Guide de référence GDL

GRAPHISOFT®

Graphisoft

Visitez le site web de Graphisoft à http://www.graphisoft.com pour la liste des distributeurs et la disponibilité des produits.

Guide de référence GDL

Le projet Nokia K2 utilisé pour l'illustration du coffret et des couvertures de manuel ArchiCAD sont la propriété de Helin & Co. Architects, Helsinki, Finlande.
Première impression.

Chapitre 1
Introduction

Le présent manuel est une référence complexe du langage de programmation GDL (Langage de Description Géometrique) de Graphisoft.

Ce manuel est recommandé pour les utilisateurs qui voudraient développer les possibilités représentées par les outils de construction et par la bibliothèque d'objets inclus avec les logiciel de Graphisoft. Il donne la description détaillée de GDL, y compris la définition de la syntaxe, des commandes et des paramètres.

1.1 L'écriture de Scripts GDL

Qu'est-ce que le GDL?

GDL est le **langage de programmation** paramétrique intégré d'ArchiCAD, semblable au BASIC. Il sert à décrire des objets solides en trois dimension et les symboles en deux dimensions qui les représentent sur le Plan: portes, fenêtres, meubles, éléments structuraux, escaliers, etc. Ces objets s'appellent **éléments de bibliothèque**.

Structure de l'Elément de bibliothèque ArchiCAD

Tout élément de bibliothèque décrit en GDL possède des **scripts**, c'est-à-dire des listes de commandes GDL qui construisent la forme 3D et le symbole 2D. Les Eléments de bibliothèque peuvent même inclure un descriptif pour le Métré dans ArchiCAD.

Les commandes du **script maître** seront exécutées avant tout script (comme s'ils étaient copiés devant les autres scripts de l'élément de bibliothèque).

Le **script 2D** contient la description paramétrique du dessin 2D. IL est possible de faire référence aux **données 2D binaires** de l'élément de bibliothèque (le contenu de la fenêtre Symbole 2D) en utilisant la commande `FRAGMENT2`. Si le script 2D est vide, les données 2D binaires seront utilisées pour afficher l'élément de bibliothèque sur le plan.

Le **script 3D** contient la description du modèle 3D. Il est possible de faire référence aux **données 3D binaires** (générées en important ou en enregistrant) avec la commande `BINARY`.

Le **script descriptif** contient les composants et descripteurs utilisés par les listes d'éléments, de composants et de zones. Il est possible de faire référence aux **données descriptives binaires** décrites dans les sections **composants** et **descripteurs** de l'élément de bibliothèque au moyen de la commande `BINARYPROP`. Si le script descriptif est vide, les données descriptives binaires seront utilisées pour la création de la liste.

Dans le **script de liste de valeurs** on peut définir des jeux de valeurs possibles pour les paramètres de l'élément de bibliothèque.

Le jeu de paramètres choisis dans la section **paramètres** sera utilisé par défaut pour l'élément de bibliothèque lors de son placement sur le plan.

L'**image de prévisualisation** est affichée dans le dialogue de l'élément de bibliothèque en naviguant dans la bibliothèque. Il est possible d'y faire référence à partir des scripts 3D et 2D au moyen des commandes `PICTURE` et `PICTURE2`.

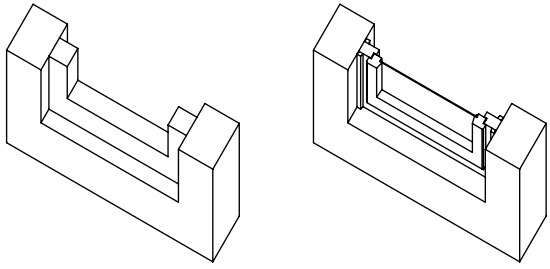
Dans la section **commentaire** on peut stocker des informations textuelles sur l'élément de bibliothèque.

ArchiCAD offre un environnement confortable pour l'écriture de scripts GDL, avec visualisation rapide et vérification de la syntaxe et des erreurs.

Analyser, décomposer et simplifier

Tous les objets, même les plus complexes, peuvent être décomposés en blocs de construction de formes géométriques simples. Avant de s'attaquer à l'écriture des scripts, il vaut mieux commencer par une courte analyse de l'objet à créer et par la définition des unités géométriques qui le composent. Une fois les composants déterminés, il est possible de les traduire dans le vocabulaire GDL. Si l'analyse est correcte, la synthèse des unités approchera la solution idéale.

Pour faire cette analyse, il est utile d'avoir des connaissances en géométrie descriptive.



Deux modèles de fenêtre

Commencez par des objets de dimensions faciles à définir et choisissez leurs formes les plus simples. Au fur et à mesure de votre apprentissage, vous pourrez vous orienter vers les formes plus complexes, en vous rapprochant de plus en plus de l'idéal.

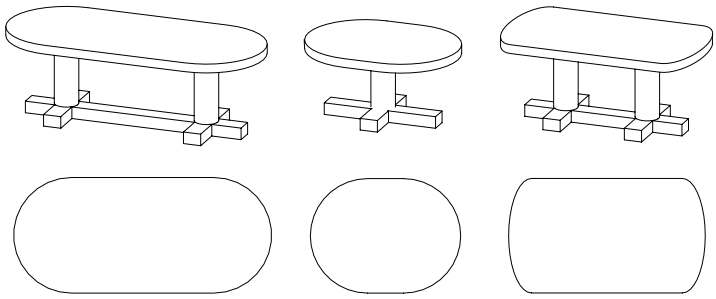
"Idéal" n'est pas forcément synonyme de "compliqué". Le degré d'élaboration de l'élément de bibliothèque idéal dépend de la nature du projet d'architecture donné. La fenêtre à gauche sur l'illustration convient parfaitement au style de la visualisation d'un modèle, tandis que celle de droite ajoute un brin de réalisme et de détail qui peuvent être utilisés ultérieurement dans la phase du projet d'exécution.

Elaboration

L'élaboration des objets paramétriques est donc fortement variée. Par exemple, les objets personnalisés utilisés à l'intérieur d'un bureau d'étude seront sans doute moins développés que ceux destinés à une distribution commerciale.

Si vos symboles n'ont que peu d'importance sur le plan et les changements de paramètres ne doivent pas être nécessairement visibles en 2D, vous pouvez délaisser totalement le script 2D paramétrique.

Même si les changements paramétriques doivent se voir en 2D, il n'est pas absolument nécessaire d'écrire un script 2D. Vous pouvez faire des modifications paramétriques dans la fenêtre du script 3D, utiliser la vue de dessus de l'objet modifié comme un nouveau symbole et enregistrer l'objet modifié sous un nouveau nom. Vous pouvez ainsi obtenir toute une série d'objets similaires dérivés du même original en en modifiant les valeurs de paramètres par défaut.



Les éléments de bibliothèque les plus complexes et les plus sophistiqués comprennent une description paramétrique 3D avec un script paramétrique 2D correspondant. Les changements apportés à leurs réglages affectent non seulement l'image de l'objet en 3D, mais aussi son apparence sur le Plan.

Premiers pas

Les besoins de votre projet, vos connaissances en programmation et en géométrie descriptive influenceront grandement à quel point vous commencerez à écrire des scripts GDL.

Ne commencez pas par les formes les plus complexes. Apprenez GDL pas à pas, essayez plusieurs possibilités et vous en tirerez rapidement profit. Nous vous recommandons d'aller du plus simple vers le plus difficile, suivant les niveaux détaillés plus loin.

Si vous connaissez déjà une langue de programmation du genre BASIC, vous pouvez commencer à vous familiariser avec GDL en lisant des scripts existants. Vous en apprendrez beaucoup en ouvrant les éléments de bibliothèque livrés avec ArchiCAD et en regardant leurs scripts 2D et 3D. Vous pouvez également enregistrer des éléments de Plan au format GDL et examiner le script ainsi obtenu.

Si vous vous n'êtes pas familiarisés avec le BASIC mais vous avez joué avec des éléments de construction, vous apprendrez facilement GDL par la pratique. Commencez par les commandes les plus simples suivant leurs effets dans la Fenêtre 3D de l'élément de bibliothèque.

Pour en savoir plus sur l'environnement de l'édition des éléments de bibliothèque dans ArchiCAD, voir la commande de menu Ouvrir Elément de bibliothèque au Chapitre 5 du Guide de référence ArchiCAD.

Commandes du premier niveau

Ces commandes sont faciles à comprendre et à utiliser. Elles ne nécessitent pas de connaissances en programmation. Vous pouvez pourtant créer de nouveaux objets très utile, même en n'utilisant que les commandes de ce premier groupe.

Formes simples

Les formes GDL sont des unités géométriques simples qui composent ensemble un élément de bibliothèque complexe. Les formes sont placées dans l'espace 3D en écrivant une instruction dans le script GDL.

Les commandes de forme consistent d'un mot-clef qui définit le type de la forme et d'un certain nombre de valeurs numériques ou de paramètres alphabétiques qui définissent ses dimensions.

Le nombre de valeurs requises varie de forme en forme.

Au début, vous pouvez éviter d'utiliser des paramètres calculées et vous en tenir aux valeurs fixes.

Les commandes de forme dont vous pouvez vous servir dès le début sont les suivantes:

En 3D :

```
BLOCK     CYLIND  SPHERE   PRISM
```

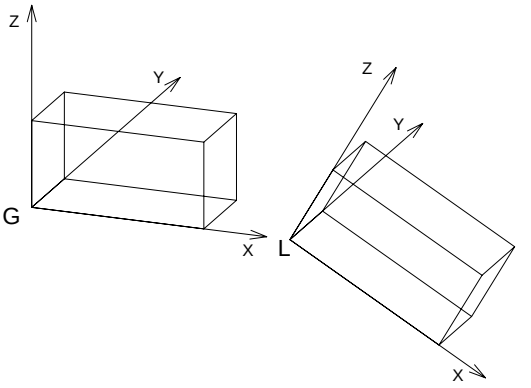
En 2D :

```
LINE2     RECT2   POLY2    CIRCLE2  ARC2
```

Les noms de ces formes s'expliquent d'eux-mêmes.

Transformation de coordonnées

Les transformations de coordonnées servent à définir la position, l'orientation et l'échelle de la forme suivante.




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

La Fenêtre 3D de l'élément de bibliothèque affiche optionnellement l'origine et la position courante globale (G) et locale (L) des axes de coordonnées pour l'objet.

Les transformations de coordonnées les plus simples sont les suivantes:

En 3D :

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

En 2D :

```
ADD2      ROT2
```

Les commandes commençant par `ADD` déplacent la forme suivante, tandis que les commandes `ROT` la font tourner autour de l'un de ses axes.

Commandes de niveau intermédiaire

Les commandes de ce groupe sont un peu plus complexe. Non qu'elles demandent des connaissances en programmation, mais parce qu'elles décrivent des formes plus complexes ou des transformations plus abstraites.

En 3D :

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

En 2D :

```
HOTSPOT2   POLY2_     TEXT2      FRAGMENT2
```

Normalement, pour ces commandes, il faut définir davantage de valeurs que pour les commandes simples. Pour certaines d'entre elles, vous devrez ajouter des valeurs d'état pour déterminer la visibilité des arêtes et des surfaces.

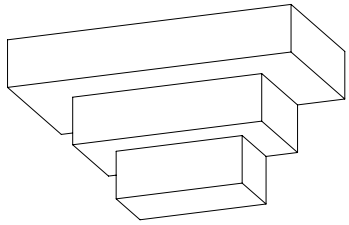
Transformations de coordonnées

En 3D :

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

En 2D :

```
MUL2
```



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

Les transformations commençant par `MUL` modifient l'échelle des formes qui les suivent, par exemple en déformant des cercles en ellipses ou des sphères en ellipsoïdes. Combinées avec des valeurs négatives, elles servent à définir des symétries. Les commandes de la seconde ligne affectent toutes les trois dimensions en même temps.

Commandes et fonctions avancées

Les commandes de ce troisième groupe représentent un degré de complexité accru, soit en raison de leur forme géométrique, soit parce qu'elles demandent de programmer effectivement dans GDL.

En 3D :

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

Il y a dans ce groupe des commandes de formes qui vous permettent de tracer un polygone spatial avec un polygone de base pour créer des surfaces courbes lisses. Certaines formes exigent également la définition de références de matières.

Avec les plans de coupe 3D, il est possible de générer des formes complexes arbitraires à partir de formes simples. Les commandes appropriées sont CUTPLANE, CUTPOLY, CUTPOLYA, CUTSHAPE et CUTEND.

En 2D :

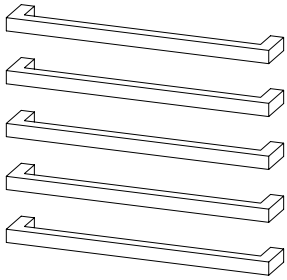
PICTURE2	POLY2_A
SPLINE2	SPLINE2_A

Instructions de contrôle de flux et de condition

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

Ces commandes sont sans doute familières pour tous ceux qui ont déjà fait de la programmation, mais elles sont tellement simples que l'on peut en comprendre le concept même si on est débutant en la matière.

Elles vous permettent de créer des parties de scripts répétitives pour placer un grand nombre de formes en écrivant peu de lignes, ou elles vous aident à prendre de décisions fondées sur des calculs précédents.



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

Paramètres

Le moment est venu où vous pouvez remplacer les valeurs numériques fixes par des noms de variables. Ceci permet de modifier l'objet en souplesse. Les variables sont accessibles dans le dialogue Options de l'élément de bibliothèque quand vous travaillez sur la Feuille de travail en Plan.

Appels de macro

Les formes GDL standard ne représentent pas une limite quelconque. Tout élément de bibliothèque existant peut devenir à son tour une forme GDL. Pour le placer, il suffit de l'évoquer par son nom dans le script et d'y transmettre les paramètres nécessaires, tout comme pour les commandes de formes standard.

GDL au niveau supérieur

Quand vous aurez bien maîtrisé les fonctions et commandes présentées plus haut, vous pourrez vous tourner vers les commandes restantes dont vous aurez sans doute besoin moins souvent.

Remarque : La quantité de mémoire installée sur votre ordinateur peut imposer une limite à la longueur de vos scripts GDL, à l'encastrement des appels de macro et au nombre de transformations.

Pour des informations additionnelles sur ce groupe de commandes, consultez le reste du manuel.

Pour obtenir des informations rapides sur les commandes disponibles et sur la structure de leurs paramètres, choisissez les commandes d'Aide appropriées dans le menu Aide.

1.2 Comment ArchiCAD génère une image 3D

La modélisation 3D dans ArchiCAD est basée sur l'arithmétique à virgule flottante, c'est à dire qu'il n'y a pas de limite pour la taille géométrique du modèle. Quelle que soit sa taille, il garde la même précision dans ses moindres détails .

Le modèle 3D que vous voyez finalement à l'écran est composé de **primitives géométriques**. Celles-ci sont stockées dans la mémoire de votre ordinateur sous forme binaire et ArchiCAD les génère conformément avec le plan que vous créez. Cette métamorphose entre les éléments de vos plans et ces données binaires est appelée la conversion 3D.

Les primitives sont les suivantes:

- tous les **nœuds** des composants de la construction

- toutes les **arêtes** liant ces nœuds

- tous les **polygones** de surface à l'intérieur des arêtes.

Les groupes de primitives sont conservés comme des '**corps**'. Les corps forment le modèle 3D. Toutes les fonctionnalités de visualisation 3D d'ArchiCAD - surfaces lisses, ombrages, matériaux brillants et transparents - sont basées sur cette structure de données.

L'espace 3D dans ArchiCAD

Le modèle 3D est créé dans un espace en trois dimensions mesuré par les axes X, Y, Z d'un **système de coordonnées maître**, dont l'origine est appelée l'**origine absolue**.

Dans ArchiCAD, vous pouvez voir l'origine absolue dans le coin inférieur gauche de votre feuille de travail en lançant l'application sans lire aucun document. De plus, l'origine absolue définit le niveau zéro auquel se réfèrent tous les étages de votre plan.

Quand, par exemple, vous placez un objet dans votre dessin, le plan horizontal définit sa position suivant les axes x et y du système de coordonnées maître. La position en z peut être réglée dans le

dialogue Options Objet. Cette position sera la base et la position par défaut du **système de coordonnées local** sur lequel est basé le script de l'objet donné.

Transformations de coordonnées

Dans le GDL, toute forme géométrique pouvant être générée par une simple commande est liée à la position actuelle du système de coordonnées. Par exemple, les BLOCKs rectangulaires sont liés à l'origine par un de leurs angles et la longueur, la largeur et la hauteur du bloc sont mesurées le long des trois axes, toujours dans la direction positive. Ainsi, la commande BLOCK ne demande que trois paramètres pour définir ses dimensions.

Que faut-il faire pour générer un autre bloc, déplacé et tourné? Avec la structure des paramètres BLOCK, ceci n'est pas possible, car elle ne possède pas les paramètres correspondants.

La solution consiste à déplacer le système de coordonnées dans la direction voulue avant d'utiliser la commande BLOCK. Avec les commandes de transformation de coordonnées, le système peut être déplacé et tourné autour de ses axes. Bien entendu, ces transformations n'affectent pas les formes déjà générées, seulement celles à créer.

L'interpréteur GDL

Quand un script GDL est exécuté, les algorithmes de l'interpréteur GDL présents dans ArchiCAD cherchent la position, la taille, l'angle de rotation, les paramètres utilisateur. Se basant sur ces informations, ils préparent un système de coordonnées local, correctement placé et orienté, prêt à recevoir les commandes GDL du script de l'objet. Chaque fois qu'une commande de forme est lue par l'interpréteur, celui-ci génère les primitives géométriques qui forment cet objet particulier.

A la fin de la lecture, le modèle binaire complet est présent en mémoire permettant ainsi de réaliser des rendus photoréalistes, ainsi que des études de luminosité.

ArchiCAD et ArchiFM contiennent un pré-compilateur et un interpréteur pour GDL. Le résultat de la passe de pré-compilation est stocké dans une partie invisible du fichier (dans la partie ressource). L'interprétation du document GDL utilise ce code, ce dispositif augmentant la vitesse d'analyse. Si le script GDL est modifié, un nouveau code est généré.

Une structure de donnée convertie d'un autre format de fichiers (par exemple, DXF, Zoom, Alias Wavefront) sera stockée dans la partie binaire des symboles ArchiCAD. Cette partie est référencée par l'instruction BINARY dans le script GDL.

L'ordre de l'analyse des scripts GDL

L'ordre dans lequel les éléments de bibliothèque placés sur le Plan sont analysés est indépendant du contrôle de l'utilisateur. L'ordre de l'analyse du script GDL est basé sur la structure interne des donnés, de plus, les opérations Annuler et Rétablir, ainsi que certaines modifications peuvent influencer cette hiérarchie. La seule exception à cette règle sont les scripts spéciaux de la bibliothèque active dont les noms commencent par "**MASTER_GDL**" ou "**MASTEREND_GDL**".

Les scripts dont le nom commence par "MASTER_GDL" sont toujours exécutés avant la conversion 3D, la création d'une Coupe/Façade, le lancement d'une création de liste et après le chargement de la bibliothèque active.

Les scripts dont le nom commence par "MASTEREND_GDL" sont exécutés après la conversion 3D, la création d'une Coupe/Façade ou d'une liste et si la bibliothèque active change (Charger bibliothèque, Ouvrir un projet, Nouveau projet, Quitter).

Ces scripts ne sont pas exécutés lorsque vous éditez des éléments de bibliothèque. Si votre bibliothèque contient un ou plusieurs fichiers de ce type, ils seront exécutés dans un ordre non défini.

Les scripts MASTER_GDL et MASTEREND_GDL peuvent inclure des définitions d'attributs, l'initialisation des variables globales utilisateur, des commandes 3D (effectives seulement sur le modèle 3D), des définitions de liste de valeurs (voir la commande VALUES dans le chapitre consacré aux Scripts non-géométriques) et des commandes spécifiques à des extensions GDL. Les attributs définis dans ces scripts seront fusionnés avec le jeu d'attributs ArchiCAD (les attributs ArchiCAD ayant les mêmes noms ne seront pas remplacés, tandis que les attributs provenant de GDL et non édités dans ArchiCAD seront toujours remplacés).

Chapitre 2

Eléments de syntaxe de base

Ce chapitre présente les éléments de base de la syntaxe de GDL, y compris les instructions, les labels, les identificateurs, les variables et les paramètres.

Les règles typographiques sont aussi expliquées en détail.

Les règles de syntaxe de GDL

GDL ne fait pas de différence entre les minuscules et les majuscules à l'exception des chaînes de caractère placées entre guillemets. La fin logique d'un script GDL est notée par une instruction `END` ou `EXIT` en fin de fichier ou par la fin physique du fichier.

Instructions Un programme GDL est composé d'instructions. Une instruction peut commencer par un mot-clé (définissant une forme GDL, des transformations de coordonnées, ou un contrôle du déroulement), par un nom de macro ou par un nom de variable suivi par '=' et une expression mathématique.

Ligne Les instructions sont dans des lignes séparées par des caractères "fin_de_ligne".
Une virgule (,) en dernière position signifie que l'instruction se poursuit sur la ligne suivante. Les deux points (:) sont utilisés pour séparer les instructions GDL dans une même ligne. Vous pouvez taper des commentaires après un point d'exclamation (!). Des lignes vides peuvent être insérées dans le document et un nombre quelconque d'espaces ou de tabulations peut séparer les opérandes des opérateurs. Après un mot-clé ou une macro, un espace ou une tabulation est obligatoire.

Label Une ligne peut commencer par un label ou étiquette. Le label est un nombre entier suivi de deux points (:). Le label est la référence de l'instruction de la ligne. Le compilateur vérifie qu'un label n'est présent qu'une seule fois. L'exécution du programme continue au label défini dans l'instruction `GOTO` ou `GOSUB`.

Caractères disponibles Le texte GDL est composé des minuscules et majuscules de l'alphabet anglais, des nombres et des caractères suivants:

```
  <espace>  _ (souligné)  ~  !  :  ,  ;  .  +  -
*  /  ^  =  <  >  #  (  )  |(barre verticale)
"  '  `  ´  ″  ‟  ’  ‘    <fin_de_ligne>
```

Chaînes de caractères Toute chaîne de caractère placée entre guillemets (",',`,´,”,’,“,‘), ou toute chaîne de caractères sans guillemets qui ne figure pas dans un script comme identificateur avec une valeur assignée (appel de macro, nom d'attribut, nom de fichier). Les chaînes sans guillemets sont converties en majuscules, il est donc recommandé d'utiliser les guillemets. La longueur maximale d'une chaîne est de 255 caractères.
Le caractère '\' a un rôle spécial. Son interprétation dépend du caractère suivant.

\\	'\' le caractère lui-même
\n	nouvelle ligne
\t	tabulation
\ *nouvelle ligne*	continuer chaîne dans la ligne suivante sans commencer une nouvelle ligne
\autres	incorrect, provoque une alerte

Exemples:

```
"Ceci est une chaîne"
“bassin 2”*1.5
'Ne pas utiliser de guillemets différents'
```

Identificateurs Les identificateurs sont des chaînes de caractères spéciales:

- pas plus de 255 caractères,
- commencent par une lettre de l'alphabet, ou par le caractère '_' ou '~',
- comprennent des lettres, chiffres et les caractères '_' ou '~'.

Les majuscules et les minuscules ne sont pas distinguées.

Les identificateurs peuvent être des mots-clefs GDL, des variables globales ou locales ou des chaînes (noms). Les mots-clefs et les noms de variables globales sont déterminés par ArchiCAD, tout autre identificateur peut être utilisé comme nom de variable.

Variables Les programmes GDL peuvent contenir des variables numériques ou constituées de caractères (définies par leurs identificateurs), des chiffres et des chaînes de caractères.
Il y a deux ensembles de variables: locales et globales.

Tous les identificateurs qui ne sont ni des mots-clefs, ni des variables globales, ni des noms d'attributs, ni des noms de macro ou de fichier sont considérés comme des variables locales. S'ils ne sont pas initialisés, leur valeur est de 0.0. Les variables locales sont empilées avec les appels de macros et l’interpréteur rend leur valeur en retour de macro. Les variables globales ne sont pas empilées pendant l’appel de macro.

Les variables globales ont des noms réservés (la liste des variables disponibles dans ArchiCAD est fournie dans l'**Annexe**). Elles ne sont pas empilées dans les appels de macro, permettant à l'utilisateur de stocker des valeurs spéciales pour le modèle et de simuler des codes de retour de macro. Les variables globales utilisateur peuvent être définies dans n'importe quel script, mais elles n'auront d'effet que dans les scripts subséquents. Pour assurer

que le script souhaité soit traité en premier, déclarez ces variables dans l'élément de bibliothèque MASTER_GDL. Les autres variables globales peuvent être utilisées dans les scripts pour communiquer avec ArchiCAD.

En utilisant la commande "=", vous pouvez assigner une valeur numérique ou de chaîne aux variables locales et globales.

Paramètres Les identificateurs figurant dans la liste des paramètres d'un élément de bibliothèque sont appelés des paramètres. Les identificateurs de type paramètre ne peuvent pas excéder les 32 caractères. A l'intérieur du script, les mêmes règles sont en vigueur que pour les variables locales.
Les paramètres des fichiers GDL de type texte seulement sont identifiés par les lettres A … Z.

Types simples Les variables, paramètres et expressions peuvent être de deux types simples: numériques et chaînes de caractères.
Expressions numériques: constantes, variables ou paramètres numériques, fonctions retournant des valeurs numériques, et toute combinaison de ces nombres dans une opération.

Expressions de type chaîne de caractères: constantes, variables ou paramètres, fonctions qui retournent une chaîne, et toute combinaison de ces chaînes dans une opération dont le résultat est une chaîne.

Types dérivés Des variables et des paramètres peuvent être aussi des matrices, et les paramètres peuvent être des listes de valeurs de type simple.
Les *Matrices* sont des tables à une ou deux dimensions de valeurs numériques et ou de chaîne de caractères, auxquelles on accède directement par index.

Les *Listes de valeur* sont des jeux de valeurs numériques ou de chaînes possibles. On peut les assigner aux paramètres dans le script de liste de valeurs de l'élément de bibliothèque ou dans le script MASTER_GDL, et apparaîtront dans la liste des paramètres comme pop-up menu.

[aaa] Les crochets signifient que les éléments contenus sont optionnels (s'ils sont en gras, ils doivent être entrés tels quels).

. . . L'élément précédent peut être répété

nomvar Le nom d'une variable GDL

chaîne Une chaîne de caractères (ne peut inclure les guillements)

CHAINE_GRAS Doit être entré dans le style affiché

CHAINE_MAJUSCULE

caractères spéciaux Doivent être entré tels quels

autre_chaîne_minuscule_dans_liste_paramètres
N'importe quelle expression GDL

Chapitre 3
Transformation de coordonnées

Ce chapitre décrit les différents types de transformation utilisés dans GDL (translation, modification de l'échelle, rotation du système de coordonnées) et la manière dont ces transformations sont interprêtées et gérées.

3.1 A propos des transformations

Dans **GDL**, tous les éléments géométriques sont liés à un système de coordonnées local. Par exemple, un angle de bloc est l'origine et ses côtés sont les plans x-y, x-z et y-z.

Le positionnement d'un élément dans la position désirée se fait en deux étapes. D'abord, il faut déplacer le système à la position désirée. Ensuite, générer l'élément. Tout mouvement, rotation, ou étirement du système de coordonnées le long ou autour d'un axe est appelé une transformation.

Les transformations sont stockées dans une pile, l'interprétation commence donc par la dernière transformation. Les scripts héritent de cette pile de transformation, ils peuvent insérer de nouveaux éléments mais ne peuvent supprimer que celles qui sont définies localement. Il est possible de supprimere une ou plusieurs transformations définies dans le script courant. En revenant d'un script, les transformations définies localement sont effacées de la pile.

3.2 Transformations pour le modèle 3D

ADDX dx
ADDY dy
ADDZ dz

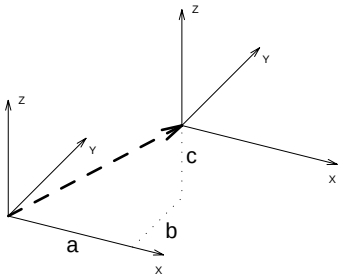
Ces commandes déplacent le système de coordonnées local de dx, dy ou dz respectivement le long de l'axe donné.

ADD dx, dy, dz

Remplace la séquence ADDX dx : ADDY dy : ADDZ dz.

Cette instruction n'a qu'une entrée dans la pile et peut donc être supprimée par DEL 1.

