント
wx: ポートの側面ベクトルのXコンポーネント
wy: ポートの側面ベクトルのYコンポーネント
wz: ポートの側面ベクトルのZコンポーネント

r: 未使用

n: 成功した場合は1、それ以外は0

断面の終了

DIM d[]
n = **APPLICATION_QUERY** (**"MEPFLEXIBLESEGMENT"**, "EndSectioning(res)", d)
断面の結果の取得。

res: 断面の解像度

d: 配列の値：
[9*k-8]: kセグメントのX位置
[9*k-7]: kセグメントのY位置
[9*k-6]: kセグメントのZ位置
[9*k-5]: kセグメントのタンジェントベクトルのXコンポーネント
[9*k-4]: kセグメントのタンジェントベクトルのYコンポーネント
[9*k-3]: kセグメントのタンジェントベクトルのZコンポーネント
[9*k-2]: kセグメントのノーマルベクトルのXコンポーネント
[9*k-1]: kセグメントのノーマルベクトルのYコンポーネント
[9*k]: kセグメントのノーマルベクトルのZコンポーネント

n: セグメントの総数

MEP ベンド

このコマンドは、フレキシブルセグメントの形状を返します。 parameter_string パラメータで対処できる4つの機能があります：

断面の開始

n = **APPLICATION_QUERY** (**"MEPBEND"**, "GetBendTypeNames()", d)
d: ベンドタイプ名（INTバージョンの例）

"Radius"
"Square Throat"
"Mitered"
"45° Throat with 45° Heel"
"45° Throat with 90° Heel"
"45° Throat with Radius Heel"
"Radius Throat with 90° Heel"
"Pleated"
"Stamped"
"Segmented"
"Segmented Standing Seam"

n: 成功した場合は1、それ以外は0

パラメータスクリプト

このコマンドは、パラメータスクリプトのさまざまな条件を返すことができます。 現在は一つの機能があります - 最初の実行の区別を返します。

進行中の最初の機会

n = **APPLICATION_QUERY** (**"parameter_script"**, "firstoccasion_in_progress", isFirstRun)

このコマンドは、現在の実行が最初の実行であるかどうか、またはいくつかのパラメータを変更するパラメータスクリプトを前回実行した結果を返します。 このコマンドには、追加のパラメータはありません。

パラメータスクリプトの一部がトリガされたイベントを実行する際の区別は重要です。 - 例： ファンクションボタンの押下を処理。

isFirstRun: 戻り値は、現在の実行が最初の実行であるかを示しています

標準プロパティ

これらのコマンドでは、[分類とプロパティ]タブページで標準プロパティのフォルダ名、パラメータ名、パラメータ値が返されます（IDとカテゴリ、リノベーション、IFCプロパティ）。

互換性：タブページは、ARCHICAD 19まで「タグとカテゴリ」と呼ばれていました。ARCHICAD 20で「カテゴリとプロパティ」に名前が変更され、ユーザー定義プロパティが追加されました。これらのクエリでは、ユーザー定義プロパティは返されません。

パラメータの順序はタブページと同じです。 これらのコマンドに2つの可能な extension_names があります：

- "OwnCustomParameters" はオブジェクトのパラメータを返します。
- "OwnCustomParameters" は親オブジェクトのパラメータを返します。

  互換性：ARCHICAD 20では、親要素（ラベルの）のデータはREQUEST "Properties_Of_Parent"でも使用可能になりました。これらのクエリでは、"COREPROPERTY"と"IFCPROPERTY"のプロパティタイプに対応するプロパティが返されます。

パラメータフォルダ名を取得

DIM folderNamesArray[] ! idString1, shortNameString1, longNameString1,
! ...
! idStringi, shortNameStringi, longNameStringi

n = **APPLICATION_QUERY** (**extension_name**, "GetParameterFolderNames()", folderNamesArray)
標準プロパティのフォルダ名を返します。

folderNamesArray: 標準プロパティのフォルダ名を含む文字配列

n / 3: フォルダ数

パラメータ名の取得

DIM parNamesArray[] ! idString1, shortNameString1, longNameString1,
! ...
! idStringj, shortNameStringj, longNameStringj

n = **APPLICATION_QUERY** (**extension_name**, "GetParameterNames(folderID)", parNamesArray)
標準プロパティの名前を返します。

GetParameterNames:
folderID: "GetParameterFolderNames"から返されたフォルダのID文字列。

parNamesArray: 標準プロパティのフォルダ名を含む文字配列

n / 3: パラメータ数

パラメータの取得

n = **APPLICATION_QUERY** (**extension_name**, "GetParameter(parID)", parValue)
標準プロパティの値を返します。

GetParameter:
parID: "GetParameterNames"から返されるパラメータのID文字列。

parValue: 標準プロパティの値を含む文字列

n: 成功した場合は1、それ以外は0

例: 標準プロパティの全ての値をクエリ

```
DIM folderNamesArray[]
n = APPLICATION_QUERY ("OwnCustomParameters", "GetParameterFolderNames()", folderNamesArray)

for i = 1 to vardim1(folderNamesArray) step 3

    DIM parNamesArray[]
    querystring = "GetParameterNames(" + folderNamesArray[i] + ")"
    n = APPLICATION_QUERY ("OwnCustomParameters", querystring, parNamesArray)

    for j = 1 to vardim1(parNamesArray) step 3

        parValue = ""
        querystring = "GetParameter(" + parNamesArray[j] + ")"
        n = APPLICATION_QUERY ("OwnCustomParameters", querystring, parValue)

    next j
next i
```

ライブラリマネージャー

このコマンドは、パラメータスクリプトのさまざまな機能を返すことができます。

IESファイル

n = **APPLICATION_QUERY** (**"LIBRARY_MANAGER"**, "IES_FILES", ies_files_list)
このコマンドは、アクティブなライブラリでロードされた.iesの拡張子ファイル名のリストを返します。

ユーザー画像ファイル

n = **APPLICATION_QUERY** (**"LIBRARY_MANAGER"**, "USER_IMAGE_FILES", image_files_list)
このコマンドは、アクティブなライブラリでロードされたユーザーが提供した画像ファイル名のリストを返します。（専用のフォルダに含まれていない画像ファイル [TIMG] *を含む名前、[BIMG] *、[UImg] *、または[HImg] *）

GDLスタイルガイド

はじめに

このドキュメントでは、主にソースコードを形式的に満たすために記述するための、GDLコーディングの標準が書かれています。 また、コンテンツのためのルールや推奨事項について説明します。 管理可能なスクリプトを生成するには、これらのルールに従う必要があります; デフォルトでは、全ての宣言型または命令文はルールがあり、ここで '推奨'（または回避可能、オプション等）を除き、明示的に記述されています。

このドキュメントは、GDLスクリプトの共通フォーマットを確立するために作成されました。 GDL言語は小文字大文字および、ほとんどのホワイトスペースを区別しません。 結果として、コーディングのトレーニングと標準が存在します。 標準が決まっていないと、プロジェクトまたは組織での作業を行う場合、とても負担になります。 以下のセクションでは、GRAPHISOFT関連製品のプログラミングを始めた方のために、グラフィソフト社の標準を記述しています。 サポートされているフォーマットはGDL言語の制約には含まれません。

命名規則

一般規則

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、ARCHICAD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）。

変数名

変数とパラメータ名は、パラメータの関数に関連する必要があります。

大小文字の混在（mixedCase）：小文字で始まります; 次に来る単語を大文字で始める必要があります。例：size, bRotAngle180, upperLeftCorner

1つまたは2つの文字の変数名を使用しないでください - 何を意味するか理解に難しくなります。

一般的なカテゴリを示すために一般的に使用される変数名に接頭辞を使用する必要があります。 変数またはパラメータの種類を検索する場合に、時間を節約することができます。 変数の意味を変更した場合は接頭辞を置き換えることを忘れないでください。

表6 変数名の接頭文字

接頭文字	意味	例
i	一般整数値 / 整数索引	iRiser
n	整数値 - 何かの量	nRiser
b	ブール値	bHandrail
st	文字列タイプ値	stPanelTypes
x	点のX座標	xRailPos
y	点のY座標	yRailPos
ペン	ペンカラー	penContour
lt	線種	ltContour
塗りつぶし	塗りつぶし種類	fillMainBody
mat	材質タイプ	matCover

アンダースコア（_）の使用は、同じスクリプト内の特徴的な変数およびパラメータに推奨されています： スクリプト内の変数に対して1つのアンダースコア（_）の接頭辞を使用し、ダブル（__）宣言し、唯一のサブルーチンセクションの中で使用される変数には2つのアンダースコア（_）の接頭辞を使用してください。 パラメータ名の先頭、または名前の単語を区切るためにアンダースコアを使用しないでください。 サブタイプからの歴史的な名前は例外です。

例

```
_iDoorTypes = iDoorTypes
! "_iDoorTypes" 変数は "iDoorTypes" パラメータから値を取得します
gosub "exampleScript"
end

"exampleScript":
    __iDoorTypes = _iDoorTypes * 3
    ! "__iDoorTypes" サブルーチン変数は "_iDoorTypes" の変数を3回値を取得し、返します
return
```

大文字の使用

- 好みに応じてコマンドは、一貫して小文字または、大文字で書く必要があります。 **GRAPHISOFTは小文字の利用をお薦めします**
- 容易にスクリプトの読み取りができるように、GDLグローバル変数は大文字で記述する必要があります。
- 次のキーワードは小文字でなければなりません：call, goto, gosub, parameters.

数式

- 次のバイナリ演算子の前後にスペースを使用する必要があります：
  - 算術：*, /, % (mod), +, -, ^ (**)
  - 論理：& (and), ¦ (or), @ (excluding or), =, <> (#), <, <=, >, >=
  - 割り当て：=

  例：

  a = (b + c % d) * e
- スペースは、次の単項演算子の前後に使用できません：
  - 下付き文字：array[25] 注記：検索機能で検索が容易に行えるように、括弧内（配列[5]） のスペースを避けてください
  - 非論理：not(x)
  - 単項マイナス、単項プラス：-x, +x
- 関数呼び出しは、パラメータリスト、およびパラメータ分離のあらゆるコンマの後ろに開き括弧の前にスペースが必要です：
  abs (signedLength)
  minimum = min (a, 25 * b, c)
- 定数と同等かテストすると（例： i = 5） 定数は2番目のオペランドである必要があります。
- ブール変数に値を代入すると論理式は括弧でする必要があります：
  bBoolValue = (i > j)
- 整数の値または変数の論理否定のブール結果を使用しないでください。 例： if not(iIntVal) thenの代わりに、iIntVal = 0 then を利用してください。 （もちろん、ブール変数や式を否定することができます、例： if not(bBoolVal) then）。
- trueまたはfalseにブール変数と式を比較しないでください; ブールまたはその否定値の値を使用します：

```
bBoolVal = 1
if bBoolVal then          ! instead of: if bBoolVal = 1
   ...
endif
if not(bBoolVal) then     ! instead of: if bBoolVal = 0
   ...
endif
```

- 複雑な式（例： ANDおよびORの両方が存在する場合）は、優先順位を明確にするために括弧する必要があります。
- ほとんど使用されない演算子の組み合わせの前後に括弧を挿入します。
- 多くの部分からなる論理式が複数の行に配置する必要があり、サブ式や論理演算子を揃える必要があります：

```
bResult = (bValue1 & bValue3 & not(bValueWithLongerName)) | ¥
        (not(bValue1) & not(bValue3) & bValueWithLongerName) | ¥
        (not(bValuel2) & bValue3 & not(bValueWithLongerName)) | ¥
        (bValue2 & not(bValue3) & bValueWithLongerName)
```

フロー制御ステートメント

if - else - endif

条件式の一行フォームを使用しないでください。

コードの可読性を向上させるためには、ネストされた文の階層を表現することが不可欠です。 以下の例は、コードブロックの推奨集計を示します。

```
if condition1 then
    statement1
    ...
    statementn
else
    statementn+1
    ...
    statementn+m
endif

if condition2 then
    if condition3 then
        ...
    else
        ...
    endif
    if condition4 then
        ...
    endif
else
    ...
endif
```

for - next, do - while, while - endwhile, repeat - until

コードの可読性を向上させるためには、ネストされた文の階層を表現することが不可欠です。 以下の例は、コードブロックの推奨集計を示します。

```
for i = initialValue to endValue
    statement1
    ...
    statementn
next i

do
    ...
    for i = initialValue to endValue
        statement1
        ...
        statementn
    next i
    ...
    bCondition = ...
    ...
while bCondition
```

サブルーチン

複数回必要とされているコードの一部は、サブルーチンにする必要があります。後の修正を容易にし、リスクを下げ、コードがより構造的になります。サブルーチンのラベルは、その機能に対応している必要があります。 名に数字を使用しないでください、それはコードが読めなくなります。 唯一のサブルーチン内で使用されると宣言された変数は、ダブルアンダースコアで始まる必要があります。
スタイル（斜体テキストは交換する必要があります）：

```
! ======================================================================
!   機能性の簡単な説明
! ----------------------------------------------------------------------
! 入力パラメータ：
!   par1：    簡単な説明（タイプ）
!   par2：    簡単な説明（タイプ）
!   ...
! 出力：
!   par1：    簡単な説明
!   ...
! 備考：
!   呼び出しに対する注意事項
!   実装の重要なポイントの説明
! ======================================================================

subroutine_title:
    ! body
return
```

右に1タビュレーターフィールドによってインデントサブルーチンの本体を記述する必要があります。
サブルーチンの終了 "リターン"を閉じるために、2行の空白行を残してください。
1行に1つのステートメントを記述する必要があります。
サブルーチンは1-2の画面幅（約80行）より長くすべきではありません。
有効性のために全ての入力パラメータを確認または、コメントの中の制限を宣言してください。
コール と パラメータキーワードは小文字です。

コメントの記述

コメントの言語は英語で書くことをお薦めします; 適切なコメントを書き込んでください。
コメントの次のスタイルを使用する必要があります：

スクリプトヘッダー

これはあくまで推奨です

```
! <連絡先のイニシャル>
! ==========================================================================
!   スクリプトの目的を記述した簡単な説明
! --------------------------------------------------------------------------
! 入力パラメータ：
!   par1：    パラメータの説明（整数）
!   par2：    パラメータの説明（1 / -1）
!   par3：    パラメータの説明（0 / 1）
!   ...
! 出力：[マクロが値を返す場合]
!   [1]:    値の説明（タイプ）
!   [2]:    値の説明（タイプ）
!   [ ... NSP]：オリジナルのスタック要素
! 備考：
!   長い説明。
!   呼び出しに対する注記
! ==========================================================================
```

全てのコードは、この後に来ることができます。

セクションの分割

```
! ==========================================================================
! セクションの名
! ==========================================================================
```

完全なコメント行の長さは80文字です。
サブルーチンのために、常にパラメータの意味と戻り値を説明する必要があります。 例： インデックスは常に範囲を示します（0または1、特別な値などから始まる）。
例 「サブルーチン」
未完成なコードが残っている場合は、常にTODOキーワードを記述してください、後に検索が容易に行えます：

```
n = 5 ! TODO：イニシャルの設定; 長さから計算されるます
```

また、現在のタブの深さで始まる、ソースコードの行の間にオプションのセクションの記述を置くことができます。 また、終了時にタブを追加することによって、ソース行の最後に短い説明を追加することができます; または、それらの複数が存在する場合は、タブで整列ができます。
いつもコメントを追加する必要があります：
- 異例のソリューション

- 他の方がコードを理解しやすくなるために役立ちます。
- 何かが禁止、または他の人にお勧めできな場合。

オプション。
コメントブレークでコードのリズムを壊さないようにしてください。
コヒーレントコードブロックをコメントするときは、次のフォーマットを利用できます：

```
! == コードブロック名 ===[
statement1
...
statementn
! == コードブロック名 ===[
```

これは見た目で、ブロックの区切りを分かり易くし、ショートカットで一致するブラケットの検索をサポートします。（例：ctrl +] Microsoft Visual Studio)
多くのコード行がある場合は「if」文の終わりをコメントが次のように if および endif の間にある場合：

```
if condition1 then
    ...
    if condition2 then
        ...
        ! many statements
        ...
    endif   ! if condition2

endif   ! if condition1
```

一部のスクリプトタイプ（フォワードおよびバックワード移行スクリプトは特に）セパレータや構造の推奨される形式を持っています。 ARCHICADライブラリ、または 「一般テクニカル標準」 を参考にご覧ください。

スクリプト構造

4文字幅のタブを使用するようにエディタを設定します。スペースを利用して、表を作成しないでください。 その代わりに、互いにインラインに式を調整するためにスペースを使用します。
行の最大長は120文字です。 ステートメントも、この数に合わせるべきです。 それを超えると、警告が表示されます。
全てのファイル名の参照は、それに応じて拡張機能、スクリプトで大文字と小文字が区別されます。
それがうまくローカライズまたはその他のスクリプトの先頭にすることができれば、複数の時間に使用される値は、使用のブロックの前に直接計算する必要があります。妥協はありません。 変数に格納することで一度だけ計算時間の複素数値または、変換スタック（add、rot、その他）を計算します。

オブジェクトスクリプトはリニアにすることで、より見やすくなります。 計算やモデル生成セグメントは複数回、またはスクリプトの読みやすさのための場合に、サブルーチンの改行をします。 二度同じことをコードを書くことを回避することは全てのプログラミング言語の重要な原則です。 冗長性は、後に多くの困難になります。

簡単に冗長性を回避することができます。小さく選択ブロック内の計算または生成コマンドのためのデータを代わりに、多数の選択のブランチを使用し準備しないようにしてください。

悪いソリューションの例

```
if bOnHomeStory then
    line_type ltContour
    fill gs_fill_type
    poly2_b 5, 3, gs_fill_pen, gs_back_pen,
        left,   0,      1,
        left,   -depth, 1,
        right,  -depth, 1,
        right,  0,      0,
        left,   0,     -1
endif
if (bOnUpperStory or bOnAboveUpper) and bDrawContBB then
    line_type ltBelow
    fill fillTypeBelow
    poly2_b 5, 3, fillPenBelow, fillBackBelow,
        left,   0,      1,
        left,   -depth, 1,
        right,  -depth, 1,
        right,  0,      0,
        left,   0,     -1
endif
```

形状の定義が重複しています! 同一のコマンド間の距離が大きかった場合はさらに悪くなる可能性があります。

良いソリューションの例

```
if bOnHomeStory then
    bPolygon = 1
    line_type ltContour
    fill gs_fill_type
    fillPen = gs_fill_pen
    fillBGPen = gs_back_pen
endif
if (bOnUpperStory or bOnAboveUpper) and bDrawContBB then
    bPolygon = 1
    line_type ltBelow
    fill fillTypeBelow
    fillPen = fillPenBelow
    fillBGPen = fillBackBelow
endif
if bPolygon then
    poly2_b 5, 3, fillPen, fillBGPen,
        left,   0,      1,
        left,   -depth, 1,
        right,  -depth, 1,
        right,  0,      0,
        left,   0,     -1
endif
```

ローカライズスクリプトを準備します。

非ローカライズ文字列には "asdf" を使用（例： マクロコール） ローカライズ文字列には `asdf` を使用（例： 文字列定数、パラメータ値）。

一般テクニカル標準

はじめに

新しいARCHICAD®のローカライズバージョンのリリースや、グラフィソフトの製品、BIMcomponents® ポータルにより、GDLオブジェクトとオブジェクトライブラリの需要が増加しつつあります。 その結果、多くの独立したサードパーティのGDLプログラマは、グラフィソフト用のライブラリまたはオブジェクトの開発が進んでいます。

GRAPHISOFTの製品同様、品質を保つために、ガイドラインは互換性のあるこれらのオブジェクトを維持するために必要だと考えます。 このドキュメントの目的はGDL開発者のための役立つヒント、サンプル、今までに説明されていなかったARCHICADの機能などを紹介することです。

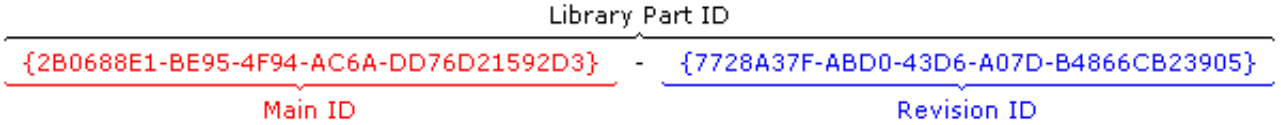
ライブラリ部品形式

ファイル拡張子。

ほとんどのGDLライブラリオブジェクトは、*.gsm拡張子で保存され、ARCHICADの中ではサブタイプによって区別されます。特別な拡張子、*.gdl（GDLスクリプトファイル） MASTER_GDL/MASTEREND_GDL ファイルもあります。 ARCHICADは文字列が "MASTER_GDL..."で始まる場合、または"MASTEREND_GDL... "とファイル名に含まれる場合、特別な方法の処理を行います。 これらのファイルは、属性の定義をロード定義するために使用したり、線種や材質の定義に使用できます（詳しくはこちらを参照： The GDL Script Analysis8ペ ー ジ）。