Exemple :




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

MULX mx
MULY my
MULZ mz

Ces commandes font subir une modification de l'échelle du système de coordonnées local le long de l'axe donné. Des valeurs mx, my, mz négatives induisent des images symétriques.

MUL mx, my, mz

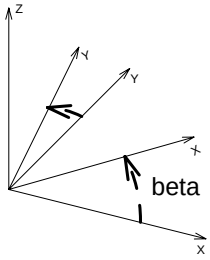
Remplace la séquence MULX mx : MULY my : MULZ mz.

Cette instruction n'a qu'une entrée dans la pile et peut donc être supprimée par DEL 1.

ROTX alphax
ROTY alphay
ROTZ alphaz

Ces commandes font subir une rotation au système de coordonnées de alphax, alphay, alphaz degrés respectivement le long de l'axe défini, dans le sens inverse des aiguilles d'une montre.

Exemple :




```
ROTZ beta
```

ROT x, y, z, alpha

Fait subir une rotation au système de coordonnées local autour de l'axe défini par le vecteur (x, y, z) de alpha degrés dans le sens inverse des aiguilles d'une montre..

Cette instruction n'a qu'une entrée dans la pile et peut donc être supprimée par DEL 1.

XFORM a_{11}, a_{12}, a_{13}, a_{14},
a_{21}, a_{22}, a_{23}, a_{24},
a_{31}, a_{32}, a_{33}, a_{34}

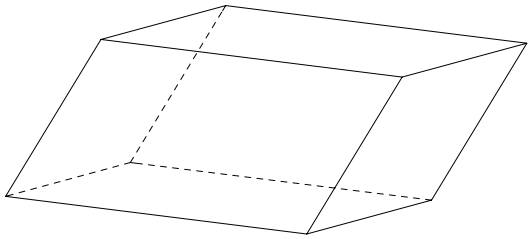
Définit une matrice de transformation complète. Utilisée normalement pour la génération automatique de code GDL. Une seule entrée dans la pile.

$x' = a_{11} * x + a_{12} * y + a_{13} * z + a_{14}$

$y' = a_{21} * x + a_{22} * y + a_{23} * z + a_{24}$

$z' = a_{31} * x + a_{32} * y + a_{33} * z + a_{34}$

Exemple:



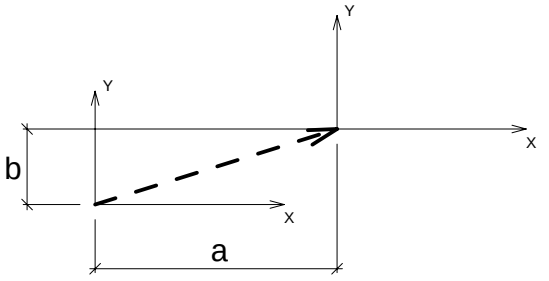
```
A=60
B=30
XFORM   2,      COS(A), COS(B)*0.6,  0,
        0,      SIN(A), SIN(B)*0.6,  0,
        0,      0,      1,           0
BLOCK   1,      1,      1
```

3.3 Transformations pour le Symbole 2D

Ces instructions sont les équivalents en 2D des transformations ADD, MUL et ROTZ utilisées en 3D.

ADD2 x, y

Exemple:

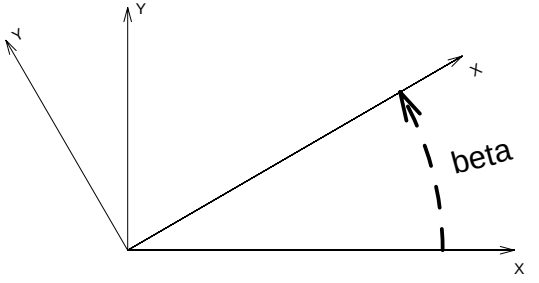



```
ADD2  a, b
```

MUL2 x, y

ROT2 alpha

Exemple:




```
ROT2 beta
```

3.4 Gestion de la pile de transformations

DEL `n [, beg_with]`

Supprime les n entrées précédentes dans la pile.

Si le paramètre beg_with n'a pas été spécifié, les n entrées précédentes sont supprimées. Le système de coordonnées local revient à la position précédente.

Si un paramètre beg_with a été spécifié, les n entrées précédentes sont supprimées en commençant par la transformation à laquelle la valeur "beg_with" fait référence. La numérotation commence par 1. Si un paramètre beg_with a été spécifié et que n est négatif, supprime vers la fin.

Si le nombre de transformations définies dans le script courant est inférieur à n, seules les transformations définies seront supprimées.

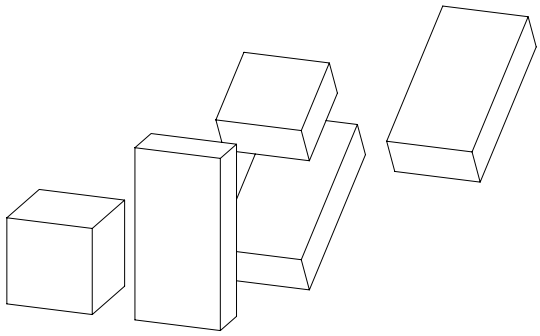
DEL TOP

Supprime toutes les transformations courantes du script courant.

NTR `( )`

Retourne le nombre courant de transformations..

Exemple:



```
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              !Deletes the ADDX 2
                                  !transformation
BLOCK       1,     0.5,   1

DEL         1,     NTR()-2        !Deletes the ADDZ 1.5
                                  !transformation
BLOCK       1,     0.5,   2

DEL         -2,    3              !Deletes the ROTX -60
                                  !and ADDY 2.5
                                  !transformations
BLOCK       1,     0.5,   2
```

Chapitre 4
Eléments plan en 3D

Les éléments plan en 3D présentés dans ce chapitre peuvent être utilisés dans les scripts 3D, permettant de définir dans l'espace 3D des points, des lignes, des arcs, des cercles et des polygons plan.

HOTSPOT x, y, z [,unID]

Point chaud 3D au point (x, y, z).

unID est l'identificateur unique du point chaud dans le script 3D. Cela peut se révéler utile si le nombre de points chauds est variable.

LIN_ x_1, y_1, z_1, x_2, y_2, z_2

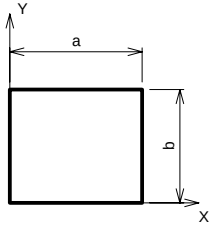
Un segment de ligne entre deux points $P_1(x_1,y_1,z_1)$ et $P_2(x_2,y_2,z_2)$.

RECT a, b

Un rectangle dans le plan x-y de côtés a et b.

Restriction de paramètres:

$a, b \geq 0$

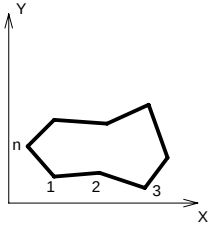


POLY n, x_1, y_1, . . . x_n, y_n

Un polygone à n arêtes dans le plan x-y. Les coordonnées du nœud$_i$ sont (x_i, y_i, 0).

Restriction de paramètres:

$n \geq 3$



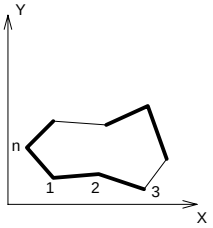
POLY_ n, x_1, y_1, $mask_1$, . . . x_n, y_n, $mask_n$

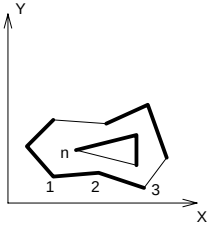
Similaire à l'instruction POLY, mais des arêtes peuvent être omises. Si $mask_i$=0, l'arête partant du nœud (x_i,y_i) est omise. Si $mask_i$=1, le nœud est affiché.

$mask_i$ = -1 est utilisé pour définir directement des trous. Voir PRISM_ pour davantage de détails.

Restriction de paramètres:

$n \geq 3$





PLANE n, x_1, y_1, z_1, . . . x_n, y_n, z_n

Un polygone avec n arêtes sur un plan arbitraire. Les coordonnées de nœud$_i$ sont (x_i, y_i, z_i). Le polygone doit trouver sur un plan pour donner un résultat correct dans les vues ombrées et les rendus, mais cette condition n'est pas vérifiée.

Restriction de paramètres:

$n \geq 3$

PLANE_ n, x_1, y_1, z_1, $mask_1$, . . . x_n, y_n, z_n, $mask_n$

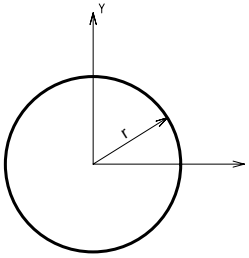
Similaire à l'instruction PLANE, mais des arêtes peuvent être omises comme pour POLY_.

Restriction de paramètres:

$n \geq 3$

CIRCLE r

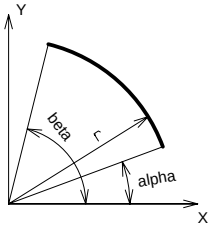
Un cercle dans le plan x-y avec son centre à l'origine et un rayon r.



ARC r, alpha, beta

Un arc (dans le mode Fil-de-fer) /secteur (dans les autres modes) sur le plan avec son centre à l'origine, de l'angle alpha à l'angle bêta et de rayon r.

Alpha et bêta sont exprimés en degrés.



Chapitre 5
Eléments 3D

Ce chapitre traite toutes les commandes pour la création des formes 3D disponibles dans GDL, des formes de base à la constitution des formes les plus complexes à partir de polylignes. Il contient aussi les éléments pour la visualisation (sources lumineuses, éléments d'image), ainsi que la définition d'un texte 3D.

Il explique aussi en détail les primitives de la structure des données 3D contenant noeuds, vecteurs, arêtes, polygones, corps et matières, suivies de l'interprétation des données binaires et des commandes liées aux plans de coupe.

5.1 Formes de base

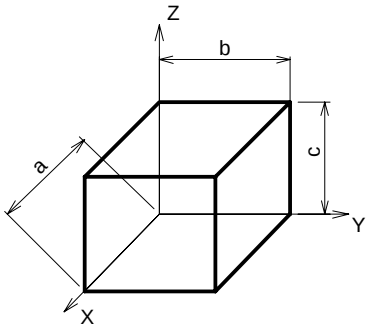
BLOCK a, b, c
BRICK a, b, c

Le premier angle du bloc est l'origine locale et les arêtes sur les axes x, y et z ont respectivement les longueurs a, b et c..

Des valeurs zéro donnent des blocs dégénérés (rectangle ou ligne).

Restriction de paramètres:

a, b, c ≥ 0

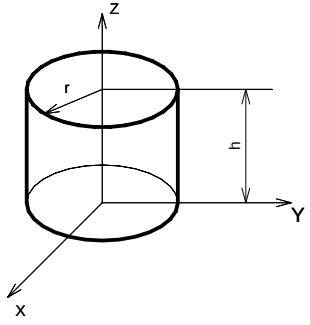


CYLIND h, r

Cylindre droit, sur l'axe z, de hauteur h et de rayon r.

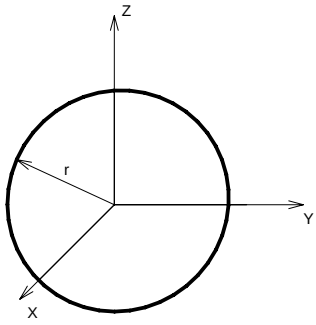
Si h est zéro, un cercle est généré dans le plan x-y.

Si r est zéro, une ligne est générée le long de l'axe z.



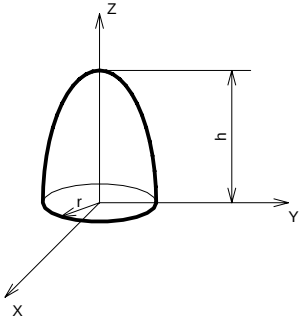
SPHERE r

Sphère centrée sur l'origine, de rayon r.



ELLIPS h, r

Demi ellipsoïde. Sa coupe transversale avec le plan x-y est un cercle centré sur l'origine, de rayon r. Le demi-axe selon z est de longueur h.



Exemple :

```
ELLIPS r, r         ! hemisphere
```

CONE h, r_1, r_2, $alpha_1$, $alpha_2$

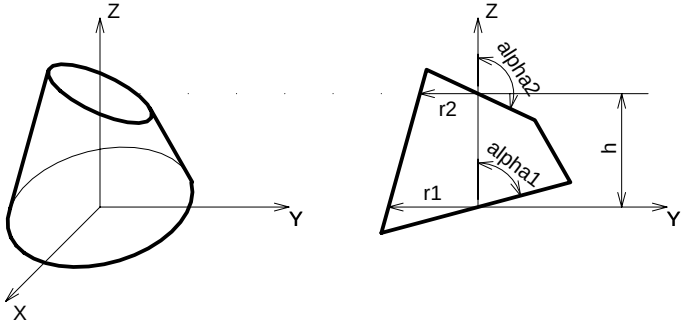
Portion de cône avec $alpha_1$ et $alpha_2$ représentant les angles d'inclinaison sur l'axe z des surfaces de dessus et de dessous, r_1 et r_2 les rayons des cercles et h la hauteur le long de z.

Si h est zéro, les valeurs de $alpha_1$ et $alpha_2$ sont ignorées et un anneau est généré dans le plan x-y.

$Alpha_1$, $alpha_2$ sont exprimés en degrés.

Restriction de paramètres:

$0 < alpha_1 < 180° \text{ and } 0 < alpha_2 < 180°$



Exemple :

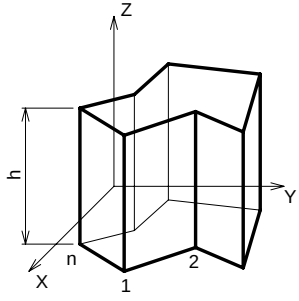
```
CONE h, r, 0, 90, 90       ! a regular cone
```

PRISM n, h, x_1, y_1, . . . x_n, y_n

Prisme droit de base polygonale dans le plan x-y (voir les paramètres de POLY). La hauteur sur l'axe z est abs(h). Une valeur négative de h signifie que le deuxième polygone est en dessous du plan x-y.

Restriction de paramètre :

$n \geq 3$

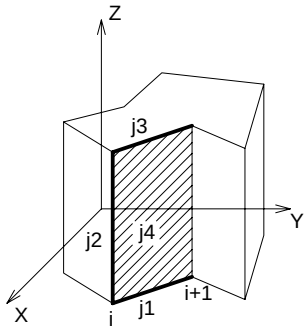


PRISM_ n, h, x_1, y_1, *mask*$_1$, . . . x_n, y_n, *mask*$_n$

Similaire à l'instruction **PRISM**, mais des arêtes ou des faces horizontales peuvent être omises.

Restriction de paramètres :

$n \geq 3$



Le nombre $mask_i$ est un nombre entier binaire (entre 0 et 15 ou 64 et 79) ou -1.

$mask_i = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 64*j_7$
où j_1, j_2, j_3, j_4, j_7 peuvent être 0 ou 1.

Les nombres j_1, j_2, j_3, j_4 représentent la présence (1) ou l'absence (0) des arêtes et des faces.

j_1 : arête horizontale inférieure

j_2 : arête verticale

j_3 : arête horizontale supérieure

j_4 : côté

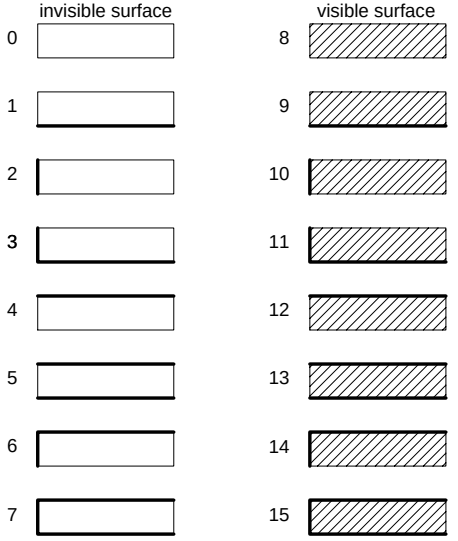
j_7 : valeur de masque spéciale additionnelle prenant uniquement effet si j_2 est égal à 1 et régit la visibilité dépendant du point de vue de l'arête verticale courante

j_2 = 0: arête verticale toujours invisible

j_2 = 1 et j_7= 1: arête verticale visible seulement si c'est un contour vu de la direction courante de la vue

j_2 = 1 et j_7 = 0: arête verticale toujours visible

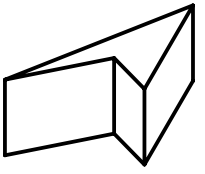
Valeurs possibles du masque (les lignes épaisses représentent les arêtes visibles) :



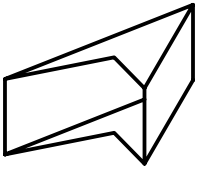
$Mask_i$ = -1 est utilisé pour définir des trous directement dans le prisme. Il marque la fin d'un contour et le début d'un trou dans le contour. Les coordonnées avant cette valeur doivent être les mêmes que les coordonnées du premier point du contour/trou. Si vous avez utilisé la valeur de masque -1, la dernière valeur de masque dans la liste des paramètres doit être -1, pour marquer la fin du dernier trou.

Les trous doivent être disjoints et les intersections internes sont interdites dans le polygone pour obtenir un résultat correct dans les rendus et les ombrages.

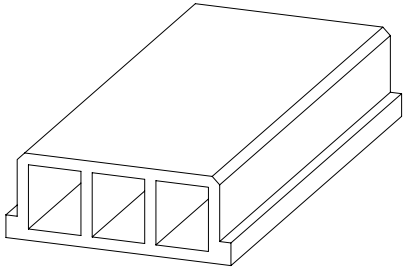
Exemples :



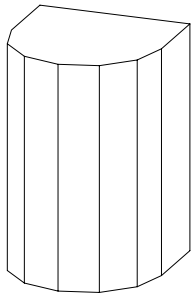
```
PRISM_ 4,1,
0,0,15,
1,1,15,
2,0,15,
1,3,15
```



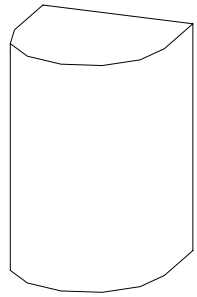
```
PRISM_ 4,1,
0,0,7,
1,1,5,
2,0,15,
1,3,15
```



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



$j_7 = 0$



$j_7 = 1$

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

CPRISM_ topmat, botmat, sidemat,
n, h, x_1, y_1, $mask_1$, . . . x_n, y_n, $mask_n$

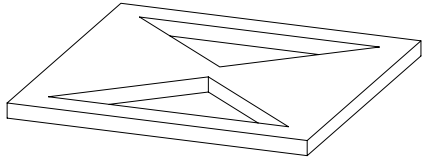
Extension de l'instruction PRISM_, les trois premiers paramètres sont utilisés pour le nom/index de matière des surfaces supérieure, inférieure et latérale. Les autres paramètres sont les mêmes que dans PRISM_.

Restriction de paramètres:

$n \geq 3$

Voir aussi l'instruction MATERIAL au Chapitre "Attributs" et la fonction IND dans l'Annexe.

Exemple :



```
CPRISM_  "Iron", 0, T_,    !"Iron" is a predefined
                           ! material.
                           ! 0 is a general
                           ! material.
                           ! T_ is  a global
                           ! variable (a material
                           ! index)
      13, 0.2,
      0,   0,   15,
      2,   0,   15,
      2,   2,   15,
      0,   2,   15,
      0,   0,   -1,  ! end of the contour

      0.2, 0.2, 15,
      1.8, 0.2, 15,
      1.0, 0.9, 15,
      0.2, 0.2, -1,  ! end of first hole
      0.2, 1.8, 15,
      1.8, 1.8, 15,
      1.0, 1.1, 15,
      0.2, 1.8, -1   ! end of second hole
```

BPRISM_ topmat, botmat, sidemat,
n, h, *radius*,
x_1, y_1, $mask_1$, . . . x_n, y_n, $mask_n$

Une prisme courbe lisse, basée sur la même structure de données que l'élément droit CPRISM_.

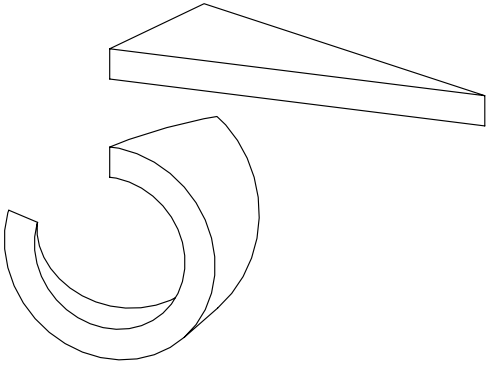
Le seul paramètre additionel est le rayon *(radius*).

Dérivée d'une CPRISM_ correspondante en courbant le plan x-y en un cylindre tangentiel à ce plan.

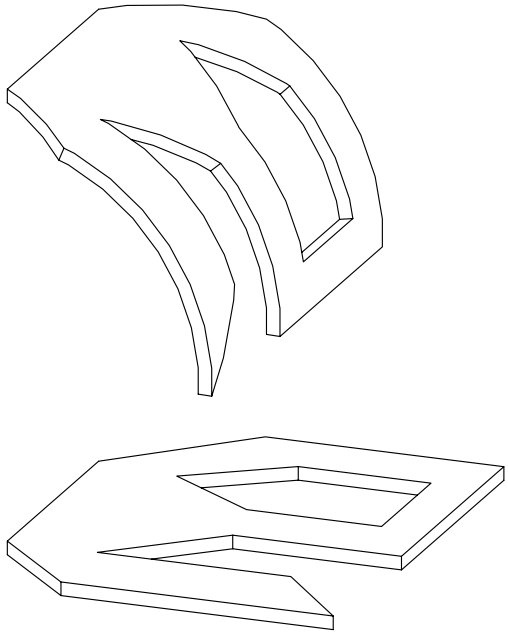
Les arêtes le long de l'axe x sont transformés en arcs circulaires; les arêtes le long de l'axe y restent horizontale; les arêtes le long de z sont de direction radiale.

Voir aussi BWALL_.

Exemples (avec les CPRISM_-s correspondants):



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



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

FPRISM_ topmat, botmat, sidemat, *hillmat*,
n, thickness, *angle*, *hill_height*,
x_1, y_1, $mask_1$,
...
x_n, y_n, $mask_n$

Similaire à l'instruction PRISM_ avec les paramètres additionnels hillmat, angle et hill_height parameters. Une partie en pente est ajoutée au haut du prisme droit.

hillmat: la matière de côté de la partie en pente

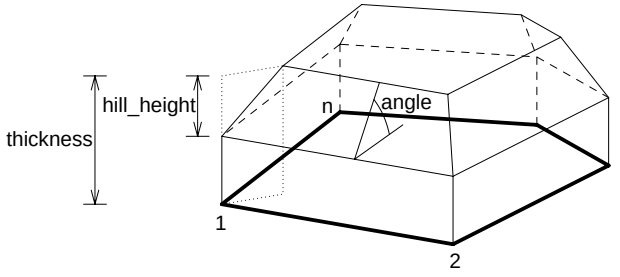
angle: l'angle d'inclinaison des arêtes latérales de la pente. Restriction: $0 \leq angle < 90$. Si angle = 0 les arêtes latérales de la pente forment, sur une vue orthogonale, un quart de cercle avec la résolution spécifiée au moyen de l'instruction RESOL.

hill_height: la hauteur de la pente. Notez que le paramètre épaisseur représente la hauteur totale du FPRISM.

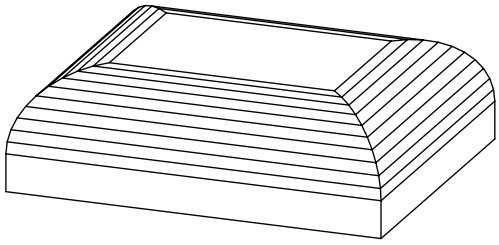
Restriction de paramètres:

$n \geq 3$, hill_height < thickness

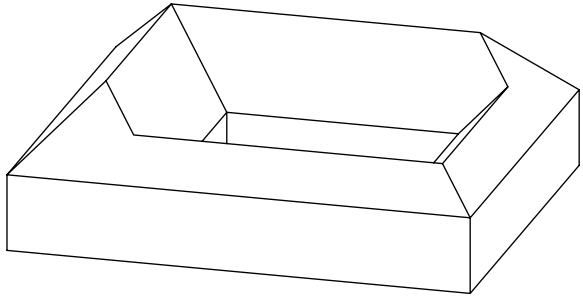
$Mask_i$ = -1 est utilisé pour définir directement des percements dans le prisme. Voir la description sous PRISM_ .



Exemples :



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



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

SPRISM_ topmat, botmat, sidemat,
n, x_b,y_b, x_e, y_e, h, *angle*,
x_1, y_1, $mask_1$, . . . x_n, y_n, $mask_n$

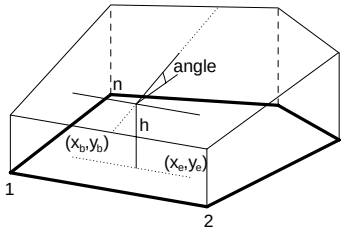
Extension de l'instruction CPRISM_ avec la possibilité d'avoir un polygone supérieur non parallèle au plan x-y. La définition du plan supérieur est similaire à la définition du plan de l'instruction CROOF_. La hauteur du prisme est définie à la ligne de référence. Les polygones supérieur et inférieur ne peuvent s'intersecter.

Paramètres additionnels:

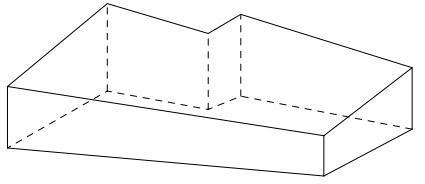
x_b, y_b, x_e, y_e: coordonnées des deux extrémités de la ligne de référence (vecteur),

angle: angle de rotation du polygone supérieur autour de la ligne de référence orientée donnée, exprimé en degrés (dans le sens contraire des aiguilles d'une montre),

Remarque: toutes les coordonnnées calculées en z des nœuds du polygone supérieur doivent être supérieures ou égales à 0.



Exemple:



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

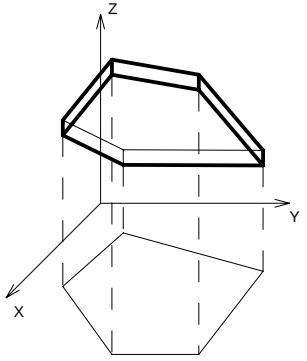
SLAB n, h, x_1, y_1, z_1, . . . x_n, y_n, z_n

Prisme oblique. Les faces latérales sont toujours perpendiculaires au plan x-y, la base est un polygone tourné autour d'un axe parallèle au plan x-y. Une valeur négative de h signifie que le deuxième polygone de base est au-dessous du premier.

L'interpréteur ne vérifie pas que les points sont vraiment sur un même plan. Les nœuds en dehors du plan donnent des effets de rendus et d'ombrages bizarres.

Restriction de paramètres:

$n \geq 3$



SLAB_ n, h, x_1, y_1, z_1, *mask*$_1$, . . . x_n, y_n, z_n, *mask*$_n$

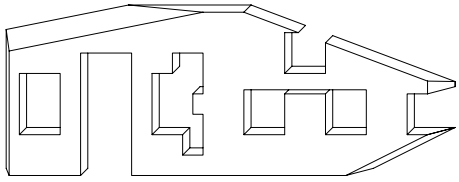
Similaire à SLAB mais des arêtes ou des faces horizontales peuvent être omises. Cette instruction est analogue à PRISM_.

CSLAB_ *topmat*, *botmat*, *sidemat*,
n, h, x_1, y_1, z_1, mask$_1$, . . . x_n, y_n, z_n, mask$_n$

Extension de l'instruction SLAB_, les trois premiers paramètres sont utilisés pour le nom/index de matière des surfaces supérieure, inférieure et latérale. Les autres paramètres sont les mêmes que dans SLAB_.

Voir aussi l'instruction MATERIAL au chapitre Attributs et la fonction IND dans l'Annexe.

CWALL_ leftmat, rightmat, sidemat,
height, x_1, x_2, x_3, x_4, t,
$mask_1$, $mask_2$, $mask_3$, $mask_4$,
n,
$xbeg_1$, $lower_1$, $xend_1$, $upper_1$, $framevis_1$,
...
$xbeg_n$, $lower_n$, $xend_n$, $upper_n$, $framevis_n$,
m,
a_1, b_1, c_1, d_1,
...
a_m, b_m, c_m, d_m



leftmat, rightmat, sidemat:

Noms/index de matière pour les surfaces gauche, droite et de côté.

Les côtés droit et gauche du mur suivent l'axe x.

Voir aussi l'instruction MATERIAL au chapitre Attributs et la fonction IND dans l'Annexe.

La ligne de référence du mur est toujours modifiée pour coïncider avec l'axes x. Les côtés du mur sont dans le plan x-z.

height :

La hauteur du mur par rapport à sa base.

x_1, x_2, x_3, x_4 :

Les extrémités projetées du mur se trouvant dans le plan x-y, comme vous le voyez plus bas. Si le mur est isolé, alors $x_1 = x_4 = 0$, $x_2 = x_3$ = longueur du mur.

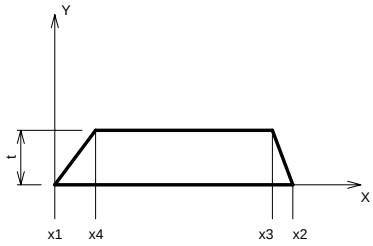
t : l'épaisseur du mur.

$t < 0$ si le corps du mur est à droite de l'axe x,

$t > 0$ si le corps du mur est à gauche de l'axe x,

les trous:

$t = 0$ le mur est représenté par un polygone et des 'cadres' sont générés autour des trous.



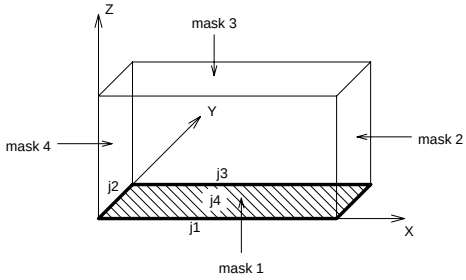
$mask_1$, $mask_2$, $mask_3$, $mask_4$:

Contrôlent la visibilité des arêtes et des polygones latéraux.

$mask_i = j_1 + 2*j_2 + 4*j_3 + 8*j_4$

où j_1, j_2, j_3, j_4 peuvent être 0 ou 1.

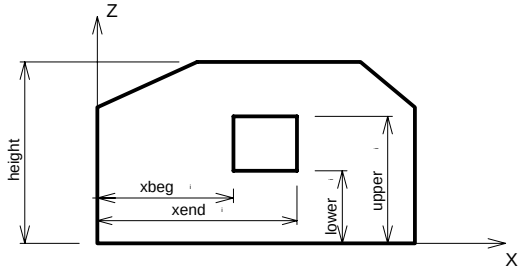
Les nombres j_1, j_2, j_3, j_4 représentent la présence (1) ou l'absence (0) des nœuds et des côtés.



n : le nombre d'ouvertures dans le mur.

$xbeg_i$, $lower_i$, $xend_i$, $upper_i$:

coordonnées des ouvertures (voir ci-dessous).

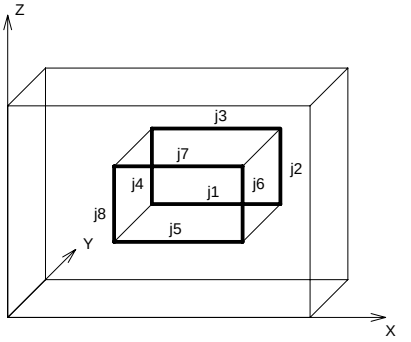


`framevis`$_i$:

1 si le arêtes du trou sont visibles, sinon 0.

Les valeurs négatives définissent la visibilité de chacune des arêtes de l'ouverture.

framevis$_i$ = - (1*j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7 + 128*j_8) où j_1 j_2 .. j_8 peuvent être 0 ou 1. Les nombres de j_1 à j_4 régissent la visibilité des arêtes de l'ouverture du côté gauche de la surface du mur, tandis que ceux de j_5 8 affectent les arêtes de droite, comme on le voit sur l'illustration.



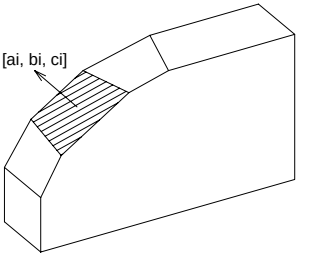
Une arête perpendiculaire à la surface du mur est visible s'il y a des arêtes visibles partant de chacune de ses extrémités.

m: le nombre des plans de coupe

a_i, b_i, c_i, d_i :

coefficients de l'équation définissant le plan de coupe [a_i*x + b_i*y + c_i*z = d_i].

Les parties sur le côté positif du plan de coupe (a_i*x + b_i*y + c_i*z > d_i) seront effacées.



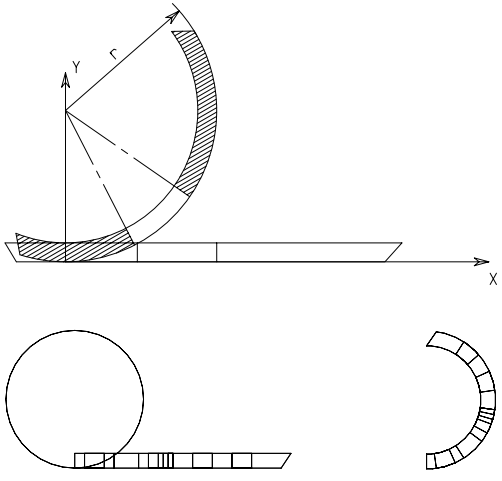
BWALL_ leftmat, rightmat, sidemat,
height, x_1, x_2, x_3, x_4, t, *radius*,
$mask_1$, $mask_2$, $mask_3$, $mask_4$,
n,
$xbeg_1$, $lower_1$, $xend_1$, $upper_1$, $framevis_1$,
. . .
$xbeg_n$, $lower_n$, $xend_n$, $upper_n$, $framevis_n$,
m,
a_1, b_1, c_1, d_1,
. . .
a_m, b_m, c_m, d_m

Mur courbe lisse, basé sur la même structure de données que l'élément de mur simple CWALL_.

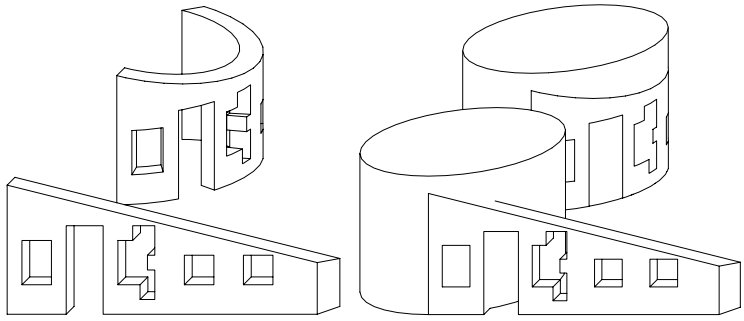
Le seul paramètre additionnel est le rayon *(radius)*.

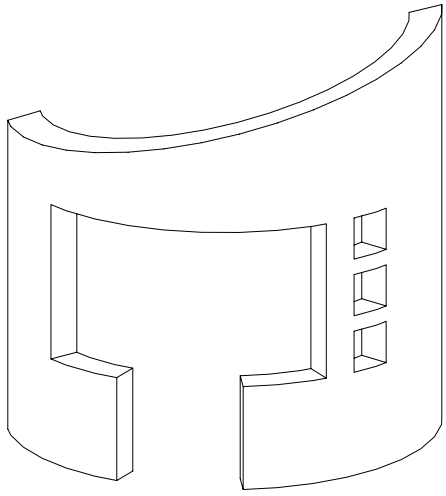
Dérivé du CWALL_ correspondant en courbant le plan x-z en un cylindre tangentiel à ce plan.

Les bords le long des axes x sont transformés en arcs circulaires, les bords le long des axes y seront de direction radiale et les bords verticaux restent verticaux. La courbature est approximée par le nombre de segments définis par la directive RESOL, comme pour les sphères et les cylindres. Voir CWALL_ pour des détails.



Exemple : un BWALL_ et le CWALL_ correspondant





```
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
```

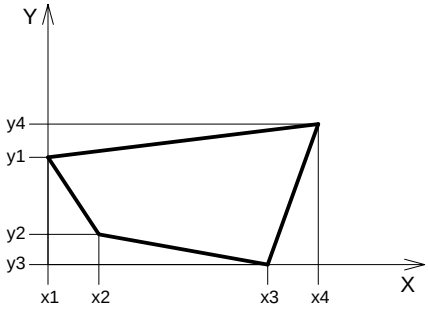
XWALL_ `leftmat, rightmat, sidevmat, sidehmat,`
`height, x1, x2, x3, x4,`
`y1, y2, y3, y4`
`t, radius`
`logheight, logoffset,`
`mask1, mask2, mask3, mask4,`
`n,`
`xbeg1, lower1, xend1, upper1, framevis1,`
`...`
`xbegn, lowern, xendn, uppern, framevisn,`
`m,`
`a1, b1, c1, d1,`
`...`
`am, bm, cm, dm,`
`status`

Définition de mur étendu basée sur la même structure de données que l'élément `BWALL`.

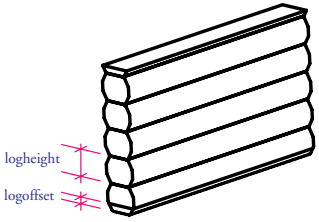
Paramètres supplémentaires :

`sidevmat, sidehmat` : le nom/index de la matière du côté vertical/horizontal

`y1, y2, y3, y4` : les points d'extrémité projetés du mur résidant sur le plan x-y, comme indiqué ci-dessous



`logheight, logoffset` : les paramètres supplémentaires pour composer un mur à partir de rondins

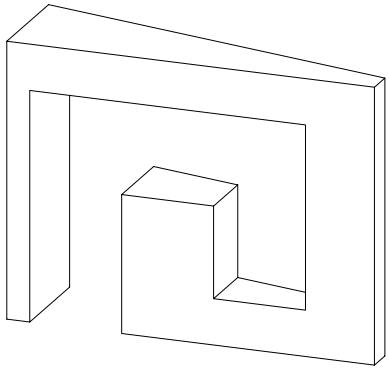


status : contrôle le comportement du mur composé de rondins

status = $j_1 + 2*j_2 + 4*j_3 + 32*j_6 + 64*j_7 + 128*j_8 + 256*j_9$

j_1 : applique la matière du côté droit sur les arêtes horizontales
j_2 : applique la matière du côté gauche sur les arêtes horizontales
j_3 : commence avec un demi-rondin
j_6 : aligne la texture sur les arêtes de mur
j_7 : double rayon sur le côté courbe
j_8 : rondin carré sur le côté droit
j_9 : rondin carré sur le côté gauche

Exemple :



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

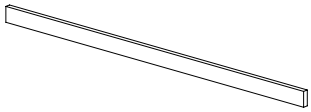
BEAM `leftmat, rightmat, sidevmat, topmat, bottommat, height, x1, x2, x3, x4, y1, y2, y3, y4, t, mask1, mask2, mask3, mask4`

Définition de poutre. Les paramètres sont similaires à ceux de l'élément XWALL_.

Paramètres supplémentaires :

topmat, bottommat :
les matières du haut et du bas

Exemple :



```
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_ topmat, botmat, sidemat,
n, x_1, y_1, x_2, y_2, height, angle, thickness,
x_1, y_1, $alpha_1$, $mask_1$,…,x_n, y_n, $alpha_n$, $mask_n$

Un pan de toiture en pente avec une rive à angle personnalisé.

topmat, botmat, sidemat:
nom/index des matières du haut, du bas et de côté

n: le nombre des nœuds du polygone de toiture

x_1, y_1, x_2, y_2: ligne (vecteur) de référence

height: la hauteur de la toiture à la ligne de référence (surface inférieure)

angle: l'angle de rotation du plan de la toiture autour de la ligne de référence orientée en degré (sens contraire aux aiguilles d'une montre)

thickness: l'épaisseur de la toiture mesurée le long de l'axe z en direction négative

x_i, y_i: les coordonnées des nœuds du polygone de toiture

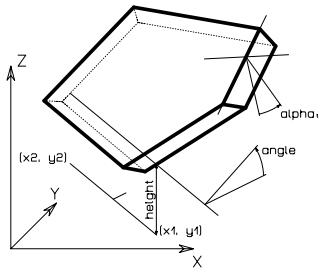
$alpha_i$: l'angle entre la face appartenant à l'arête i de la toiture et le plan perpendiculaire au plan de la toiture; $-90 < alpha_i < 90$. En regardant dans la direction de l'arête du polygone de toiture correctement orientée, l'angle de rotation dans le sens contraire aux aiguilles d'une montre est positif.

Les arêtes du polygones de toiture sont correctement orienté si, dans une vue de dessus, le contour a la séquence contraire au sens des aiguilles d'une montre et les trous ont une séquence dans le sens des aiguilles d'une montre.

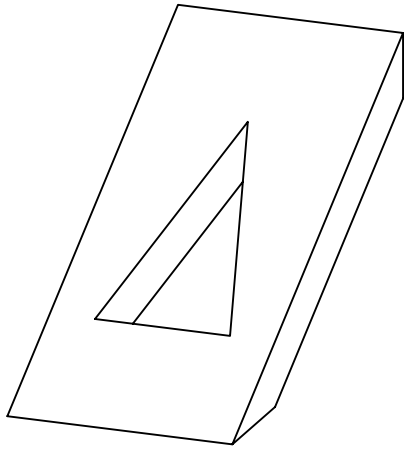
$mask_i$: définit la visibilité des arêtes de la toiture, ses valeurs sont identiques à celles de PRISM_

Restriction de paramètres:

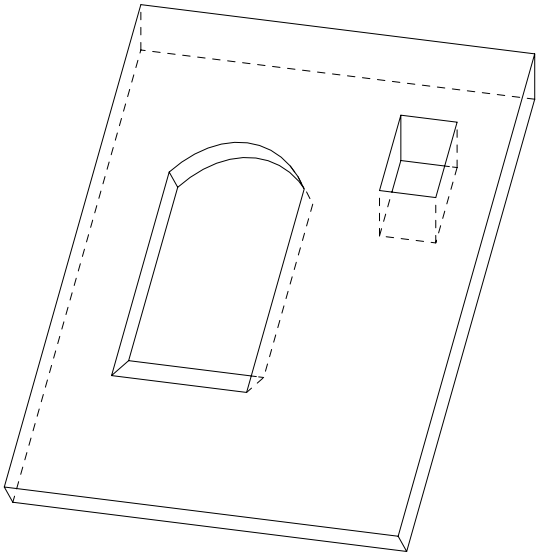
$n \geq 3$



Exemples:



```
CROOF_    1, 1,     1,     ! materials
       9,
       0,     0,
       1,     0,     ! reference line (x1,y1) (x2,y2)
       0.0,                ! height
       -30,                ! angle
       2.5,                ! thickness
       0,     0,     -60,   15,
       10,    0,     0,     15,
       10,    20,    -30,   15,
       0,     20,    0,     15,
       0,     0,     0,     -1,
       2,     5,     0,     15,
       8,     5,     0,     15,
       5,     15,    0,     15,
       2,     5,     0,     -1
```



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

MESH a, b, m, n, mask,
z_{11}, z_{12}, . . . z_{1m},
z_{21}, z_{22}, . . . z_{2m},
. . .
z_{n1}, z_{n2}, . . . z_{nm}

Maille simple basée sur un filet équidistant. Les côtés du rectangle de base sont a et b; les points m et n sont placés le long des axes x et y respectivement; z_{ij} est l'altitude du nœud.

Masquage

mask = j_1 + $4*j_3$ + $16*j_5$ + $32*j_6$ + $64*j_7$

où j_1, j_3, j_5, j_6, j_7 peuvent être 0 ou 1.

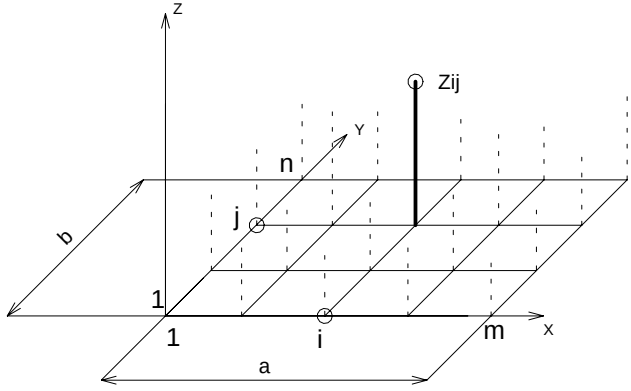
j_1 (1): la surface de base est présente.

j_3 (4): les surfaces de côté sont présentes.

j_5 (16): les arêtes de base et de côté sont visibles.

j_6 (32): les arêtes du haut sont visibles.

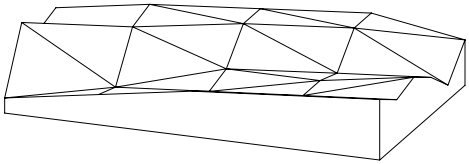
j_7 (64): les arêtes de la surface du haut sont visibles, la surface du haut n'est pas lisse.



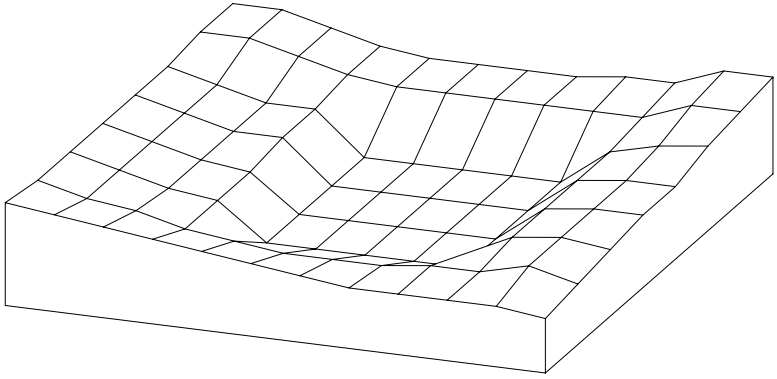
Restrictions de paramètres:

m ≥ 2, n ≥ 2

Exemples:



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



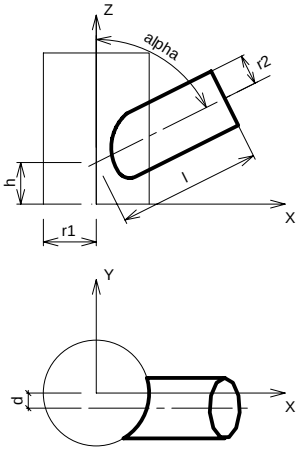
```
MESH 90,100, 12,8, 1+4+16+32+64,
 17,16,15,14,13,12,11,10,10,10,10, 9,
 16,14,13,11,10, 9, 9, 9,10,10,12,10,
 16,14,12,11, 5, 5, 5, 5, 5,11,12,11,
 16,14,12,11, 5, 5, 5, 5, 5,11,12,12,
 16,14,12,12, 5, 5, 5, 5, 5,11,12,12,
 16,14,12,12, 5, 5, 5, 5, 5,11,13,14,
 17,17,15,13,12,12,12,12,12,12,15,15,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

ARMC r_1, r_2, l, h, d, alpha

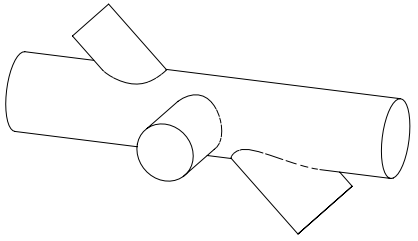
Une partie de tube partant d'un autre tube; les paramètres sont présentés sur la figure (les lignes de pénétration sont également calculées et dessinées). Alpha est en degrés.

Restriction de paramètres:

$r_1 \geq r_2 + d$
$r_1 \leq l * \sin(alpha) - r_2 * \cos(alpha)$



Exemple :



```
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
```

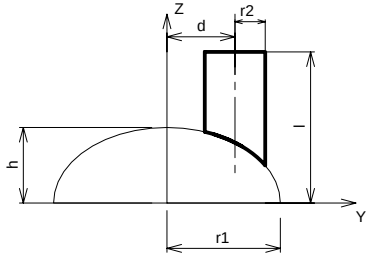
ARME l, r_1, r_2, h, d

Partie de tube partant d'une ellipsoïde dans le plan y-z; les paramètres sont comme sur la figure (les lignes de pénétration sont également calculées et dessinées).

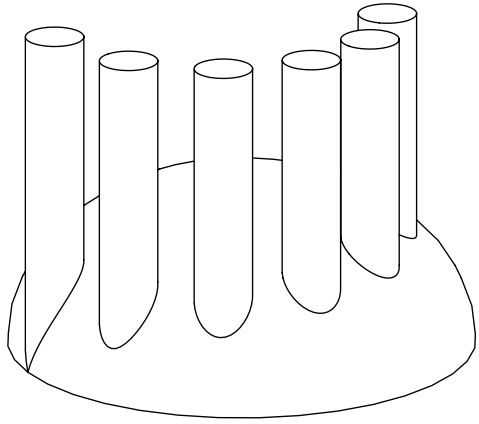
Restriction de paramètres:

$r_1 \geq r_2 + d$

$l \geq h * (1 - (r_2 - d)^2 / r_1^2)$



Exemple:



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

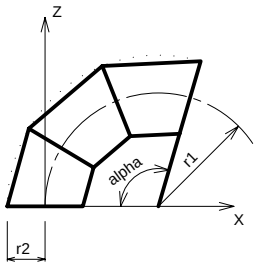
ELBOW r_1, alpha, r_2

Tube coudé dans le plan x-z. Le rayon de l'arc est r_1, l'angle est alpha et le rayon du tube est r_2.

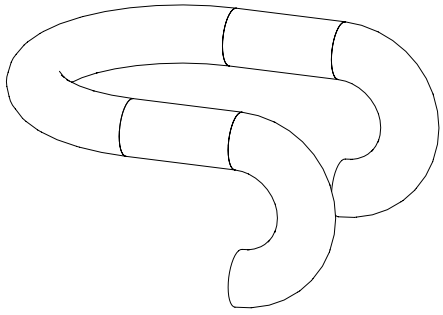
Alpha est en degrés.

Restriction de paramètres:

$$r_1 > r_2$$



Exemple:



```
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
```

5.2 Formes générées à partir de polylignes

Ces éléments permettent de créer des formes 3D complexes en utilisant une polyligne et une règle intégrée. Il est possible de faire subir de rotations, des projections et des translations à la polyligne donnée. Les corps obtenus sont une généralisation d'éléments existants, comme PRISM_ ou CYLIND.

Formes générées à partir d'une seule polyligne:

EXTRUDE

PYRAMID

REVOLVE

Formes générées à partir de deux polylignes:

RULED

SWEEP

TUBE

TUBEA

La première polyligne est toujours dans le plan x-y. Les points sont définis par deux coordonnées, la troisième valeur est l'état (voir ci-dessous). La seconde polyligne (pour RULED et SWEEP) est une courbe spatiale. Les nœuds sont définis par trois coordonnées.

Forme générée à partir de quatre polylignes:

COONS

Forme générée à partir de plusieurs polylignes:

MASS

Restrictions générales pour les polylignes:

Les nœuds consécutifs ne doivent pas coïncider (excepté RULED).

La polyligne ne doit pas se couper elle-même (non vérifié, mais le calcul des lignes cachées et l'ombrage seront incorrects).

Les polylignes peuvent être ouvertes ou fermées. Dans le dernier cas, le premier nœud doit être répété à la fin de l'instruction.

Masquage

Les valeurs de masque sont utilisées pour activer ou désactiver les surfaces caractéristiques et/ou les arêtes de la forme 3D. Les valeurs de masque sont spécifiques à chaque élément.

mask = $j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7$

où j_1, j_2, j_3, j_4, j_5, j_6, j_7 peuvent être 0 ou 1.

Les nombres j_1, j_2, j_3, j_4 indiquent que les surfaces sont présentes (1) ou omises (0).

Les nombres j_5, j_6, j_7 indiquent que les arêtes sont visibles (1) ou invisibles (0).

j_1 : surface de base
j_2 : surface du haut
j_3 : surface de côté
j_4 : autre surface de côté
j_5 : arêtes de la base
j_6 : arêtes du dessus
j_7 : coupe transversale/arêtes de surface visibles, surface non lisse

Pour activer toutes les surfaces et arêtes, la valeur du masque doit être de 127.

Etat

Les valeurs d'état sont utilisées pour définir qu'un point donné de la polyligne laissera ou non une trace en tournant.

0 :	les arêtes latérales et arcs latitudinaires partant du nœud sont tous visibles.
1 :	les arêtes latérales et arcs latitudinaires sont seulement utilisés pour afficher le contour.
-1 :	pour EXTRUDE seulement: marque la fin du polygone ou trou qui le comprend et signifie que le nœud suivant sera le premier nœud d'un autre trou.

Pour créer une forme 3D lisse, régler toutes les valeurs d'état à 1. Utiliser état = 0 pour créer une faîte.

Les codes d'état additionnels pour les arcs et les segments tangentiels sont décrits sous "Code d'état additionnels pour polylignes". Les autres valeurs sont réservés pour des développements à venir.

EXTRUDE n, dx, dy, dz, mask, x_1, y_1, s_1, . . . x_n, y_n, s_n

Prisme général utilisant une polyligne comme base dans le plan x-y. Le vecteur de déplacement entre bases est (dx, dy, dz).

Généralisation des instructions PRISM et SLAB. La polyligne de base n'est pas obligatoirement fermée et les arêtes latérales ne sont pas perpendiculaires au plan x-y. La polyligne de base peut comprendre des trous, tout comme PRISM_. Les arêtes de contour peuvent être visibles ou non.

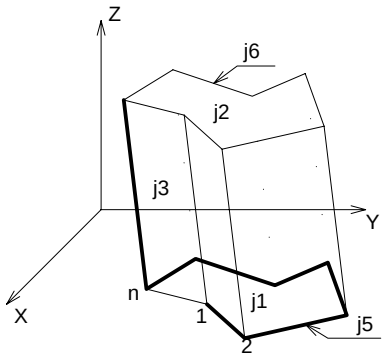
n : le nombre des nœuds de la polyligne.

mask : contrôle la présence des polygones inférieur et supérieur et, dans le cas d'une polyligne ouverte, du polygone latéral.

s_i : état des arêtes latérales ou marque la fin du polygone ou du trou.

Restriction de paramètres:

```
n > 2
```



Masquage

mask = $j_1 + 2*j_2 + 4*j_3 + 16*j_5 + 32*j_6$

j_1 (1): la surface de base est présente.

j_2 (2): la surface du haut est présente.

j_3 (4): la surface latérale (de fermeture) est présente.

j_5 (16): les arêtes de la base sont visibles.

j_6 (32): les arêtes du haut sont visibles.

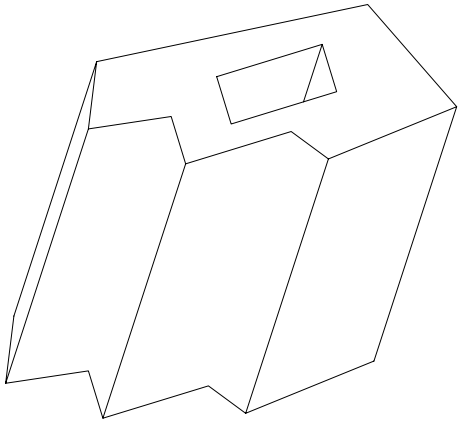
Etat

0 : les arêtes latérales partant du nœud sont toutes visibles.

1 : les arêtes latérales partant du nœud sont utilisées pour afficher le contour.

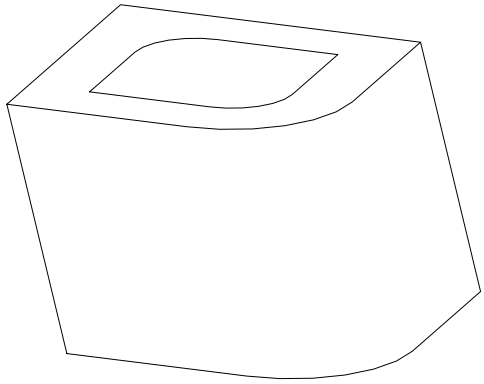
-1 : marque la fin du polygone ou trou qui le comprend et signifie que le nœud suivant sera le premier nœud d'un autre trou.