識別

識別子



IDは長い二つの部分、それぞれ36進文字で構成されています。 最初の36文字は、**メインID** を表し、最後の36文字は、**リビジョンID** IDを表します。

- ライブラリ部品を初めて保存するときにメインIDが作成されます。 また、ライブラリの部品は、「名前を付けて保存」コマンドを使用して再保存されている場合に変更されます。
- ライブラリ部品を初めて保存するときにリビジョンIDも作成されていますが、ライブラリの部品は、「保存」コマンドを使って再保存されている場合に変更されます。 LP_XMLConverterツールを使用すると、コンパイルがリビジョンIDのみを変更し、メインIDは変更されません。

これはメインIDがその機能にライブラリの部分を識別し、リビジョンIDは、オブジェクトのリビジョンを区別するのに役立ちますことを意味します。それでは、実際にこれを見てみましょう。

ライブラリ部品の識別

ARCHICADのオブジェクトを配置すると、プログラムはIDによる参照で格納し、IDを持たないオブジェクト場合には、オブジェクト名のみを考慮します（ARCHICADの8以前に保存したライブラリ部品と.gdlファイル）。 ARCHICADの8以前のバージョンのライブラリ部品には、GUIDが無かったため。 そのようなライブラリ部品がファイル内で検出された場合は、ARCHICADはIDに対して0を割り当てます。

ライブラリをロードする場合、ARCHICADは既にプロジェクト内に配置されたオブジェクトにロードされたライブラリ部品を一致させるため、次の階層の基準を使用しています：

- 格納されたIDが有効の場合：

- ARCHICADは両方のIDで正確な一致を取得しようとします
- それに失敗した場合は、ARCHICADは始めにメインIDの部分で一致を探します。
- それでも一致しない場合は、他の要素の移行テーブの値をチェックし代替を探します。
- ARCHICADの12以前に保存されたファイルをロードする場合は、ARCHICADはライブラリ部品名によって一致させようとします。

• 格納されたIDが0の場合は、識別手順は、名前のみで一致させようとします。

全てのライブラリの部品は、その呼ばれるマクロが「GUIDのルックアップテーブルが含まれているように、配置された要素のマクロを探していたときと同じ処理が実行されます。 マクロ呼び出しを含むオブジェクトを保存するときに、当然、この表は現在ロードされライブラリ内の名前ベースの検索を使用して収集されます。

ライブラリオブジェクトの正確なGUIDを知る方法

この方法を使用するには、サブタイプ階層ダイアログウィンドウを知らなければなりません このダイアログでは、ツリービューで現在ロードされているライブラリのサブタイプの階層を見ることができます。 主な属性 - 名前、バージョン、ID、ファイルの場所、オブジェクトはテンプレートまたは配置可能であるかフラグが示し、選択したライブラリの部品のウィンドウの下部に表示されます。

このダイアログは、3コンテキストで表示されます：

- サブタイプ別にオブジェクトを開く...（ファイルメニュー）
- サブタイプを選択...（ライブラリ部品編集ウィンドウ）
- 全てのオブジェクトを配置（特殊メニュー）

当然のことながら、ライブラリ部品のXML形式でのIDを読み取ることができます（位置：xpointer (/Symbol/@UNID)). これを取得するには、LP_XMLConverterツールを使用します。

互換性の問題

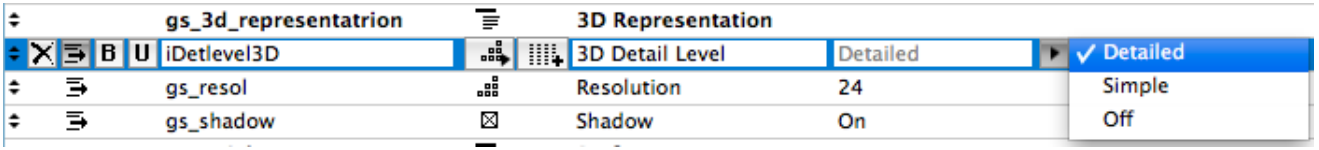
重要な原理は、メインIDは、ライブラリ利用者に一定の機能性を表しているということです。 古いライブラリ部品で既に使用されているメインIDを使用して、新しいライブラリ部品を発行する場合は、新しいライブラリと古いプロジェクトを読み込むときに、古い配置要素は、新しいオブジェクトに置き換えされることを意味します。 これは、オブジェクトのパラメータと機能に変更がないことのようで、ユーザーの期待に反します。 名前または古いパラメータの関数を変更したい場合は、新しいメインIDを生成し、あいまいさや予期しないデータの損失を回避するために、移行スクリプトを使用しています。新しいメインIDは一意かつ、ライブラリ内の他のIDと同一でないことを確認してください。

オブジェクト名の変更を行っても、MainIDに変更がない場合、ARCHICADとの互換性はなくなりませんので、注意してください。 同じように、既存のライブラリ部品に新しい名前（新しいID）を与えると、互換性は無くなります。

この問題はローカライズされたライブラリにも影響します。 あなたはパラメータを制御する文字列型を使用している場合は、関連する値は、国バージョン間で異なります。 例：問題をよく理解していないと、ドイツ版の図面をデンマーク版でロードすると、いくつかのコントロールパラメータには意味が無いので、生成された要素が変更されます。 2つの解決法があります。簡単な方法は、ドイツとデンマークのライブラリがお互いに行うには、結果的に局在化をメインのIDを変更するには、何もないことを宣言することです。 2つ目の方法は、文字列として機能する整数型制御パラメータを作成することです（こちらをご覧ください、 VALUES）。 これらの整数パラ

メータは行列式で、文字列説明は入力方法です（したがって、ローカライズが異なる可能性がありますが、本当の意味のままとなります）。 スクリプトを書くときに、整数パラメータ値を使用してください。

次のコードの例では、文字列型として動作する詳細レベルの整数パラメータを紹介します：




```
! マスタースクリプト：
dim stDetlevel3DDesc [3]
stDetlevel3DDesc[1]=`Detailed`
stDetlevel3DDesc[2]=`Simple`
stDetlevel3DDesc[3]=`Off`

! iDetlevel3D constants
DETLEVEL3D_DETAILED = 1
DETLEVEL3D_SIMPLE   = 2
DETLEVEL3D_OFF      = 3

! パラメータスクリプト：
values{2} "iDetlevel3D"  DETLEVEL3D_DETAILED, stDetlevel3DDesc[1],
                         DETLEVEL3D_SIMPLE,   stDetlevel3DDesc[2],
                         DETLEVEL3D_OFF,      stDetlevel3DDesc[3]
```

移行要素

これは、移行スクリプトを使用して、ライブラリの部品の古い、新しい、更新されたバージョン（新しいメインID）との間のリンクを維持することができます（「上位移行スクリプト」，「下位移行スクリプト」） および 「移行テーブル」。

これらのスクリプトでは、どのライブラリ部品が代替し、新しいオブジェクトのパラメータ値を古いパラメータ値からどのように更新するかが定義されています（新旧のメインIDで関連付けされています）。 移行が唯一の特定のパラメトリックな条件下で行われるためには、ルールを設定することができます。 移行の対象がこれらを満たしている場合は、アップグレードまたはダウングレードが可能ですが、それ以外の場合は、オプションはございません。

移行の対象は、パラメータ設定に応じて、1つまたは複数の後継（または下位互換性）を有すること一般的には可能です。 ゾーンスタンプを移行に関しては少し異なります。 ゾーンスタンプの一つのタイプは、多くのゾーンのカテゴリにリンクすることができます。 それぞれのカテゴリには、ゾーンスタンプの1種類のみを使用することができます。ゾーンスタンプが移行では、カテゴリは保持されます。 以降のルートが分岐する場合、1つのカテゴリに対して、2つの異なるスタンプをリンクすることが可能です（パラメータ設定に応じて、"Zone Old" が "Zone New 1"にアップグレード、または "Zone New 2"）。 移行プロセスに関する有効な結果ですが、ARCHICADでは一貫性のない状況です。これを避けるために、直接的な方法で、ゾーンスタンプの移行をお薦めします。

一般スクリプトの問題

数値型 - 高精度

ARCHICADの9以前では、全ての数値は浮動小数点値として内部的に格納され、結果的に不正確な値でした。 これは、整数値が不正確に格納されていたことを意味します。 ARCHICAD9からは整数が使用され、GDLパラメータの型が正しく整数として内部的に格納されます。

Parameter types internally stored as an Integer:

Integer,
Boolean,
Material,
Line type,
Fillpattern,
Pencolor,
Intensity (Light)

Parameter types internally stored as a Floating-point number:

Length,
Angle,
Real,
RGB Color component (Light)

GDLの変数はまだ型定義を必要としません、値からの変換の間に決定されたタイプを変数にロードされます。 数値演算子の出力はタイプがあります。 この情報についてはGDLマニュアルを参考にしてください。

プログラマは等価演算子で整数型を比較することができます。 ARCHICAD 9から、プログラマが直接等価演算子を使用して、整数値と浮動小数点値を比較しようとすると、警告が現れます。 浮動小数点の平等、比較のための数値は、比較の精度を意味する小さなイプシロン値を使用します。 浮動小数点数と整数の平等、比較のために round_int 関数を使用します。

異なる数値型の間の等価性のテストのサンプルの方法を以下に説明します：

```
iDummy = 1 * 2
if iDummy = 2 then
   ! 有効な比較、trueの場合、これらのステートメントが実行されます
   ...
endif
```

```
dDummy = 1.5 + 0.5
if dDummy = 2 then
   ! trueとは限らない場合は、そのような比較を信用してはいけません
   ...
endif


dDummy = 1.1 * 2
if dDummy = 2.2 then
   ! trueとは限らない場合は、そのような比較を信用してはいけません
   ...
endif


! EPS = 0.0001 -> マスタースクリプト内
dDummy = 1.1 * 2
if abs (dDummy - 2.2) < EPS then
   ! 有効な比較、trueの場合、これらのステートメントが実行されます
   ...
endif


dDummy = 1.5 * 2
if round_int (dDummy) = 3 then
   ! 有効な比較、trueの場合、これらのステートメントが実行されます
   ...
endif
```

三角関数

GDLスクリプトでは、さまざまな三角関数が必要な場合があります。 以下の関数は、GDLから直接利用が可能です： cos, sin, tan, acs, asn, atn.
以下のように他の全ての機能を容易に派生することができます。

Secant Sec(X) = 1 / cos(X)
Cosecant Cosec(X) = 1 / sin(X)
Cotangent Cotan(X) = 1 / tan(X)
Inv. Sine Arcsin(X) = atn(X / Sqr(-X * X + 1))
Inv. Cosine Arccos(X) = atn(-X / sqr(-X * X + 1)) + 2 * atn(1)
Inv. Secant Arcsec(X) = atn(X / sqr(X * X - 1)) + sgn((X) -1) * 2*atn(1)
Inv. Cosecant Arccosec(X) = atn(X / sqr(X*X - 1)) + (sgn(X) - 1) * 2*atn(1)
Inv. Cotangent Arccotan(X) = atn(X) + 2 * atn(1)
Hyp. Sine HSin(X) = (exp(X) - exp(-X)) / 2
Hyp. Cosine HCos(X) = (exp(X) + exp(-X)) / 2
Hyp. Tangent HTan(X) = (exp(X) - exp(-X)) / (exp(X) + exp(-X))
Hyp. Secant HSec(X) = 2 / (exp(X) + exp(-X))
Hyp. Cosecant HCosec(X) = 2 / (exp(X) - exp(-X))
Hyp. Cotangent HCotan(X) = (exp(X) + exp(-X)) / (exp(X) - exp(-X))
Inv. Hyp. Sine HArcsin(X) = log(X + sqr(X * X + 1))
Inv. Hyp. Cosine HArccos(X) = log(X + sqr(X * X - 1))
Inv. Hyp. Tangent HArctan(X) = log((1 + X) / (1 - X)) / 2
Inv. Hyp. Secant HArcsec(X) = log((sqr(-X * X + 1) + 1) / X)
Inv. Hyp. Cosecant HArccosec(X) = log((sgn(X) * sqr(X * X + 1) +1) / X)
Inv. Hyp. Cotangent HArccotan(X) = log((X + 1) / (X - 1)) / 2

注記：

Logarithm to base N LogN(X) = log(X) / log(N)

GDL警告

他のプログラミング言語と同様に、GDLには従うべきの構文とロジックがあります。構文に誤りがある場合は、エラーメッセージが表示されます。 スクリプトの実行時に混乱または予期せぬエラーが発生した場合は、GDLの警告が送られます。

[オプション]→[作業環境]→[モデル再構築オプション]で、どこにそれらのメッセージを送るかを選択できます：

- **エラーメッセージ表示による割り込み**：問題発生時にダイアログがポップアップ表示されます
- **レポートの書き出し**：このメッセージはレポートウィンドウに表示されます

何をメッセージとして送るかを選択できます：この設定はライブラリ開発メニューの"**ライブラリ部品スクリプトの警告を確認**"にあります。 有効にされた場合、エラー以外にもWHERE設定に応じて警告がレポートされます。

警告メッセージを表示したいときを選択することもできます。この設定はライブラリ開発メニューの"**GDLメッセージを常に送信**"で行うことができます。 この機能をオンにすると、GDLが実行されるたびに、警告とエラーが強制的にレポートされます。オフにしたままの場合は、通常の警告レポートのままになります。

これらのスイッチを組み合わせて利用すると、問題が生じる可能性もありますのでご注意ください： 例えば、**"GDLメッセージを常に送信"** と**"エラーメッセージ表示による割り込み"** の両方を有効にすると、編集可能なホットスポットを移動やポップアップダイアログなどの実行を妨げる可能性があります。

現在のパラメータ設定でスクリプトの問題生じている場合は、ライブラリ部品エディタで**"スクリプトを確認"**をオンにすると、常に警告またはポップアップウィンドウでエラーメッセージを表示させます。 PRINTコマンドを使用するか、GDLデバッガを利用することで見つけにくい問題を探し当てることができます。

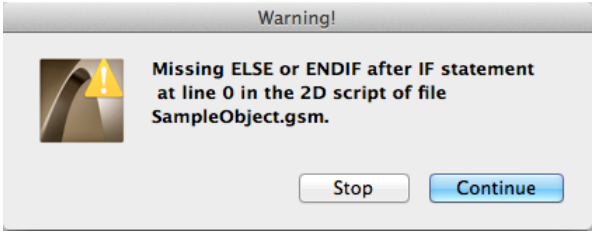
GDLの警告の行番号は、問題を含むスクリプトを参照しています。

解析エラーは、細心の注意を持って取り扱わなければなりません。 最初の行は解析が不可能になっていることを示していますが、実際の問題は前の行にある可能性もあります。

例

インタプリタは、endif で最初の不足している文を検出し、そこに停止します; 問題は4行目にあるのは明らかですが、本当は endif のが不足しているところにあります。

```
if condition1 then
    if condition2 then
        ! do something

    ! do something -'endif'が不足してます
else
    ! a potentially long code block
endif
```



開発された最新の警告メッセージの例：

警告メッセージ	可能な説明
文字列として単純なパラメータを再宣言	オブジェクト内の単純なパラメータを指定し、コールされるマクロ内の配列として使用
未定義parentId "id"がUI_PAGE定義で使用されています	タブページの階層で欠落しているparent ID
パラメータスクリプトで使用されている、ビュー/プロジェクトに依存するグローバル"globalName"	参照：「グローバル変数」
パラメータスクリプトで使用されている、"requestName"を要求	参照：「REQUESTオプション」
パラメータスクリプトで使用されている、アプリケーションクエリ "applicationQueryName"	参照：「アプリケーションクエリオプション」
おそらく不要なパラメータの型の変更	パラメータは、元の型でサポートされていない値を受け取ります

ホットスポットとHotline ID

ホットスポット/hotline/hotarc識別の目的

ARCHICADではホットスポット/hotline/hotarc識別は連動寸法をサポートするために導入しました。 この機能を使用して、寸法記入項目はGDLオブジェクトのホットスポット/hotlineのいずれかを参照することができます。 ホットスポット/hotlineの数がオブジェクトの異なるパラメータ化状態の間で変更されたとき、それは重要な問題となります。

旧ホットスポット/hotlineの問題

プログラマがホットスポット/hotline/hotarc IDを指定しない場合は - または0に設定した場合、ARCHICADが継続的に序数を増やし割り当てます。 このソリューションは、静的オブジェクトに対して正しいが、一部のホットスポット/hotlineを表示またはパラメータセットアップの間で非表示にするときの寸法の問題を引き起こします。 IDは再配置されますので変更し、断面図上のの関連寸法項目は無くなります。

正しいホットスポット/hotline/hotarc スクリプト

これら全ての理由から、オブジェクト内のホットスポット/hotlineに固定IDを割り当てる必要があります。 個別に制御可能な機能のホットスポット/hotlineのために広い間隔を確保することにより行うことができます。

それでは階段を例として見てみましょう。境界ホットスポット/hotlineは[1-100]の間隔を使用することができ、手摺りは[200～299]間隔、蹴り上げは[1000～]を使用することにします。 これは手摺りの寸法は、蹴り上げの数に変更または下部の接点が複雑になっても、壊れないことを保証します （ホットスポット/hotlineを使用することで）。

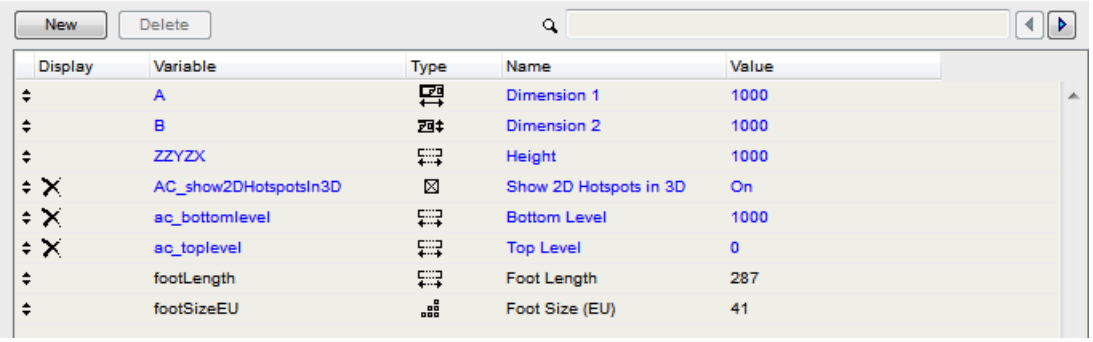
ホットスポットの有効化

ARCHICAD 8以降からライブラリ部品の編集可能なホットスポットを使用できます。 機能は、一つの可能性を除き ホットスポットベースの編集コマンド に記載されています。

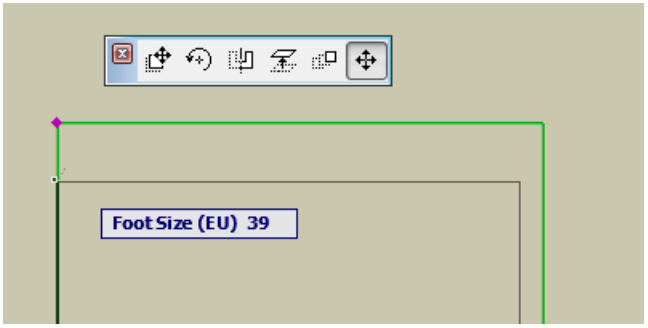
場合によってでは、編集されたものとは異なるパラメータを表示したい場合があります。下記のサンプルコードをご覧ください：

編集可能なホットスポットの例 - 下駄箱

靴のサイズをと靴のサイズをメートルで必要です。 そのために、2つのパラメータを作成し、パラメータスクリプトでそれらを接続してください。 当然のことながら、説明パラメータの型が異なります（例： テキスト）。 強調する編集パラメータは footLength footSizeEU - 表示パラメータはパラメータスクリプトを介してアップデートする必要があります。



2D編集



パラメータスクリプト

```
DIM lengthValues[10]
DIM sizeValues[10]
for i = 1 to 10
    sizeValues[i] = i + 35
    lengthValues[i] = (i + 35) * 0.007
next i


values "footLength" lengthValues
values "footSizeEU" sizeValues

if GLOB_MODPAR_NAME = "footLength" then
    parameters footSizeEU = round_int (footLength / 0.007)
else
    if GLOB_MODPAR_NAME = "footSizeEU" or GLOB_MODPAR_NAME = "" then
        parameters footLength = footSizeEU * 0.007
    endif
endif
```

2Dスクリプト

```
rect2 0, 0, footLength * 0.4, footLength   ! または現実的な靴のモデル

hotspot2 0, 0,          1, footLength, 1 + 256, footSizeEU
hotspot2 0, footLength, 2, footLength, 2, footSizeEU
hotspot2 0, -0.1,       3, footLength, 3
```

GDLの実行コンテキスト

ARCHICADはGDLオブジェクトは、それが表示されるかで使用されているコンテキストについて知ることができます。 次のグローバル変数は、この目的のために使用されます：