Exemples:



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

            2,     0,     0,
            3,     2,     0,
            4,     0,     0,
            3,     -2,    0,
            2,     0,     -1
```



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

             C,           C,           0,
             D+S*SIN(0),  R-S*COS(0),  1,
             D+S*SIN(15), R-S*COS(15), 1,
             D+S*SIN(30), R-S*COS(30), 1,
             D+S*SIN(45), R-S*COS(45), 1,
             D+S*SIN(60), R-S*COS(60), 1,
             D+S*SIN(75), R-S*COS(75), 1,
             D+S*SIN(90), R-S*COS(90), 1,
             A-C,         B-C,         0,
             R-S*COS(90), E+S*SIN(90), 1,
             R-S*COS(75), E+S*SIN(75), 1,
             R-S*COS(60), E+S*SIN(60), 1,
             R-S*COS(45), E+S*SIN(45), 1,
             R-S*COS(30), E+S*SIN(30), 1,
             R-S*COS(15), E+S*SIN(15), 1,
             R-S*COS(0),  E+S*SIN(0),  1,
             C,           C,           -1
```

PYRAMID n, h, mask, x_1, y_1, s_1, . . . x_n, y_n, s_n

Pyramide basée sur une polyligne dans le plan x-y. Le sommet de la pyramide est en (0, 0, h).

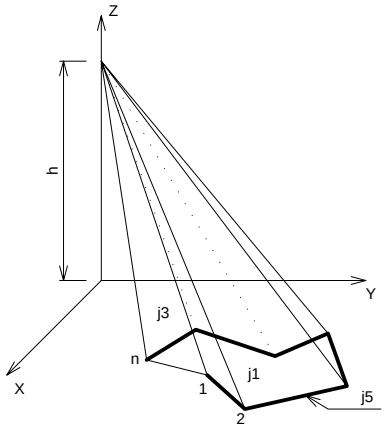
n : nombre de nœuds de la polyligne.

mask : contrôle la présence du polygone inférieur et (dans le cas d'une polyligne ouverte) du polygone latéral.

s_i : état des arêtes latérales.

Restriction de paramètres:

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



Masquage

mask = j_1 + 4*j_3 + 16*j_5

où j_1, j_3, j_5 peuvent être 0 ou 1.

j_1 (1): la surface de base est présente.

j_3 (4): la surface latérale (de fermeture) est présente.

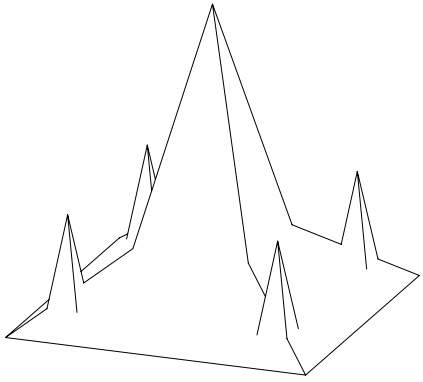
j_5 (16): les arêtes de la base sont visibles.

Etat

0 : les arêtes latérales partant du nœud sont toutes visibles.

1 : les arêtes latérales partant du nœud sont utilisées pour afficher le contour.

Exemple :



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

100:
PYRAMID     4,          1.5,        21,
            -0.25,      -0.25,      0,
            0.25,       -0.25,      0,
            0.25,       0.25,       0,
            -0.25,      0.25,       0
RETURN
```

REVOLVE n, alpha, mask, x_1, y_1, s_1, . . . x_n, y_n, s_n

Surface générée par la rotation d'une polyligne définie dans le plan x-y autour de l'axe x.

n: nombre de nœuds de la polyligne.

alpha: angle de la rotation en degrés.

mask: contrôle la présence des polygones inférieur, supérieur et (dans le cas d'un angle alpha < 360°) latéraux.

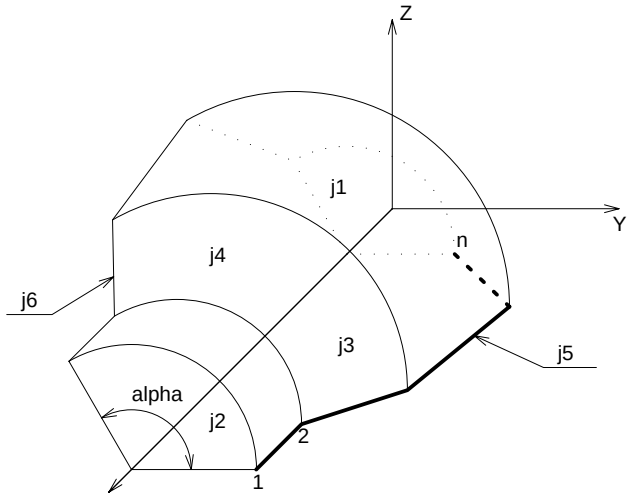
s_i: état des arcs latitudinaires.

Restriction de paramètres:

$n \geq 2$

$y_i \geq 0.0$

y_i et $y_i + _1$ (la valeur y de deux nœuds voisins) ne peuvent être zéro en même temps.



Masquage

mask = $j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7$

où $j_1, j_2, j_3, j_4, j_5, j_6, j_7$ peuvent être 0 ou 1.

j_1 (1): la surface de base est présente.

j_2 (2): la surface de dessus est présente.

j_3 (4): la surface latérale est présente à l'angle initial.

j_4 (8): la surface latérale est présente à l'angle d'arrivée.

j_5 (16): les arêtes de la surface latérale à l'angle initial sont visibles.

j_6 (32): les arêtes de la surface latérale à l'angle d'arrivée sont visibles

j_7 (64): les arêtes de la coupe transversale sont visibles, la surface n'est pas lisse.

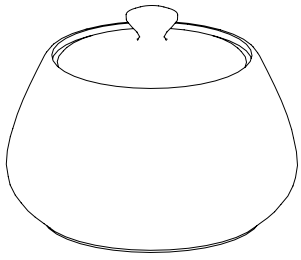
Etat

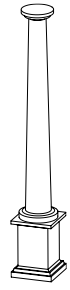
0 : les arcs latitudinaires partant du nœud sont tous visibles.

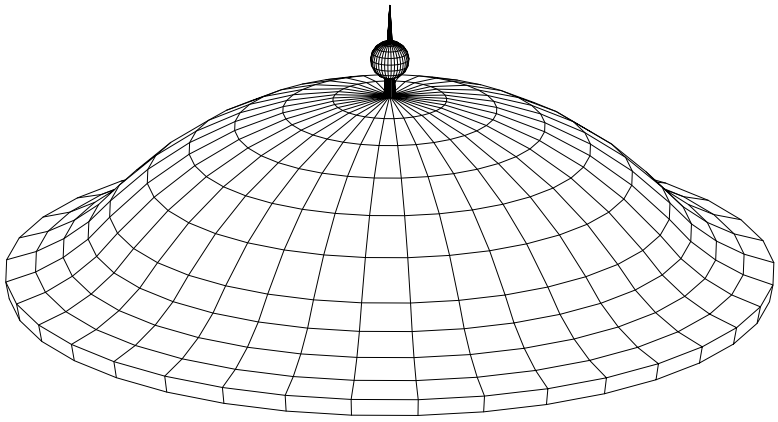
1 : les arcs latitudinaires partant du nœud sont utilisés pour afficher le contour.

2 : dans le rendu photoréaliste (à condition d'utiliser le moteur ArchiCAD ou Z-buffer), en définissant de surfaces lisses, l'arête latitudinale appartenant à ce point définit une rupture. Cette solution est l'équivalent de la définition de nœuds additionnels; le calcul est exécuté par le compilateur. L'algorithme du rendu photoréaliste reste inchangé. Si vous utilisez un autre mode de projection 3D, cette valeur a le même effet que la valeur 0.

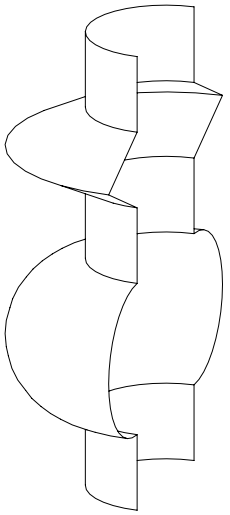
Exemples :







```
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
```



sans code d'état 2

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

avec code d'état 2

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

RULED n, mask,
u_1, v_1, s_1, . . . u_n, v_n, s_n,
x_1, y_1, z_1, . . . x_n, y_n, z_n

RULED est une surface basée sur une courbe plan et une courbe spatiale ayant le même nombre de nœuds. De simples segments joignent les nœuds correspondants des deux polylignes.

C'est le seul élément GDL qui permette que des nœuds voisins se recouvrent.

n : nombre des nœuds de polyligne des deux courbes.

mask : contrôle l'existence des polygones de base, de dessus et de côté, ainsi que la visibilité des arêtes des polylignes génératrices. Le polygone de côté joint les premier et dernier nœuds des courbes, si l'une ou l'autre n'est pas fermée.

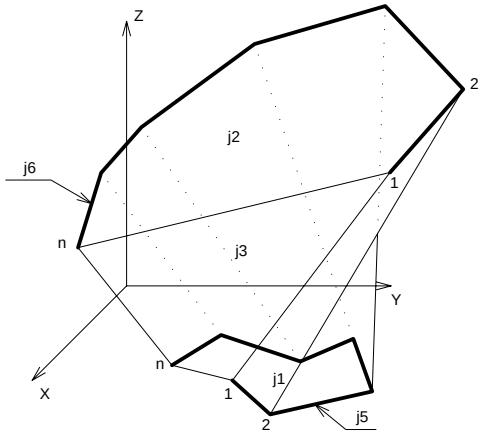
u_i, v_i : coordonnées des nœuds de la courbe plan.

s_i : état des arêtes latérales.

x_i, y_i, z_i : coordonnées des nœuds de la courbe spatiale.

Restriction des paramètres :

$n > 1$



Masquage

mask = $j_1 + 2*j_2 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7$

où j_1, j_2, j_3, j_5, j_6, j_7 peuvent être 0 ou 1.

j_1 (1): la surface de base est présente.

j_2 (2): la surface de dessus est présente (sans effet si la surface de dessus n'est pas une surface plan).

j_3 (4): la surface de côté est présente (un quadrangle plan ou deux triangles).

j_5 (16): les arêtes de la courbe plan sont visibles.

j_6 (32): les arêtes de la courbe spatiale sont visibles.

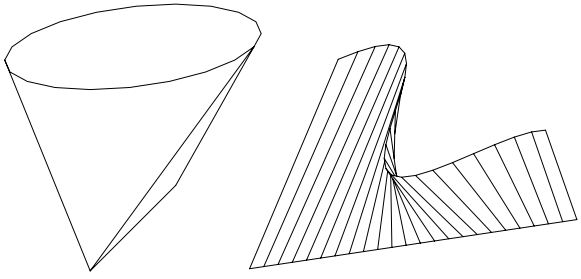
j_7 (64): les arêtes de la surface sont visibles, la surface n'est pas lisse.

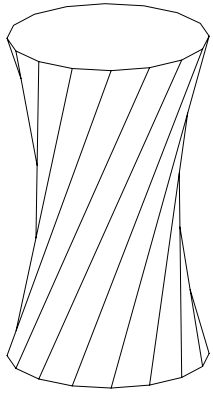
Etat

0 : les arêtes latérales partant du nœud sont toutes visibles.

1 : les arêtes latérales partant du nœud sont utilisées pour afficher le contour.

Exemples :





```
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
```

SWEEP n, m, alpha, scale, mask,
u_1, v_1, s_1, . . . u_n, v_n, s_n,
x_1, y_1, z_1, . . . x_m, y_m, z_m

Surface générée par une polyligne en balayant la trajectoire d'une courbe spatiale.

La polyligne peut être ouverte ou fermée. Il est possible de lui faire subir une rotation ou une multiplication sur son propre plan.

Le plan de la polyligne suit la trajectoire de la courbe spatiale. La courbe spatiale part obligatoirement du plan x-y. Si cette condition n'est pas remplie, la courbe est déplacée le long de l'axe z pour partir du plan x-y.

L'intersection (x_i, y_i, z_i) par point est perpendiculaire au segment entre les points (x_{i-1}, y_{i-1}, z_{i-1}) et (x_i, y_i, z_i).

SWEEP peut être utilisée pour modeler le bec d'une théière ou d'autres formes complexes.

n :	nombre des nœuds de la polyligne.
m :	nombre des nœuds de la trajectoire.
alpha :	angle de la rotation de la polyligne sur son propre plan, d'un nœud de la trajectoire au nœud suivant.
scale :	facteur d'incrément d'échelle de la polyligne d'une nœud de la trajectoire au nœud suivant.
mask :	contrôle l'existence des surfaces et arêtes des polygones de base et de dessus.
u_i, v_i :	coordonnées des nœuds de la polyligne de base.
s_i :	état des arêtes latérales.
x_i, y_i, z_i :	coordonnées des nœuds de la trajectoire de la courbe.

Restriction de paramètres :

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

Masquage

mask = $j_1 + 2*j_2 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7$

où $j_1, j_2, j_3, j_5, j_6, j_7$ peuvent être 0 ou 1.

j_1 (1): la surface de base est présente.

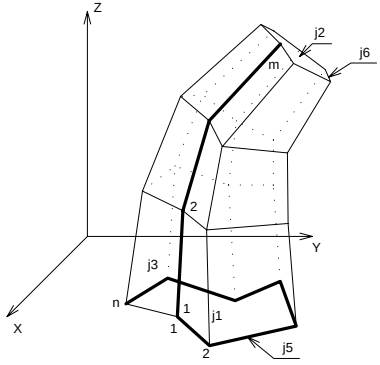
j_2 (2): la surface du haut est présente.

j_3 (4): la surface latérale est présente.

j_5 (16): les arêtes de la base sont visibles.

j_6 (32): les arêtes du haut sont visibles.

j_7 (64): les arêtes de l'intersection sont visibles, la surface est articulée.

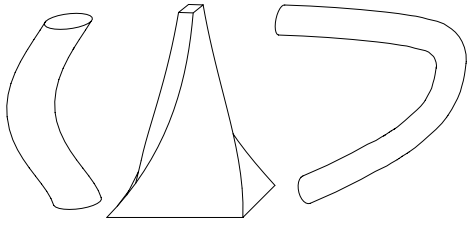


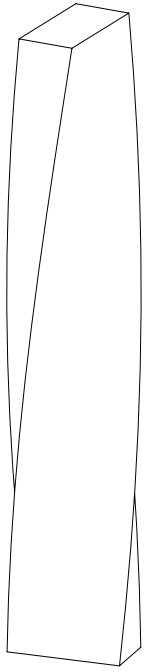
Etat

0 : les arêtes latérales partant du nœud sont toutes visibles.

1 : les arêtes latérales partant du nœud sont utilisées pour afficher le contour.

Exemples :





```
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 n, m, mask,
u_1, w_1, s_1,
. . .
u_n, w_n, s_n,
x_1, y_1, z_1, $angle_1$,
. . .
x_m, y_m, z_m, $angle_m$

Surface générée par une polyligne balayant la trajectoire d'une courbe spatiale sans déformation de la coupe transversale génératrice. Les surfaces de connexion internes peuvent subir une rotation dans le plan U-W du système de coordonnées instantané U-V-W.

axe V: approximation de la tangente de la courbe génératrice au point correspondant,

axe W perpendiculaire à l'axe V et pointant vers le haut par rapport à l'axe z local,

axe U perpendiculaire aux axes V et W et forme avec eux un système de coordonnées cartésiennes.

Si l'axe V est vertical, la direction W n'est pas définie correctement. L'axe W du nœud de trajectoire précédent est utilisé pour déterminer une direction horizontale.

Le polygone de coupe transversale du tube mesuré au milieu des segments de la trajectoire est toujours égal au polygone de base (u_1, w_1, ... u_n, w_n). Les polygones de coupe aux points de jonction sont situés sur le plan bisecteur des segments joints. Le polygone de base doit être fermé.

n : nombre des nœuds de la polyligne.

m: nombre des nœuds de la trajectoire.

u_i, w_i : coordonnées des nœuds de la polyligne de base.

s_i : état des arêtes latérales.

x_i, y_i, z_i : coordonnées des nœuds de la courbe de la trajectoire.

(La trajectoire comprend deux points de plus qu'il n'y a de sections générées. Le premier et le dernier point déterminent la position dans l'espace de la première et de la dernière surface appartenant au TUBE. Le rôle de ces points est uniquement de déterminer la normale des surfaces, ce ne sont pas réellement des nœuds de la trajectoire. L'orientation des surfaces est la même que celle des surfaces qui

auraient été générées aux nœuds les plus proches des deux extrémités si le TUBE avait continué dans les sens indiqués par ces points.)

$angle_i$ angle de rotation de l'intersection.

Masquage

mask = $j_1 + 2*j_2 + 16*j_5 + 32*j_6 + 64*j_7$

où j_1, j_2, j_5, j_6, j_7 peuvent être 0 ou 1.

j_1 (1): la surface de base est présente.

j_2 (2): la surface finale est présente.

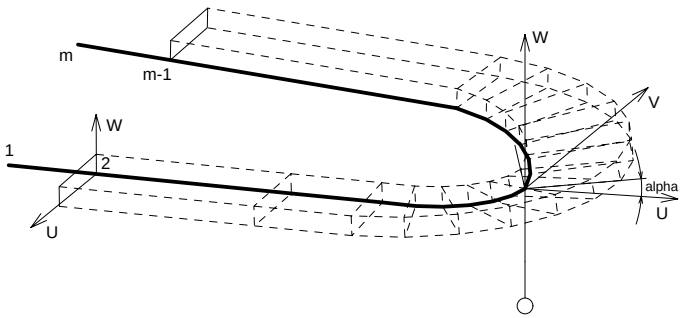
j_5 (16): les arêtes de la base (à x_1, y_1, z_1) sont visibles.

j_6 (32): les arêtes finales (à x_m, y_m, z_m) sont visibles.

j_7 (64): les arêtes de la coupe transversale (sauf les lignes de connexion des faces sur un même plan) sont visibles, la surface est articulée.

Restrictions de paramètres :

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

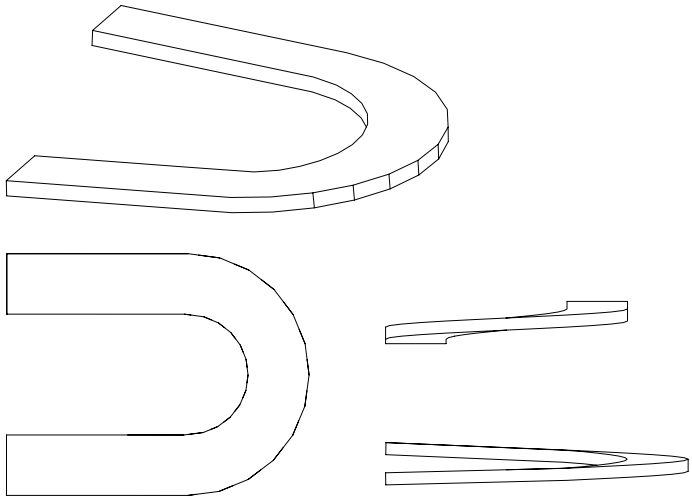


Etat

0 : les arêtes latérales partant du nœud sont toutes visibles.

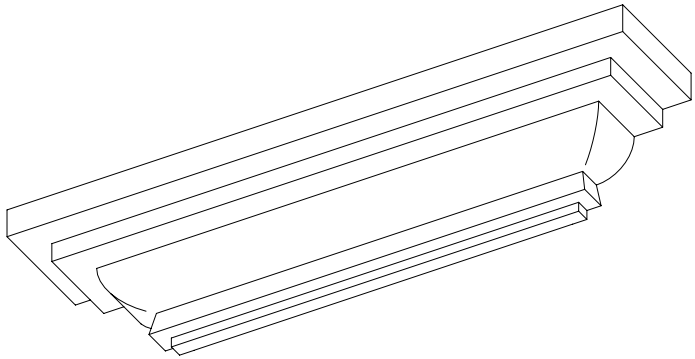
1 : les arêtes latérales partant du nœud sont utilisées pour afficher le contour.

Exemples:

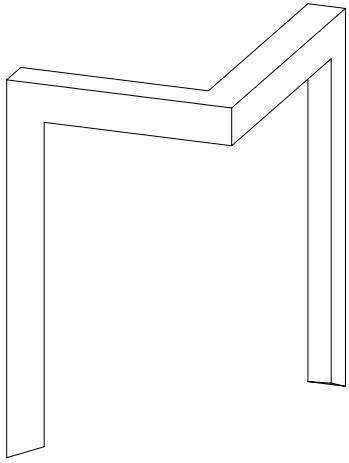


```
TUBE   4,              18,             16+32,
       2.0,            0.0,            0,
       0.0,            0.0,            0,
       0.0,            0.4,            0,
       2.0,            0.4,            0,

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



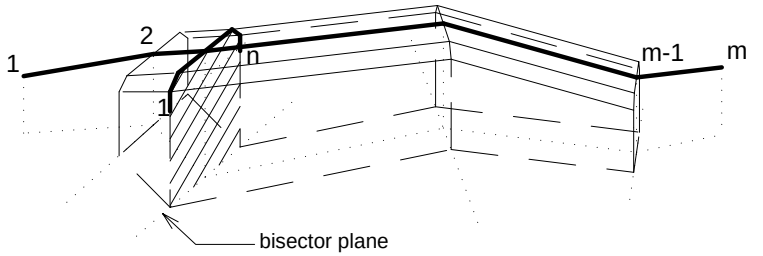
```
TUBE        14,         6,          1+2+16+32,
            0,          0,          0,
            0.03,       0,          0,
            0.03,       0.02,       0,
            0.06,       0.02,       0,
            0.05,       0.0699,     0,
            0.05,       0.07,       1,
            0.05,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



```
TUBE        3,     7,     16+32,
            0,     0,     0,
            -0.5,  0,     0,
            0,     0.5,   0,

            0.2,   0,     -0.2,        0,
            0,     0,     0,           0,
            0,     0,     5,           0,
            3,     0,     5,           0,
            3,     4,     5,           0,
            3,     4,     0,           0,
            3,     3.8,   -0.2,        0
```

TUBEA n, m, mask,
u_1, w_1, s_1,
. . .
u_n, w_n, s_n,
x_1, y_1, z_1,
. . .
x_m, y_m, z_m

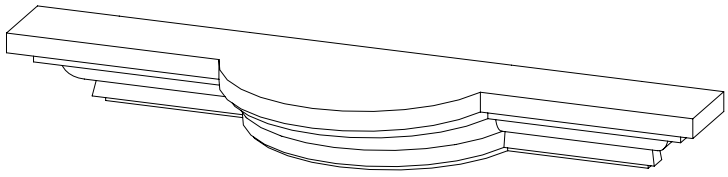


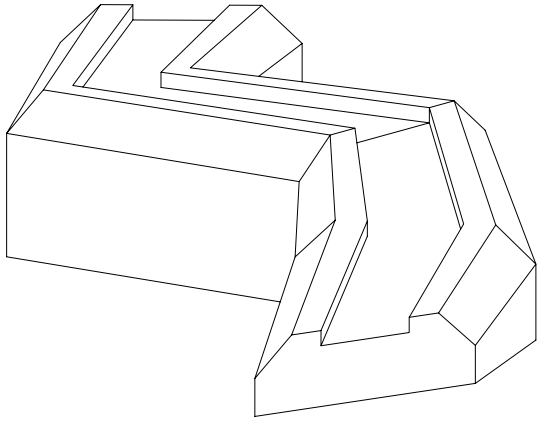
TUBEA est une surface générée par une polyligne balayant la trajectoire d'une courbe spatiale avec un algorithme différent de celui de l'instruction TUBE.

Le polygone de coupe généré à chaque point de jonction de la courbe de la trajectoire est égal au polygone de base (u_1, w_1, ... u_n, w_n) et est situé dans le plan bisecteur des projections des segments joints au plan x-y local. Le polygone de base peut être ouvert: dans ce cas, les polygones de coupe seront générés de manière à atteindre le plan x-y local, comme c'est le cas des surfaces REVOLVE.

La coupe transversale du tube mesurée au milieu des segments de la trajectoire peut être différente du polygone de base.

Exemples :





```
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 n, m, mask,
$x1_1$, $y1_1$, $z1_1$, . . . $x1_n$, $y1_n$, $z1_n$,
$x2_1$, $y2_1$, $z2_1$, . . . $x2_n$, $y2_n$, $z2_n$,
$x3_1$, $y3_1$, $z3_1$, . . . $x3_m$, $y3_m$, $z3_m$,
$x4_1$, $y4_1$, $z4_1$, . . . $x4_m$, $y4_m$, $z4_m$

Surface de Coons générée à partir de quatre courbes de bordure.

Masquage

mask = $4*j_3 + 8*j_4 + 16*j_5 + 32*j_6 + 64*j_7$

où j_3, j_4, j_5, j_6, j_7 peuvent être 0 ou 1.

j_3 (4): arêtes de la 1ère bordure (x1, y1, z1) sont visibles.

j_4 (8): arêtes de la 2ème bordure (x2, y2, z2) sont visibles.

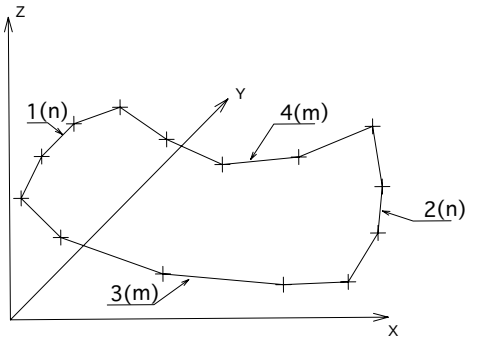
j_5 (16): arêtes de la 3ème bordure (x3, y3, z3) sont visibles.

j_6 (32): arêtes de la 4ème bordure (x4, y4, z4) sont visibles.

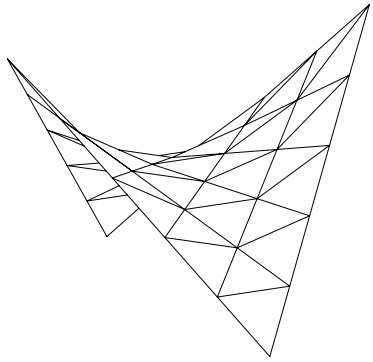
j_7 (64): arêtes de surface sont visibles, surface non lisse.

Restrictions de paramètres :

n, m > 1



Exemples:

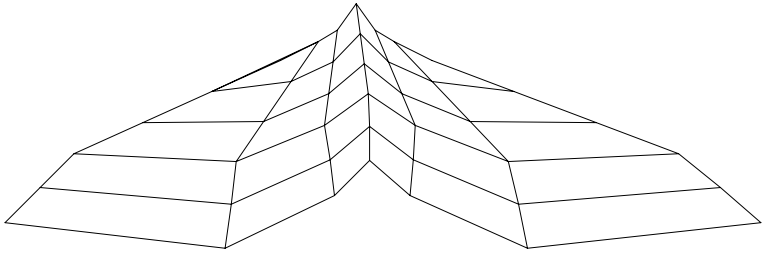


```
COONS  6,     6,     4+8+16+32+64,
       !1st boundary, n=6
       0,     0,     5,
       1,     0,     4,
       2,     0,     3,
       3,     0,     2,
       4,     0,     1,
       5,     0,     0,

       !2nd boundary, n=6
       0,     5,     0,
       1,     5,     1,
       2,     5,     2,
       3,     5,     3,
       4,     5,     4,
       5,     5,     5,

       !3rd boundary, m=6
       0,     0,     5,
       0,     1,     4,
       0,     2,     3,
       0,     3,     2,
       0,     4,     1,
       0,     5,     0,

       !4th boundary, m=6
       5,     0,     0,
       5,     1,     1,
       5,     2,     2,
       5,     3,     3,
       5,     4,     4,
       5,     5,     5
```



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

       !3rd boundary, m=6
       1,            2,            0,
       2,            4,            -0.5,
       3,            6,            -1,
       4,            8,            -1.5,
       5,            9,            -1.8,
       6,            10,           -2,
       !4th boundary, m=6
       1,            -2,           0,
       2,            -4,           -0.5,
       3,            -6,           -1,
       4,            -8,           -1.5,
       5,            -9,           -1.8,
       6,            -10,          -2
```

MASS topmat, botmat, sidemat, n, m, mask, h,
x_1, y_1, z_1, s_1,
. . .
x_n, y_n, z_n, s_n,
x_{n+1}, y_{n+1}, z_{n+1}, s_{n+1},
. . .
x_{n+m}, y_{n+m}, z_{n+m}, s_{n+m}

topmat, botmat, sidemat:
nom/index des matières de dessus, de dessous et de côté

n : le nombre des nœuds dans le polygone de masse

m : le nombre des nœuds sur les arêtes

h: la hauteur de la jupe (peut être négative)

x_i, y_i, z_i : les coordonnées des nœuds

s_i : similaire à l'instruction PRISM_

Masquage

mask = j_1 + 4*j_3 + 16*j_5 + 32*j_6 + 64*j_7
où j_1, j_3, j_5, j_6, j_7 peuvent être 0 ou 1.

j_1 (1): la surface de base est présente

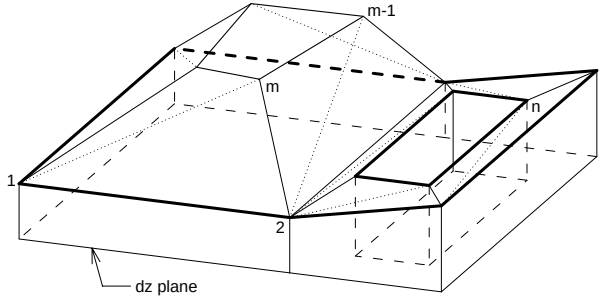
j_3 (4): la surfaces de côté est présente

j_5 (16): les arêtes de base et de côté sont visibles

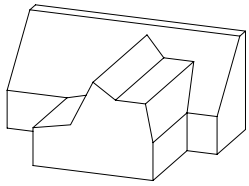
j_6 (32): les arêtes de dessus sont visibles

j_7 (64): les arêtes de dessus sont visibles, la surface supérieure n'est pas lisse

Restrictions de paramètres :
n ≥ 3, m ≥ 0



Exemple :



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

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

5.3 Eléments pour la visualisation

LIGHT `red, green, blue, shadow, radius, alpha, beta, angfalloff, dist1, dist2, distfalloff`

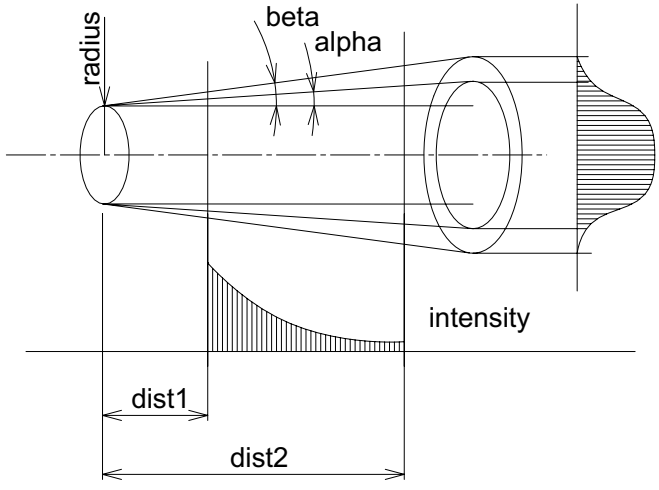
Une source lumineuse irradiant une lumière de couleur [rouge, vert, bleu] de l'origine locale le long de l'axe x local. La lumière est projetée parallèlement à l'axe x d'une source ponctuelle ou circulaire. Elle a son intensité maximale à l'intérieur d'une portion de cône d'angle alpha et est de zéro à l'angle bêta du cône. Ceci est contrôlé par le paramètre angfalloff. (La valeur zéro donne à la lumière un contour prononcé, les valeurs plus élevées signifient une transition plus souple.) L'effet de la lumière est limitée le long de l'axe par les valeurs de coupe dist1 et dist2. Le paramètre distfalloff contrôle la décroissance de l'intensité en fonction de la distance. (La valeur zéro signifie une intensité constante, les valeurs plus élevées sont utilisées pour une dégradation plus forte.)

Les transformations GDL n'affectent que le point de départ et la direction de la lumière**.**

Le paramètre shadow détermine si la lumière donne des ombres.

0 : lumière ne donne pas d'ombres

1 : lumière donne des ombres



Restriction de paramètres :

`alpha ≤ beta ≤ 80°`

Les combinaisons de paramètres suivantes ont des significations spéciales:

`rayon = 0, alpha = 0, bêta = 0`

Une lumière ponctuelle, irradiant de la lumière dans toutes les directions, et ne donnant pas d'ombres.

Les paramètres shadow et angfalloff sont ignorés, des valeurs zéro sont supposées.

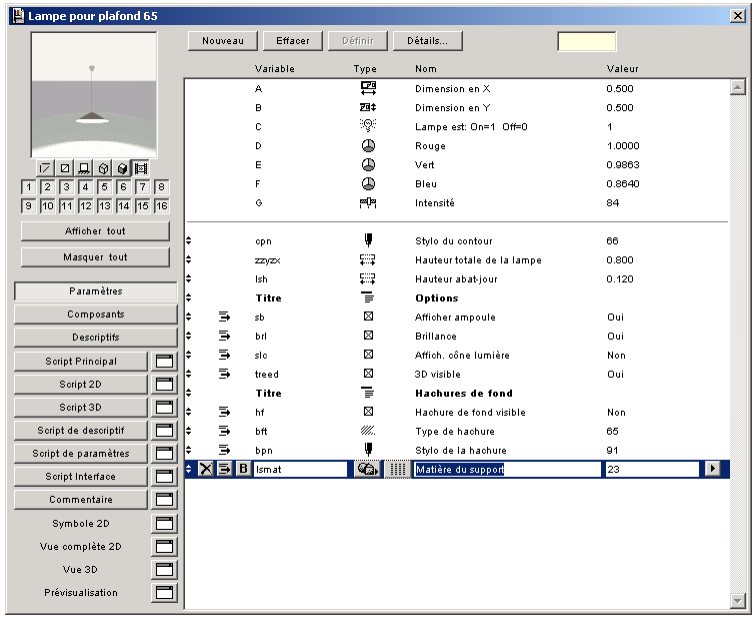
`rayon > 0, alpha = 0, bêta = 0`

Une lumière directionnelle

Exemple:

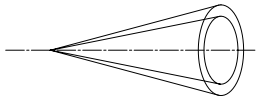
```
LIGHT 1.0,0.2,0.3,          ! RGB
 1,                         ! shadow on
 1.0,                       ! radius
 45.0,60.0,                 ! angle1, angle2
 0.3,                       ! angfalloff
 1.0,10.0,                  ! dist1, dist2
 0.2                        ! distfalloff
```

Le dialogue dans ArchiCAD d'un élément de bibliothèque de type Lampe:

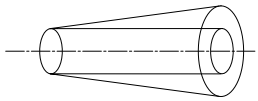


Une partie du script GDL correspondant:

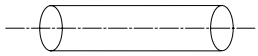
```
IF C = 0 GOTO 10
LIGHT G/100*D, G/100*E, G/100*F,    !RGB
...
10:
```



r = 0, alpha > 0, beta > 0



r > 0, alpha = 0, beta > 0



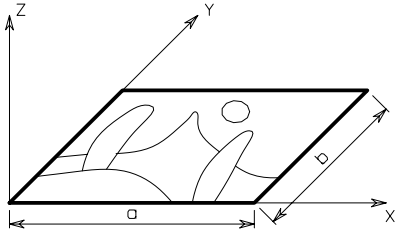
r > 0, alpha = 0, beta = 0

Types de lumière utilisant des paramètres alpha et bêta différents

PICTURE `expression, a, b, mask`

Un élément d'image pour le Rendu photoréaliste.

Une expression de type texte signifie un nom de fichier. Une expression numérique signifie l'index d'une image conservée à l'intérieur de l'élément de bibliothèque. L'index 0 a une valeur spéciale, il fait référence à l'image de prévisualisation de l'élément de bibliothèque. D'autres images ne peuvent être conservées dans un élément de bibliothèque qu'au cas où vous enregistrez le projet ou des éléments sélectionnés contenant des images sous forme d'Objet ArchiCAD.
L'image du fichier référé est cadré dans un rectangle traité comme un `RECT` par toutes les autres méthodes de projection 3D.



mask = alpha + distortion

alpha : contrôle canal alpha

0 : ne pas utiliser le canal alpha, image est un rectangle

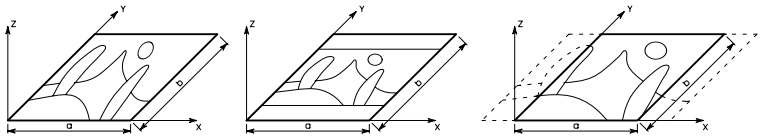
1 : utiliser le canal alpha, des parties de l'image peuvent être transparentes.

distortion : contrôle la distortion

0 : cadrer l'image dans le rectangle donné

2 : cadrer image au milieu du rectangle en utilisant les proportions naturelles de l'image

4 : remplir le rectangle avec l'image en position centrale, en utilisant les proportions naturelles de l'image



5.4 Elément de texte

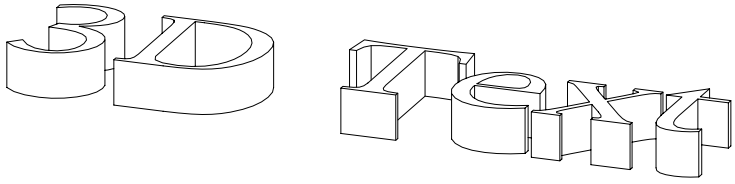
TEXT d, 0, expression

La représentation en 3D dans le style défini de la valeur d'une expression de type caractère ou numérique. Voir DEFINE STYLE et SET STYLE au Chapitre.

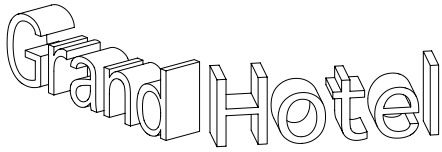
d: épaisseur des caractères en mètres

Dans cette version de GDL, le second paramètre est toujours zéro.

Exemples:



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



```
name = "Grand"

ROTX 90
ROTY -30
TEXT 0.003, 0, name

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

Remarque : Pour être compatible avec le script GDL 2D, la hauteur de caractère est toujours interprétée en millimètres dans les instructions DEFINE STYLE.

5.5 Eléments de primitive

Les primitives de la structure 3D sont VERT, VECT, EDGE, PGON, et BODY. Les corps sont représentés par des surfaces et par les connexions entre elles. L'information pour exécuter une Coupe 3D vient de l'information de connexion.

L'indexation commence à 1 et chaque corps ou instruction BASE remet les index à 1. Pour chaque arête, les index des polygones adjacents sont stockés (maximum 2). Les arêtes ont une orientation du premier point donné vers le second.

Les polygones sont des listes d'arêtes orientées avec un index pour chaque arête. Ces nombres peuvent être négatifs, ce qui signifie que les arêtes sont utilisées dans la direction opposée. Les polygones peuvent comprendre des trous. Dans la liste des arêtes, un zéro indique un nouveau trou. Les trous ne peuvent inclure eux-mêmes de nouveaux trous. Une arête peut appartenir à aucun, à un ou à deux polygones. L'orientation du polygone est correcte si, dans le cas des formes fermées, l'arête a une orientation différente pour les deux polygones.

Les vecteurs de normale des polygones sont stockés à part. Dans le cas des corps fermés, ils pointent de l'intérieur vers l'extérieur du corps. L'orientation de la liste d'arêtes est dans le sens inverse des aiguilles d'une montre (sens trigonométrique), si vous la regardez de l'extérieur. L'orientation des trous est opposé au polygone parent. Les vecteurs normaux d'un corps ouvert doivent pointer vers le même côté du corps.

Pour déterminer l'intérieur et l'extérieur d'un corps, il doit être fermé. Une définition simple d'un corps fermé est la suivante: chaque arête est adjacente à exactement deux polygones.

L'efficacité des algorithmes de coupes, de lignes cachées et de rendu est moindre dans le cas des corps ouverts. Tous les éléments 3D complexe avec des paramètres réguliers sont des corps fermés dans la structure de données 3D interne.

La recherche des lignes de contour est basée sur les bits d'état des arêtes et de leurs polygones adjacents. Ceci est automatiquement défini pour les éléments incurvés complexes, mais c'est à vous de spécifier correctement ces bits dans le cas des éléments de primitives.

Dans le cas d'une définition simplifiée (PGON.ivect = 0 ou PGON.état < 0) les primitives auxquelles d'autres font référence précèdent leur référence. Dans ce cas, l'ordre recommandé est:

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

La recherche des polygones adjacents à ces arêtes est faite pendant l'exécution de l'instruction BODY.

La numérotation des VERTs, EDGEs, VECTs et PGONs est relative à la dernière instruction BASE explicite ou implicite.

Les valeurs d'état sont utilisées pour stocker certaines informations sur les primitives. Chaque bit a une signification indépendante dans l'état mais il y a des exceptions.

Des valeurs données peuvent être additionnées entre elles. D'autres combinaisons que celles données ci-dessous sont strictement réservées à l'usage interne. L'état par défaut est zéro.

VERT `x, y, z`

Un nœud dans l'espace x-y-z, défini par trois coordonnées.

TEVE `x, y, z, u, v`

Extension de l'instruction VERT, comprenant une définition des coordonnées d'une texture. Peut remplacer l'instruction VERT s'il est nécessaire d'inclure les coordonnées de texture définies par l'utilisateur au lieu de l'automatisme d'ArchiCAD (voir instruction COOR).

x, y, z: les coordonnées d'un nœud

u, v: les coordonnées de texture du nœud

Les coordonnées (u, v) de chacun des nœuds du corps doivent être spécifiée et chaque nœud doit avoir au moins une coordonnée de texture. Si des instructions VERT et TEVE sont mélangées dans une définition de corps, les coordonnées (u, v) resteront sans effet.

Remarque: les coordonnées (u, v) de texture n'affectent que le rendu photoréaliste, mais non pas le mapping des hachures vectorielles.

VECT `x, y, z`

Définition du vecteur de normale d'un polygone par trois coordonnées. Dans le cas d'une définition simplifiée (PGON.ivect = 0) ces instructions peuvent être omises.

EDGE $vert_1$, $vert_2$, $pgon_1$, $pgon_2$, status

Définition d'une arête.

$vert_1$, $vert_2$: index des extrémités.

Les index $vert_1$ et $vert_2$ doivent être différents et référencés aux VERTs précédemment définis.

$pgon_1$, $pgon_2$:
index des polygones voisins. Zéro et des valeurs négatives ont la signification suivante:

0 : arête de côté ou isolée.

<0 : ArchiCAD recherche les voisins possibles.

Bits d'état:

1	arête invisible.
2	arête d'une surface courbe.

Bits réservés pour usage ultérieur:

4	première arête d'une surface courbe (avec 2 seulement).
8	dernière arête d'une surface courbe (avec 2 seulement)
16	l'arête est un arc de cercle
32	premier segment d'un arc (avec 16 seulement)
64	dernier segment d'un arc (avec 16 seulement).

PGON `n, ivect, status, edge`$_1$`, edge`$_2$`, . . . edge`$_n$

n : nombre d'arêtes dans la liste des arêtes

ivect : indice du vecteur de normale. Il doit faire référence à un `VECT` défini préalablement.

Si ivect = 0, ArchiCAD calcule le vecteur de normale pendant l'analyse.

Les index d'arête edge$_1$, edge$_2$, . . . edge$_n$ doivent se référer à des EDGEs préalablement définis. Une valeur de 0 signifie le début ou la fin d'une définition de trou.

Un index négatif change à l'opposée dans le polygone la direction du vecteur normal ou arête stockée. (Le vecteur ou l'arête stockée ne change pas; d'autres polygones peuvent s'y référer en utilisant l'orientation originale avec un index positif.)

Bits d'état:

1	polygone invisible
2	polygone d'une surface courbe
16	polygone concave
32	polygone avec trou(s)
64	trou(s) convexe(s) (avec 32 seulement)

Bits réservés pour usage ultérieur:

4	premier polygone d'une surface courbe (avec 2 seulement).
8	dernier polygone d'une surface courbe (avec 2 seulement).

Si une valeur d'état est négative, ArchiCAD calcule l'état du polygone (comme polygone concave ou polygone avec trous).

n = 0 est possible pour des cas spéciaux.

PIPG `filename, a, b, mask, n, ivect, status, edge`$_1$`, edge`$_2$`, . . . edge`$_n$

Définition de polygone d'image. Les quatre premiers paramètres sont les mêmes que dans l'élément `PICTURE`, les autres sont les mêmes que dans l'élément `PGON`.

COOR `wrap, vert`$_1$`, vert`$_2$`, vert`$_3$`, vert`$_4$

Système de coordonnées local d'un `BODY` pour mapping de hachure ou de texture.

wrap : mode + type de projection

Modes de wrapping:

1 : plan

2 : boîte

3 : cylindrique

4 : sphérique

5 : même que le mode cylindrique, sauf que dans le rendu, les surfaces de dessus et de dessous auront un mapping circulaire.

Type de projection:

256 : la hachure commence toujours à l'origine du système de coordonnées local

1024 : projection de texture quadratique (recommandée)

2048 : la projection linéaire de texture est fondée sur la distance moyenne

4096 : la projection linéaire de texture est fondée sur une triangulation normale.

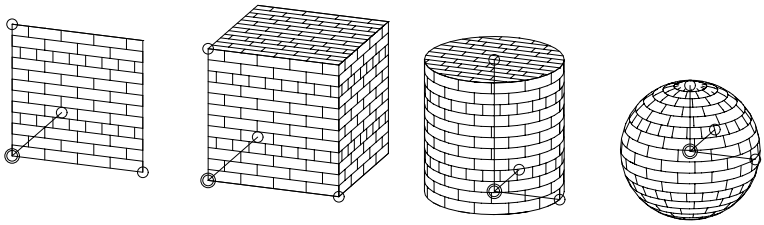
Les trois dernières valeurs n'ont d'effet qu'avec la définition de coordonnées personnalisées pour la texture (voir instruction `TEVE`).

vert$_1$: index d'un `VERT`, représentant l'origine du système de coordonnées local.

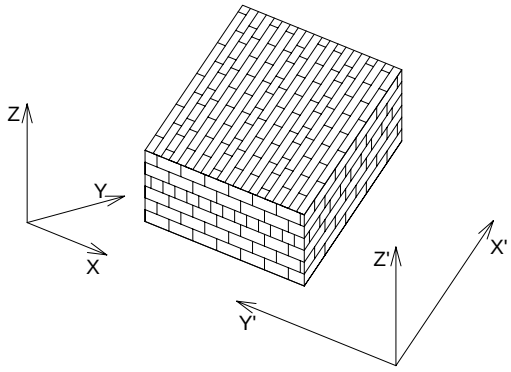
vert$_2$, vert$_3$, vert$_4$:

index des `VERT`s définissant les trois axes de coordonnées.

Placez un signe moins (-) devant l'index de VERT s'ils sont utilisés seulement pour définir le système de coordonnées local.



Exemple pour les axes de texture personnalisées :




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

BODY `status`

Placement d'un corps avec les primitives précédentes.

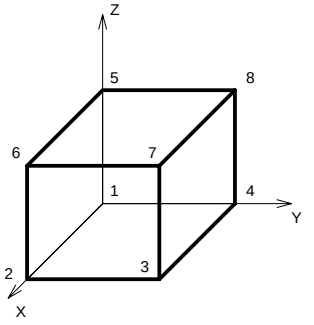
Bits d'état:

1	corps fermé
2	corps incluant une ou plusieurs surface(s) courbe(s)
4	modèle de surface: si le corps est coupé, aucune surface n'est créée sur le plan de coupe.
32	le corps est toujours accompagné d'ombre, indépendamment de l'algorithme de présélection automatique
64	le corps n'a jamais d'ombre

Si ni 32, ni 64 ne sont définis, la présélection automatique d'ombre s'effectue. Voir SHADOW.

Si l'état est négatif, ArchiCAD recherche l'état du corps.

Exemple :



1: Description complète

```
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: (pas de référence directe aux polygones, ArchiCAD les calcule)

```
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

Remet à zéro les compteurs pour les instructions d'éléments géométriques de bas niveau (VERT, VECT, EDGE et PGON). Implicitement exécutée après toute définition d'élément composé.

5.6 Utiliser les données 3D binaires

BINARY `mode [, section]`

Commande spéciale pour inclure des objets binaires dans une macro **GDL**. Un ensemble de nœuds, de vecteurs, d'arêtes, de polygones, de corps et de matières est lu dans une section spéciale du fichier Elément de bibliothèque. Ils sont transformés selon les transformations courantes et fusionnés dans le modèle 3D. Les données de la section binaire ne sont pas éditables par l'utilisateur.

mode:

- 0: les réglages courants de `PEN` et de `MATERIAL` sont appliqués.
- 1: les réglages courants de `PEN` et de `MATERIAL` n'ont pas d'effet. L'élément de bibliothèque sera affiché avec les définitions de couleurs et de matières stockées. L'apparence de la surface est constante.
- 2: les réglages de `PEN` et de `MATERIAL` stockés sont utlilisés, les matières non définies sont remplacées par les réglages courants.
- 3: les réglages de `PEN` et de `MATERIAL` stockés sont utlilisés, les matières non définies sont remplacées par les attributs stockés par défaut.

section: index de la partie binaire, entre 1 et 16

En utilisant 0 comme index de section, la référence est faite simultanément à tous les éléments binaires existants.

Seules des sections avec une valeur d'index de 1 peuvent être enregistrées dans GDL, les commandes `BINARY` sans argument de section font également référence à cet index. Les autres index de section sont utilisés par les outils externes (StairMaker, ArchiSite, VisualGDL, etc.).

Si vous ouvrez des fichiers avec une structure de donnée différente de celle d'ArchiCAD (par exemple DXF, ZOOM) leur description 3D sera convertie au format binaire.

Enregistrer est possible dans la fenêtre d'édition de l'élément de bibliothèque avec la commande *Enregistrer sous...* Si la case de

contrôle *Format binaire* est marquée, le **GDL** de l'article de bibliothèque sera remplacé par une description binaire.

Enregistrer le modèle 3D en format binaire après une opération de coupe 3D va enregistrer un modèle tronqué, vous permettant de créer des formes coupées.

Vous ne pouvez enregistrer l'élément de bibliothèque en format binaire que si son modèle 3D a déjà été généré, c.à.d. que vous avez visualisé au moins une fois la vue 3D de l'élément.

Remplacer la description **GDL** de l'article de bibliothèque par une description binaire, réduit considérablement le temps de conversion 3D de l'élément.

En revanche, la description binaire 3D est non paramètrable et plus volumineux qu'un programme algorithmique **GDL**.

5.7 Plans de coupe 3D

```
CUTPLANE  [x, y, z [, side]]
              [stmt1
              stmt2
              ...
              stmtn]
  CUTEND
```

ou

```
CUTPLANE  angle
              [stmt1
              stmt2
              ...
              stmtn]
  CUTEND
```

Crée un plan de coupe et supprime les parties coupées des formes englobées. CUTPLANE peut avoir un nombre de paramètres varié.

Si CUTPLANE a:

Zéro paramètre: plan x-y

1 paramètre: plan de coupe traverse l'axe x, *l'angle* est fermé par le plan de coupe et le plan x-y

2 paramètres: plan de coupe parallèle à l'axe z, traversant les axes x et y aux valeurs données

3 paramètres: traverse les axes x, y et z aux valeurs données

4 paramètres: trois premiers paramètres comme ci-haut,

side = 0: supprime les parties au-dessus du plan de coupe (défaut)

side = 1: supprimer les parties sous le plan de coupe; dans le cas de x-y, x-z, y-z, les parties dans la direction négative de l'axe.

Sans le paramètre *side*, la commande supprime les parties au-dessus du plan de coupe. Si les trois premiers paramètres définissent le plan x-y, x-z ou y-z (par exemple, 1.0, 1.0, 0.0 définit le plan x-y), les parties dans la direction positive du troisième axe seront supprimées.

Un nombre quelconque de commandes diverses peuvent être ajoutées entre CUTPLANE et CUTEND. Les macros aussi peuvent contenir des commandes CUTPLANE.

Les paramètres de CUTPLANE se réfèrent au système de coordonnées courant.

Les transformations entre CUTPLANE et CUTEND n'ont pas d'effet sur le plan de coupe donné, mais les CUTPLANEs suivants seront transofrmés. Il importe donc d'utiliser toutes les transformations nécessaires pour préparer CUTPLANE, puis de supprimer ces transformations avant de défnir les formes à couper.

Les couples de commandes CUTPLANE-CUTEND peuvent être emboîtées même dans des cycles. Si le dernier CUTEND manque, le CUTPLANE correspondant sera exécuté pour toutes les formes jusqu'à la fin du script.

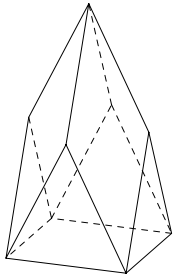
Dans une macro, CUTPLANE affecte uniquement les formes définies dans la macro donnée, même si CUTEND manque.

Si une macro est appelée entre CUTPLANE et CUTEND, les formes de la macro seront coupées.

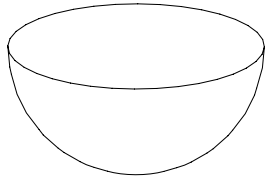
Les réglages de matière, de couleur et de hachure courants sont effectifs pour les surfaces coupées.

- Si CUTPLANE n'est pas fermé par CUTEND, toutes les formes peuvent être entièrement supprimées. C'est pourquoi les messages concernant les CUTENDs manquants apparaissent.
- Si les transformations utilisées uniquement pour positionner le CUTPLANE ne sont pas supprimées, vous pourriez penser que CUTPLANE est dans une mauvaise position, alors que ce sont les formes qui ont été déplacées.

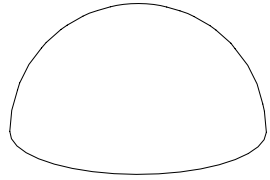
Exemples :



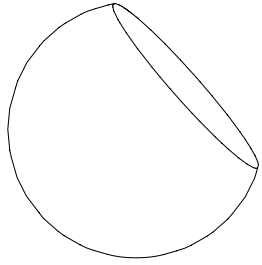
```
CUTPLANE      2,     2,     4
CUTPLANE      -2,    2,     4
CUTPLANE      -2,    -2,    4
CUTPLANE      2,     -2,    4
       ADD            -1,    -1,    0
       BRICK          2,     2,     4
       DEL            1
CUTEND
CUTEND
CUTEND
CUTEND
```



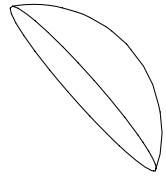
```
CUTPLANE
   SPHERE 2
CUTEND
```



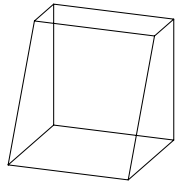
```
CUTPLANE 1, 1, 0, 1
   SPHERE 2
CUTEND
```



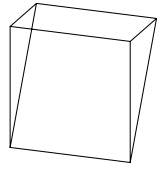
```
CUTPLANE 1.8, 1.8,
         1.8
   SPHERE 2
CUTEND
```



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



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



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

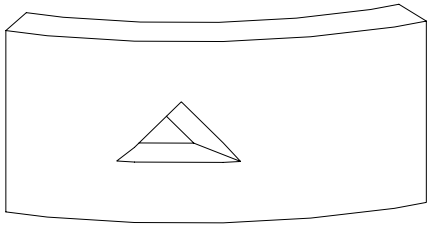
CUTPOLY n,
x_1, y_1, . . . x_n, y_n
[, x, y, z]
[stmt1
stmt2
...
stmtn]

CUTEND

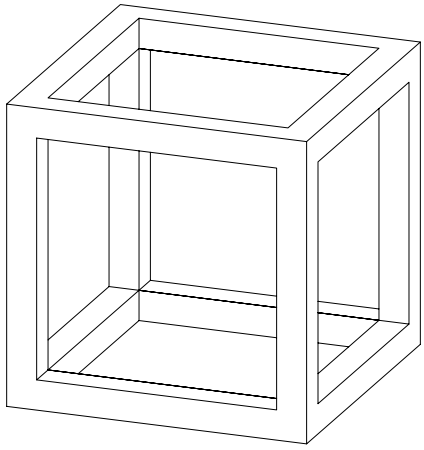
Similaire à la commande CUTPLANE. Les paramètres de CUTPOLY font référence au système de coordonnées courant. Le polygone doit être convexe et ne peut s'intersecter lui-même. La direction de la coupe est l'axe Z. Il est également possible de spécifier un autre vecteur (x, y, z).

Les paramètres définissent un "tube" infini. Le polygone est la coupe transversale du tube et la direction de la coupe est la direction du tube. Tout ce qui se trouve à l'intérieur du tube sera supprimé.

Exemples :

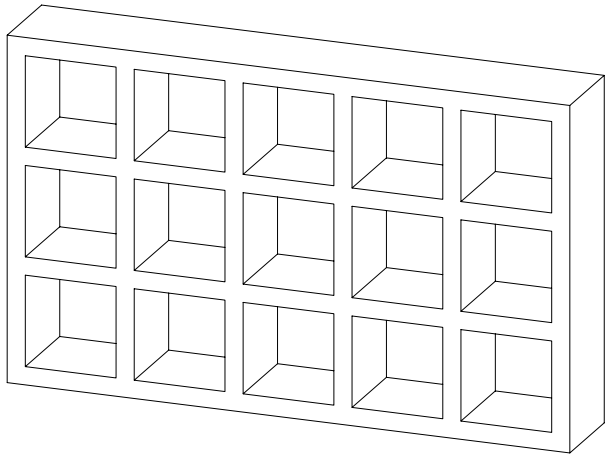