- GLOB_VIEW_TYPE 有効ビューを決定
- GLOB_PREVIEW_MODE 有効プレビューを決定
- GLOB_FEEDBACK_MODE 編集コンテキスト表示
- GLOB_SEO_TOOL_MODE ソリッド編集コンテキスト表示

可能な値については、「一般環境情報」と以下のリストを参照してください：

GLOB_VIEW_TYPE = 2 - 2D、平面図

モデルは標準的な2Dの平面図に表示されます。 3Dスクリプトでは、モデルが project2D コマンドで2D投影されていることを意味します。 オブジェクトの主な用途です - この2Dモデルは常に正しく効率的でなければなりません。

もし、GLOB_FEEDBACK_MODE = 1の場合、 3Dモデルは、オブジェクトのホットスポットの編集中に、3D平面図上のフィードバック線を介して表示されます。 このモデルは、ユーザーインタラクションを通じて瞬時に何度も描かれます。これは、モデルはオブジェクトの本質的な部分を表すべきであることを意味します。 注記、テキスト（text2 コマンドで生成されたテキスト）は、出力を減速するため、フィードバックモードで更新されません。

GLOB_VIEW_TYPE = 3 - 3Dビュー

3Dモデルは標準的な3Dモデルウィンドウに表示され、レンダリングにも利用されます。 このビューでは、オブジェクトの内部の詳細は見られることはないので、省略してもかまいません。 オブジェクトの2番めに大切な用途です - この3Dモデルは常に正しく効率的でなければなりません。 このターゲットタイプは、正しい外観を要求します。

もし、GLOB_FEEDBACK_MODE = 1の場合、 3Dモデルは、オブジェクトのホットスポットの編集中に、3Dモデルウィンドウ上のフィードバック線を介して表示されます。 このモデルは、ユーザーインタラクションを通じて瞬時に何度も描かれます。これは、モデルはオブジェクトの本質的、可視部分を表すべきであることを意味します。

GLOB_VIEW_TYPE = 4 - 断面図 or GLOB_VIEW_TYPE = 5 - 立面図

3Dモデルは標準的な断面図/立面図に表示されます。 これらのビューでは、オブジェクトが他の全てのビュータイプに不要な内部の詳細を生成する必要があります。

もし、GLOB_FEEDBACK_MODE = 1の場合、 3Dモデルは、オブジェクトのホットスポットの編集中に、断面/立面図ウィンドウ上のフィードバック線を介して表示されます。 このモデルは、ユーザーインタラクションを通じて瞬時に何度も描かれます。これは、モデルはオブジェクトの本質的、可視部分を表すべきであることを意味します。

GLOB_VIEW_TYPE = 6 - 3Dドキュメント

3Dモデルは標準的な不等角投影に図面として表示されます。 これはドキュメント、3D上での寸法に使われます。

GLOB_VIEW_TYPE = 7 - 詳細図面

モデルは詳細図ウィンドウで使用されます。 モデルは、結果的に他のビューよりも詳細なものができます。 スクリプトタイプから派生した情報の2Dおよび3Dモデルは区別されません。

GLOB_VIEW_TYPE = 8 - レイアウト

モデルはプリント表示のレイアウトウィンドウで使用されます。 モデルは、その印刷の外観を示すべきです。 スクリプトタイプから派生した情報の2Dおよび3Dモデルは区別されません。

もし、GLOB_FEEDBACK_MODE = 1の場合、 Dモデルは、オブジェクトのホットスポットの編集中に、レイアウトウィンドウ上のフィードバック線を介して表示されます。 このモデルは、ユーザーインタラクションを通じて瞬時に何度も描かれます。これは、モデルはオブジェクトの本質的、可視部分を表すべきであることを意味します。

GLOB_VIEW_TYPE = 9 - 計算 もしくは GLOB_PREVIEW_MODE = 2 - リスト

3Dモデルは、リストエンジンによって表面、体積計算のために使用されます。 このコンテキストは、リストのためにモデルの変更を行ための適切な場所である。 例： 塗装される表面と必要な塗料の量を上げるために、余分なボディを生成することができます。モデル生成の計算とリストで任意の結果を得るため、2つのグローバルの組み合わせを使用してください。

GLOB_PREVIEW_MODE = 1 - 設定ダイアログ

モデルは、オブジェクト設定ダイアログのプレビューボックスに表示されます。 スクリプトタイプから派生した情報の2Dおよび3Dモデルは区別されません。 オブジェクトは限られたプレビュー画面のサイズを考慮して、高速で、荒目のプレビューを提供する必要があります。

GLOB_SEO_TOOL_MODE = 1 ソリッド編集のオペレータとして作成

生成された3Dモデルはソリッドオペレーション（CSG） パラメータとして使用されます。 オブジェクトのスペースの要求は、オブジェクト自体よりも大きい場合に役立ちます。 例： スラブから階段を減算するときに、階段が歩行者のための穴をカットしていることを期待します。 これを達成するには、このコンテキストでは、階段歩行スペースを含むモデルを生成する必要があります。

ARCHICADのコミュニケーション値

ARCHICADのとライブラリ部品との間のパラメータ値フローの二つの方向があります。 1つ目の方向はARCHICADが、そのコンテキストの属性に関するライブラリの部品に通知することを意味します（例えばプロジェクトや窓が内に配置される壁の厚さのスケール）。 2つ目はライブラリ部品が、オブジェクトの直接のコンテキストの中で何かを変更するためにはARCHICADに命令自体について何かを答える時です（例： 壁をカットして時の深さ）。

ARCHICADからの情報フロー

ARCHICADのからの情報には3つのチャネルがあります： 事前に定義された名前と直接コールされた値のグローバル変数およびパラメータ。

グローバル変数

グローバル変数は、現在のプロジェクトの設定にし、オブジェクトの配置状況に応じてのARCHICADによって決まります。全てのグローバルが全てのコンテキストとビューには埋められませんので、ご注意ください。

グローバル変数の完全なリストと特定のスクリプトの関連する制限については、「グローバル変数」を参照してください。

固定名オプションパラメータ

情報を提供するためのARCHICADの新しい方法は、固定名のオプションパラメータの方法があります。 指定されたライブラリに、任意のオプションのパラメータに一致する名前と型を持つパラメータがある場合、ARCHICADはその機能に応じてその値を設定します。

ARCHICADの定義されたライブラリ部品パラメータを学習するには、「ARCHICADに設定されたパラメータ」 の「固定名パラメータ」を参考にしてください。

新規リクエストとアプリケーションクエリ

ほとんど使用されない、特別な情報については、ライブラリ部品はスクリプトの中で要求をコールしたアプリケーションクエリを使用しています。 グローバル変数とは違い、実際に実行するスクリプトが含まれている時に、戻り値を与えます。 ほとんどの場合、パラメータスクリプト内での要求とクエリ、マスタースクリプトをパラメータスクリプトとして実行することを避けるべきです。 これらのスクリプトが使用されると、返された値や関数の有効性は保証されません。

オプション、パラメータスクリプトの互換性と構文については、「REQUESTオプション」と「アプリケーションクエリオプション」を参照してください。

ライブラリ部品からの情報

ARCHICADが正しくライブラリ部品を使用するには、特定の情報を必要とします。これらの情報は、機能およびコンテキストに依存し、事前に定義された名前と機能を持つパラメータとして組み込みこまれたARCHICADのサブタイプに格納されています。組み込みこまれたARCHICADサブタイプに加え、いくつかの機能は固定名のオプションパラメータである必要がります。

組み込まれたサブタイプの固定パラメータと「ARCHICADに読み込まれたパラメータ」 の 「固定名パラメータ」を参考にしてください。

モデル表示オプション、ライブラリグローバル

図面内のライブラリ部品の表示は、現在のビューに依存してもよい。

内部モデル表示オプション

ビューの内部設定はGDLグローバル変数を介して利用可能です（例： GLOB_SCALE、GLOB_STRUCTURE_DISPLAY）、要求のオプション（例： "window_show_dim"、"door_show_dim"、"floor_plan_option"、" view_rotangle"）。

ライブラリグローバル表示オプション

ARCHICAD 13以降はライブラリから表示オプションを定義できます。 これらのオプションは、各ビューに格納され、それらはそれに応じて返されます。

次の特性/パラメータ/オプションは、ビューに依存されたライブラリグローバルに保存してください：

- 開口線の表示/非表示
- 最小スペースの表示/非表示
- 統一するために個別に変更されるべきではない、ペンおよび他のビュー属性（例： 最小スペース）
- 特定の付属品要素の表示/非表示（例： ドアノブ、ハンドル）
- オブジェクトグループの2Dシンボルタイプ設定

ビュー依存ライブラリグローバルに格納する必要がありません： プロジェクト全体における一般的な値、地域における一般的な値、それぞれのオブジェクトに設定する必要のある値。

モデル表示オプションにタブページを追加するには、ライブラリグローバル設定 から派生したライブラリ部品を作成する必要があります。（GUID: {709CC5CC-6817-4C56-A74B-BED99DDB5FFA}) サブタイプ。 このオブジェクトは、パラメータとして所望のグローバルオプションが含まれている必要があり、タブページのためのユーザーインターフェース定義を持つ必要があります。 UIの幅は既存のパネルに合わせて、600ピクセルに設定してください。UIの高さは自由に定義可能です。 接続パラメータまたはユーザーインターフェース要素のパラメータ·スクリプトを有することができます。

LIBRARYGLOBAL コマンドは、独自のライブラリのグローバル設定の値を照会し、配置可能要素に使用できる現在のビュー設定に応じてるオブジェクトです。

スクリプトタイプの特定の問題

マスタースクリプト

マスタースクリプトを記述するとき、それがARCHICAD上で各スクリプトの実行前にテストする必要があります。 これは次のことを意味します：

- パラメータ定義の配置とマスタースクリプトで複数のスクリプトによる計算に使用することは、良いアイデアです： ファイルサイズが減少し、要素が容易に変更できるようになります。
- libpartの評価にかかる不必要な時間を回避するには、ここには共通の計算のみを置いてください（マスタースクリプトはそれぞれ、全てのスクリプトの前に評価されことを忘れないでください）。
- 有効性の理由から、マスタースクリプトでパラメータバッファを使用しないでください。
- マスタースクリプト内の end コマンドを入れないでください; そうでない場合、ARCHICADは、スクリプトの残りの部分を実行しません。

2Dスクリプト

実行コンテキスト

2Dスクリプトは、2Dモデルが生成された時に実行されます：

- 2D図面
- 2D編集フィードバック
- オブジェクト設定ダイアログウィンドウの2Dプレビュー
- レイアウト図面
- レイアウト図面フィードバック

建築設計の大部分は、2Dで行われるので、このモデルは最も重要です。 ホットスポットを介して編集しているときに、正確な外観は、高速生成時間と適切な機能の要求を意味します。

一般的な推奨事項

オブジェクトが変更可能にするために、フラグメントおよびバイナリ2Dフォーマットを使用しないようにしてください。

2Dスクリプトは、2Dシンボルよりも多くのカスタマイズが可能です。 バイナリ2Dシンボルでは、湾曲塗りつぶしが正しく引き伸ばされていませんが、2Dスクリプトでこの問題に直面しません。

線および塗りつぶし特性の定義

ARCHICADの9以降はGDLから、複数のメインカテゴリの線と塗りつぶしを選ぶことができます。 線とポリゴンセグメントは輪郭、内部または一般として定義できます; 塗りつぶしは切断、表面、作図に定義できます。 これらのカテゴリはARCHICADのユーザーマニュアルに記載されています、GDLオブジェクトで使用する方法を見てみましょう。

線や塗りつぶしの正しいプロパティを設定すると、スクリプトから表示オプション依存性をなくすことができます。 以前は、設定された表示オプションに応じて、いくつかの内側の線で描画するための条件を追加する必要がありました。 内線を定義し試してみましょう。ARCHICADは表示するか、表示オプションによってコントロールされるでしょうか。
それでは、定義例を要約するウィンドウの2Dスクリプトからの抜粋を見てみましょう：

```
! ===== 下端 =====

line_property 0    ! 一般の線

! 上から見た下端 -> 表面塗りつぶし
poly2_b{2} 4, 1 + 2 * (gs_fillSillCover > 0) + 4 + 64, ...
...

! ===== 壁セグメント / 空洞クロージャ =====

line_property 1    ! inner lines
line2 ...
...

line_property 2    ! wall contours
line2 ...
...

! 壁セグメントは切断表示 -> 切断塗りつぶし
poly2_b{2} 4, 2 + 4 + 8 + 16 + 32, ...

! ===== 窓枠 =====

line_property 0    ! 一般の線

! 窓枠の側面は切断 -> 切断塗りつぶし
poly2_b{2} 4, 1 + 2 * (gs_fillFrames > 0) + 4 + 32, ...
...
```

3Dスクリプト

実行コンテキスト

3Dスクリプトは、3Dモデルが生成された時に実行されます：

- 3Dウィンドウ（ワイヤフレーム、陰線処理、ソリッドモデル）
- project2 が3Dモデルを2Dに投影で使用した時は2D図面

- 2D断面 - 詳細
- 3D編集フィードバック - 速度を最適化
- 3Dでソリッドオペレーションのための演算子 - 目的の機能のためにデザイナーに依頼
- リストのための表面と体積計算
- オブジェクト設定ダイアログウィンドウの3Dプレビュー
- project2 が3Dモデルを2Dに投影で使用した時はレイアウト図面
- レイアウト図面フィードバック

一般的な推奨事項

オブジェクトが変更可能にするために、バイナリフォーマットを使用しないようにしてください。

陰線ビュー内のオブジェクトの可視性を制御するためのステータスコードを使用してください。 曲面の輪郭線を可視化。 可能な場合、不要な線を非表示にします。

可能な限り、編集可能なホットスポットの代わりに、固定のものを定義します。

後で修正を容易にするために del top コマンドを使用しないでください。

常に3Dスクリプトの終了時にグローバル座標を復元し、オブジェクト上でさらに変更を容易にするために end コマンドでそれに従ってください。

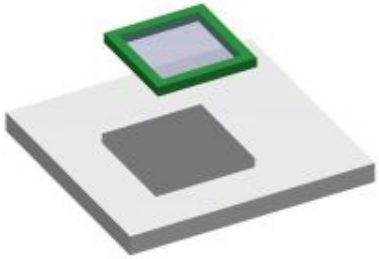
モデル透過形状

インターナルレンダリングエンジンで正しい影を作るために、オブジェクトの固体と透明部分との間にbody -1 コマンドを使用します（例： ガラリ窓）。

表7 透過形状の例

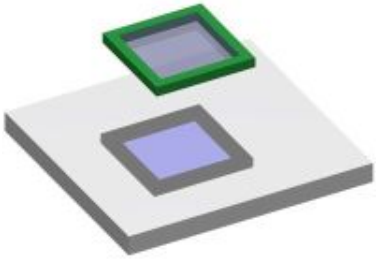
不正解

```
prism_ 10, 0.1,
   0,   0,   15,
   1,   0,   15,
   1,   1,   15,
   0,   1,   15,
   0,   0,   -1,
   0.1, 0.1, 15,
   0.9, 0.1, 15,
   0.9, 0.9, 15,
   0.1, 0.9, 15,
   0.1, 0.1, -1


material "blueglass"

prism_ 5, 0.1,
   0.1, 0.1, 15,
   0.9, 0.1, 15,
   0.9, 0.9, 15,
   0.1, 0.9, 15,
   0.1, 0.1, -1
```



正解

```
prism_ 10, 0.1,
   0,   0,   15,
   1,   0,   15,
   1,   1,   15,
   0,   1,   15,
   0,   0,   -1,
   0.1, 0.1, 15,
   0.9, 0.1, 15,
   0.9, 0.9, 15,
   0.1, 0.9, 15,
   0.1, 0.1, -1

body -1

material "blueglass"

prism_ 5, 0.1,
   0.1, 0.1, 15,
   0.9, 0.1, 15,
   0.9, 0.9, 15,
   0.1, 0.9, 15,
   0.1, 0.1, -1
```



テクスチャマッピング

テクスチャマッピングは、オブジェクト上で正しく適用された場合、必ず確認してください。 デフォルトARCHICADのテクスチャマッピング処理で、良好な結果が得られない場合は、正しい方法を設定するために coor コマンドを使用します。 下記の例をご覧ください：

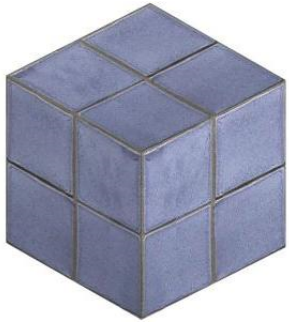
表8 ランダムに正しく整列されたタイリング。

ランダムテクスチャ

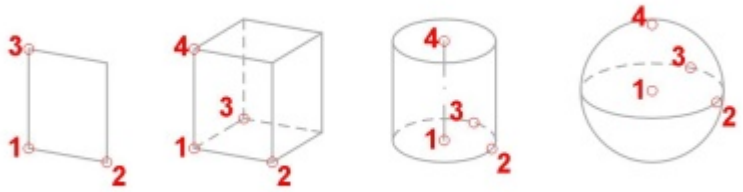
```
define texture "owntile" "T.jpg",
        1, 1, 128+256, 0

define material "tilemat" 21,
        0.7, 0.7, 1,
        0.15, 0.95, 0, 0.0,
        0, 0,
        ind (fill, ""), 1,
        ind (texture, "owntile")

material tilemat

block 1, 1, 1
```

整列テクスチャ

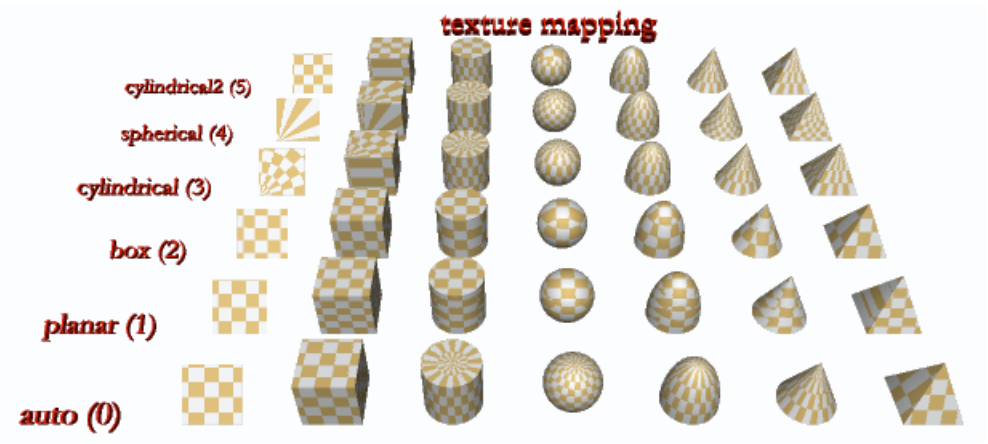
```
define texture "owntile" "T.jpg",
        1, 1, 128+256, 0

define material "tilemat" 21,
        0.7, 0.7, 1,
        0.15, 0.95, 0, 0.0,
        0, 0,
        ind (fill, ""), 1,
        ind (texture, "owntile")

material tilemat

block 1, 1, 1

base
vert 0, 0, 0
vert 1, 0, 0
vert 0, 1, 0
vert 0, 0, 1

coor 2 + 256, -1, -2, -3, -4
```



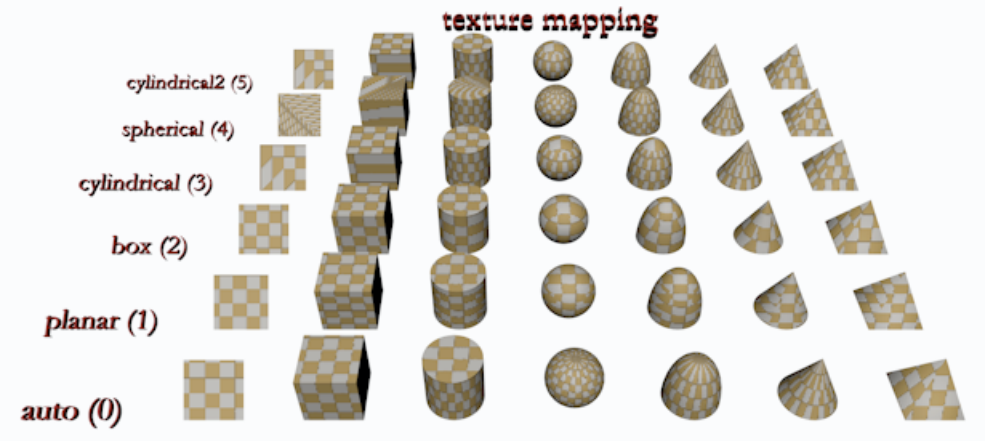
一般的には、別の形状は、body -1 コマンドを使用した、異なるテクスチャ座標系が必要です。
異なるテクスチャマッピングモードを使用するときは、vert または teveコマンドを使用して、正しい軸の定義の行う必要があります。
ノードの順序を以下に示します。



vert または teve コマンドによって定義されたノード間の異なる距離を設定することにより、テクスチャを歪ませます。
別のレンダリングエンジンを利用した場合、わずかに異なる結果になることがありますので注意してください、以下はその例です。
インターナルエンジン：