```
ROTX   90
MULZ   -1
CUTPOLY      3,
             0.5,   1,
             2,     2,
             3.5,   1,
             -1.8,  0,     1
      DEL    1
      BPRISM_ "Red brick", "Red brick", "Face
brick",
             4,     0.9,   7,
             0.0,   0.0,   15,
             6.0,   0.0,   15,
 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



```
A=1.0
D=0.1
CUTPOLY      4,
             D,     D,
             A-D,   D,
             A-D,   A-D,
             D,     A-D

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

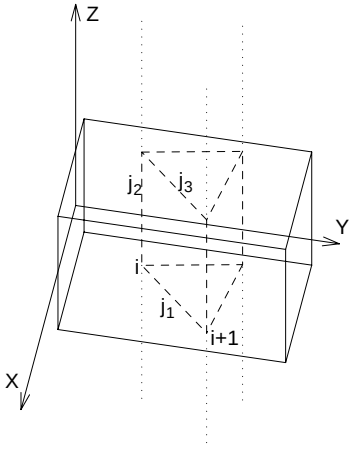


```
ROTX 90
FOR I=1 TO 3
      FOR J=1 TO 5
          CUTPOLY  4,
             0,     0,     1,     0,
             1,     1,     0,     1
          ADDX             1.2
      NEXT J
      DEL    5
      ADDY   1.2
NEXT I
DEL NTR()-1
ADD    -0.2,  -0.2,  0
BRICK  6.2,   3.8,   1
FOR K=1 TO 15
      CUTEND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

CUTPOLYA n, status, d,
x_1, y_1, $mask_1$, . . . x_n, y_n, $mask_n$
[, x, y, z]
[stmt1
stmt2
...
stmtn]

CUTEND

Similaire à la définition CUTPOLY, mais avec la possibilité de contrôler la visibilité des arêtes des polygones générés. La forme de la coupe est un tube infini d'un côté, avec une coupe transversale polygonale définie. L'extrémité de la forme de coupe ne peut pas entrer dans le corps.



codes d'état:

1: utiliser les propres attributs du corps pour les polygones et arêtes générés

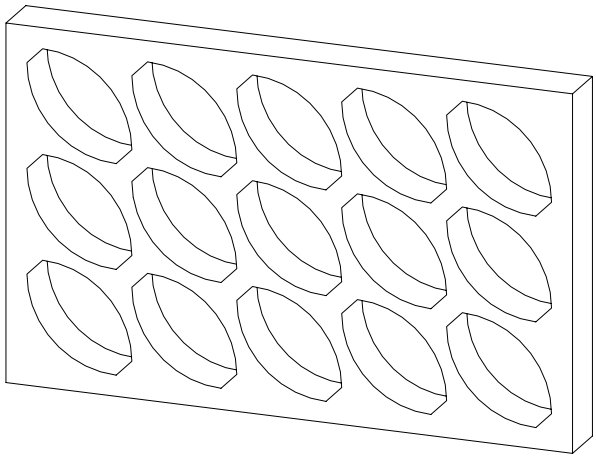
2: les polygones coupés générés seront traités comme des polygones normaux

d: la distance entre l'origine locale et l'extrémité du tube infini sur un côté
d = 0 signifie une coupe avec un tube infini des deux côtés

$mask_i$: similaire à l'instruction PRISM_

$mask_i = j_1 + 2 * j_2 + 4 * j_3$

Exemple :



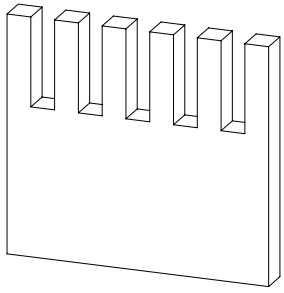
```
ROTX 90
FOR I=1 TO 3
       FOR J=1 TO 5
           CUTPOLYA 6,     1,     0,
                    1,     0.15,  5,
                    0.15,  0.15,  900,
                    0,     90,    4007,
                    0,     0.85,  5,
                    0.85,  0.85,  900,
                    0,     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 d
[stmt1
stmt2
...
stmtn]
CUTEND

Si d = 0.0 la forme de coupe est le plan X-Y. La coupe supprime la partie au-dessus du plan X-Y.

d < 0.0 signifie une coupe en forme de L. La partie au-dessus du plan X-Y (x 0) sera supprimée.

d > 0.0 signifie une coupe en forme de U. Similairement à la coupe en forme de L, la partie au-dessus du plan X-Y (0 ≤ x ≤ d) sera supprimée.



```
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
```

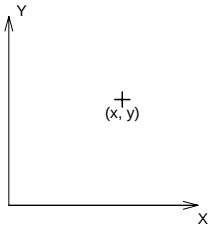
Chapitre 6
Formes 2D

Ce chapitre présente les commandes pour générer des formes 2D, allant des simples formes comme des lignes et arcs jusqu'aux polygones et splines, ainsi que la définition d'éléments de texte 2D. Il couvre également la manière dont les données binaires sont traitées en 2D, ainsi que la projection dans la vue 2D de la forme créée en 3D, afin d'assurer la cohérence entre les aspects 2D et 3D. D'autres commandes permettent de placer des éléments graphiques dans les listes d'éléments créés pour les calculs.

6.1 Eléments de dessin

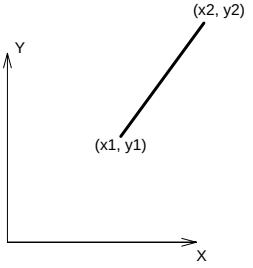
HOTSPOT2 x, y [, unID]

unID est l'identificateur unique du point chaud dans le script 2D. Cela peut se révéler utile si le nombre de points chauds est variable.



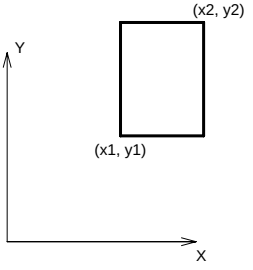
LINE2 x_1, y_1, x_2, y_2

Définit une ligne entre deux points.



RECT2 x_1, y_1, x_2, y_2

Définit un rectangle par deux noeuds.

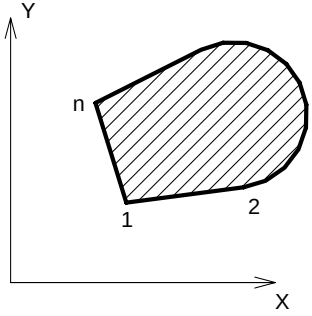


POLY2 n, framefill, x_1, y_1, ... x_n, y_n

Un polygone ouvert ou fermé avec un nombre n d'arêtes.

Restriction de paramètres:

$n \geq 2$



framefill = $j_1 + 2*j_2 + 4*j_3$

où j_1, j_2, j_3 peuvent être 0 ou 1.

j_1 (1): contour seulement

j_2 (2): hachure seulement

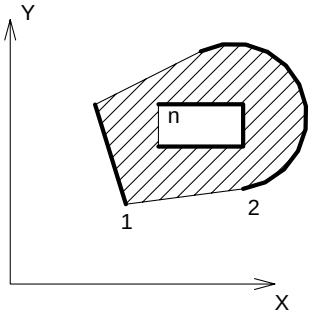
j_3 (4): fermer un polygone ouvert.

POLY2_ n, framefill, x_1, y_1, *mask*$_1$, ... x_n, y_n, *mask*$_n$

Identique à l'instruction POLY2, sauf que des arêtes peuvent être omises. Si $mask_i = 0$, l'arête partant du nœud (x_i, y_i) sera omise. Si $mask_i = 1$, le nœud est affiché. $Mask_i = 1$ sert à définir direcement des trous. Voir PRISM_ pour les détails.

Restriction de paramètres:

$n \geq 2$



framefill = $j_1 + 2*j_2 + 4*j_3$

où j_1, j_2, j_3 peuvent être 0 ou 1.

j_1 (1): contour seulement

j_2 (2): hachure seulement

j_3 (4): fermer un polygone ouvert.

$mask_i$: 0: le segment suivant est invisible

1: le segment suivant est visible

-1: fin d'un contour

POLY2_A `n, framefill, fillpen,`
`x1, y1, mask1, ..., xn, yn, maskn`

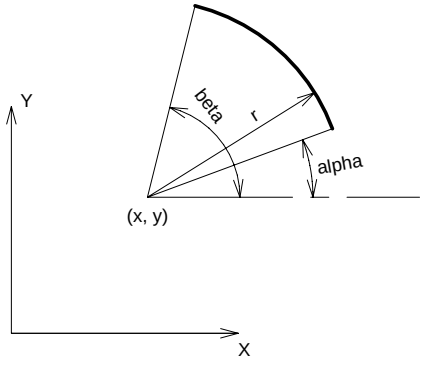
POLY2_B `n, framefill, fillpen,`
`fillbkgdpen,`
`x1, y1, mask1, ..., xn, yn, maskn`

Version avancée de la commande `POLY2_` avec des paramètres additionnels: la couleur de la hachure et la couleur de fond de la hachure. Tous les autres paramètres sont identiques à ceux de l'instruction `POLY2_`.

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

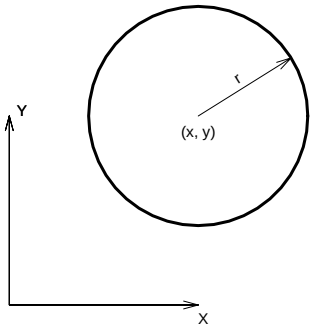
Un arc avec son centre en (x, y) entre les angles alpha et bêta, avec un rayon r.

Alpha et bêta sont en degrés.



CIRCLE2 `x, y, r`

Un cercle avec son centre en (x, y) et un rayon r.

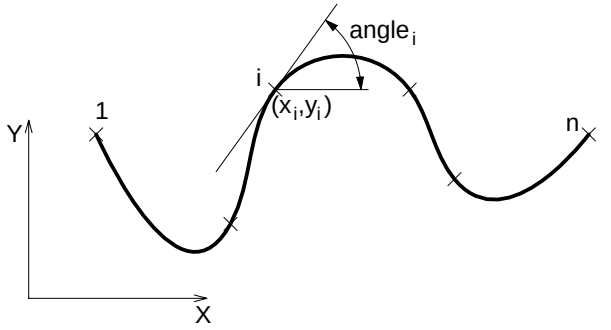


SPLINE2 `n, status, x1, y1, angle1,…, xn, yn, anglen`

Restriction de paramètres:

n 2

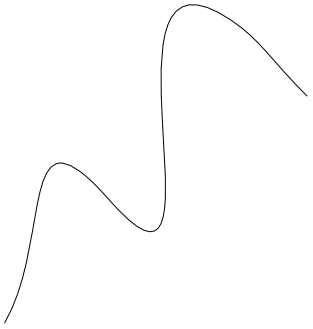
Spline avec un nombre n de points de contrôle. La tangente du point de contrôle (x_i, y_i) est définie par l'$angle_i$, l'angle avec l'axe x est en degrés.



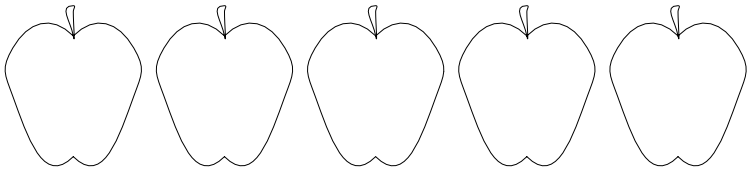
Bits d'état:

0:	défaut
1:	spline fermée, le dernier nœud est connecté au premier pour fermer la spline
2:	spline lissée automatiquement, la valeur du paramètre angle entre le premier et le dernier nœud n'est pas utilisé pour générer la spline. Un algorithme interne sera utilisé pour lisser la spline automatiquement.

Exemples:



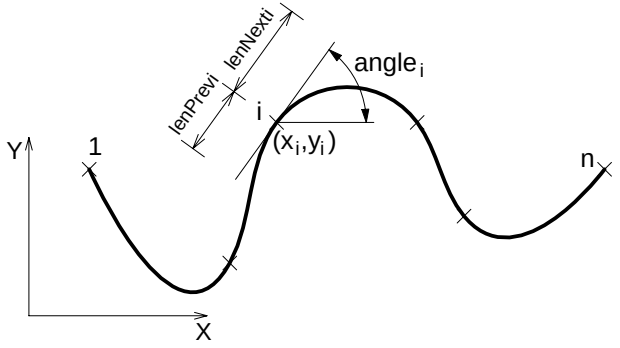
```
SPLINE2      5,     2,
       0,     0,     60,
       1,     2,     30,
       1.5,   1.5,   -30,
       3,     4,     45,
       4,     3,     -45
```



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

SPLINE2_A `n, status,`
x_1, y_1, $angle_1$, $lenPrev_1$, $lenNext_1$,
...
x_n, y_n, $angle_n$, $lenPrev_n$, $lenNext_n$

Extension de l'instruction `SPLINE2` (courbe de Bézier), utilisée principalement dans la génération automatique du script 2D en raison de sa complexité (pour détails, voir le Guide de référence ArchiCAD).



Bits d'état:

0: défaut

1: spline fermée, le dernier nœud est connecté au premier pour fermer la spline

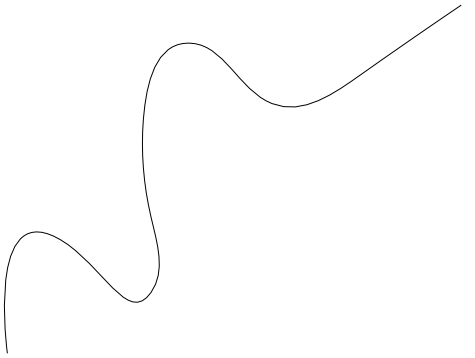
2: spline lissée automatiquement, les valeur des paramètres angle, $lenPrev_i$ et $lenNext_i$ des nœuds entre le premier et le dernier nœud ne sont pas utilisés pour générer la spline. Un algorithme interne sera utilisé pour lisser la spline automatiquement.

x_i, y_i: coordonnées des points de contrôle

$lenPrev_i$, $lenNext_i$: longueurs de tangente des points de contrôle précédent et suivant

$angle_i$: angle de direction de tangente

Exemple:



```
SPLINE2_A    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 `expression, a, b, mask`

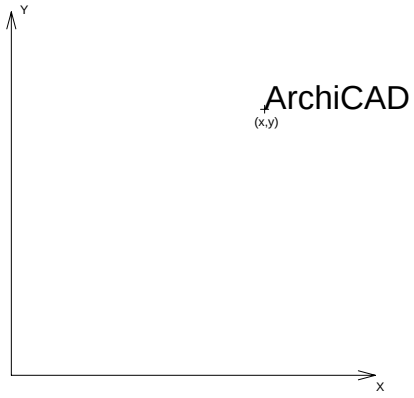
Cette instruction peut être utilisée en 2D similairement à la commande `PICTURE` disponible en 3D. Contrairement au 3D, les valeurs de masque n'ont pas d'effet en 2D.

Une expression de type texte signifie un nom de fichier. Une expression numérique signifie l'index d'une image conservée à l'intérieur de l'élément de bibliothèque. L'index 0 a une valeur spéciale, il fait référence à l'image de prévisualisation de l'élément de bibliothèque. D'autres images ne peuvent être conservées dans un élément de bibliothèque qu'au cas où vous enregistrez le projet ou des éléments sélectionnés contenant des images sous forme d'Objet ArchiCAD.

6.2 Elément texte

TEXT2 `x, y, expression`

La valeur d'une expression calculée de type numérique ou texte est placée dans le style défini aux coordonnées x, y. Voir aussi les commandes [SET] `STYLE` et `DEFINE STYLE`.



6.3 Utiliser les données binaires 2D

FRAGMENT2 `fragment_index, use_current_attributes_flag`

Le fragment avec l'index donné est inséré dans la Vue complète 2D avec les transformations courantes.

use_current_attributes_flag :

0: Le fragment apparaît avec la couleur, le type de ligne et la hachure définis

1: Les réglages courants du script sont utilisés au lieu de la couleur, du type de ligne et de la hachure du fragment

FRAGMENT2 **ALL**, `use_current_attributes_flag`

Tous les fragments sont insérés dans la Vue complète 2D avec les transformations courantes.

use_current_attributes_flag :

0: Les fragments apparaissent avec la couleur, le type de ligne et la hachure définis

1: Les réglages courants du script sont utilisés au lieu de la couleur, du type de ligne et de la hachure des fragments

6.4 Projections 3D placées en 2D

PROJECT2 `projcode, angle, method`

Crée une projection du script 3D dans le même élément de bibliothèque et ajout les lignes générées au symbole paramétrique 2D.

projcode:

3: Vue de dessus
4: Vue latérale
6: Axonométrie frontale
7: Axonométrie isométrique
8: Axonométrie monométrique
9: Axonométrie dimétrique

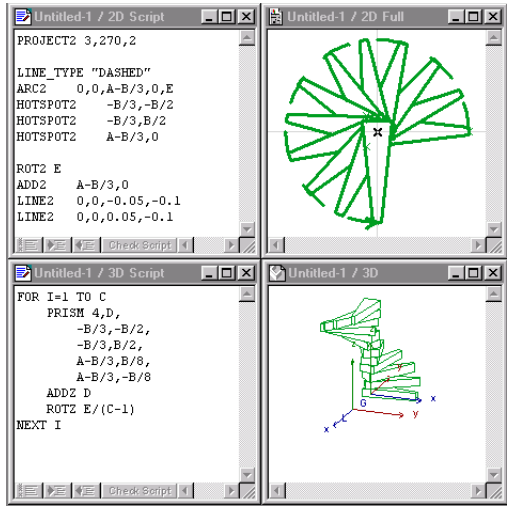
-3: Vue de dessous
-6: Axon. frontale de dessous
-7: Axon. isométrique de dessous
-8: Axon. monométrique de dessous
-9: Axon. dimétrique de dessous

angle: angle d'azimut défini dans le dialogue Points de vue.

method:

1: fil-de-fer
2: lignes cachées analytiques

Exemple:



6.5 Dessins dans la liste

Les commandes qui suivent n'ont d'effet que lors de la création d'une liste d'éléments.

Les éléments de bibliothèque de type descriptitf associés à un élément de bibliothèque de type Objet, Porte, Fenêtre ou Lampe placé sur le plan et dont le script 2D comprend ces commandes, font référence aux parties 2D et 3D de cet élément de bibliothèque. Ceci est une référence virtuelle, résolue pendant la création de la liste en utilisant le script 2D ou 3D des éléments actuellement listés.

DRAWING2 `[expression]`

Crée, en fonction de la valeur de l'expression, un dessin de l'élément de bibliothèque (expression = 0 par défaut) ou de l'étiquette de l'élément (expression = 1) associé à l'objet propriétaire contenant cette commande.

DRAWING3 `projcode, angle, method`

Similiairement à `PROJECT2`, crée une projection du script 3D de l'élément de bibliothèque associé à l'élément de bibliothèque de type descriptif comprenant cette commande. Tous ses paramètres sont identiques à ceux de `PROJECT2`.

Chapitre 7

Codes d'état additionnels pour polylignes plan

Les codes de masque et d'état suivants permettent de créer des segments et des arcs dans les polyligne plan en utilisant des contraintes spéciales.

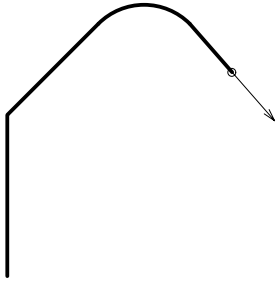
Les polylignes plan avec codes de masque et d'état à leurs nœuds servent de base à bien des éléments GDL:

POLY_	$mask_i$
PRISM_	$mask_i$
CPRISM_	$mask_i$
BPRISM_	$mask_i$
FPRISM_	$mask_i$
SPRISM_	$mask_i$
CROOF_	$mask_i$
EXTRUDE	s_i
PYRAMID	s_i
REVOLVE	s_i
SWEEP	s_i
TUBE	s_i
TUBEA	s_i
POLY2_	$mask_i$
POLY2_A	$mask_i$
POLY2_B	$mask_i$

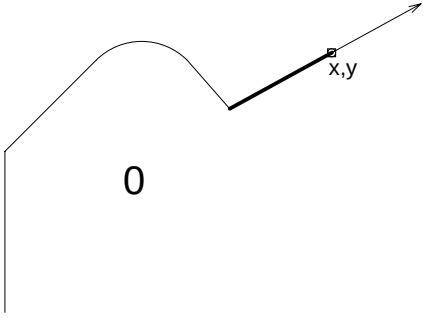
La résolution des arcs est régie par les directives décrites au chapitre "Attributs".

Dans le cas de l'instruction POLY2_, si la résolution est supérieure à 8, des arcs véritables seront générés, sinon, tous les arcs créés seront segmentés.

Les codes masque/état additionnels suivants permettent de créer des segments et des arcs dans la polyligne en utilisant des contraintes spéciales. Ils font référence au segment ou à l'arc suivant. Les codes de masque et d'état initiaux n'ont d'effet qu'à condition de l'avoir spécifié (un "+s" doit être inclus après le code additionnel).

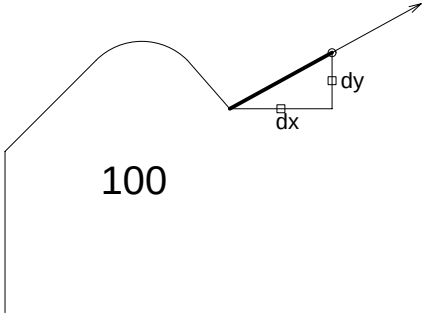


Partie précédente de la polyligne: la position courante et la tangente sont définies.



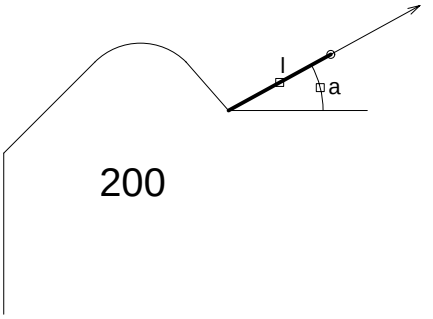
Segment par extrémité absolue

```
x, y, s,
      où 0 ≤  s ≤ 100
```



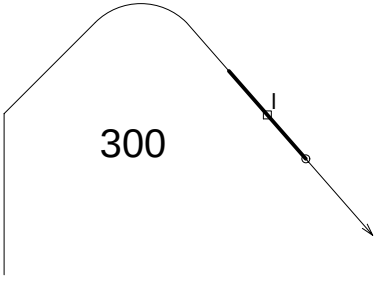
Segment par extrémité relative

```
dx, dy, 100+s,
      où 0 ≤  s ≤  100
```



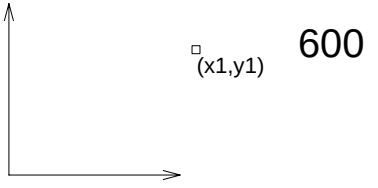
Segment par longueur et direction

```
l, a, 200+s,
      où 0 ≤ s ≤ 100
```



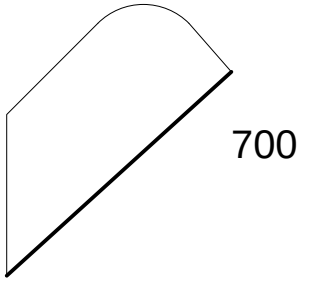
Segment tangentiel par longueur

```
l, 0, 300+s,
      où 0 ≤  s ≤  100
```



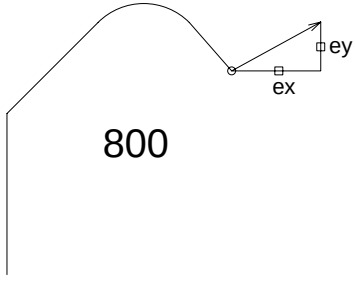
Définir point de départ

```
x1, y1, 600,
```



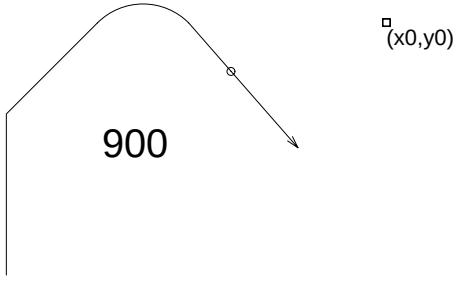
Fermer polyligne

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



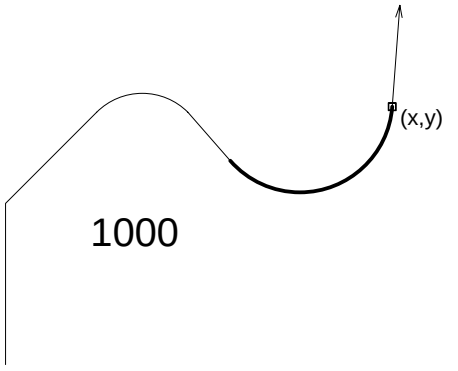
Définir tangente

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



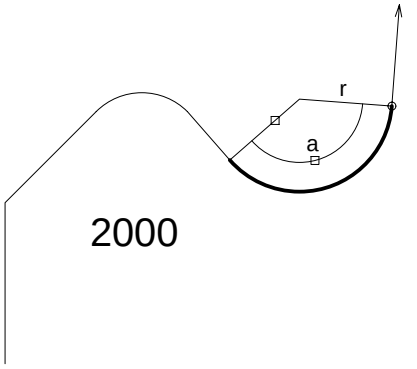
Définir centre

```
x0, y0, 900,
```



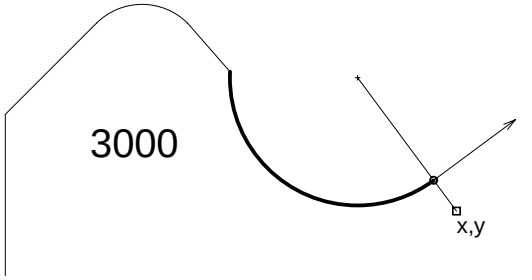
Arc tangentiel à extrémité

```
x, y, 1000+s,
      où 0 ≤  s ≤ 100
```



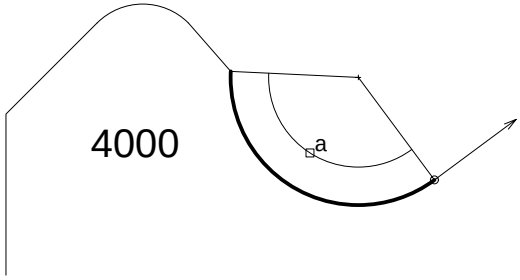
Arc tangentiel par rayon et angle

```
r, a, 2000+s,
      où 0 ≤ s ≤ 100
```



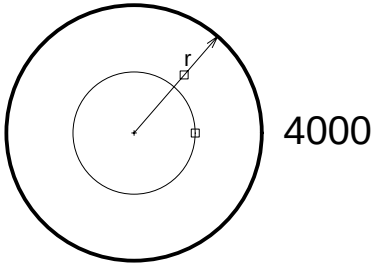
Arc utilisant centre et point sur le rayon final

```
x, y, 3000+s,
      où 0 ≤ s ≤ 100
```



Arc utilisant centre et angle

`0, a, 4000+s,`
`où 0 ≤ s ≤ 100`



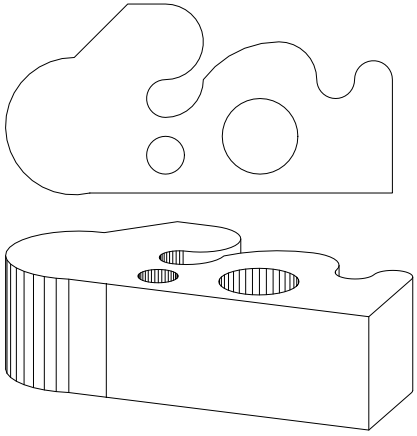
Cercle utilisant centre et rayon

`r, 360, 4000+s,`
`où 0 ≤ s ≤ 100`

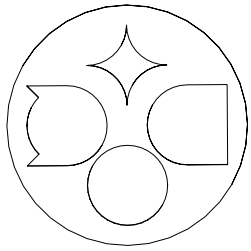
Dans ce cas l'état s fait référence au cercle entier.