C4Dエンジン：



複雑な表面または歪んだテクスチャの正しいテクスチャマッピングは coor および teve コマンドでモデル化することができます。この方法では、表面モデルのみを作成することができます。 ARCHICADには直接的なテクスチャ仕様は存在しません。材質定義の一部としてテクスチャを定義することができます。 このテクスチャはレンダリングエンジンとOpenGLで使用されています、しかしOpenGLでは限られたテクスチャマッピングが実装がされています、標準3Dエンジンではテクスチャマッピングはありません。

TEVE コマンドでは、平面ポイント（u、v）、空間的な幾何学的な点（x、y、z）をマッピングすることができます：

- （x, y, z） ローカル座標系でメートルで測定されています。
- （u, v） は無限テクスチャ空間内の単位で測定される。1単位は、その方向におけるテクスチャの程度と同じくらいの長さです。

負の値または1以上の値を u または v に与えることができます。

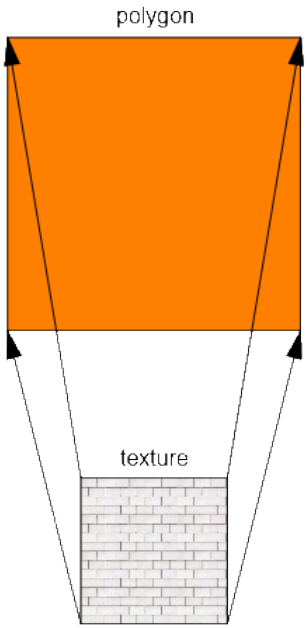
例1を参照してください：

表9 Teve 例1：歪みのないマッピング

プログラム

```
base

teve  0, 0, 1,    0, 0
teve  2, 0, 1,    1, 0
teve  0, 2, 1,    0, 1
teve  2, 2, 1,    1, 1
teve  0, 0, 1,    1, 1

edge  1, 2, -1, -1, 0
edge  2, 4, -1, -1, 0
edge  4, 3, -1, -1, 0
edge  3, 1, -1, -1, 0

set material 92

pgon 4, 0, 0, 1, 2, 3, 4
coor 1024, 1, 2, 3, -5

body -1
```

論理



結果

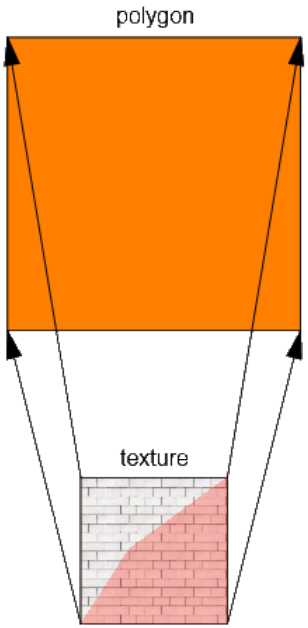
非正規のマッピングを行う場合は、レンダリングエンジンは、モデル空間内の形状にテクスチャ空間における形状に適合します：

表10 Teve 例1：歪みのあるマッピング

プログラム

```
base

teve  0, 0, 1,    0,    0
teve  2, 0, 1,    1,    0
teve  0, 2, 1,    0.3,  0.5
teve  2, 2, 1,    1,    1
teve  0, 0, 1,    1,    1

edge  1, 2, -1, -1, 0
edge  2, 4, -1, -1, 0
edge  4, 3, -1, -1, 0
edge  3, 1, -1, -1, 0

set material 92

pgon 4, 0, 0, 1, 2, 3, 4
coor 1024, 1, 2, 3, -5

body -1
```

論理



結果

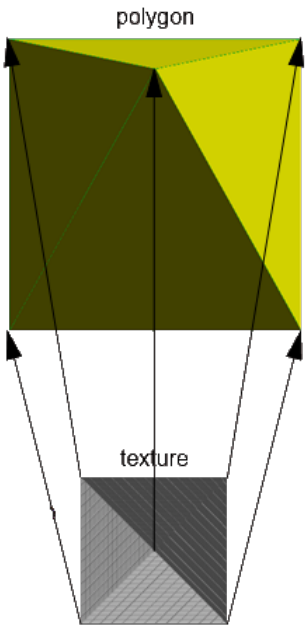
この例でわかるように同じことが、本当の3Dボディにも当てはまります：

表11 Teve 例1：ピラミッド上の歪みのあるマッピング

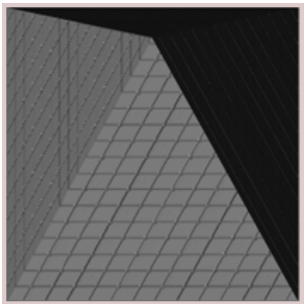
プログラム

```
base

teve  0, 0, 1,    0,    0 ! 1
teve  2, 0, 1,    2,    0 ! 2
teve  2, 2, 1,    2,    2 ! 3
teve  2, 2, 1,    0,    2 ! 4
teve  1, 1, 3,    1,    1 ! 5

edge  1, 2, -1, -1, 0
edge  2, 4, -1, -1, 0
edge  4, 3, -1, -1, 0
edge  3, 1, -1, -1, 0

set material 92

pgon 3, 0, 0, 1, 6, -5
coor 1024, -6, -7, -8, -9

body -1
```

論理



結果



モデルの頂点に対して1つだけテクスチャの頂点を割り当てることができることに注意してください。 テクスチャの頂点をポリゴンごとに割り当てることはできません。 それは利点であり、欠点でもあります。

画像要素

モデルの複雑な部品を単一の画像に置き換えることも可能です。 この方法は木や低木のために使用することができます。

そのファイル名で呼ばれる外部の画像を使用して、ファイル拡張子を省略しないでください。

pictureコマンドを使用して3Dモデル内の画像を配置すると、ポリゴンが表面の画像を使用して作成されます。 ポリゴンの材質は、レンダリングの結果に影響します。 これを考慮すると、マット面を使用する必要があります - 色は画像に応じて選択することができます。

```
define material "pictmat" 2,
        1, 1, 1             ! RGB

material "pictmat"

picture "filename.extension", a, b, mask
```

最初の画像は、光沢面上の画像を示している - 望ましくない副作用を観察できます。 希望の結果 - 次の写真では、正確に設定された材質のテクスチャを見ることができます

表12 透過画像

光沢面	マット面
	

透明画像の場合 - 上記の木のようにベース材質より正確な定義を考慮する必要があります。例を参照してください：

```
define material "pictmat" 0,
        1, 1, 1,            ! RGB
        0.5, 0.8, 0, 0,
        0, 0,
        0, 0, 0,
        0, 0, 0,
        0

material "pictmat"

picture "filename", a, b, mask
```

グループ操作：

グループ操作はGDLでソリッド編集を可能にします。 間違った使い方をすると、逆にリスクが生まれます。
重要な点は、別の物の中にグループを配置してはならないということです。 このような状況では、下のソーススニペットのように新しいグループを定義する必要があります：

subtractionResult = subgroup ("sub_operand_1", "sub_operand_2")

パラメータスクリプト

実行コンテキスト

パラメータスクリプトは次のケースで実行されます：

- オブジェクト設定ダイアログウィンドウを開く
- オブジェクト設定ダイアログウィンドウでパラメータの値を変更
- 編集可能ホットスポットを使用してパラメータの値を変更（フィードバックを発生中も）
- 従来のホットスポットを使用してオブジェクトをストレッチ
- 移行ライブラリをロード（AC18より）

パラメータスクリプトは実行されるかもしれません：

- SYMB_POS_X/SYMB_POS_Yにオブジェクトが参照された場合、オブジェクトをドラッグ
- ゾーンを更新 は必要に応じて、影響を受けたゾーンのパラメータスクリプトを実行します

パラメータスクリプトは実行されません：

- 再構築
- スケールの変更
- フロアの変更

複数選択を編集すると、意図しないパラメータ値も変更されることがあります。

パラメータスクリプトは、単一のユーザーインタラクションに複数回実行することができることに注意してください。 理由としては、パラメータスクリプトはパラメータの値を変更することができることであり、パラメータスクリプトは再度実行することが必要だからです。 したがって、それはあなたが実行のカーディナリティを予測することはできないかもしれないので、パラメータのスクリプトで一つのパラメータ値を増加させるのは意味がありません。

パラメータスクリプトの実行はリニアで、必ずしも複数のではありません。 パラメータスクリプトの開始が1回のみになるよう強制することができます。**[パラメータスクリプトを一度だけ実行]** オプション（オブジェクトの[互換性オプション]パネルにあります）をオンにします。複数回行う必要がないことが確実な場合に実行してください。 これでオブジェクトの反応が早くなり、時間とリソースの節約につながります。

一般的な推奨事項

パラメータスクリプト内のパラメータを制御する場合は、追加のパラメータの順序に従うようにしてみてください。

次を利用して、パラメータ間の関係を定義することができます： GLOB_MODPAR_NAME 値（最終更新パラメータの名前を含む）。半径と直径の両方を設定可能な円オブジェクトを作成できます（1つがパラメータリストを介して、もう一つが編集可能ホットスポット）。 パラメータの有効範囲を定義するには、このオプションを使用しないでください - 代わりに values コマンドを使用してください。

values コマンドを使用して、全てのパラメータの有効な値の範囲を定義します。
parameters コマンドを使用して、パラメータ、スクリプト内の特定の状態にあるパラメータの値をリセットする場合は、同じようなステートメントは、マスタースクリプトに入れておく必要があります。 パラメータスクリプトがシステムによって実行されていない場合に、オブジェクトの正しい表示を保ちます。例：

```
! parameter script
if bCondition then
    yy = 1
    parameters yy = yy
endif

! master script
if bCondition then yy = 1
```

フォントタイプ名

文字列パラメータが必要な場合は、stFont 命名、テキストのフォントの種類を設定するために、プラットフォームに依存しない解決法で行うには、次の値リストの定義を使用します。

```
DIM fontNames[]
request ("FONTNAMES_LIST", "", fontNames)
values "stFont" fontNames, CUSTOM
```

ARCHICAD_Library_Master.gsm オブジェクトでこれを行う場合は、同じ"stFont"パラメータを使用して、ロードされた全てのライブラリ部品は自動的に同じ値リストを受け取ります。
欠落または予期しないフォントの種類を扱いに、CUSTOM値は必要です。

配列パラメータに制限を設定

配列パラメータは、均一なデータのために使用されるべきです。例： 全ての配列要素は関連する項目です。
配列パラメータの全てのコンポーネントを制限するためのサンプルコードスニペット gridXPosition 範囲 [1, 5] 、UIでの使い方：

```
! parameter script
values "gridXPosition" range [1, 5]

! UI script
for i = 1 to nGridLines     ! nGridLines: number of lines in the array parameter
    ui_infield{3}   gridXPosition[i], xPos, yPos, infieldWidth, infieldHeight

    yPos = yPos + diffY
next i
```

ユーザーインターフェイススクリプト

実行コンテキスト

ユーザーインターフェイススクリプトは1つのコンテキストにのみ表示されます： オブジェクト設定ダイアログウィンドウのユーザーインターフェイスタブページ。

このスクリプトは、ダイアログウィンドウの初期化時に、各ユーザーインタラクションやパラメータの変更後に実行されます。

一般的な推奨事項

パラメータリストに代わるデフォルトとして、カスタム設定ページを上部のUIセレクタに表示したい場合は、 **デフォルトとして設定** ボタンを押してください（またはXMLの「StatBits」セクションに「STBit_UIDefault」ビットを追加してください）。 それ以外の場合はパラメータリストは、最初のタブになります。階層ページの場合、GDLエディタ/ UIウィンドウで **階層ページ** ボタンを押してください （またはXMLの「StatBits」セクションに「STBit_UIUseHierarchicalPages」ビットを追加してください）。

最小文字の文字にはスタイリングが適応されないので、注意してください。 アウトライン および シャドウ のスタイルは、Windowsプラットフォームには影響しません。

ARCHICADは、オペレーティングシステムとのダイアログで使用されているフォントに一致しようとします。 Windows上でグラフィカルUIをスクリプトすると、テキストの周りに多くのスペースを残します。MacOSの場合は、テキストが切り捨てられることがあります。

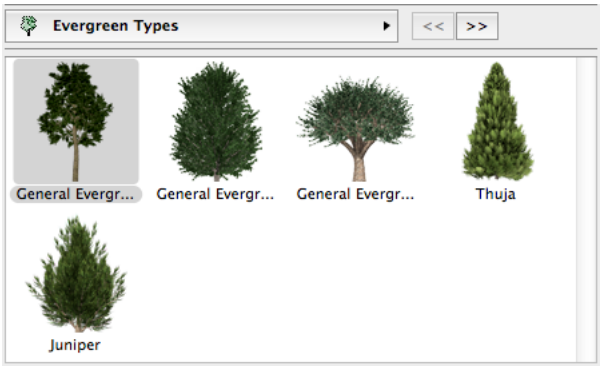
サムネイルコントロール画像

ui_infield コマンドを使用して値リストのサムネイルビューフィールドを定義する場合には、次のことに注意してください。 全てのパラメータ値のため、同じサイズのサムネイルを使用すべきです（空の値を含む）。 サムネイルは同じサイズである必要があります、そうでない場合は画像が歪んでしまいます。 1つの画像ファイルにサムネイルを組み立てるために、ARCHICADの図形ツールを使用することをお勧めします。

表13 画像のInfield

入力画像

出力画像



1つのオブジェクトのみで使用されているユーザーインターフェイス画像は、ライブラリ部品ファイル自体に統合されるべきです。LP_XMLConverter tool を利用することで可能です。

そのファイル名によって参照される外部の画像を使用する場合は、ファイルの拡張子を省略しないでください。 これで、同じ名前を持つ画像やオブジェクトから生じるエラーを回避します。

オブジェクト自体のマクロフォルダにインターフェーススクリプトで使用される、または埋め込まれた全ての画像を保管してください。外部画像を使用： file_dependence コマンドを追加し、オブジェクトとアーカイブ形式で保存されていることを確認します。

タブページの取り扱い

ARCHICAD18から、新しい階層ページングオプションが、タブページ選択で利用可能になりました。 これはオブジェクトにパラメータを追加して、「UI_PAGE」、**階層ページ** パラメータを介してアクセスされます。 個別のポップアップタブページコントロールは、カスタムUIフィールドの上に表示されます。 使用可能なページの順序と階層は、ページのIDによって定義することができます。ルートIDは常に-1です。 UIページで「旧式」タブページセレクタを設定して、利用することもまだ可能です。

それでは、スクリプト例を見てみましょう：

```
! Master Script

! TabIDs
TABID_ROOT      = -1
TABID_PAGE_1    = 50
TABID_PAGE_2    = 60

dim uiUsedPageIDs[][2]
dim uiUsedPageNames[][2]


idxPage = 1

uiUsedPageNames[idxPage][1] = "PageName_1"
uiUsedPageNames[idxPage][2] = "pageIconName_1.png"

uiUsedPageIDs[idxPage][1]   = TABID_PAGE_1
uiUsedPageIDs[idxPage][2]   = TABID_ROOT    ! Parent Page ID

idxPage = idxPage + 1

uiUsedPageNames[idxPage][1] = "PageName_2"
uiUsedPageNames[idxPage][2] = "pageIconName_2.png"

uiUsedPageIDs[idxPage][1]   = TABID_PAGE_2
uiUsedPageIDs[idxPage][2]   = TABID_PAGE_1  ! Parent Page ID



file_dependence "pageIconName_1.png"
file_dependence "pageIconName_2.png"
file_dependence "pageIconName_3.png"


! Parameter Script

dim pageValues[]
for i = 1 to vardim1(uiUsedPageIDs)
    pageValues[i]= uiUsedPageIDs[i][1]
next i

values "gs_ui_current_page" pageValues
```

```
! UI Script

ui_dialog "Custom Settings Title"
ui_current_page gs_ui_current_page

for i = 1 to vardim1(uiUsedPageIDs)
    if uiUsedPageIDs[i][1] = TABID_PAGE_1 then
        ui_page uiUsedPageIDs[i][1],    uiUsedPageIDs[i][2],
                uiUsedPageNames[i][1],  uiUsedPageNames[i][2]
        if gs_ui_current_page = TABID_PAGE_1 then
            gosub "pageSubroutinTitle_1"
        endif
    endif

    if uiUsedPageIDs[i][1] = TABID_PAGE_2 then
        ui_page uiUsedPageIDs[i][1],    uiUsedPageIDs[i][2],
                uiUsedPageNames[i][1],  uiUsedPageNames[i][2]
        if gs_ui_current_page = TABID_PAGE_2 then
            gosub "pageSubroutinTitle_2"
        endif
    endif
next i



! ======================================================================
! Call User Interface Macro's TabPages
! ======================================================================

call "ui_customMacro" parameters all    uiUsedPageIDs   = uiUsedPageIDs,
                                        uiUsedPageNames = uiUsedPageNames

! ======================================================================
end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end
! ======================================================================
```

```
! ==========================================================================
! UI Page Subroutines
! ==========================================================================

"pageSubroutinTitle_1":
    ! UI Page 1 description
return


"pageSubroutinTitle_2":
    ! UI Page 2 description
return
```

動的項目のサムネイルコントロール

ARCHICAD 10以降は、新しい動的方法でリンクのコントロール項目と値リスト項目が使用できます。 この方法を使用すると、パラメータスクリプトへのパラメータ値の有効性の論理をローカライズすることができます - コントロールが使用可能な値のセットを採用します。 この動的リンクはui_infield{3}および、ui_infield{4}で使用できます。 旧式の静的なリンクは、静的な機能で動作します（ui_infieldおよび、ui_infield{2}を使用）。

動的方法の2つのコンポーネント：

1. 全ての可能な値のオプションを使用してユーザーインターフェースコントロールを定義します。

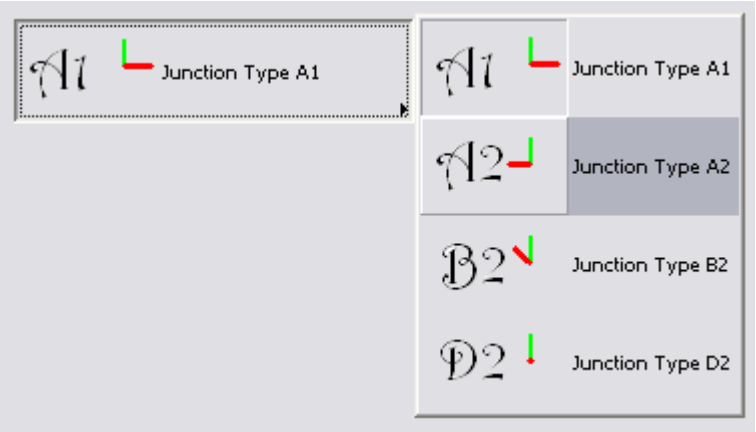
例では、2行4列を含むインデックス画像を使用して、ポップアップメニューコントロール（メソッド=2）を示します。サンプルコントロールは、8つの可能な値をサポートします。

```
ui_infield{3} iJunctionType, xColumn1-10, 44, 200, 50,
    2, 3, 8, 2,
    70, 45, 70, 45,
    1, `Junction Type A1`, 2,
    2, `Junction Type B1`, 4,
    3, `Junction Type C1`, 1,
    4, `Junction Type D1`, 3,
    5, `Junction Type A2`, 5,
    6, `Junction Type B2`, 7,
    7, `Junction Type C2`, 6,
    8, `Junction Type D2`, 8
```

2. 与えられた状況下でのパラメータの使用可能な値のリストを設定します。

```
if iLeftNeighbour = 1 then
    values "iJunctionType" 1, 3, 4, 6
else
    if iRightNeighbour = 1 then
        values "iJunctionType" 2, 5, 7, 8
    else
        values "iJunctionType" 1, 5, 7
    endif
endif
```

コントロールの結果は下の画像に表示されています。（iLeftNeighbour = 0, iRightNeighbour = 1）



透過UI画像

ARCHICAD 10では新しい方法で、アルファ層を含む透明の画像を処理が導入されました。 次のコントロールが正しくアルファレイヤーの写真を処理します：

- ui_pict
- ui_infield{3}, method = 1（サムネイルビューコントロール）
- ui_infield{3}, method = 2（アイコンとテキスト付きポップアップ）
- ui_infield{3}, method = 3（アイコン付きポップアップ）
- ui_infield{3}, method = 4（アイコンラジオプッシュボタン）
- ui_infield{4}, method = 1（サムネイルビューコントロール）

- ui_infield{4}, method = 2（アイコンとテキスト付きポップアップ）
- ui_infield{4}, method = 3（アイコン付きポップアップ）
- ui_infield{4}, method = 4（アイコンラジオプッシュボタン）