Toutes les valeurs angulaires sont en degrés. Les coordonnées omises marquées par 0 (pour les codes 300, 700, 4000) peuvent avoir n'importe quelle valeur.

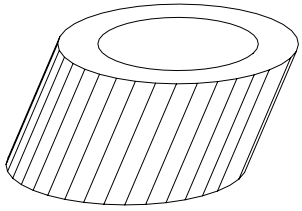
Exemples:



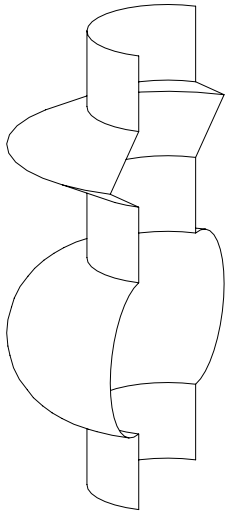
```
EXTRUDE       21,    0,     0,     3, 1+2+4+16+32,
 0,    0,      0,
 7,    0,      0,
 7,    3,      1,
 6,    3,      1000,   ! tangential arc to endpoint
 5,    3,      1001,   ! tangential arc to endpoint
 1,     90,    2000,   ! tangential arc by radius
                       !  and angle
 2,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



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



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



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

Chapitre 8
Attributs

La première partie de ce chapitre couvre les directives qui influencent l'interprétation des instructions GDL subséquentes. Les directives peuvent définir le lissage des éléments cylindriques, le mode de représentation dans la vue 3D, ou la création d'un attribut (couleur, matière, texte, style, ect.) préalable à la première référence.

La définition d'attributs est traitée dans la deuxième partie du chapitre. Cette particularité permet aux utilisteurs de personnaliser les matières, textures, hachures, types de lignes et styles de texte assignés aux objets qui ne sont pas présents dans l'ensemble d'attribut courant de votre projet.

8.1 Directives

Ces directives influencent l'interprétation des instructions **GDL** subséquentes. Leur influence reste valide jusqu'à la prochaine directive ou la fin du document. Les documents appelés héritent des réglages courants, les changements ont une influence locale. Au retour du script les réglages sont remis aux valeurs d'avant l'appel de la macro.

Directives de scripts 2D et 3D

[LET] `varnam = n`

Affectation de valeur. La directive `LET` est optionnelle. La variable mémorise la valeur évaluée de n.

RADIUS `rmin, rmax`

Définit le lissage des éléments cylindriques et des arcs dans les polylignes.

Un cercle de rayon r est représenté:

si r ≤ rmin, par un hexagone,

si r ≥ rmax, par un polygone à 36 arêtes,

si rmin < r < rmax, par un polygone à (6+30*(r–rmin)/(rmax–rmin)) arêtes.

La conversion des arcs est proportionnelle.

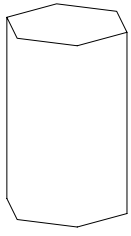
Après une instruction `RADIUS`, les instructions `RESOL` et `TOLER` précédentes perdent leur effet.

Restriction de paramètres:

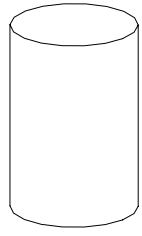
```
rmin ≤ rmax
```

Exemples:

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



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



RESOL n

Définit le lissage des éléments cylindriques et des arcs dans les polylignes. Les cercles sont convertis en polygones réguliers à n côtés.

La conversion des arcs est proportionnelle.

Après une instruction RESOL, les instructions RADIUS et TOLER précédentes perdent leur effet.

Restriction de paramètres:

```
n ≥ 3
```

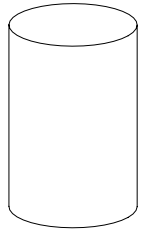
Défaut:

```
RESOL 36
```

Exemples:

```
RESOL  5
CYLIND 3.0, 1.0
```

```
RESOL  36
CYLIND 3.0, 1.0
```



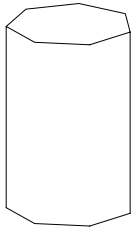
TOLER d

Définit le lissage des éléments cylindriques et des arcs dans les polylignes. L'erreur d'approximation de l'arc (à savoir la plus grande distance entre l'arc théorique et la corde générée) sera plus petite que d.

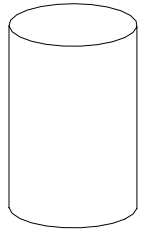
Après une instruction TOLER, les instructions RADIUS et RESOL précédentes perdent leur effet.

Exemples:

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



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



Les directives RADIUS, RESOL et TOLER définissent le lissages des éléments cylindriques 3D (CIRCLE, ARC, CYLIND, SPHERE, ELLIPS, CONE, ARMC, ARME, ELBOW, REVOLVE) et des arcs dans les polylignes 2D utilisant des courbes (voir chapitre "Codes d'état additionnels pour les polylignes plan").

PEN n

Définit la couleur actuelle.

Restriction de paramètres :

0 < n ≤ 99

Défaut :

```
PEN 1
```

s'il n'y a pas d'instruction PEN dans le script.

(Pour les Eléments de bibliothèque, ArchiCAD lit les valeurs par défaut dans les réglages de l'Elément de bibliothèque. Si le script se réfère à un index non existant, PEN 1 devient le réglage par défaut.)

[SET] STYLE name_string
[SET] STYLE index

Tous les textes générés auront ce jusqu'à la prochaine instruction SET STYLE.

L'index est une constante se référant à une pile de styles dans la structure de données interne d'ArchiCAD. Cette pile est modifiée durant l'analyse **GDL** et peut aussi être modifiée depuis le programme. L'utilisation de l'index au lieu du nom du style est seulement recommandée avec une utilisation préalable de la fonction IND décrite plus haut..

Défaut:

```
SET STYLE 0
```

(police de l'application, taille 5 mm, ancrage = 1, normal) s'il n'y a pas d'instruction SET STYLE dans le script.

Voir aussi la fonction IND dans l'Annexe.

Directives de scripts 3D seulement

MODEL WIRE
MODEL SURFACE
MODEL SOLID

Définit le mode de représentation dans le document courant.

MODEL WIRE : fil-de fer seulement, ni surfaces, ni volumes.

Les objets sont transparents.

MODEL SURFACE, MODEL SOLID : La génération des surfaces de section est fondée sur la relations des surfaces de côté, ce qui fait que les deux méthodes génèrent la même structure de données 3D interne. Les objets sont opaques.

La seule différence devient visibles après avoir coupé une partie du corps:

MODEL SURFACE: l'intérieur des corps sera visible,

MODEL SOLID: de nouvelles surfaces peuvent apparaître.

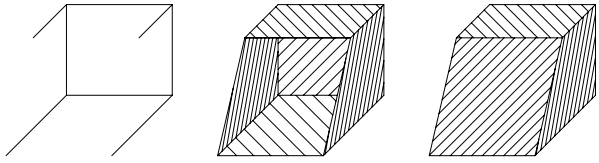
Défaut:

```
MODEL SOLID
```

Les trois blocs suivants illustrent les trois méthodes de modelage :

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

Après les avoir coupés avec un plan :



[SET] MATERIAL name_string
[SET] MATERIAL index

Toutes les surfaces générées représenteront cette matière jusqu'à la prochaine instruction MATERIAL. Les surfaces des corps BPRISM_, CPRISM_, FPRISM_, SPRISM_, CSLAB_, CWA mLL_, BWALL_, XWALL_, CROOF_ et MASS font exception à cette règle.

L'index est une constante se référant à une pile de matières dans la structure de données interne d'ArchiCAD. Cette pile est modifiée durant l'analyse **GDL** et peut aussi être modifiée depuis le programme. L'utilisation de l'index au lieu du nom de la matière est seulement recommandée avec une utilisation préalable de la fonction IND.

L'index 0 a une signification spéciale: les surfaces utilisent la couleur de stylo actuel et elles ont une apparence mate.

Défaut:

```
MATERIAL 0
```

s'il n'y a pas d'instruction MATERIAL dans le script.

(ArchiCAD lit les valeurs par défaut dans les réglages de l'Elément de bibliothèque. Si le script se réfère à un index non existant, MATERIAL 0 devient le réglage par défaut.)

Voir aussi la fonction IND dans l'Annexe.

SECT_FILL fill, fillbkgdpen, fillpen, contourpen

Définit les hachures utilisées pour les éléments 3D générés par la suite dans la fenêtre Coupe/Façade.

fill : le numéro d'index de la hachure

fillbkgdpen : le numéro de stylo du fond de la hachure

fillpen : le numéro de stylo de la hachure

contourpen : le numéro de stylo du contour

SHADOW keyword1[, keyword2]

Contrôle les ombres des éléments dans le Rendu photoréaliste.

keyword1: ON, AUTO ou OFF

keyword2: ON ou OFF

ON: tous les éléments porteront des ombres dans tous les cas

OFF: aucun élément ne portera d'ombre dans aucun cas

AUTO: les ombrages seront déterminés automatiquement

Régler SHADOW OFF pour les parties cachées fait gagner du temps et de la place.

Régler SHADOW ON assure que même de petits détails porteront des ombres.

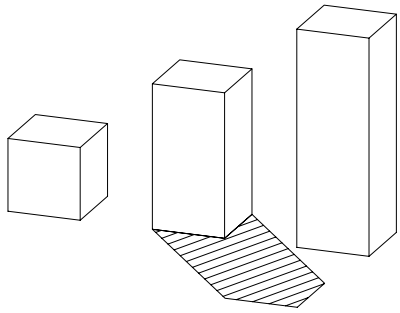
Le second mot-clef optionnel contrôle l'apparence des ombres sur les surfaces.

SHADOW keyword1, OFF désactive les ombres vectorisées sur les surfaces suivantes.

SHADOW keyword1, ON active de nouveau les ombres vectorisées.

Défaut:

SHADOW AUTO



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

Directives de scripts 2D seulement

DRAWINDEX nr

Définit l'ordre de dessin des éléments du script 2D. Les éléments dont le numéro *drawindex* est inférieur seront dessinés en premier.

Restriction de paramètre : 0 < n° <= 50

(Dans la version actuelle de GDL, seules les valeurs 10, 20, 30, 40 et 50 de *drawindex* sont valables. Les autres valeurs seront arrondies pour les atteindre.)

En l'absence de toute directive DRAWINDEX, l'ordre de dessin par défaut est le suivant.

1) Figures
2) Hachures
3) Lignes
4) Eléments textuels

[**SET**] **FILL** name_string
[**SET**] **FILL** index

Tous les polygones 2D générés auront cette hachure jusqu'à la prochaine instruction SET FILL.

L'index est une constante se référant à une pile de hachures dans la structure de données interne d'ArchiCAD. Cette pile est modifiée durant l'analyse **GDL** et peut aussi être modifiée depuis le programme. L'utilisation de l'index au lieu du nom de la hachure est seulement recommandée avec une utilisation préalable de la fonction IND décrite plus haut.

Défaut:

```
SET FILL 0
```

hachure vide, s'il n'y a pas d'instruction SET FILL dans le script. Voir aussi la fonction IND dans l'Annexe.

[**SET**] **LINE_TYPE** name_string
[**SET**] **LINE_TYPE** index

Toutes les lignes 2D générées (lignes, arcs, lignes chaînées) auront ce type de ligne jusqu'à la prochaine instruction SET LINE_TYPE.

L'index est une constante se référant à une pile de types de lignes dans la structure de données interne d'ArchiCAD. Cette pile est modifiée durant l'analyse **GDL** et peut aussi être modifiée depuis le programme. L'utilisation de l'index au lieu du nom du type de ligne

est seulement recommandée avec une utilisation préalable de la fonction `IND`.

Défaut:

```
SET LINE_TYPE 1
```

ligne continue, s'il n'y a pas d'instruction `SET LINE_TYPE` dans le script. Voir aussi la fonction `IND` dans l'Annexe.

8.2 Définition d'attributs

Dans ArchiCAD, on peut créer des attributs en utilisant les dialogue de type de Matière, de Hachure et de Ligne. Ces attributs plan peuvent être références à partir de n'importe quel script GDL.
Il est également possible de définir des attributs dans les scripts GDL. Il y a deux cas possibles:

1. Définition d'attributs dans le script `MASTER_GDL`. Ce script est interprété quand la bibliothèque contenant le script est chargée en mémoire. Les attributs `MASTER_GDL` sont fusionnés avec les attributs du plan; les attributs ArchiCAD ayant les mêmes noms n'étant pas remplacés. Quand `MASTER_GDL` a été chargé, les attributs qui y ont été définis peuvent servir de référence pour n'importe quel script.
2. Définition d'attributs dans les éléments de bibliothèque. Les matières et textures définies de cette manière peuvent être utilisées dans ces scripts et leurs scripts dérivés. Les motifs de hachure et les types de ligne définis et utilisés dans les scripts 2D ont le même comportement que s'ils avaient été définis dans le script `MASTER_GDL`.

   La commande *Vérifier GDL Scripts* dans le dialogue principal de l'Elément de bibliothèque permet de vérifier si les paramètres de matière, de hachure, de type de ligne et de style sont corrects.

   Si une matière, une hachure, un type de ligne ou un style est différent dans l'interprétation 3D de l'élément de bibliothèque de celle que vous souhaitiez et aucun message d'alerte n'apparaît, cela veut probablement dire que les valeurs des paramètres sont incorrectes. La commande *Vérifier GDL Scripts* génère des messages détaillés pour trouver ces paramètres.

Définition de matière

DEFINE MATERIAL name type, m_1, m_2, . . . m_n

Tout script **GDL** peut contenir des définitions de matières préalables à la première référence à ce nom de matière. La matière ainsi définie peut être utilisée seulement dans le script où elle a été définie et dans ses scripts de seconde génération.

name : nom de la matière.

type : type de la matière. Le nombre concret (n) des paramètres qui définit la matière est différent dépendant du type. La signification des paramètres et leurs limites sont expliquées dans l'exemple.

0 : définition générale, n=16

1: définition simple, n=9 (Les paramètres supplémentaires sont des constantes ou sont calculées à partir des valeurs données.)

2-7: types de matière prédéfinis, n=3
Ces trois valeurs sont les composantes RVB de la couleur de surface. Les autres paramètres sont des constantes ou sont calculées à partir de la couleur.

2: mat
3: métal
4: plastique
5: verre
6: luisant
7: constant

10: définition générale avec paramètre hachure, n=17

11: définition simple avec paramètre hachure, n=10

12-17: types de matière prédéfinis avec paramètre hachure, n=4

20: définition générale avec paramètres hachure, index couleur de hachure et index de texture, n=19

21: définition simple avec paramètres hachure, index couleur de hachure et index de texture, n=12

22-27: types de matière prédéfinis avec paramètres hachure, index couleur de hachure et index de texture, n=6

Interprétations spéciales des types 20-27:

Si le numéro du stylo est zéro, les hachures vectorielles seront générées avec le stylo actif.

La valeur zéro permet de définir des matières sans hachure vectorielle ou texture.

Exemples :

```
DEFINE MATERIAL "water" 0,
  0.5284, 0.5989, 0.6167,
! surface RGB [0.0..1.0]
  1.0, 0.5, 0.5, 0.9,
! ambient, diffuse, specular, transparent
! coefficients [0.0..1.0]
  20,
! shining [0.0..100.0]
  1,
! transparency attenuation [0.0..4.0]
  0.5284, 0.5989, 0.6167,
! specular RGB [0.0..1.0]
  0, 0, 0,
! emission RGB [0.0..1.0]
  0.0
! emission attenuation [0.0..65.5]

DEFINE MATERIAL "asphalt" 1,
  0.1995, 0.2023, 0.2418
! surface RGB [0.0..1.0]
  1.0, 1.0, 0.0, 0.0,
! ambient, diffuse, specular, transparent
! coefficients [0.0..1.0]
  0,
! shining [0..100]
  0
! transparency attenuation [0..4]

DEFINE MATERIAL "matte red" 2,
  1.0,  0.0,  0.0
! surface RGB [0.0..1.0]

DEFINE MATERIAL "Red Brick" 10,
  0.878294, 0.398199, 0.109468,
  0.58, 0.85, 0.0, 0.0,
   0,
   0.0,
  0.878401, 0.513481, 0.412253,
  0.0, 0.0, 0.0,
  0,
  IND(FILL, "common brick")
! fill index
```

```
DEFINE MATERIAL     "Yellow Brick+*" 20,
       1,     1,     0,
! surface RGB [0.0 .. 1.0]
       0.58,    0.85,  0,  0,
! ambient, diffuse, specular, transparent
! coefficients [0.0 .. 1.0]
       0,
! shining [0.0 .. 100.0]
       0,
! transparency attenuation [0.0 .. 4.0]
       0.878401,    0.513481,    0.412253,
! specular RGB [0.0 .. 1.0]
       0,     0,     0,
! emission RGB [0.0 .. 1.0]
       0,
! emission attenuation [0.0 .. 65.5]
       IND(FILL, "Brick Jointing 25x75"), 61,
 IND(TEXTURE, "Brick")
! Fill index, color index, texture index
```

DEFINE TEXTURE `name,  filename, x, y, mask, angle`

Les scripts **GDL** peuvent inclure des définitions de texture préalables à la première référence à ce nom de texture. La texture ainsi définie peut être utilisée seulement dans le script où elle a été définie et dans ses scripts de seconde génération.

name: nom de la texture

filename: nom du fichier d'image

x: largeur logique de la texture

y: hauteur logique de la texture

mask: $j_1 + 2 * j_2 + 4 * j_3 + 8 * j_4 + 16 * j_5 +$
$32 * j_6 + 64 * j_7 + 128 * j_8 + 256 * j_9$

où j_1, j_2, j_3, j_4, j_5, j_6, j_7, j_8, j_9 peuvent être 0 ou 1

Contrôles du canal alpha ($j_1 \ldots j_6$):

j_1: canal alpha change la transparence de la texture

j_2: Bump mapping ou perturbation de la normale de surface.

Bump mapping utilise le canal alpha pour déterminer l'amplitude de la normale de surface.

j_3: canal alpha change la couleur diffuse de la texture

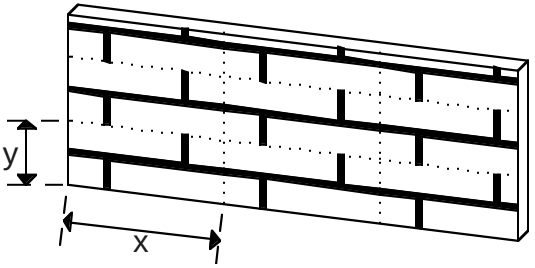
j_4: canal alpha change la couleur spéculaire de la texture

j_5: canal alpha change la couleur ambiente de la texture

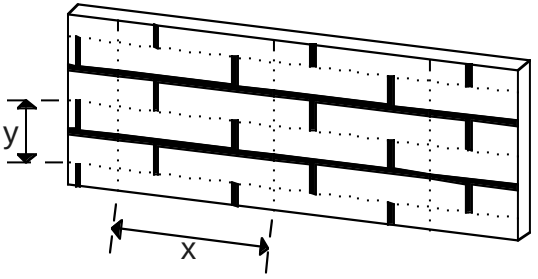
j_6: canal alpha change la couleur de surface de la texture

Contrôles de connexion (j_7… j_9):

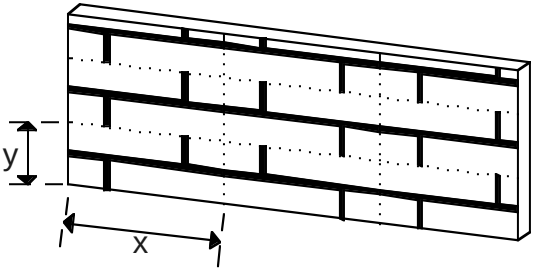
Si la valeur est zéro, le mode normal est sélectionné :



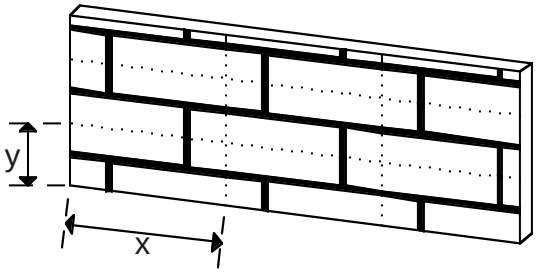
j_7: texture déplacée de manière aléatoire



j_8: symétrie en direction x



j_9: symétrie en direction y



angle: angle de la rotation de l'orientation naturelle.

Exemple :

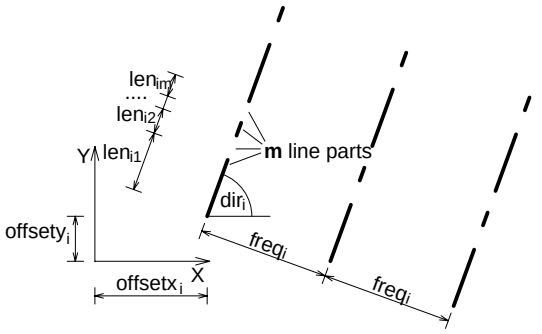
```
DEFINE TEXTURE  "Brick" "Brick.PICT",
 1.35, 0.3, 256+128, 35.0
```

Définition de hachure

DEFINE FILL name pat_1, pat_2, pat_3, pat_4, pat_5, pat_6, pat_7, pat_8,

spacing, angle, n,
$freq_1$, dir_1, $offsetx_1$, $offsety_1$, m_1,
len_{11}, . . . len_{m1},
...
$freq_n$, dir_n, $offsetx_n$, $offsety_n$, m_n,
len_{n1}, . . . len_{nm}

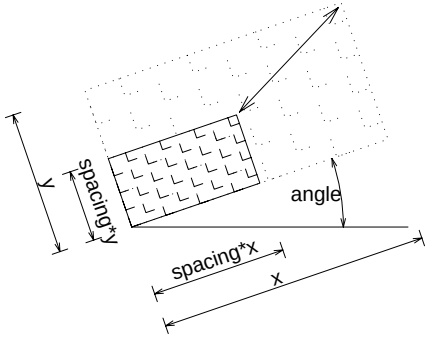
Les scripts GDL peuvent inclure des définitions de hachure préalables à la première référence à ce nom de hachure. La hachure ainsi définie peut être utilisée seulement dans le script où elle a été définie et dans ses scripts de seconde génération.



name : nom de la hachure.

pat_1, pat_2, pat_3, pat_4, pat_5, pat_6, pat_7, pat_8 :

définition de motif, huit chiffres entre 0 et 255 représentant des valeurs binaires. Définit le motif bitmap de la hachure.



spacing : espacement de la hachure - définit un facteur d'échelle global pour la hachure entière. Toutes les valeurs seront multipliées par ce chiffre dans les directions x et y.

angle : angle de rotation globale en degrés

n : nombre de lignes de la hachure

$freq_i$: fréquence de la ligne (la distance entre deux lignes est égale à spacing * $freq_{i)}$

dir_i : angle de direction de la ligne en degrés

$offsetx_i$,

$offsety_i$: décalage de la ligne par rapport à l'origine

m_i, nombre des parties de la ligne

len_{ij} longueur des parties de la ligne (longueur réelle égale à spacing * len_{ij}). Les parties de ligne sont des segments et des espaces qui se suivent. La première partie de ligne est un segment. La longueur zéro représente un point.

Le *motif bitmap* est seulement défini par les paramètres pat_1...pat_8 et est utilisé par ArchiCAD si l'option Options/Options d'affichage/ Hachures de polygones/Motifs bitmap a été choisie. Pour le définir, choisissez l'unité la plus petite de la hachure et représentez-la sous

formes de points et d'espaces vides sur une grille rectangulaire de 8x8. Les huit paramètres de motif sont la représentation décimale des valeurs binaires sur les lignes de la grille (un point est représenté par 1 et un espace vide par 0).

La *hachure vectorielle* (Options/Options d'affichage/Hachures de polygones/Hachures vectorielles) est définie par la seconde partie de la définition sous forme de collection lignes pointillées répétées avec une fréquence ($freq_i$). Chaque ligne de la collection est décrite par sa direction (dir_i), son décalage par rapport à l'origine ($offsetx_i$, $offsety_i$) et la définition de ligne pointillée qui contient des segments et des espaces qui se suivent avec une longueur définie (len_{ij}).

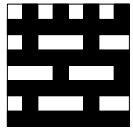
Remarque : Il n'est possible de définir en GDL que des hachures simples, et non pas des hachures symbole.

Exemple :

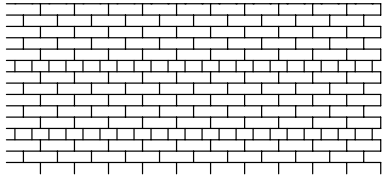
```
DEFINE FILL "brick"          85,    255,   136,   255,
                             34,    255,   136,   255,
       0.08333,      0.0,          4,
       1.0,          0.0,   0.0,   0.0,       0,
       3.0,          90.0,  0.0,   0.0,       2,
       1.0,          1.0,
       3.0,          90.0,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

Motif bitmap :

Motif :	Valeur binaire :	Vue :
pat1 = 85	01010101	• • • •
pat2 = 255	11111111	••••••••
pat3 = 136	10001000	• •
pat4 = 255	11111111	••••••••
pat5 = 34	00100010	• •
pat6 = 255	11111111	••••••••
pat7 = 136	10001000	• •
pat8 = 255	11111111	••••••••

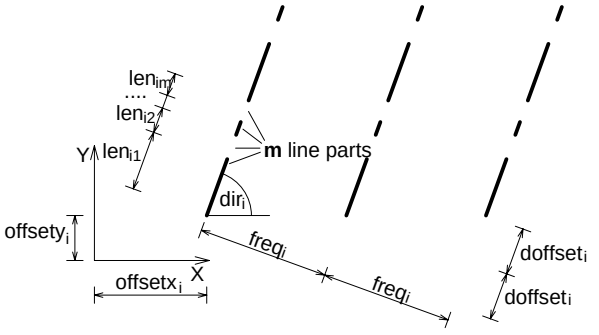


Hachure vectorielle :

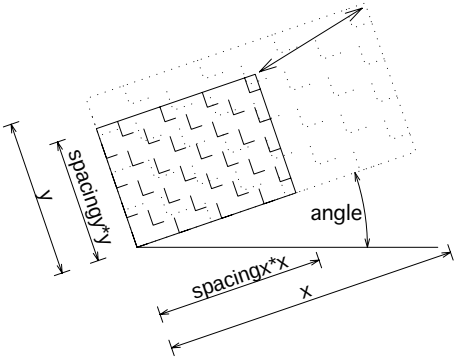


Définition de hachure avancée

DEFINE FILLA name pat_1, pat_2, pat_3, pat_4, pat_5, pat_6, pat_7, pat_8,
spacingx, spacingy, angle, n,
$freq_1$, $doffset_1$, dir_1, $offsetx_1$, $offsety_1$, m_1,
len_{11}, . . . len_{m1},
...
$freq_n$, $doffset_n$,dir_n, $offsetx_n$, $offsety_n$, m_n,
len_{n1}, . . . len_{nm}



Extension de l'instruction **DEFINE FILL.** Paramètres additionnels :



spacingx : facteur d'espacement en x

spacingy : facteur d'espacement en y

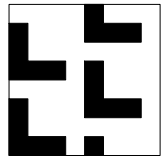
Les deux paramètres définissent un facteur d'échelle global pour la hachure entière. Toutes les valeurs en direction x seront multipliées par spacingx et toutes les valeurs en direction y seront multipliées par spacingy.

doffset$_i$: le décalage du commencement de la ligne de hachurage similaire suivante, mesuré le long de la direction de la ligne. Chaque ligne de la série sera dessinée à la distance définie par freq$_i$ avec un décalage de doffset$_i$. La longueur véritable du décalage sera l'espacement multiplié par doffset$_i$.

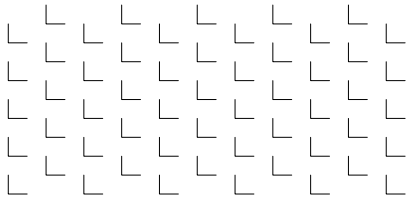
Exemple :

```
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
```

Motif bitmap :

Motif :	Valeur binaire :		Vue :
pat1 = 8	00001000	.	
pat2 = 142	10001110	• •••	
pat3 = 128	10000000	•	
pat4 = 232	11101000	••• •	
pat5 = 8	00001000	.	
pat6 = 142	10001110	• •••	
pat7 = 128	10000000	•	
pat8 = 232	11101000	••• •	

Hachure vectorielle :



Définition de type de ligne

DEFINE LINE_TYPE name spacing, n, len_1, . . . len_n

Les scripts **GDL** peuvent inclure des définitions de type de ligne préalables à la première référence à ce nom de type de ligne. Le type de ligne ainsi défini peut être utilisé seulement dans le script où il a été défini et dans ses scripts de seconde génération.