UIのフォントサイズ

静的テキストを使用する場合は、次の点に注意してください（ui_style コマンドとの組み合わせ）。

そのため、対象となるオペレーティングシステムの違いにより、WindowsとMacではフォントサイズに違いが生じます。 副作用として、最小文字 フォントサイズでは、Windows上の 小文字 よりも少し大きくなります。 一般的なルールとして、常に両方のプラットフォーム上のユーザーインターフェースの、オーバーラップとクリッピングの確認テストをしてください。

さらに、太字、斜体、および下線のような特別なスタイルは、最小文字 サイズとの組み合わせでは使用できません。 アウトラインとシャドウはもう使用されなくなった、古いMacintoshのスタイルです。

2つの画像は、異なるサイズやスタイルの静的なテキストの外観を示しています。

Windows上：

	small	xsmall	large
normal	ui_style 0,0	**ui_style 1,0**	ui_style 2,0
bold	**ui_style 0,1**	ui_style 1,1	**ui_style 2,1**
italic	*ui_style 0,2*	**ui_style 1,2**	*ui_style 2,2*
underline	ui_style 0,4	**ui_style 1,4**	ui_style 2,4

Mac上：

	small	xsmall	large
normal	ui_style 0,0	ui_style 1,0	ui_style 2,0
bold	**ui_style 0,1**	ui_style 1,1	**ui_style 2,1**
italic	*ui_style 0,2*	ui_style 1,2	*ui_style 2,2*
underline	ui_style 0,4	ui_style 1,4	ui_style 2,4

上位移行スクリプト

実行コンテキスト

ARCHICADの以前のバージョンで保存され、プロジェクトが更新され、それ以降のバージョン/更新ライブラリ（ARCHICAD 15以降）で開いたときに上位移行スクリプトが実行されます。この新しいライブラリは、手動またはライブラリマネージャで統合オプションを使用してロードすることができます。 オブジェクトの配置されたインスタンスに対して、新規、変更されたメインID,有効な上位移行スクリプトが新しいライブラリにある場合、自動的にARCHICADが置き変えます。 スクリプトの実行が成功した場合、古い要素を新しいものに交換します。

このスクリプトは、機能が失われたり、外観に大きな変化をすることなく、古いものに基づいて新しいパラメータを設定することを目的としています。

一般的な推奨事項

スクリプトの最初の行には「actualGuid」変数にFROM_GUIDグローバル変数（移行する元のオブジェクトの主要なIDが含まれています） を埋めます。 メンテナンス性を確保するために、次の構造を使用してください。

スクリプトの残りの部分は、GUIDの変更に対して一つ一つサブルーチンコールに分割されます。 全てのブロックは、オブジェクトの移行テーブルの対応する行と下位移行スクリプト内のブロックを持つ必要があります。 メインIDの最新の変更は常にこのスクリプトの最後の呼び出しでなければなりません。 各ブロックのIDを（_startID）で開始し、（_endID）で終わるものは、サブルーチンの移行ロジックで定義します。ブロックの終わりでは新しい"_endID"を常に"actualGuid"変数にたいして設定します（または空のIDを設定、アップグレードプロセスがオブジェクトの前のブロックのバージョンで停止することを意味します）。例 ：

```
actualGUID = FROM_GUID

! ==============================================================================
! Subroutines
! ==============================================================================

    _startID    = "AAAA-AAAA-...AAA"
    _endID      = "BBBB-BBBB-...BBB"
gosub "migrationstepname_FWM"

! ==============================================================================
! Set Migration GUID
! ==============================================================================

setmigrationguid actualGUID

! ==============================================================================
end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end
! ==============================================================================

! ==============================================================================
! migrationstepname
! ==============================================================================
"migrationstepname_FWM":
    if actualGuid = _startID then
        newParameter = oldParameter
        parameters newParameter = newParameter
        actualGuid = _endID
    endif
return
```

下位移行スクリプト

実行コンテキスト

ARCHICADの以前のバージョンにプロジェクトを保存するときに、下位移行スクリプトが実行されます。 ライブラリ部品の現在のバージョンが、以前のバージョンの同等より異なるメインIDを有する場合、オブジェクトの移行スクリプトが評価されます。 結果として、可能であれば、libpartがダウングレードされるか（項目の主な機能には影響しません）、完全に失われます（この場合、以前のバージョンでは"欠落"の点が表示されます）。 現在のバージョンで導入された新しい機能セットは、以前のバージョンに比べて大きな変化を表している場合、後者が起こります。

成功した下位移行プロセスは、主要な機能ロスや外観に変化を避けるための方法で、オブジェクトのパラメータを変換します。

一般的な推奨事項

スクリプトの最初の行は、継続制御変数を有効に設定します。メンテナンス性を確保するために、次の構造を使用してください。

スクリプトの残りの部分は、サブルーチンコールに分割されます。メインIDの一つ変化が1サブルーチンです。 全てのサブルーチンは、オブジェクトの移行テーブルの対応する行と上位移行スクリプト内の関連するサブルーチンを持つ必要があります。 メインIDを変更し、**最新のステップバック**では、このスクリプトの**最初のサブルーチン** である必要があります。

各サブルーチンの開始で、ターゲットGUIDがチェックされます。空でない場合、スクリプトは呼ばれた順序で実行します。 下位移行は1つ前のバージョンまで可能なので、ターゲットGUIDは一度だけ設定する必要があります（以前のバージョンのオブジェクトを分離するために、移行を分ける場合を除く）。

サブルーチンの最後に、「targetGuid」変数にデスティネーションIDを設定します。 変数に空のIDを設定した場合、ダウングレードプロセスを取り消すことがあります。 「targetGuid」はTO_GUIDグローバル変数（変換中のターゲット要素のメインIDを含む）を、移行プロセスの最初の部分に一致する場合に完了します。

タイトルごとまたはサブルーチンの移行ステップの簡単な説明を追加することを強くお勧めします。 サブルーチンの上位移行スクリプトペアに対して同じタイトルを使用する必要があります。

オブジェクトが退化するステージに達した場合、次を使用して配置されたオブジェクトのIDを設定する必要があります：
setmigrationguid
移行で空のIDを返す場合は、要素は以前のバージョンで開いているプロジェクトから失われます。

```
targetGUID = TO_GUID

! ==============================================================================
! Subroutines
! ==============================================================================

gosub "migrationstepname_BWM"

! ==============================================================================
! Set Migration GUID
! ==============================================================================

setmigrationguid targetGUID

! ==============================================================================
end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end ! end
! ==============================================================================


! ==============================================================================
! migrationstepname
! ==============================================================================
"migrationstepname _BWM":
    if targetGUID # "" then
        bMigrationSuccess = 1
        if bMigrationSuccess = 1 then
            oldParameter = newParameter
            parameters oldParameter = oldParameter
        else
            targetGuid = ""
        endif
    endif
return
```

移行テーブル

オブジェクトのメインIDを変更するたびに、要素の移行テーブルに古いIDを記入する必要があります。 各行に前のIDとのARCHICADバージョン番号が含まれます（または、2つのバージョンで1回以上変更された場合は0）。 上位移行中に、プログラムは、移行プロセスで使用可能な要素を事前に選択し、IDリストをスキャンします。 下位移行中に、このリストをスキャンするプログラムは、以前のARCHICADバージョンと同等のバージョンのものを選択します。 このテーブルの各行は、少なくとも一つの上位移行スクリプトの対応するサブルーチンと下位移行スクリプトを持っている必要があります。

マクロの作成

頻繁に使用される機能を収集し、マクロにしてみてください。 多くのオブジェクトからマクロオブジェクトを呼び出すと、ライブラリのサイズが削減し、冗長性を減らすことにより健全なオブジェクトになります。

しかし、以前の抽象的なレベルに小さな機能のマクロを作成しないようにしてください。 例えば1mx1mx1mのブロックを生成するために"block_1x1x1"マクロを作成しないでください。 これは不必要なマクロのコール数が増加し、それが透明性を悪化させる可能性があります。

マクロとして.gdlは使用しないでください、マクロオブジェクトを使用してください。

マクロを呼び出すと、常にcall キーワードを使用して、クォーテションマーク間のマクロの名を入れてください（例：call "m_rail_wired"）。 マクロ名がアーカイブファイルから欠落しているマクロを回避するために、パラメータであるマクロ呼び出しを作成しないでください。 ARCHICADはデフォルトのマクロのみをアーカイブファイルに保存します。 （ワークアラウンド：end ステートメントの後にマクロとして全てのパラメータ値を呼び出します）

パラメータバッファを使用した時は注意してください。 使用するには、そのコンテンツをスクリプトの最初に保存してください。 唯一定義された値（リターン値）が、スクリプトの終わりまでに、バッファにあることを確認してください。

マクロのリターンパラメータ

ARCHICAD 10から、マクロは呼び出し側オブジェクトにパラメータを返すことができます。コールする側では、次のパラメータを使用して、戻り値を集めることができます： returned_parameters キーワードが変数リストに続きます。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で指定された、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、全ての後の値はその中に格納されます。

マクロオブジェクトでは、end および exit コマンドは呼び出し元のオブジェクトに戻りする必要な値を定義します。下の例を参照してください。

Advanced "parameters all"

ARCHICADの10から parameters all キーワードが、マクロに渡す追加のパラメータを指定することができます。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。マクロは、この場合にもパラメータを返すことができます。

速いマクロの呼び出し

呼び出し元オブジェクトとマクロの間で転送するパラメータ値の速度はARCHICADの10で改善されました。マクロ呼び出しの速度の向上の利用に関するヒントをご覧ください 「速度の問題」。

マクロ呼び出しの例

呼び出し側オブジェクトのスクリプト。

```
call "myMacro" parameters all extraParam = 1
call "myMacro" parameters returned_parameters realWidth
call "myMacro" parameters all extraParam = 1 returned_parameters realWidth
call "myMacro" parameters all returned_parameters realWidth
```

マクロのスクリプト。

```
realWidth = 2
end realWidth
```

下位変換の問題

ARCHICAD 19以降から、3D関連のビュー、視点を開くために必要な全ての計算は、バックグラウンド処理として実行されます。

サポートされたビューポイント：

- 3Dウィンドウ
- 断面図
- 立面図
- 展開図　（"境界面積"または"ゾーンを認識して適合"が有効の場合以外）
- 3Dドキュメント

バックグラウンド処理が成功した場合、ビューを開くには数秒で行うことができます。 **しかし、プランファイルに非スレッドセーフなライブラリ部品またはオブジェクトが存在する場合は、バックグラウンドの計算を無効にすることができます：**

- ゾーン
- テキストエンジンオペレーションを含むオブジェクト（スタイルとスタイル定義のコマンドで設定されている以外）
- 次の要求を使用しているオブジェクト："CUSTOM_AUTO_LABEL"、"ZONE_COLUS_AREA", "MATCHING_PROPERTIES", "ASSOCEL_PROPERTIES" "STYLE_INFO"、"TEXTBLOCK_INFO"、"FONTNAMES_LIST"
- 変数マクロ、要求、非スレッドセーフのマクロを使用するオブジェクト。Project2コマンドまたはシンボル塗りつぶし定義は、非スレッドセーフのマクロコールとしてカウントされます。

GDLアドオンとバックグラウンド処理はアドオンそのものに関係します。

決定的アドオン（バックグラウンド処理に影響しません）：

- ポリゴン操作
- Propertyアドオン
- 読み取り専用モードで使用された場合で、有効なライブラリ内のロードされたファイル：テキストまたはデータ I/O アドオン、XMLアドオン

非決定的アドオン（バックグラウンド処理を無効）：

- DateTimeアドオン

- FileManagerアドオン
- 読み取り専用モードで使用されない場合で、有効なライブラリ内のロードされていないファイル：テキストまたはデータ I/O アドオン、XMLアドオン

オブジェクトスクリプトは静的に検査され、ライブラリ部品が現在の設定でobstacle機能自身が実行されなくても、バックグラウンド変換は無効になります。

バックグラウンド処理でロードされたライブラリ部品の互換性を確認するには、ライブラリ開発メニューの"ライブラリ部品のスレッドセーフを確認"コマンドを利用します。

速度の問題

図面の生成が遅くなるので、project2 コマンドを使用しないようにしてください。

より高速な3D再生を行うために最小限にモデル内の面の数を減らします。 RESOL、TOLERおよびRADIUS をコマンドを曲面の分割のコントロールに使用します。

クローズド形状はオープン形状に比べて、3Dで高速に生成されます（例： シリンダーは開放管よりも高速です）。

マスタースクリプトのスクリプトは、各スクリプトタイプの前に実行されていることを考慮してますので、ここでスクリプト特定の計算を入れないでください。 これは複数のスクリプトが必要とする一般的な計算のための場所です。

ドアや窓のスクリプトでは不必要な切断は避けてください （wallhole および wallniche）。

合理的な場合は、整数値とオペレーションを使用しください。これらは、浮動小数点演算よりもはるかに高速です。

文字列操作を使用して最小限に抑えるようにしてください。

マクロを呼び出す場合、call コマンドの後にパラメータの順番で使用してください。call "myMacro" parameters all マクロのパラメータの順番と呼び出しオブジェクトが似ている場合は高速です。 マクロの呼び出しで文字列型のパラメータを転送しないようにしてください。可能な限り数値型を使用してください。

Windows-Macintosh互換性

GDLオブジェクトやライブラリはプラットフォームに依存しないことがGRAPHISOFTによって検討されていますが、手動でWindowsからMacintoshにオブジェクトを移動すると、次のような問題が発生します：

- オブジェクトおよび、一覧表テンプレートで、Windowsフォントがデフォルトの Macintoshフォントに置き換えられます、逆も同様。
- テキストタイプのリストファイル（listset.txt、listkey.txt、list templates、その他）は、改行を失う可能性があるので動作しません（通常は非utf-8コードテキスト）。

バイナリライブラリとプラットフォームの変更

上記の問題を回避するには、一番目のプラットフォーム上でライブラリの.plaアーカイブファイルを保存し、2番目のプラットフォームで抽出します。このやり方で非utf-8ファイルは正しく変換されます。

GDLでの画像とHDPIのサポート

ARCHICAD 21からHDPIサポートが実際に導入され、OS Xデバイスで使用することができます。この機能では、スケーラブルベクターグラフィックス（.svg）ソース画像を使用して、解像度の異なる同じ画像のバージョンを含む多重表現.tiff画像を作成します。ARCHICADの実行中に、使用可能な100%、150%、200%の設定から現在の表示デバイスに適した画像解像度が決定されます。 Windowsプラットフォームでは、100%がデフォルトの解像度です。ただし、.svgファイルから作成された画像は、Windowsプラットフォームでは若干見た目が異なります。

このオプションは、LP_XMLConverterツールからのみ使用することができます。ライブラリ変換中に、.svgソース画像ファイルが自動的に.tiff画像に変換されます。.gsmオブジェクトは.svg画像を処理することができません。そのため、スクリプトの画像名を参照する文字列に.tiff拡張子が含まれているか確認してください（または拡張子を省略してください）。

全ての解像度でベクター画像を必ずテストして（.tiff画像は画像エディタで解像度別に確認することができます）、スケーリング後に画像が不鮮明にならないようにしてください（ソース.svgのフルピクセルに可能なかぎり揃えられた線を使用します。.png画像でも同様です）。

GDLPict、Picture、InfoPict .xmlセクションの.svg**標準画像**（画像がバイナリライブラリ部品自体にコンパイルされる場合）の構文要件は、次のとおりです：

- MIME属性は"image/svg"。MIMEタイプとソース画像の拡張子が異なる場合、警告が発生して変換が終了します（MIME is "image/svg", but the image file's extension is not svg）。
- SectionFlags属性は"1"。このフラグは.tiff変換を開始します。画像が.svgでフラグが異なる場合、変換中に警告が表示されます（SectionFlags should be "1" in case of an svg image）。

GDLPict、Picture、InfoPict .xmlセクションを参照するその他の画像は、変更されません。ソース画像の拡張子がパスに正しく指定されているか確認してください。

オブジェクトスクリプト内で名前によって直接参照される**非標準画像**は、LP_XMLConverterツールで同様に処理されます：

- .svg画像は、_images フォルダではなくライブラリソースの一部である必要があります。
- 変換では、.svgソース画像に対応する.tiffがまったく同じ場所に作成されます。.svgソース画像はバイナリライブラリの個別のフォルダにもコピーされ（_svg_ソース名拡張子）、.xml比較ワークフローを反転するソース.xmlから.gsmへの変換をサポートします。この追加フォルダは、LP_XMLConverterツールの-excludesvgオプションで回避することができます。

.svgから.tiffへの変換が失敗するあらゆるケースで、ライブラリ構築中に警告またはエラーが発生します。

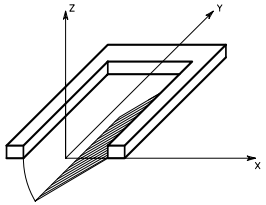
対象とLP_XMLConverterツールのチュートリアルと例については、『GDL Center Tips and Tricks [https://gdl.graphisoft.com/tips-and-tricks/how-to-use-the-lp_xmlconverter-tool]』ガイドを参照してください。

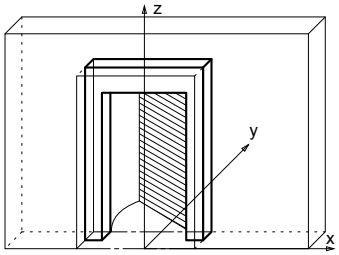
建具

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。

位置

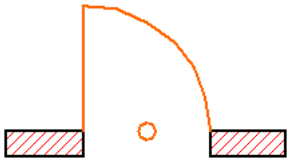
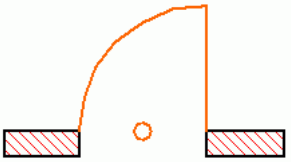
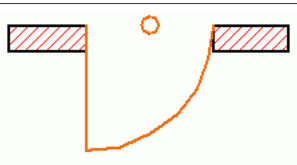
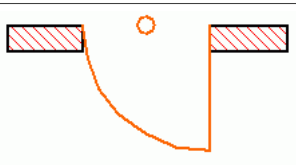
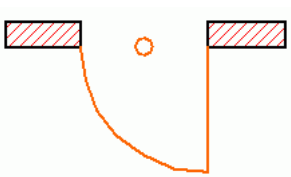
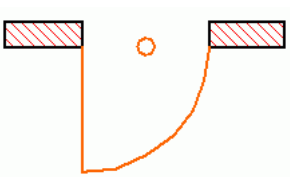
ドアが正しく定義されている場合、次のように挿入されます：挿入点の右をクリックするとドア扉は同じ側面の右側に開きます。 窓が正しく定義されている場合、挿入時にクリックした側面が外側を意味します。

開口位置は8ついずれかの形を選べます。 これらは、GDLの3つのグローバル変数で表されます：

- 3DでYZ平面または、2DでY軸にミラーリング（SYMB_MIRRORED）
- 壁の長手方向軸でミラーリング（180度回転：SYMB_ROTANGLE）
- 反転（WIDO_REVEAL_SIDE）

通常、各ウィンドウで、それらの条件に異なった方法で反応する必要があります。 仕様ではオブジェクトの各部分どのように動作するかが明確である必要があります。 例： ドアの扉は変更が加わると移動しますが、空洞クロージャの場合は動きません。 一貫性のある

ライブラリ部品を維持するには、いくつかの変換で組み合わせを使用する必要があります。 外壁側（反転）を変更すると、ライブラリの部分がミラーされ、公称のフレーム厚の値によって移動されます。
簡略されたドアの8つの状態のイラスト - 円が原点を示します。

グローバル変数 1.	図面の例 1.	グローバル変数 2.	図面の例 2.
WIDO_REVEAL_SIDE = 0 SYMB_MIRRORED = 0 SYMB_ROTANGLE = 0		WIDO_REVEAL_SIDE = 0 SYMB_MIRRORED = 1 SYMB_ROTANGLE = 0	
WIDO_REVEAL_SIDE = 1 SYMB_MIRRORED = 0 SYMB_ROTANGLE = 180		WIDO_REVEAL_SIDE = 1 SYMB_MIRRORED = 1 SYMB_ROTANGLE = 180	
WIDO_REVEAL_SIDE = 1 SYMB_MIRRORED = 0 SYMB_ROTANGLE = 0		WIDO_REVEAL_SIDE = 1 SYMB_MIRRORED = 1 SYMB_ROTANGLE = 0	
WIDO_REVEAL_SIDE = 0 SYMB_MIRRORED = 0 SYMB_ROTANGLE = 180		WIDO_REVEAL_SIDE = 0 SYMB_MIRRORED = 1 SYMB_ROTANGLE = 180	

ARCHICADによって行われる、自動変換を元に戻すサンプルコード：

```
! 2D script
bRotated = round_int (SYMB_ROTANGLE) = 180
if bRotated then
 rot2 180
endif
if SYMB_MIRRORED then
 mul2 -1, 1
endif
if WIDO_REVEAL_SIDE exor bRotated then
 add2 0, WALL_THICKNESS
endif

! 3D script
bRotated = round_int (SYMB_ROTANGLE) = 180
if bRotated then
 roty 180
endif
if SYMB_MIRRORED then
 mulx -1
endif
if WIDO_REVEAL_SIDE exor bRotated then
 addz -WALL_THICKNESS
endif
```

反転とミラーリングは全てのドアと窓で可能ですが、メーカーのライブラリ等の実在する商品を反転することは、そもそも間違っていますのでご注意ください。 この場合、スクリプトはARCHICADが行ったミラーリングを元に戻す必要があります。

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
  - 曲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
  - - カスタムの壁の開口部の切り取り
  - WALLHOLE2
  - 壁ポリゴンの拡張
  - WALLBLOCK2
  - WALLLINE2

- WALLARC2

直線壁の矩形の建具

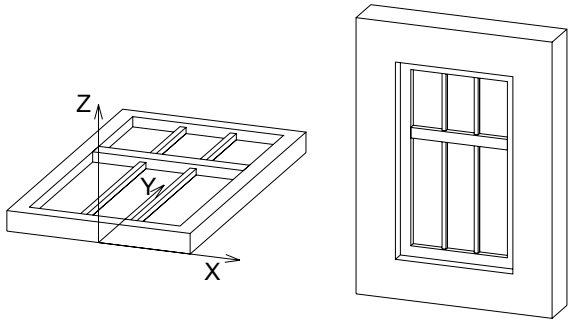
これはドアと窓を作成する最も簡単な方法です。 できる限り、PRISM_やRECTなどの単純なGDLコマンドを使用することをお勧めします。

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、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:



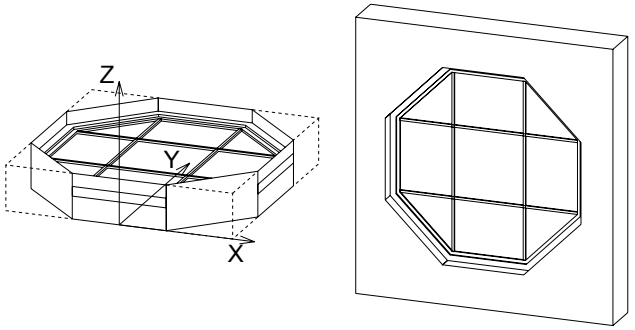
```
a=0.9: b=1.5: c=0.1: d=0.08
e=0.08: f=0.9: g=0.03: h=3
PRISM_ 10, c,
       -a/2, 0, 15, a/2, 0, 15,
       a/2, b, 15, -a/2, b, 15,
       -a/2, 0, -1,
       -a/2+d, d, 15, a/2-d, d, 15,
       a/2-d, b-d, 15, -a/2+d, b-d, 15,
       -a/2+d, d, -1
ADD -a/2+d, f, 0
BRICK a-2*d, e, c
ADD -g/2, -f+d, c/2
GOSUB 1
ADDZ -g
GOSUB 1
DEL 2
MATERIAL "Glass - Blue"
ADD 0, -f+d, c/2
RECT a-2*d, f-d
ADDY f-d+e
RECT a-2*d, b-f-e-d
END

1:
   FOR i=1 TO h-1
      ADDX (a-2*d)/3
      BLOCK g, f-d, g
      ADDY f+e-d
      BLOCK g, b-f-d-e, g
      DEL 1
   NEXT i
   DEL h-1
   RETURN
```

3D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ください。



- 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]

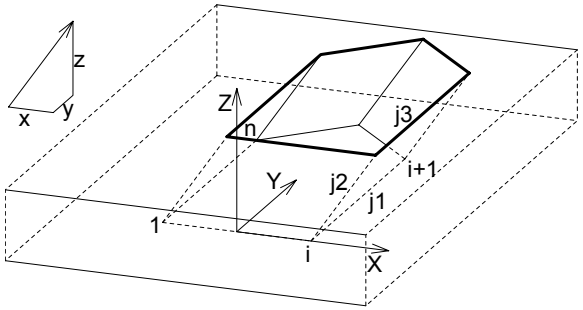
n: ポリゴンの節点の数

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」ステートメントと同様。
maski = j_1 + 2*j_2 + 4*j_3 + 64*j_7, ここで、 各 jiフラグは0または1をとります.

x, y, z: 任意の方向ベクトル（デフォルトは建具のZ軸）

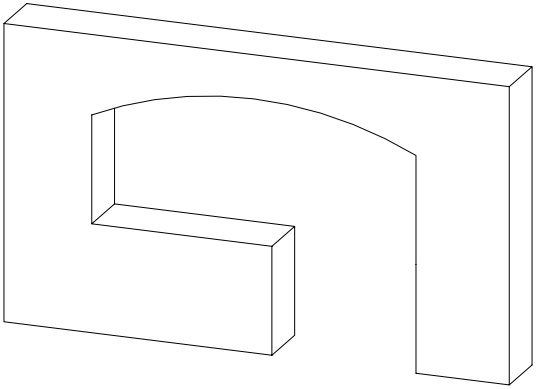


このコマンドを建具の3Dスクリプトに使用して、配置先の壁にカスタム穴を切り取ることができます。 現在の壁の3D生成中に、全ての建具の3Dスクリプトがモデルを生成しないで解析され、WALLHOLEコマンドをまとめてコールします。 WALLHOLEコマンドがある場合、スクリプト内で定義されているポリゴン切断面と方向のチューブ形状で、現在の壁が切り取られます。 建具には複数のWALLHOLEを使用できるので、同じ建具に交差するような複数の開口部を切り取ることもできます。 建具の3DスクリプトでWALLHOLEコマンドが見つかると、その建具には矩形の開口部は生成されません。

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。 ミラー変形は予期しない方法での切り取り方向に影響を及ぼします。より簡単な結果を得るには、 「WALLNICHE」コマンドを使用します。

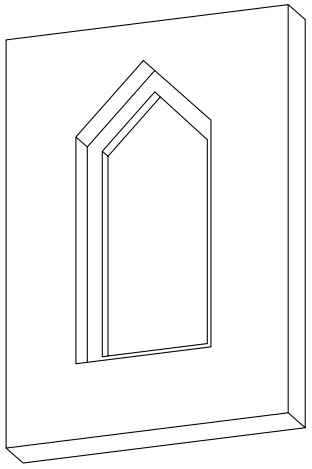
例 1:



```
RESOL 72
l1 = 2.7: l2=1.2
h1=2.1: h2=0.3: h3=0.9
r = ((l1/2)^2+h2^2)/(2*h2)
a = ATN((l1/2)/(r-h2))
WALLHOLE 5, 1,
        -l1/2, h3,    15,
        l1/2,  h3,    15,
        l1/2,  h1-h2, 13,
        0,     h1-r,  915,
        0, 2*a, 4015
WALLHOLE 4, 1,
        l1/2-l2, 0,  15,
        l1/2,    0,  15,
        l1/2,    h3, 15,
        l1/2-l2, h3, 15
```

例 2:

```
WALLHOLE 5, 1,
        -0.45, 0, 15,
        0.45, 0, 15,
        0.45, 1.5, 15,
        0, 1.95, 15,
        -0.45, 1.5, 15
PRISM_ 12, 0.1,
        -0.45, 0, 15,
        0.45, 0, 15,
        0.45, 1.5, 15,
        0, 1.95, 15,
        -0.45, 1.5, 15,
        -0.45, 0, -1,
        -0.35, 0.1, 15,
        0.35, 0.1, 15,
        0.35, 1.45, 15,
        0, 1.80, 15,
        -0.35, 1.44, 15,
        -0.35, 0.1, -1
```



WALLNICHE

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

「CUTFORM」と同様。

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: ウェッジ形状の切り取りボディ。 ウェッジ上端の方向は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 = $j_1 + 2*j_2 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9$, ここで、 各 jiフラグは0または1をとります.

j_1: 生成されたポリゴンと辺にボディの属性を使用
j_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
j_4: 切り取りの制限を定義します（j4で）

j5: 切り取りの制限を定義します（j5で）
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。
j9: 切り取り形状にはカスタム側面材質 （mati） があります。

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

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。

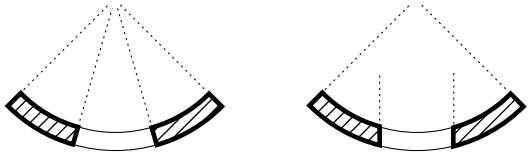
切り取りが半無限の場合、切り取りボディはrx, ry, rzで定義された方向に沿ったdの距離から始まり、半無限切り取りの方向はrx, ry, rzで定義された方向とは逆になります。

mati: 切り取り形状の側面材質 （ステータスj9 = 1の場合）

mask: 切り取りボディの辺の可視性を定義します。
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、壁に開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。

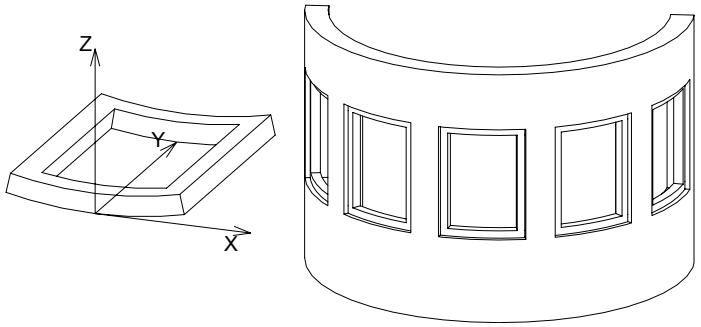


上図左側の壁の穴は、プログラムに自動的に建具用の穴を切り取らせた場合に作成されます。 この場合、側面は半径方向と一致します。 右側の穴は、建具オブジェクトの3Dスクリプトで「WALLHOLE」コマンドを使用して切り取られています。 これらの点を考慮してオブジェクト自体を作成する必要があります。

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: 壁の曲線に沿ったフレーム付き窓

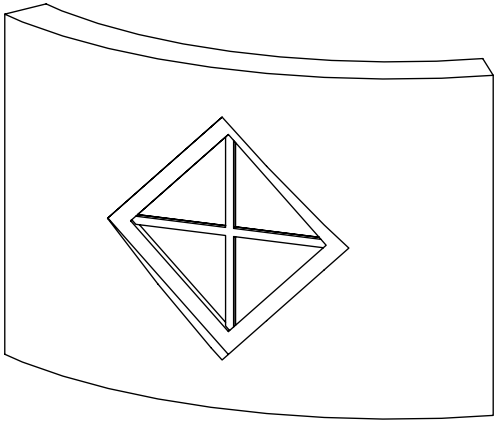


```
RESOL 72
ROTX -90 : MULY -1
C= 0.12 : Z=360*A/(2*WIDO_ORIG_DIST*PI)
Y= 360*C/(2*WIDO_ORIG_DIST*PI) : A1= 270+Z/2 : A2=270-Z/2
GOSUB "curved_horizontal_frame"
ADDZ B
MULZ -1
GOSUB "curved_horizontal_frame"
DEL 2
ADDZ C
GOSUB "vertical_frame"
MULX -1
GOSUB "vertical_frame"
END
"curved_horizontal_frame":
   PRISM_ 9, C,
         cos(A2)*R_, SIN(A2)*R_+R_, 11,
         cos(A2+Y)*R_, sin(A2+Y)*R_+R_, 13,
         0, R_, 900,
         0, Z-2*Y, 4009,
         cos(A1)*R_, sin(A1)*R_+R_, 11,
         cos(A1)*(R_-0.1), sin(A1)*(R_-0.1)+R_, 11,
         cos(A1-Y)*(R_-0.1), sin(A1-Y)*(R_-0.1)+R_, 13,
         0, -(Z-2*Y), 4009,
         cos(A2)*(R_-0.1), sin(A2)*(R_-0.1)+R_, 11
   RETURN
"vertical_frame":
   PRISM_ 4, B-2*C,
         cos(A2)*R_,        sin(A2)*R_+R_, 10,
         cos(A2+Y)*R_,       sin(A2+Y)*R_+R_, 15,
         cos(A2+Y)*(R_-0.1), sin(A2+Y)*(R_-0.1)+R_, 10,
         cos(A2)*(R_-0.1),   sin(A2)*(R_-0.1)+R_, 10
   RETURN
```

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:



```
wFrame=0.1: wDivider=0.025
Z=A/2-SQR(2)*wFrame: Y=A/2-SQR(2)*wFrame-wDivider
ADDY A/2
WALLHOLE 4, 1,
	0,	-A/2, 15,
	A/2,	0,	15,
	0,	A/2,	15,
	-A/2, 0,	15
PRISM_ 10, 0.1,
	0,	-A/2, 15,
	A/2,	0,	15,
	0,	A/2,	15,
	-A/2, 0,	15,
	0,	-A/2, -1,
	0,	-Z,	15,
	Z,	0,	15,
	0,	Z,	15,
	-Z,	0,	15,
	0, -	Z,	-1
ADDZ 0.02
GOSUB "cross_divider"
ADDZ 0.03
GOSUB "cross_divider"
ADDY -Z
SET MATERIAL "Glass-Blue"
ROTZ 45
RECT SQR(2)*Z, SQR(2)*Z
END
```

```
"cross_divider":
   PRISM_ 16, 0.03,
          0, -Z, 15,
          wDivider, -Y, 15,
          wDivider, -wDivider, 15,
          Y, -wDivider, 15,
          Z, 0, 15,
          Z, wDivider, 15,
          wDivider, wDivider, 15,
          wDivider, Y, 15,
          0, Z, 15,
          -wDivider, Y, 15,
          -wDivider, wDivider, 15,
          -Y, wDivider, 15,
          -Z, 0, 15,
          -Y, -wDivider, 15,
          -wDivider, -wDivider, 15,
          -wDivider, -Y, 15
   RETURN
```

2D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}」と同じです。

fill_control:

fill_control = 2*j_2 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7, ここで、 各 jiフラグは0または1をとります.
j_2: ポリゴンの表面塗りつぶしを描画
j_4: ローカル塗りつぶし向き
j_5: 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）
j_6: 切断塗りつぶし （デフォルトは作図塗りつぶし）
j_7: 表面塗りつぶし （j6 = 0の場合のみ、 デフォルトは作図塗りつぶし）

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:

distortion_flags = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8, 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。

j_1-j_7： 「POLY2_B{5}」ステートメントと同様。

j_8: 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、コマンドで生成されたポリゴンを切り取ります。一方、同一オブジェ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、コマンドで生成された線を切り取ります。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、「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。 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、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、「ARC2」と同じです。

平面図から作成されるGDL

平面図をGDLスクリプトまたはライブラリ部品として保存すると、GDL要素になります。 このGDLスクリプトをカスタムライブラリ部品のテンプレートとして使用できます。

キーワード

共通のキーワード

FILE_DEPENDENCE
MOD
AND
OR
EXOR
FOR
TO
STEP
NEXT
DO (に DO - WHILE, に WHILE - ENDWHILE)
WHILE (に DO - WHILE, に WHILE - ENDWHILE)
ENDWHILE
REPEAT
UNTIL
IF (に IF - GOTO, に IF - THEN - ELSE - ENDIF)
THEN (に IF - GOTO, に IF - THEN - ELSE - ENDIF)
GOTO (に IF - GOTO, に GOTO)
GOSUB (に IF - GOTO, に GOSUB)
ELSE
ENDIF
RETURN
END
EXIT
BREAKPOINT

FILLTYPES_MASK (に DEFINE FILL, に DEFINE FILLA, に DEFINE SYMBOL_FILL, に DEFINE SOLID_FILL, に DEFINE EMPTY_FILL, に DEFINE LINEAR_GRADIENT_FILL, に DEFINE RADIAL_GRADIENT_FILL, に DEFINE TRANSLUCENT_FILL, に DEFINE IMAGE_FILL, に VALUES)
PROFILETYPES_MASK
DICT
DIM
PUT

GET
USE
NSP
CALL
RETURNED_PARAMETERS
DEFAULT
PRINT

HASKEY
REMOVEKEY
VARDIM1
VARDIM2
PARVALUE_DESCRIPTION
ABS
CEIL
INT
FRA
ROUND_INT
SGN
SQR
ACS
ASN
ATN
COS
SIN
TAN
PI
EXP
LGT
LOG
NOT
MIN
MAX
RND
BITTEST
BITSET

REQ
REQUEST
IND
APPLICATION_QUERY
LIBRARYGLOBAL
STR
STR{2}
SPLIT
STW
STRLEN
STRSTR
STRSUB
STRTOUPPER
STRTOLOWER
OPEN
INPUT
VARTYPE
OUTPUT
CLOSE
INITADDONSCOPE
PREPAREFUNCTION
CALLFUNCTION
CLOSEADDONSCOPE

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
CONT
FILTER
GDLBIN
HIP_ROOFS
LIN
LINE
MIGRATIONWARNING
NOD
NODE
ORIGO
PARS
PAUSE
PLOTMAKER
PLOTTER
RECT_
REF
SFLINE
TET
TETRA
TRI
WALL_
VOCA
UI_OK
UI_CANCEL

3D専用

ADDX
ADDY
ADDZ
ADD
MULX
MULY
MULZ
MUL
ROTX
ROTY
ROTZ
ROT

XFORM

BLOCK
BRICK
CYLIND
SPHERE
ELLIPS
CONE
PRISM
PRISM_
CPRISM_
CPRISM_{2}
CPRISM_{3}
CPRISM_{4}
BPRISM_
FPRISM_
HPRISM_
SPRISM_
SPRISM_{2}
SPRISM_{3}
SPRISM_{4}
SLAB
SLAB_
CSLAB_
CWALL_
BWALL_
XWALL_
XWALL_{2}
XWALL_{3}
BEAM
CROOF_
CROOF_{2}
CROOF_{3}
CROOF_{4}
MESH
ARMC

ARME
ELBOW
EXTRUDE
PYRAMID
REVOLVE
REVOLVE{2}
REVOLVE{3}
REVOLVE{4}
REVOLVE{5}
RULED
RULED{2}
RULEDSEGMENTED
RULEDSEGMENTED{2}
SWEEP
TUBE
TUBE{2}
TUBEA
COONS
COONS{2}
MASS
MASS{2}
POLYROOF
POLYROOF{2}
POLYROOF{3}
POLYROOF{4}
EXTRUDEDSHELL
EXTRUDEDSHELL{2}
EXTRUDEDSHELL{3}
REVOLVEDSHELL
REVOLVEDSHELL{2}
REVOLVEDSHELL{3}
REVOLVEDSHELLANGULAR
REVOLVEDSHELLANGULAR{2}
REVOLVEDSHELLANGULAR{3}
RULEDSHELL
RULEDSHELL{2}

RULEDSHELL{3}
TEXT
BODY
BASE
NURBSCURVE2D
NURBSCURVE3D
NURBSSURFACE
NURBSVERT
NURBSEDGE
NURBSTRIM
NURBSTRIMSINGULAR
NURBSFACE
NURBSFACE{2}
NURBSLUMP
NURBSBODY
POINTCLOUD
CUTPLANE
CUTEND (に CUTPLANE, に CUTPLANE{2}, に CUTPLANE{3}, に CUTPOLY, に CUTPOLYA, に CUTSHAPE)
CUTPLANE{2}
CUTPLANE{3}
CUTPOLY
CUTPOLYA
CUTSHAPE
CUTFORM
CUTFORM{2}
GROUP
ENDGROUP
ADDGROUP
ADDGROUP{2}
ADDGROUP{3}
SUBGROUP
SUBGROUP{2}
SUBGROUP{3}
ISECTGROUP
ISECTGROUP{2}
ISECTGROUP{3}

ISECTLINES
PLACEGROUP
KILLGROUP
SWEEPGROUP
SWEEPGROUP{2}
SWEEPGROUP{3}
SWEEPGROUP{4}
SWEEPGROUP{5}
CREATEGROUPWITHMATERIAL
BINARY
WALLNICHE

HOTSPOT
HOTLINE
HOTARC
LIN_
RECT
POLY
POLY_
PLANE
PLANE_
CIRCLE
ARC
LIGHT
PICTURE
RICHTEXT
VERT (に VERT, に VERT{2})
TEVE
VECT
EDGE
PGON
PGON{2}
PGON{3}
PIPG
COOR
COOR{2}

COOR{3}
MODEL
WIRE
SURFACE
SOLID
MATERIAL (に [SET] MATERIAL, に IND)
BUILDING_MATERIAL (に [SET] BUILDING_MATERIAL, に IND)
SECT_FILL
SECT_ATTRS
SECT_ATTRS{2}
SHADOW
ON
OFF
AUTO
DEFINE MATERIAL (に DEFINE MATERIAL, に DEFINE MATERIAL BASED_ON)
BASED_ON
DEFINE TEXTURE
TEXTURE
WALLHOLE

2D専用

ADD2
MUL2
ROT2

LINE2
RECT2
POLY2
POLY2_
POLY2_A
POLY2_B
POLY2_B{2}
POLY2_B{3}
POLY2_B{4}
POLY2_B{5}
POLY2_B{6}

ARC2
CIRCLE2
SPLINE2
SPLINE2A
TEXT2
RICHTEXT2
FRAGMENT2
PROJECT2
PROJECT2{2}
PROJECT2{3}
PROJECT2{4}
DRAWING2
DRAWING3
DRAWING3{2}
DRAWING3{3}
WALLHOLE2
WALLHOLE2{2}
WALLBLOCK2
WALLBLOCK2{2}
WALLLINE2
WALLARC2

HOTSPOT2
HOTLINE2
HOTARC2
PICTURE2
PICTURE2{2}
LINE_PROPERTY
DRAWINDEX
FILL (に [SET] FILL, に IND)
LINE_TYPE (に [SET] LINE_TYPE, に IND)
DEFINE FILL
DEFINE FILLA
DEFINE SYMBOL_FILL
DEFINE SOLID_FILL
DEFINE EMPTY_FILL

DEFINE LINEAR_GRADIENT_FILL
DEFINE RADIAL_GRADIENT_FILL
DEFINE TRANSLUCENT_FILL
DEFINE IMAGE_FILL
DEFINE LINE_TYPE
DEFINE SYMBOL_LINE

2Dおよび3D用

DEL (に DEL, に DEL TOP)
TOP
NTR
ADDITIONAL_DATA (に LIGHT, に DEFINE MATERIAL BASED_ON)
LET
RADIUS
RESOL
TOLER
PEN
SET (に [SET] STYLE, に [SET] MATERIAL, に [SET] BUILDING_MATERIAL, に [SET] FILL, に [SET] LINE_TYPE)
STYLE (に [SET] STYLE, に IND)
DEFINE STYLE
DEFINE STYLE{2}
PARAGRAPH
ENDPARAGRAPH
TEXTBLOCK
TEXTBLOCK_
PROFILE_ATTR

非ジオメトリックスクリプト

特性スクリプト

DATABASE_SET
DESCRIPTOR
REF DESCRIPTOR
COMPONENT
REF COMPONENT
BINARYPROP

SURFACE3D
VOLUME3D
POSITION
WALLS
COLUMNS
BEAMS
DOORS
WINDOWS
OBJECTS
CEILS
PITCHED_ROOFS
LIGHTS
HATCHES
ROOMS
MESHES
DRAWING

パラメータスクリプト

VALUES
CUSTOM (に VALUES, に UI_INFIELD{4})
RANGE
VALUES{2}
PARAMETERS (に PARAMETERS, に CALL)
LOCK
ALL (に LOCK, に HIDEPARAMETER, に CALL)
HIDEPARAMETER

インターフェイススクリプト

UI_DIALOG
UI_PAGE
UI_CURRENT_PAGE
UI_BUTTON (に UI_BUTTON, に UI_TOOLTIP)
UI_PREV
UI_NEXT
UI_FUNCTION
UI_LINK

UI_PICT_BUTTON (に UI_PICT_BUTTON, に UI_TOOLTIP)
UI_SEPARATOR
UI_GROUPBOX
UI_PICT (に UI_PICT, に UI_TOOLTIP)
UI_STYLE
UI_OUTFIELD (に UI_OUTFIELD, に UI_TOOLTIP)
UI_INFIELD (に UI_INFIELD, に UI_TOOLTIP)
UI_INFIELD{2} (に UI_INFIELD{2}, に UI_TOOLTIP)
UI_INFIELD{3} (に UI_INFIELD{3}, に UI_TOOLTIP)
UI_INFIELD{4} (に UI_INFIELD{4}, に UI_TOOLTIP)
UI_CUSTOM_POPUP_INFIELD (に UI_CUSTOM_POPUP_INFIELD, に UI_TOOLTIP)
UI_CUSTOM_POPUP_INFIELD{2} (に UI_CUSTOM_POPUP_INFIELD{2}, に UI_TOOLTIP)
UI_RADIOBUTTON (に UI_RADIOBUTTON, に UI_TOOLTIP)
UI_RADIOBUTTON{2}
UI_PICT_RADIOBUTTON
UI_PICT_RADIOBUTTON{2}
UI_PICT_PUSHCHECKBUTTON
UI_PICT_PUSHCHECKBUTTON{2}
UI_TEXTSTYLE_INFIELD
UI_TEXTSTYLE_INFIELD{2}
UI_LISTFIELD (に UI_LISTFIELD, に UI_TOOLTIP)
UI_LISTITEM (に UI_LISTITEM, に UI_TOOLTIP)
UI_LISTITEM{2} (に UI_LISTITEM{2}, に UI_TOOLTIP)
UI_CUSTOM_POPUP_LISTITEM (に UI_CUSTOM_POPUP_LISTITEM, に UI_TOOLTIP)
UI_CUSTOM_POPUP_LISTITEM{2} (に UI_CUSTOM_POPUP_LISTITEM{2}, に UI_TOOLTIP)
UI_TOOLTIP
UI_COLORPICKER
UI_COLORPICKER{2}
UI_SLIDER
UI_SLIDER{2}

上位および下位移行スクリプト

SETMIGRATIONGUID
STORED_PAR_VALUE
DELETED_PAR_VALUE
NEWPARAMETER

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「GDL Text 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)

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。

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

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、追加/修正）
- WO（上書き）。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。

- 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）。
- Browse：既存のデータファイルを検索します（[開く]ダイアログ）

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なりません。 全てのスクリプトから、ロードされたライブラリから読むためにデータファイルを開くことは可能ですが、書き込みはパラメータ、ユーザーインターフェース、および特性スクリプトでのみ可能です。

SEPARATOR、MODE、DIALOG 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, ...])

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。 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、正常に読み込まれた値の数です。

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。

channel: チャネル値、接続を識別するのに使用されます。

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）

vari: 読み取りレコード項目を受け取る変数

例:

```
! 最初の列の 3 つの値の入力
nr = INPUT (ch1, "key1", 1, v1, v2, v3)

PRINT nr, v1, v2, v3
```

データベースに値を書き込む

OUTPUT channel, recordID, fieldID, expr1 [, expr2, ...]

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。

ライブラリでロードされたデータファイルの修正は、パラメータ、ユーザーインターフェース、および特性スクリプトでのみ可能です。

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。

expri: 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。

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_string, 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)

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。

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。

規則子は、次のように日時の値に置き換えられます。

%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
%b	省略形の月名
%B	省略なしの月名
%m	月、10進数で表示（01～12）
%d	10進数による1か月の日付（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）
%M	分、10進数で表示（00～59）
%S	秒、-59進数で表示（00～59）
%P	12時間時計のAM/PM表示

%c	01:35:56 PM Wednesday, March 27, 1996 という形式での日時
%x	Wednesday, March 27, 1996形式での日付
%X	01:35:56 PM形式での時刻
%a	省略形の曜日名
%A	省略なしの曜日名
%w	曜日、10進数で表示（0 （日曜日）～6 （土曜日））
%j	年間日付、10進数で表示（001～366）
%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	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）
%Z	設定可能な場合タイムゾーンを出力
%%	% 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)

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。

channel: チャネル値、接続を識別するのに使用されます。

datetimestr: 文字列タイプ値

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のパラメータはコンマ（,）で区切る必要があります。

1. parameter: FILES/FOLDERS: 検索対象。
2. parameter (optional): 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: 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。
folder = OPEN ("FileMan", "c:¥", "FOLDERS")

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: 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
n = INPUT (folder, 0, 0, fileName)
成功した場合、nは1です。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

```
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)
```

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。

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取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。

作成ファイルは、相対パスで指定されている場合、アプリケーションデータフォルダのサブフォルダに配置されます。 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)
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。

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
　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、 (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。 全てのスクリプトから、ロードされたライブラリから読むためにデータファイルを開くことは可能ですが、書き込みはパラメータ、ユーザーインターフェース、および特性スクリプトでのみ可能です。
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。
NEWLINE: 改行文字の定義。有効値 :
- CR (Carriage return, 0x0D)
- LF (Line feed, 0x0A)
- CRLF (Carriage return + Line feed, 0x0D0x0A)

Windowsのような改行は、こちらを使用します "NEWLINE = CRLF"
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, NEWLINE = LF"

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, ...])
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。

channel: チャネル値、接続を識別するのに使用されます。

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

例:

```
nr = INPUT (ch1, 1, 1, v1, v2, v3) ! 最初の列の 3 つの値の入力
! 最初の行の最初の列から
PRINT nr, v1, v2, v3
```

値を書き込む

OUTPUT channel, recordID, fieldID, expr1 [, expr2, ...]

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。

ライブラリでロードされたデータファイルの修正は、パラメータ、ユーザーインターフェース、および特性スクリプトでのみ可能です。

channel: チャネル値

recordID: 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。

fieldID: 役割はなし。この値は使われません。

expr1: 出力値

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値で識別されたファイルを閉じます。

channel: チャネル値

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 "close all"
   endif

"close all":
   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 計算ガイド」を参照してください。

特性データベースを開く

OPEN ("PROP", "database set name", "[database files]")

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。

database set name: 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: 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。 ファイルの順番は固定されています：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。

例 1:

```
channel = OPEN ("PROP", "sample",
      "'ArchiCAD_Library_KEY.txt', 'ArchiCAD_Library_COMP.txt', 'ArchiCAD_Library_DESC.txt', 'ArchiCAD_Library_UNIT.txt'")
```

これは、上記のファイル（これらは、ARCHICAD 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）。

例 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。

特性データベースを閉じる

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。

特性データベースに入力

INPUT (channel_number, "query type", "field list", variable1 [, ...])

channel_number: 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: 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。 有効フィールド：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。有効フィールド：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。 有効フィールド：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。 有効フィールド：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。 有効フィールド：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。いずれかのフィールドはデータベース内で有効でない場合（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。

field list: 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：数値 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：数値 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: 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。

例 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>

例 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.

例 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）

特性データベースに出力

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。

- 要素： 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。
- テキスト： 文字シーケンス 子ノードを持つことはできません。
- コメント： 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。

	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）

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。

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

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」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。全てのスクリプトから、ロードされたライブラリから読むためにデータファイルを開くことは可能ですが、書き込みはパラメータ、ユーザーインターフェース、および特性スクリプトでのみ可能です。

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。

n = 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, ...: 戻り値を受け取る変数のリスト（省略可能）

INPUTの命令：

- GetLastError： 最後の操作の結果を検索します。
  recordID："GetLastError"
  fieldID：無視されます。
  戻り値：
  var1：エラーコード /OK
  var2：エラーまたはOKの説明テキスト
- NewPositionDesc： 新規の位置記述項目の要求
  recordID："NewPositionDesc"

fieldID：無視されます。

戻り値：var1：新規の位置記述項目（最初はルートを参照）

- **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

  戻り値：var1：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）
- **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

  var1：無視されます。

  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。
- **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（または 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。

  正確なパスを指定するには、Path movemodeを使用する必要があります。このmovemode後、必要なパスのみを表示する必要があります。

  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。に戻る。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。

  移動モード：

  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。

  検索モード：

  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。

n = INPUT (ch, "MoveToNode FromPrevSibling K* ELEM+EREF 2", posDesc, name, val, type)

- **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
- **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"

  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。

  fieldID：位置記述項目

戻り値：

var1：子ノードの数

- **NumberofAttributes：** 要素ノードの属性の数を返します。

  recordID："NumberofAttributes attrname"

  attrname：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。

  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）

  戻り値：

  var1：属性の数
- **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 ：*

  attrnumber：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
- **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。

  var1：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。

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。

- CreateDocument：
  recordID："CreateDocument"
  fieldID：無視されます。
  var1：ドキュメントの名前 これは、 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。
- 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）。
- 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。
- 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。
- 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。

- **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特定の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。
- **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。
- **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
  var1：無視されます。
  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。
- **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
- **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。

- **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
- **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。
  var1：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。
- **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「XMLドキュメントを開く」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。

表14 エラーコードおよびメッセージ（OPENファンクションで使用可能な戻り値）

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）は許可されていません"
-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

このアドオンは入力ポリゴンとその上で行われる操作に基づいて、結果のポリゴンを計算します。

互換性：ARCHICAD 21で導入されました： ポリラインへの操作もあります。

入力ポリゴンはアドオンに渡されたときの名前によって識別され、以前に定義されたコンテナに格納します。 結果ポリゴンが自動的にアドオンによって名前が付けられ、第2ターゲットコンテナに格納します 入力および結果ポリゴンは、このように異なるコンテナに格納します。

さらに多くの数の輪郭を持つ複数ポリゴンは、単一の操作で作成することができます。 これらはターゲットコンテナに個々のポリゴンとして管理されます。 その結果、これらのポリゴンは引き続き、ポリゴン操作でアクセスすることができます。 原理はソリッド図形コマンドと同じです（「ソリッド図形コマンド」を参照）。 入力ポリゴンは隣接してなければなりません。

ポリゴンは複数の輪郭によって定義され、個々は連続および接続された頂点のシーケンスです。 最初の輪郭が境界の外にあります。 その後の輪郭の全てが内側にあり重なっていないこと、そして最初のポリゴンの切り取りを作成します。

ポリラインを閉じる必要はありませんが、複数の輪郭を持つことはできません。

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。

コンテナの管理

CreateContainer
新規コンテナを作成します。
PREPAREFUNCTION ch, "CreateContainer", "myContainer", ""

DeleteContainer
既存のコンテナを削除します。
PREPAREFUNCTION ch, "DeleteContainer", "myContainer", ""

EmptyContainer
既存のコンテナを空にします。
PREPAREFUNCTION ch, "EmptyContainer", "myContainer", ""

SetSourceContainer
コンテナをソースコンテナとして設定します。
PREPAREFUNCTION ch, "SetSourceContainer", "mySourceContainer", ""

SetDestinationContainer
コンテナを出力先コンテナとして設定します。
PREPAREFUNCTION ch, "SetDestinationContainer", "myDestinationContainer", ""

ポリゴン/ポリラインの管理
ジオメトリは2つのデータ形式、配列または辞書でやり取りすることができます。互換性：ARCHICAD 23で導入されました。
完全に同じジオメトリを2つの方法で保存し、取り出すことができます。

配列
配列形式は3つの分離した配列です（任意の名前を持つことができ、関数パラメータリスト内での位置によって識別されます）。

contourArray: ポリゴンのみに必要、ポリラインには不要。 **vertArray**で各輪郭の最後の頂点のインデックスを含む配列。
"GetContourEnds"から戻された"Store"で指定。

vertArray: ポリゴン/ポリラインの全ての輪郭を記述する全ての頂点を含む配列 "Store"と"StorePolyline"で使用するための2次元配列、"GetVertices"および"GetPolylineVertices"から戻された1次元（平坦化された2次元）配列。

inhEdgeInfosArray: 呼び出し側で定義され、辺にアタッチされた情報を含むオプション配列。 これには**vertArray**と同じ数の頂点を含める必要があります。 "Store"と"StorePolyline"で指定され、"GetInhEdgeInfos"と"GetPolylineInhEdgeInfos"から返されます。

辞書

辞書形式は扱いがより簡単です（任意の名前を持つことができ、関数パラメータリスト内での位置によって識別されます）。

.isClosed: （ブール）1 - ポリゴンまたは閉じたポリライン、0 - 開いたポリライン（追加の辺として指定された最後の点）

.useEdgeInfo: （ブール、省略可能）1に設定すると、PolyOperationsには.edgeInfoキーを使用する必要があり、そうでない場合は無視されます

.defaultInhEdgeInfo: （整数、省略可能）"Store"のdefaultInhEdgeInfoパラメータに対応します

.contour: （辞書）ポリゴン輪郭のデータを含み、**vertArray**に対応します

.contour.edges[n]: （配列）ポリゴンの各辺に対する埋め込み辞書を含みます

.contour.edges[n].type: （整数）0 - 直線、1 - 曲線（円弧）

.contour.edges[n].begPoint: （配列）辺の開始点に対する埋め込み辞書

.contour.edges[n].begPoint.x / .y: （フロート）座標

.contour.edges[n].arcAngle: （角度）辺曲線の中心角、正は反時計回り、負は時計回り（直線辺には設定されません）

.contour.edges[n].edgeInfo: （整数、省略可能）辺にアタッチされる情報

.holes[m]: （配列、省略可能）内部の穴のデータを含み、.contourと同様に、穴が存在する場合のみ設定されます

一部のグローバル変数は、PolyOperationsと互換性のある辞書構造を使用します（OPENING_SYMBOL_GEOMETRY.polygon2Dなど）。

"StoreDictPolygon"と"StoreDictPolyline"には同じデータ形式を使用できます。

ポリラインに.holes[]は指定せず、ポリゴンは常に.isClosed = 1です。

保存

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。

```
PREPAREFUNCTION ch, "Store", "poly1", nVertices, nContours,
    vertArray, contourArray [, defaultInhEdgeInfo, inhEdgeInfosArray]
```

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、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。 このデータを使用して複雑な操作の後に新規作成された辺を参照できます。（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）

注釈：

- ポリゴンは穴と湾曲辺持つことができますが、それらの湾曲辺は円形状の円弧でなければなりません。
- ポリゴンは、各辺の追加データにリンクできます。
- 最初の頂点は常に、全ての輪郭に対し繰り返す必要があります。 この表現では三角形は4つの頂点を持ち、最初と最後の頂点が同一です。
- 最初の輪郭は主要輪郭なので、他のものを含まなければなりません。

StorePolyline

指定されたパラメータを持つポリライン「polyline1」を、現在のソースコンテナに保存します。

PREPAREFUNCTION ch, "StorePolyline", "polyline1", nVertices,
vertArray, [, defaultInhEdgeInfo, inhEdgeInfosArray]

polyline1: 保存されるポリラインの名前

nVertices: 頂点の総数

vertArray: ポリラインを記述するnVertices項目を全て含む配列。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。

defaultInhEdgeInfo: 継承された辺情報のうちの1つ。 この情報は、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。 このデータを使用して複雑な操作の後に新規作成された辺を参照できます。（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）

注釈：

- ポリラインは、各辺の追加データにリンクできます。

- 閉じたポリラインを定義するには、最後の頂点座標を最初の頂点座標と同じにする必要があります。開いているポリラインと閉じたポリラインの動作は、オフセットまたは分割されている場合は異なります。

StoreDictPolygon

辞書として記述されたポリゴン「poly1」を、現在のソースコンテナに保存します。互換性：ARCHICAD 23で導入されました。
PREPAREFUNCTION ch, "StoreDictPolygon", "poly1", PolyOpPolygon
poly1: 保存されるポリゴンの名前
PolyOpPolygon: 辞書としてのポリゴン

StoreDictPolyline

辞書として記述されたポリライン「polyline1」を、現在のソースコンテナに保存します。互換性：ARCHICAD 23で導入されました。
PREPAREFUNCTION ch, "StoreDictPolyline", "polyline1", PolyOpPolyline
polyline1: 保存されるポリラインの名前
PolyOpPolyline: 辞書としてのポリライン

Dispose

ポリゴン/ポリライン「poly1」を、コンテナ「myContainer」から削除します。
PREPAREFUNCTION ch, "Dispose", "poly1", "myContainer"

ポリゴン/ポリライン操作の設定

これらのコマンドは、データが辞書形式と配列形式のどちらで指定されているかに関係なく、同じように機能します。

HalfPlaneParams

「PolyCut」操作で使用する2Dに、半平面の記述を設定します。
PREPAREFUNCTION ch, "HalfPlaneParams", "", ca, cb, cc
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

OffsetParams

「OffsetEdge」、「ResizeContour」、「PolylineOffsetVectors」操作で使用されるオフセットパラメータを設定します。

PREPAREFUNCTION ch, "OffsetParams", "", itemIdx, offsetValue

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。 「PolylineOffsetVectors」操作の場合常に1。

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。

MultipleEdgeOffsetParams

「OffsetMultipleEdges」および「PolylineOffsetVectors」操作で使用されるオフセットパラメータを設定します。

PREPAREFUNCTION ch, "MultipleEdgeOffsetParams", "", nOffset, offsetArray

nOffset: オフセットの辺の数。

offsetArray: オフセットする辺を記述するnOffset項目を含む配列。（edgeIndex, offsetValue）レコードの2次元配列。 edgeIndexは、1から始まる辺のインデックス、offsetValueは変換距離。 開いているポリラインの場合、（辺の方向に見て）左に向かって正、右に向かって負。

PolylineOffsetVectors

「OffsetPolylineWithVectors」操作で使用される終了オフセットパラメータを設定します。「OffsetParams」や「MultipleEdgeOffsetParams」を別途設定する必要があります。

PREPAREFUNCTION ch, "PolylineOffsetVectors", "", sx, sy, ex, ey

sx, sy: 頂点オフセット方向ベクトルを開始します。 （互換性：バージョンARCHICAD 21までは単位ベクトルである必要があります。）

ex, ey: 頂点オフセット方向ベクトルを終了します。 （互換性：バージョンARCHICAD 21までは単位ベクトルである必要があります。）

ポリゴン/ポリライン操作

これらの機能は、データが辞書形式と配列形式のどちらで指定されているかに関係なく、同じように動作します。

以下のポリゴン操作では、「poly1」、poly2」ポリゴンおよび「polyline1」ソースポリラインが、ソースコンテナ内にあります。 この操作で作成されたポリゴン/ポリラインは、「resPolygonID」または「resPolylineID」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。これらの名前は一般的に配列で返されます。

+ - /

「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。

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "poly1 OP poly2", "", resPolyIDArray)
```

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

ClipPolyline

ポリラインとポリゴンを切り取ります。

```
dim resPolyIDArray[]
nPline = CALLFUNCTION (ch, "ClipPolyline", "polyline1 poly1", resPolyIDArray)
```

polyline1: ソースコンテナのポリラインの名前。

poly1: ソースコンテナのクリップポリゴンの名前。

resPolyIDArray: 生成されたポリラインIDの配列。

CopyPolygon

ソースコンテナから出力先コンテナにポリゴン/ポリラインをコピーします。

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "CopyPolygon", "poly1", resPolyIDArray)
```

Regularize

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。

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "Regularize", "poly1", resPolyIDArray)
```

以下の場合に、ポリゴンは有効になります。

- 最初の境界に他の全てが含まれている
- 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）
- 自己交差していない
- 面積が0でない
- 長さが0の辺がない

PolyCut

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, "PolyCut", "poly1", resPolyIDArray)
```

OffsetEdge

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, "OffsetEdge", "poly1", resPolyIDArray)
```

OffsetMultipleEdges

ポリゴンの複数の辺を、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、「MultipleEdgeOffsetParams」コマンドで設定する必要があります。結果は正規化されます。

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "OffsetMultipleEdges", "poly1", resPolyIDArray)
```

OffsetPolyline

ポリラインの全ての辺を垂直に変換します。

変換オフセットは、「OffsetParams」コマンドで設定する必要があります。

```
dim resPolyIDArray[]
nPline = CALLFUNCTION (ch, "OffsetPolyline", "polyline1", resPolyIDArray)
```

OffsetPolylineWithVectors

開いているポリラインの全ての辺を垂直に変換し、ベクトルに沿って終点を移動します。

変換ベクトルおよびオフセットは「PolylineOffsetVectors」および「OffsetParams」コマンドで設定する必要があります。

```
dim resPolyIDArray[]
nPline = CALLFUNCTION (ch, "OffsetPolylineWithVectors", "polyline1", resPolyIDArray)
```

閉じているポリラインに入力アーギュメントエラーを指定します。

ResizeContour

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。

```
dim resPolyIDArray[]
nPgon = CALLFUNCTION (ch, "ResizeContour", "poly1", resPolyIDArray)
```

CentreOfGravity

ポリゴンの重心を計算します。互換性：ARCHICAD 22で導入されました。

```
n = CALLFUNCTION (ch, "CentreOfGravity", "poly1", x, y)
```

戻り値：

n: 重心を正常に定義できた場合は1、ポリゴンが自己交差している場合など、正常に定義できない場合は0。

x, y: 重心の座標。

結果として生成されたポリゴン/ポリラインを取得する

配列

GetSourcePolygons, GetSourcePolylines

現時点のソースコンテナから全てのポリゴン名/ポリライン名を取得します。

```
dim resPolyIDArray[], resPolylineIDArray[]
nPgon = CALLFUNCTION (ch, "GetSourcePolygons", "", resPolyIDArray
nPline = CALLFUNCTION (ch, "GetSourcePolylines", "", resPolylineIDArray
```

GetDestinationPolygons, GetDestinationPolylines

現時点の出力先コンテナから全てのポリゴン名/ポリライン名を取得します。

```
dim resPolyIDArray[], resPolylineIDArray[]
nPgon = CALLFUNCTION (ch, "GetDestinationPolygons", "", resPolyIDArray)
nPline = CALLFUNCTION (ch, "GetDestinationPolylines", "", resPolylineIDArray
```

GetVertices, GetPolylineVertices

ポリゴン/ポリライン操作を呼び出した後に、生成されたポリゴン/ポリラインの頂点を取得します。

「polygonID」または「polylineID」という名前のポリゴン/ポリラインが出力先ポリゴンコンテナに配置されます。

```
dim resVertices[], resPolylineVertices[]
nVertices = CALLFUNCTION (ch, "GetVertices", polygonID, resVertices)
nVertices = CALLFUNCTION (ch, "GetPolylineVertices", polylineID, resPolylineVertices)
```

GetContourEnds

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。

「polygonID」という名前のポリゴンが出力コンテナに配置されます。

```
dim contArr[]
nContours = CALLFUNCTION (ch, "GetContourEnds", polygonID, contArr)
```

GetInhEdgeInfos, GetPolylineInhEdgeInfos

ポリゴン/ポリライン操作を呼び出した後に、生成されたポリゴン輪郭/ポリライン情報を取得します。

```
dim inhEdgeInfosArr[], polylineInhEdgeInfosArr[]
nEdgeInfos = CALLFUNCTION (ch, "GetInhEdgeInfos", polygonID, inhEdgeInfosArr)
nEdgeInfos = CALLFUNCTION (ch, "GetPolylineInhEdgeInfos",
                           polylineID, polylineInhEdgeInfosArr)
```

「polygonID」または「polylineID」という名前のポリゴン/ポリラインが出力先ポリゴンコンテナに配置されます。

辞書

GetSourceDictPolygon, GetSourceDictPolyline

ソースポリゴン/ポリラインを辞書として取得。互換性：ARCHICAD 23で導入されました。

「polygonID」または「polylineID」という名前のポリゴン/ポリラインがソースコンテナに配置されます。

```
dict PolyOperationsPolygon, PolyOperationsPolyline
CALLFUNCTION (ch, "GetSourceDictPolygon", polygonID, PolyOperationsPolygon
CALLFUNCTION (ch, "GetSourceDictPolyline", polylineID, PolyOperationsPolyline
```

GetDestinationDictPolygon, GetDestinationDictPolyline

ポリゴン/ポリライン操作を呼び出した後に、生成されたポリゴン/ポリラインを辞書として取得します。互換性：ARCHICAD 23で導入されました。

「polygonID」または「polylineID」という名前のポリゴン/ポリラインが出力先ポリゴンコンテナに配置されます。

```
dict PolyOperationsPolygon, PolyOperationsPolyline
CALLFUNCTION (ch, "GetDestinationDictPolygon", polygonID, PolyOperationsPolygon)
CALLFUNCTION (ch, "GetDestinationDictPolyline", polylineID, PolyOperationsPolyline
```

チャネルを閉じる

チャネル「ch」を閉じます。保存されている全てのポリゴン/ポリラインを削除します。

```
CLOSEADDONSCOPE (ch)
```

自動テキストガイド

それはGDL自身の一部ではありません。ARCHICADはGDL出力で見つかった、全ての参照を自動テキストフィールドに代替します。例： <PROJECTSTATUS>をtext2コマンドのパラメータ文字列に書くと、ARCHICADは実際の値に置き換えます。 これらはGDLでは見えません – その結果、テキストのサイズと他の属性を測ることができません。

プロジェクト情報キーワード

```
PROJECTNAME
PROJECTNUMBER
PROJECTSTATUS
```

DATEOFISSUE
SITEFULLADDRESS

SITEADDRESS1
SITEADDRESS2
SITEADDRESS3
SITECITY
SITESTATE

SITEPOSTCODE
SITECOUNTRY
KEYWORDS
NOTES
ARCHITECTNAME

ARCHITECTPOSITION
CADTECHNICIAN
ARCHITECTCOMPANY
ARCHITECTFULLADDRESS
ARCHITECTADDRESS1

ARCHITECTADDRESS2
ARCHITECTADDRESS3
ARCHITECTCITY
ARCHITECTSTATE
ARCHITECTPOSTCODE

ARCHITECTCOUNTRY
ARCHITECTEMAIL
ARCHITECTPHONE
ARCHITECTFAX
ARCHITECTWEB

CLIENTNAME
CLIENTCOMPANY
CLIENTFULLADDRESS
CLIENTADDRESS1
CLIENTADDRESS2

CLIENTADDRESS3
CLIENTCITY
CLIENTSTATE
CLIENTPOSTCODE

CLIENTCOUNTRY

CLIENTEMAIL
CLIENTPHONE
CLIENTFAX

一般

SHORTDATE
LONGDATE
TIME
FILENAME
FILEPATH
LASTSAVEDAT
LASTSAVEDBY

レイアウト自動テキスト

LAYOUTNAME
LAYOUTID
SUBSETNAME
SUBSETID
LAYOUTNUMBER
NUMOFLAYOUTS

図面自動テキスト

DRAWINGNAME
DRAWINGID
DRAWINGSCALE
ORIGINALSCALE
MAGNIFICATION
RENOVATIONFILTER

参照タイプ自動テキスト

LAYOUTNAME_R

LAYOUTID_R
SUBSETNAME_R
SUBSETID_R
DRAWINGNAME_R
DRAWINGID_R
DRAWINGSCALE_R
ORIGINALSCALE_R
MAGNIFICATION_R
FILENAME_R
FILEPATH_R
LAYOUTNUMBER_R
RENOVATIONFILTER_R

マーカータイプ自動テキスト

MARKERSHEETNUMBER_R
MARKERDRAWINGNUMBER_R
MARKERSHEETNUMBER90_R
MARKERDRAWINGNUMBER90_R
MARKERSHEETNUMBER110_R
MARKERDRAWINGNUMBER110_R
BACKREFSHEETNUMBER_R

変更関連の自動テキスト

CHANGEID
CHANGEDESCRIPTION
REVISIONID
ISSUEID
ISSUEDESCRIPTION
ISSUEDATE
ISSUEDBY

レイアウト改訂関係の自動テキスト

CURRENTREVISIONID
CURRENTISSUEID
CURRENTISSUEDESCRIPTION
CURRENTISSUEDATE

CURRENTISSUEDBY

索引

GDLコマンドの構文リスト

A

ABS (x)

ACS (x)

ADD dx, dy, dz

ADD2 x, y

ADDGROUP (g_expr1, g_expr2)

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

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

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

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

ADDX dx

ADDY dy

ADDZ dz

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

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

CALL macro_name_string [,]
PARAMETERS [ALL][name1=value1, ..., namen=valuen][[,]
RETURNED_PARAMETERS r1, r2, ...]

B

C

...
xn, yn, alphan, sn, matn

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

CREATEGROUPWITHMATERIAL (g_expr, repl_directive, pen, material)

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

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

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

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

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

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

CUTEND

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

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

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

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

CUTSHAPE d [, status]
[statement1 statement2 ... statementn]
CUTEND

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

CUTFORM{2} n, method, status,
rx, ry, rz, d,

x1, y1, mask1 [, mat1],
...
xn, yn, maskn [, matn]

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

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

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

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

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

CUTSHAPE d [, status]
[statement1 statement2 ... statementn]
CUTEND

CWALL_ left_material, right_material, side_material,
height, x1, x2, x3, x4, t,

D

spacing_x, spacing_y, angle, n,
frequency1, directional_offset1, direction1,
offset_x1, offset_y1, m1,
length11, ..., length1m,
...
frequencyn, directional_offsetn, directionn,
offset_xn, offset_yn, mn,
lengthn1, ..., lengthnm

DEFINE IMAGE_FILL name image_name [[,] FILLTYPES_MASK fill_types]
part1, part2, part3, part4, part5, part6, part7, part8,
image_vert_size, image_hor_size, image_mask, image_rotangle

DEFINE LINEAR_GRADIENT_FILL name [[,] FILLTYPES_MASK fill_types]

DEFINE LINE_TYPE name spacing, n,
length1, ..., lengthn

DEFINE MATERIAL name type,
surface_red, surface_green, surface_blue
[, ambient_ce, diffuse_ce, specular_ce, transparent_ce,
shining, transparency_attenuation
[, specular_red, specular_green, specular_blue,
emission_red, emission_green, emission_blue, emission_att]]
[, fill_index [, fillcolor_index, texture_index]]

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

DEFINE RADIAL_GRADIENT_FILL name [[,] FILLTYPES_MASK fill_types]

DEFINE SOLID_FILL name [[,] FILLTYPES_MASK fill_types]

DEFINE STYLE name font_family, size, anchor, face_code

DEFINE STYLE{2} name font_family, size, face_code

DEFINE SYMBOL_FILL name [,][FILLTYPES_MASK fill_types,]
pat1, pat2, pat3, pat4, pat5, pat6, pat7, pat8,
spacingx1, spacingy1, spacingx2, spacingy2,
angle, scaling1, scaling2, macro_name [,] PARAMETERS [name1
= value1, ..., namen = valuen]

E

```
    statementn+1
    statementn+2
    ...
    statementn+m]
ENDIF

PARAGRAPH name alignment, firstline_indent,
        left_indent, right_indent, line_spacing [,
        tab_position1, ...]
    [PEN index]
    [[SET] STYLE style1]
    [[SET] MATERIAL index]
    'string1'
    'string2'
    ...
    'string n'
    [PEN index]
    [[SET] STYLE style2]
    [[SET] MATERIAL index]
    'string1'
    'string2'
    ...
    'string n'
    ...
ENDPARAGRAPH

WHILE condition DO
    [statement1
    statement2
    ...
    statementn]
ENDWHILE

EXIT [v1, v2, ..., vn]

EXP (x)

EXTRUDE n, dx, dy, dz, mask,
        x1, y1, s1,
```

...
xn, yn, sn

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

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

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

F

G

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

H

HASKEY (dictionary.key)

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

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

HOTARC r, alpha, beta, unID

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

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

HOTLINE2 x1, y1, x2, y2, unID

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

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

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

I

IF condition THEN label
IF condition GOTO label
IF condition GOSUB label

IF condition THEN label
IF condition GOTO label
IF condition GOSUB label

IF condition THEN label
IF condition GOTO label

K

L

M

N

O

P

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

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

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

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

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

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

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

POLYROOF defaultMat, k, m, n,
offset, thickness, applyContourInsidePivot,

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

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

POLYROOF{3} defaultMat, mask, k, m, n,
offset, thickness, totalThickness, applyContourInsidePivot,
z_1, ..., z_k,
pivotX_1, pivotY_1, pivotMask_1,
roofAngle_11, gableOverhang_11, topMat_11, bottomMat_11,
...
roofAngle_1k, gableOverhang_1k, topMat_1k, bottomMat_1k,

PROJECT2 projection_code, angle, method

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

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

PROJECT2{4} projection_code, angle,
useTransparency, statusParts,
numCutplanes,
cutplaneHeight1, ..., cutplaneHeightn,

method1, parts1,
cutFillIndex1,
cutFillFgPen1, cutFillBgPen1,
cutFillOrigoX1, cutFillOrigoY1, cutFillDirection1,
cutLinePen1, cutLineType1,
projectedFillIndex1,
projectedFillFgPen1, projectedFillBgPen1,
projectedFillOrigoX1, projectedFillOrigoY1,
projectedFillDirection1,
projectedLinePen1, projectedLineType1,
...
method(numCutplanes+1)), parts(numCutplanes+1),
cutFillIndex(numCutplanes+1),
cutFillFgPen(numCutplanes+1), cutFillBgPen(numCutplanes+1),
cutFillOrigoX(numCutplanes+1), cutFillOrigoY(numCutplanes+1),
cutFillDirection(numCutplanes+1),
cutLinePen(numCutplanes+1), cutLineType(numCutplanes+1),
projectedFillIndex(numCutplanes+1),
projectedFillFgPen(numCutplanes+1), projectedFillBgPen(numCutplanes+1),
projectedFillOrigoX(numCutplanes+1), projectedFillOrigoY(numCutplanes+1),
projectedFillDirection(numCutplanes+1),
projectedLinePen(numCutplanes+1), projectedLineType(numCutplanes+1)

PUT expression [, expression, ...]

R

x_n, y_n, s_n

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

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

REVOLVEDSHELLANGULAR{3} topMat, bottomMat,
sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
n, status, offset, thickness, flipped, trimmingBody, alphaOffset, alpha,
segmentationType, nOfSegments,
preThickenTran_11, preThickenTran_12, preThickenTran_13,
preThickenTran_14,
preThickenTran_21, preThickenTran_22, preThickenTran_23,
preThickenTran_24,

profile2Tran_21, profile2Tran_22, profile2Tran_23, profile2Tran_24
profile2Tran_31, profile2Tran_32, profile2Tran_33, profile2Tran 34
generatrixFirstIndex_1, generatrixSecondIndex_1,
...
generatrixFirstIndex_g, generatrixSecondIndex_g

RULEDSHELL{2} topMat, bottomMat,
sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
n, m, g, status,
offset, thickness, flipped, trimmingBody,
preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
firstpolyX_1, firstpolyY_1, firstpolyS_1,
...
firstpolyX_n, firstpolyY_n, firstpolyS_n,
secondpolyX_1, secondpolyY_1, secondpolyS_1,
...
secondpolyX_m, secondpolyY_m, secondpolyS_m,
profile2Tran_11, profile2Tran_12, profile2Tran_13, profile2Tran_14
profile2Tran_21, profile2Tran_22, profile2Tran_23, profile2Tran_24
profile2Tran_31, profile2Tran_32, profile2Tran_33, profile2Tran 34
generatrixFirstIndex_1, generatrixSecondIndex_1,
...
generatrixFirstIndex_g, generatrixSecondIndex_g

RULEDSHELL{3} topMat, bottomMat,
sideMat_1, sideMat_2, sideMat_3, sideMat_4, defaultMat,
n, m, g, status,
offset, thickness, flipped, trimmingBody,
preThickenTran_11, preThickenTran_12, preThickenTran_13, preThickenTran_14,
preThickenTran_21, preThickenTran_22, preThickenTran_23, preThickenTran_24,
preThickenTran_31, preThickenTran_32, preThickenTran_33, preThickenTran_34,
firstpolyX_1, firstpolyY_1, firstpolyS_1,
...
firstpolyX_n, firstpolyY_n, firstpolyS_n,
secondpolyX_1, secondpolyY_1, secondpolyS_1,

S

T

```
    ...
    statementn+m]
ENDIF
FOR variable_name = initial_value TO end_value [ STEP step_value ] NEXT variable_name
TOLER d
DEL TOP
TUBE n, m, mask,
        u1, w1, s1,
        ...
        un, wn, sn,
        x1, y1, z1, angle1,
        ...
        xm, ym, zm, anglem
TUBEA n, m, mask,
        u1, w1, s1,
        ...
        un, wn, sn,
        x1, y1, z1,
        ...
        xm, ym, zm
TUBE{2} top_material, bottom_material, cut_material,
        n, m, mask,
        u1, w1, s1, mat1,
        ...
        un, wn, sn, matn,
        x1, y1, z1, angle1,
        ...
        xm, ym, zm, anglem
```

U

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

[UI_TOOLTIP tooltiptext]

UI_DIALOG title [, size_x, size_y]

UI_GROUPBOX text, x, y, width, height

UI_INFIELD "name", x, y, width, height [,
method, picture_name,
images_number,
rows_number, cell_x, cell_y,
image_x, image_y,
expression_image1, text1,
...
expression_imagen, textn]

UI_INFIELD "name", x, y, width, height [, extra parameters ...]
[UI_TOOLTIP tooltiptext]

UI_INFIELD{2} name, x, y, width, height [,
method, picture_name,
images_number,
rows_number, cell_x, cell_y,
image_x, image_y,
expression_image1, text1,
...
expression_imagen, textn]

UI_INFIELD{2} name, x, y, width, height [, extra parameters ...]
[UI_TOOLTIP tooltiptext]

UI_INFIELD{3} name, x, y, width, height [,
method, picture_name,
images_number,
rows_number, cell_x, cell_y,
image_x, image_y,
expression_image1, text1, value_definition1,
...
[picIdxArray, textArray, valuesArray,
...]
expression_imagen, textn, value_definitionn]

V

W

innerRadius,
x1, y1, s1,
...
xn, yn, sn

WALLHOLE n, status,
x1, y1, mask1,
...
xn, yn, maskn
[, x, y, z]

WALLHOLE2 n, fill_control, fill_pen, fill_background_pen,
fillOrigoX, fillOrigoY, fillAngle,
x1, y1, s1,
...
xn, yn, sn

WALLHOLE2{2} n, frame_fill, fillcategory, distortion_flags,
fill_pen, fill_background_pen,
fillOrigoX, fillOrigoY,
mxx, mxy, myx, myy,
innerRadius,
x1, y1, s1,
...
xn, yn, sn

WALLLINE2 x1, y1, x2, y2

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

DO [statment1
statement2
...
statementn]
WHILE condition

X

mask1, mask2, mask3, mask4,
n,
x_start1, y_low1, x_end1, y_high1,
sill_depth1, frame_shown1,
...
x_startn, y_lown, x_endn, y_highn,
sill_depthn, frame_shownn,
m,
a1, b1, c1, d1,
...
am, bm, cm, dm,
status

XWALL_{3} left_material, right_material, vertical_material, horizontal_material,
height, x1, x2, x3, x4,
y1, y2, y3, y4,
t, radius,
log_height, log_offset,
mask1, mask2, mask3, mask4,
n,
x_start1, y_low1, x_end1, y_high1,
sill_depth1, frame_shown1,
...
x_startn, y_lown, x_endn, y_highn,
sill_depthn, frame_shownn,
m,
a1, b1, c1, d1,
...
am, bm, cm, dm,
status