Les scripts **GDL** peuvent inclure des définitions de type de ligne préalables à la première référence à ce nom de type de ligne. Le type de ligne ainsi défini peut être utilisé seulement dans le script où il a été défini et dans ses scripts de seconde génération.

name : nom du type de ligne.

spacing : espacement

n : nombre des parties de la ligne

len_i : longueur des parties de la ligne (longueur réelle égale à spacing * len_i). Les parties de ligne sont des segments et des espaces qui se suivent. La première partie de ligne est un segment. La longueur zéro représente un point.

Remarque : Seuls les types de ligne simples (pointillés) peuvent être définis, et non pas les lignes symbole.

Exemple :

```
DEFINE LINE_TYPE "line - - ." 1,
 6, 0.005, 0.002, 0.001, 0.002, 0.0, 0.002
```

Définition de style

```
DEFINE STYLE  name font_family, size, anchor, facecode
DEFINE STYLE  name PLOTMAKER, size, anchor, slant
DEFINE STYLE  name PLOTTER, size, anchor, slant
```

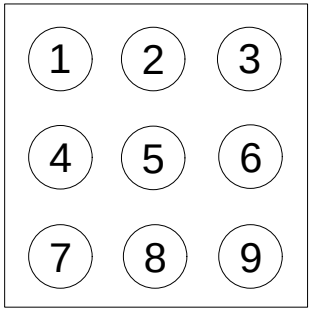
Les scripts **GDL** peuvent inclure des définitions de style préalables à la première référence à ce nom de style. Le style ainsi défini peut être utilisé seulement dans le script où il a été défini et dans ses scripts de seconde génération.

name : nom du style

font_family : nom de la police utilisée (par exemple, Geneva)

size : hauteur en mm des caractères.

anchor : code du point d'ancrage dans le texte



facecode : combinaison des valeurs suivantes:

0	normal
1	**gras**
2	*italique*
4	souligné
8	contour
16	ombre

slant : inclinaison en degrés (pour les polices PLOTMAKER et PLOTTER seulement)

Chapitre 9
Scripts non géométriques

En plus des fenêtres de script 2D et 3D qui définissent l'apparence de l'Objet GDL, d'autres scripts sont disponibles pour y ajouter des informations complémentaires. Ce sont le script de Descriptif utilisé dans les calculs quantitatifs, le script de Paramètres contenant toutes les valeurs possibles de différents paramètres et le script d'Interface utilisateur pour personnaliser la saisie des paramètres. Les commandes disponibles dans ces scripts sont détaillées sur les pages suivantes.

9.1 Le script de Descriptif

Les éléments de bibliothèque possèdent une fenêtre GDL réservée au script de Descriptif. Ce script permet de paramétrer les propriétés des éléments de bibliothèque et de définir par une directive leur place dans la liste de composants. Avec quelques commandes, il est possible de définir dans le script les descriptifs, les composants et les caractéristiques qui étaient présents dans les fenêtres de Descriptifs des versions précédentes d'ArchiCAD. Il est également possible de faire référence à des descriptifs et à des composants dans des bases de données externes. La longueur des codes ne peut dépasser les 32 caractères.

Toutes les autres commandes GDL qui ne génèrent pas une forme peuvent être utilisées dans le script de Descriptif.

DATABASE_SET `set_name [, desc_name, comp_name, unit_name, key_name, crit_name, listset_name]`

Définition ou sélection de l'ensemble de base de données. Si cette commande est placée dans un script MASTER_GDL, elle définit un ensemble de base de données contenant des fichiers Descriptif, Composant, Unité, Clé, Critères et Liste.

Le nom de cet ensemble de base de données peut alors être référencé à partir des scripts de Propriétés à l'aide de la même commande avec le paramètre set_name comme seule directive. Pour cela, sélectionnez l'ensemble de base de données auquel se rapporte les REF Composant et Descriptif. Par défaut, le nom de l'ensemble de base de données est "Default Set". Il sera utilisé si aucun autre ensemble est sélectionné. Par défaut, les noms des fichiers d'ensemble de base de données sont : DESCDATA, COMPDATA, COMPUNIT, LISTKEY, LISTCRIT, LISTSET.

Les scripts peuvent comprendre un nombre indéfini de sélections DATABASE_SET.

set_name :	le nom de l'ensemble de base de données
desc_name :	le nom du fichier de données Descriptif
comp_name :	le nom du fichier de données Composant
unit_name :	le nom du fichier de données Unité
key_name :	le nom du fichier de données Clé
crit_name :	le nom du fichier Critères
listset_name :	le nom du fichier Liste

DESCRIPTOR `name [,code, keycode]`

Définition de descriptif local. Le script peut comprendre un nombre illimité de `DESCRIPTOR`s.

name : La chaîne peut s'étendre sur plusieurs lignes. Les nouvelles lignes sont définies par le caractère '\n' et les tabulations par '\t'. Ajouter '\' à la fin d'une ligne permet de continuer la chaîne dans la ligne suivante sans ajouter une nouvelle ligne. A l'intérieur de la chaîne, si le caractère '\' est doublé (\\), il perd sa fonction de contrôle est signifie simplement '\'. La longueur de la chaîne ne peut excéder les 255 caractères (caractères de ligne nouvelle inclus): les caractères en trop seront simplement coupés par le compilateur. Si vous avez besoin d'un texte plus long, utilisez plusieurs `DESCRIPTOR`s.

code : chaîne, définit un code pour le decriptif

keycode : chaîne, fait référence à un clef dans une base de données externe. La clef sera assignée au descriptif.

REF DESCRIPTOR `code [, keycode]`

Référence par code et chaîne de code-clef à un descriptif se trouvant dans une base de données externe.

COMPONENT `name, quantity, unit [, prop_with, code, keycode, unitcode]`

Définition de composant local. Le script peut comprendre un nombre illimité de `COMPONENT`s.

name : le nom du composant (max. 128 caractères)

quantity : quantité, une expression numérique

unit : la chaîne utilisée pour la description de l'unité

prop_with : un code entre 1 et 6. En créant une liste, la quantité de composant défini au-dessus sera automatiquement multipliée par une valeur calculée pour l'élément listé courant

1: pièce
2: longueur
3: surface A
4: surface B
5: surface
6: volume

code : chaîne, définit un code pour le composant

keycode : chaîne, fait référence à une clef dans une base de données externe
La clef sera assignée au composant.

unitcode : chaîne, référence à une unité dans une base de données externe qui régit le format de sortie de la quantité du composant. Remplace la chaîne d'unité définie localement.

REF COMPONENT `code [, keycode [, num_expr]]`

Référence par code et chaîne de code-clef à un composant se trouvant dans une base de données externe. La valeur servant à la multiplication dans la base de données des composants peut être remplacée par l'expression numérique définie ici.

BINARYPROP

Référence aux données de propriétés binaires (composants et descriptifs) définies dans la section Composant/Descriptifs de l'élément de bibliothèque.

Les directives DATABASE_SET n'ont aucune influence sur les données binaires.

SURFACE3D ( )
VOLUME3D ( )

Ces fonctions retournent la surface et le volume de la forme 3D de l'élément de bibliothèque.

> **Attention**: Si vous placez plusieurs formes au même endroit avec les même paramètres, ces fonctions vous fourniront la somme totale de la surface et du volume de toutes ces formes.

POSITION position_keyword

N'a d'effet que dans la Liste de composants.

Modifie le type seulement de l'élément auquel les descriptifs et composants suivants sont associés. S'il n'y a pas de telle directive dans le script de Descriptif, les descriptifs et les composants seront listés avec leurs type d'élément par défaut. Les mots-clef sont les suivants:

WALLS
COLUMNS
BEAMS
DOORS
WINDOWS
OBJECTS
CEILS
PITCHED_ROOFS
LIGHTS
HATCHES
ROOMS
MESHES

La directive reste effective pour tous les DESCRIPTORs et COMPONENTs jusqu'à la défition d'une autre directive. Le script peut comprendre un nombre illimité de directives.

Exemple:

```
DESCRIPTOR   "\tPainted box.\n\t Properties:\n\
 \t\t - swinging doors\n\
 \t\t - adjustable height\n\
 \t\t - scratchproof"
REF DESCRIPTOR   "0001"
s = SURFACE3D ()     ! wardrobe surface
COMPONENT  "glue", 1.5, "kg"
COMPONENT  "handle", 2 * c, "nb"  ! c number of
doors
COMPONENT  "paint", 0.5 * s, "kg"
POSITION   WALLS
REF COMPONENT   "0002"
```

DRAWING

Fait référence au dessin décrit dans le script 2D du même élément de bibliothèque. Vous pouvez l'utiliser pour placer des dessins dans le Métré.

9.2 Le script de Paramètres

Les listes de valeurs sont des jeux de valeurs numériques ou de chaîne possible. Elles peuvent être appliquées aux paramètres définis dans le script de Paramètres pour les valeurs de l'élément de bibliothèque ou dans le script MASTER_GDL. Le type de paramètre doit être une liste de valeur de type simple. La compatibilité du type est vérifiée par le compilateur.

Le script de Paramètres est interprété chaque fois qu'une valeur de paramètre de type liste de valeurs doit être modifiée, et les valeurs possibles définies dans le script apparaîtront dans un pop-up menu. La commande de définition de liste est:

VALUES `"name" val_def`$_1$ `[, val_def`$_2$`, ..., val_def`$_n$`]`

name : le nom du paramètre

val_def_i : la définition de la valeur, soit

val_i : l'expression numérique ou alphanumérique, soit

CUSTOM : mot clé, signifiant qu'il est possible d'entre n'importe quelle valeur personnalisée, soit

RANGE `left_delimiter [exp`$_1$`], [exp`$_2$`] right_delimiter [`**`STEP`** `stepbegval, stepval] :`
la définition de l'intervalle, avec un pas optionnel

left_delimiter : **[** ou **(**, '[' soit '>=', '(', soit '>'

exp_1 : l'expression de la limite inférieure

exp_2 : l'expression de la limite supérieure

right_delimiter : **]** ou **)**, ']' soit '<=', ')' soit '<'

stepbegval : la valeur de départ

stepval : la valeur du pas

Exemples :

```
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
```

PARAMETERS `name`$_1$ `= val`$_1$ `[, name`$_2$ `= val`$_2$`, ..., name`$_n$ `= val`$_n$`]`

name$_i$: le nom du paramètre

val$_i$: la nouvelle valeur du paramètre

Les valeurs de paramètres d'un élément de bibliothèque peuvent être modifiées par le script de paramètres à l'aide de cette commande.

La modification ne sera effective que pour l'interprétation suivante. Les commandes figurant dans les macros font référence aux paramètres du processus appelant. Si le paramètre est une liste de valeurs, la valeur choisie peut être une valeur existante, la valeur personnalisée ou la première valeur de la liste de valeurs.

Par ailleurs, la variable alphanumérique globale `GLOB_MODPAR_NAME` contient le nom du dernier paramètre modifié par l'utilisateur.

LOCK `name`$_1$ `[, name`$_2$`, ..., name`$_n$`]`

Verrouille le paramètre nommé dans la boîte de dialogue de configuration. Les paramètres verrouillés apparaissent en grisé dans la boîte de dialogue. Leur valeur ne peut être modifiée par l'utilisateur.

9.3 Le script d'Interface utilisateur

A l'aide de quelques commandes **GDL**, vous pouvez définir un nouvel onglet dans le dialogue d'options des éléments de bibliothèque pour personnaliser l'interface des paramètres.

Si vous cliquez sur le bouton "Définir par défaut" dans l'éditeur d'élément de bibliothèque, l'interface personnalisée sera utilisée par défaut dans la configuration de l'élément.

Les paramètres personnalisés ne sont pas automatiquement masqués dans la liste originale. Ils peuvent être masqués manuellement dans l'éditeur d'élément de bibliothèque.

L'origine du système de cordonnées est située dans le coin supérieur gauche. Les tailles et la valeur des coordonnées sont mesurées en pixels.

UI_DIALOG `title [, sizex, sizey]`

Définit le titre de la boîte de dialogue. Dans la version actuelle de ArchiCAD, la taille de la zone disponible est fixée à 310 x 266 pixels. Les paramètres sizex et sizey ne sont pas utilisés.

Limitation : Le script d'interface ne peut comprendre qu'une seule commande `UI_DIALOG`.

UI_PAGE `pagenum`

Directive de page, définit la page sur laquelle les éléments d'interface sont placés. La numérotation des pages commence à " 1 ". Le basculement entre les pages peut être défini en créant deux boutons à l'aide des commandes `UI_NEXT` et `UI_PREV`.

En l'absence d'une commande `UI_PAGE` dans le script d'interface, chaque élément sera placé sur la première page par défaut.

Avertissement : Toute rupture dans la continuité de la liste des pages provoque l'insertion d'une nouvelle page sans boutons, ce qui exclut toute possibilité de passer à une autre page.

UI_BUTTON `type, text, x, y, width, height`

Définition des boutons de la page. Les boutons peuvent être utilisés pour passer d'une page à l'autre. Ils ne sont pas générés automatiquement. Ils doivent donc être définis pour chaque page.

type : **UI_PREV** : si ce bouton est activé, la page précédente est affichée
UI_NEXT : si ce bouton est activé, la page suivante est affichée

text : le texte qui doit figurer sur le bouton

x, y : la position du bouton

width, height : la largeur et la hauteur exprimées en pixels

UI_GROUPBOX `text, x, y, width, height`

Un cadre de groupement est un séparateur rectangulaire. Il peut être utilisé pour grouper visuellement des paramètres associés.

text : le titre du cadre

x, y : la position du coin supérieur gauche

width, height : la largeur et la hauteur exprimées en pixels

UI_SEPARATOR x_1, y_1, x_2, y_2

Génère une ligne de séparation. Ce séparateur peut uniquement être vertical ($x_1 = x_2$) ou horizontal ($y_1 = y_2$).

x_1, y_1 : les coordonnées du point de départ

x_2, y_2 : les coordonnées du point d'extrémité

UI_PICT `expression, x, y [, width, height]`

Elément d'image dans la boîte de dialogue. Le fichier d'image doit figurer dans l'une des bibliothèques chargées.

expression : le nom du fichier ou le numéro d'index de l'image enregistrée dans l'élément de bibliothèque. L'index 0 fait référence à l'image d'aperçu de l'élément de bibliothèque.

x, y : la position du coin supérieur gauche de l'image

width, height : la hauteur et la largeur optionnelles exprimées en pixels ; la hauteur et la largeur originales de l'image seront utilisées par défaut.

UI_STYLE `fontsize, facecode`

Tous les `UI_OUTFIELD` et `UI_INFIELD` générés selon ce mot clé représentent ce style jusqu'à la modification de l'`UI_STYLE`.

fontsize : l'une des valeurs suivantes :
0 petit
1 très petit
2 grand

facecode : Similaire à la définition `STYLE`, mais les valeurs ne peuvent pas être combinées.

UI_OUTFIELD `text, x, y, width, height`

Génère un texte statique.

text : le texte concret

x, y : la position du coin supérieur gauche du bloc de texte

width, height : la largeur et la hauteur du texte exprimées en pixels

UI_INFIELD
```
"name", x, y, width, height [,
versionFlag, pictName, nrImages, nrRows,
cellX, cellY, imageX, imageY,
imageExp1, text1,
...,
imageExpn, textn]
```

Génère un champ de texte éditable ou un menu déroulant pour l'entrée du paramètre. Un menu déroulant est généré si le type de paramètre est une liste de valeur, une matière, une hachure, un type de ligne ou une couleur de stylo.

Si les paramètres optionnels de la commande sont présents, les listes de valeurs peuvent être affichées en alternance comme des champs miniatures de visualisation. Ces champs affichent les images spécifiées et les textes associés et ils sont munis de barres de défilement sur leurs côtés. Ils vous permettent de sélectionner un seul élément à la fois, à l'instar des menus déroulants.

Le script d'interface est rétabli avec la nouvelle valeur après la modification d'un quelconque paramètre.

name : le nom du paramètre

x, y : la position du texte, du menu déroulant ou du contrôle

width, height : la largeur et la hauteur exprimées en pixels

versionFlag : réservé, toujours " 1 "

pictName : le nom du fichier d'image contenant une matrice des images concaténées ou une chaîne vide

nrImages : le nombre d'images dans la matrice

nrRows : le nombre de lignes de la matrice

cellX, cellY : la largeur et la hauteur d'une cellule dans le champ miniature de visualisation, y compris l'image et le texte

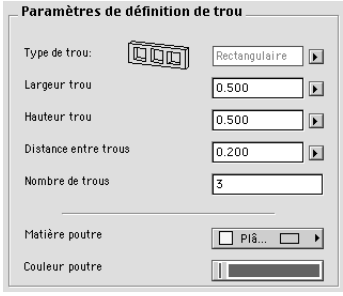
imageX, imageY :
la largeur et la hauteur de l'image dans la cellule

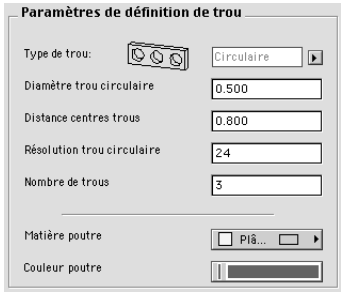
imageExpi : l'index de l'image i-th dans la matrice ou nom du fichier de l'image individuelle

texti : le texte figurant dans la cellule no. i

Exemple :

```
IF c THEN
   UI_DIALOG "Paramètres de définition de trou"
   UI_OUTFIELD "Type de trou:",15,40,180,20
   UI_INFIELD "D",190,40,105,20
   IF D="Rectangulaire" THEN
     UI_PICT "rect.pict",110,33,60,30
     UI_OUTFIELD "Largeur trou",15,70,180,20
     UI_INFIELD "E", 190,70,105,20
     UI_OUTFIELD "Hauteur trou",15,100,180,20
     UI_INFIELD "F", 190,100,105,20
     UI_OUTFIELD "Distance entre trous",15,130,180,20
     UI_INFIELD "G", 190,130,105,20
   ELSE
     UI_PICT "circle.pict",110,33,60,30
     UI_OUTFIELD "Diamètre trou circulaire",15,70,180,20
     UI_INFIELD "J", 190,70,105,20
     UI_OUTFIELD "Distance centres trous",15,100,180,20
     UI_INFIELD "K", 190,100,105,20
     UI_OUTFIELD "Résolution trou circulaire",15,130,180,20
      UI_INFIELD "M", 190,130,105,20
   ENDIF
   UI_OUTFIELD "Nombre de trous",15,160,180,20
     UI_INFIELD "I", 190,160,105,20
ENDIF
UI_SEPARATOR 50,195,250,195
UI_OUTFIELD "Matière poutre", 15,210,180,20
UI_INFIELD "MAT", 190,210,105,20
UI_OUTFIELD "Couleur poutre", 15,240,180,20
UI_INFIELD "P", 190,240,105,20
```





Chapitre 10
Expressions et fonctions

Tous les paramètres mathématiques des formes GDL peuvent être le résultat de calculs. Ainsi, vous pouvez définir que la hauteur du cylindre soit cinq fois le rayon du cylindre ou, avant de définir un cube, déplacer le système de coordonnées dans chaque direction de la moitié de la taille du cube, de façon à avoir l'origine originale au centre du cube plutôt qu'à son angle inférieur gauche.

Pour définir ces calculs, GDL offre un grand nombre d'outils mathématiques: expressions, opérateurs et fonctions.

10.1 Expressions

Vous pouvez écrire des expressions composées avec des instructions **GDL**. Les expressions peuvent être de type numérique ou chaînes de caractères. Elle peuvent être composées de constantes, de variables et d'appels de fonction connectés par des opérateurs. Les parenthèses (( )) (precedence 1) (priorité 1) sont utilisées pour transgresser la priorité par défaut des opérateurs.

Les variables de type *simple* peuvent être de simples valeurs numériques ou de texte, à l'intérieur d'un même script, et peuvent être utilisées dans les expressions de type numérique et chaîne de caractères respectivement. Les opérations ayant pour résultat un texte ne peuvent être utilisées directement comme noms de macro dans les appels de macro ou comme noms d'attribut dans les définitions de matière, de hachure, de ligne ou de style. Les variables texte seront traitées comme telles et peuvent être utilisées chaque fois qu'un texte est requis. Si, par la suite, la même variable se voit assigner dans le même script une valeur numérique, elle ne pourra être utilisée que dans les expressions numériques jusqu'à ce qu'une valeur de chaîne soit définie pour elle de nouveau. Le type des expressions est vérifié pendant la pré-compilation.

GDL supporte les matrices d'une ou de deux dimensions. Les variables deviennent une *matrice* après une instruction de déclaration spécifiant leurs dimensions maximum:

DIM *var1* **[**dim_1**]**, *var2* **[**dim_1**][**dim_2**]**, ...

Après le mot-clef `DIM`, il peut y avoir un nombre quelconque de noms de variables séparés par des virgules. *var1* et *var2* sont les noms de matrice, les chiffres entre guillemets représentent les dimensions de la matrice (constantes numériques). On ne peut pas utiliser des expressions de variable comme dimensions.

Les paramètres des éléments de bibliothèque peuvent être des matrices. Leurs dimensions sont spécifiées dans le dialogue de l'élément de bibliothèque. Il n'est pas nécessaire de déclarer les matrices de paramètre dans le script. En faisant référence à un élément de bibliothèque en utilisant une instruction `CALL`, le paramètre doit être une matrice avec les mêmes dimensions.

Les éléments de ces matrices peuvent être référencées n'importe où dans le script, mais les variables après la déclaration seulement:

var1 **[**num_expr**]** or *var1*

var2 **[**num_expr1**][**num_expr2**]** or *var2*

Mettre un nom de matrice sans les valeurs actuelles d'index veut dire que vous faites référence à la matrice entière, ce qui est accepté dans certains cas (instructions `CALL, PRINT, LET, PUT, REQUEST, INPUT, OUTPUT`).
On peut utiliser des éléments de matrice dans n'importe quelle expression numérique ou de chaîne. Ils peuvent avoir des valeurs numériques ou de chaîne. L'indexation commence à 1, et toute expression numérique peut être utilisé comme index.
Il n'est pas possible d'utiliser le nom d'une matrice plus loin dans le même script comme nom de variable script.
Si la valeur de l'index courant est supérieure à la dimension déclarée du paramètre, un message d'erreur est affiché.

Exemples d'expressions numériques:

```
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
```

Exemples d'expressions de chaîne de caractères:

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

Exemples d'expressions utilisant des valeurs de matrice:

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

10.2 Opérateurs

Dans la liste ci-dessous, les opérateurs sont indiqués en ordre décroissant de priorité. L'évaluation d'une expression commence avec l'opérateur de plus haute priorité et de gauche à droite.

Opérateurs arithmétiques

^ (ou **)	Puissance	priorité 2
*	Multiplication	priorité 3
/	Division	priorité 3
MOD (or **%**)	Modulo (reste)	priorité 3
	X MOD Y = X - Y * INT (X/Y)	
+	Addition	priorité 4
-	Soustraction	priorité 4

Remarque: + (addition) peut être utilisée pour les expressions de type caractère: le résultat est la concaténation des chaînes de caractères.

Opérateurs relationnels

=	Egal à	priorité 5
<	Inférieur à	priorité 5
>	Supérieur à	priorité 5
<=	Inférieur ou égal à	priorité 5
>=	Supérieur ou égal à	priorité 5
<> (or #)	Différent de	priorité 5

Les opérateurs relationnels peuvent être utilisés avec deux expressions de type caractère. Le résultat est 1 ou 0 numérique. Cette fonction fait la différence entre majuscules et minuscules.

Opérateurs booléens

AND (ou **&**)	Et logique	priorité 6
OR (ou |)	Ou inclusif logique	priorité 7
EXOR (ou **@**)	Ou exclusif logique	priorité 8

GDL utilise uniquement les nombres à virgule flottante, tandis que les opérateurs booléens utilisent des nombres réels. Cela veut dire que 0.0 vaut “faux”, alors que toute autre valeur vaut “vrai”. La valeur d’une expression logique est aussi un réel, 1.0 pour “vrai” et 0.0 pour “faux”.

10.3 Fonctions

Fonctions arithmétiques

ABS (x) Retourne la valeur absolue de x.

CEIL (x) Retourne la plus petite valeur intégrale qui n'est pas inférieure à x.

Exemple: `CEIL`(1.23) = 2; `CEIL` (-1.9) = -1

INT (x) Retourne la partie entière de x.
(par ex. INT(1.23) = 1, INT(-1.23) = -2).

FRA (x) Retourne la partie fractionnaire de x.
(par ex. FRA(1.23) = 0.23, FRA(-1.23) = 0.77).

SGN (x) Retourne +1.0 si x est positif, -1.0 si x est négatif, sinon 0.0.

SQR (x) Retourne la racine carrée de x.

Fonctions circulaires

Ces fonctions utilisent les degrés en arguments (`COS, SIN, TAN`) et en valeurs de retour (`ACS, ASN, ATN`).

ACS (x) Retourne l'arc cosinus de x.
($-1.0 \leq x \leq 1.0$; $0° \leq ACS(x) \leq 180°$).

ASN (x) Retourne l'arc sinus de x.
($-1.0 \leq x \leq 1.0$; $-90° \leq ASN(x) \leq 90°$).

ATN (x) Retourne l'arc tangente de x.
($-90° \leq ATN(x) \leq 90°$).

COS (x) Retourne le cosinus de x.

SIN (x) Retourne le sinus de x.

TAN (x) Retourne la tangente de x.

PI Retourne la constante de Ludolph. (= 3.1415926).

Fonctions transcendentales

EXP (x) Retourne la x ^{e} puissance de e. (e = 2.7182818).

LGT (x) Retourne le logaritme base 10 de x.

LOG (x) Retourne le logarithme naturel de x.

Fonction booléenne

NOT (x) Retourne faux (=0.0) si x est vrai (0.0) et vrai (=1.0) si x est faux (=0.0). (Négation logique).

Fonctions statistiques

MIN (x_1,x_2, . . . x_n) Retourne le plus petit des arguments. Le nombre des arguments n'est pas fixe.

MAX (x_1,x_2, . . . x_n) Retourne le plus grand des arguments. Le nombre des arguments n'est pas fixe.

RND (x) Retourne une valeur aléatoire entre 0.0 et x (x > 0.0).

Fonctions de chaîne

STR `(numeric_expression, len, frac)`
STR `(formatstring, numeric_expression)`

La première forme de la fonction crée une chaîne à partir de la valeur courante de l'expression numérique. Le nombre minimum des caractères numériques de la chaîne est len, tandis que frac représente les nombres placés après la virgule flottante.

Exemple:

```
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
```

Dans le second cas, formatstring peut être soit une variable, soit une constante. Si le format est vide, il est interprété en mètres, avec une précision de trois décimales (0 entier affiché)..

Le formatstring est comme suit:

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

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

none	la justification à droite (par défaut)
-	la justification à gauche
+	le signe + explicite
space	espace au lieu d'un signe +
'*' 0	arrondi désactivé (par défaut)
'*' 1	arrondi .5
'*' 2	arrondi .25
'*' 3	arrondi .1
'*' 4	arrondi .01

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

'#'	ne pas afficher entier zéro

flags (pour ffi, fdi, fi):

'0'	afficher 0 pouce

field_width**:** entier décimal non signé

le nombre minimum de caractères à générer

precision: entier décimal non signé

le nombre de places fractionnaires à générer

conv_spec (spécification de conversion):

e	- format exponentiel (mètre)
m	- mètre
mm	- millimètre
cm	- centimètre
ffi	- pieds et pouces fractionnaires
fdi	- pieds et pouces décimaux
df	- pieds décimaux
fi	- pouces fractionnaires
di	- pouces décimaux

pour surfaces:

sqm	- mètres carrés
sqcm	- centimètres carrés
sqmm	- millimètres carrés
sqf	- pieds carrés
sqi	- pouces carrés

pour angles :

dd	- degrés décimaux
dms	- degrés, minutes, secondes
gr	- grads
rad	- radians
surv	- unités

pour volumes :

cum	mètre cube
l	litre
cucm	centimètre cube
cumm	millimètre cube
cuf	pied cube
cui	pouce cube
cuy	yard cube
gal	gallon

Exemples:

```
h = 23
nr = 0.345678

TEXT2  0, h, STR ("%m", nr)             !0.346
TEXT2  0, h-1, STR ("%#10.2m", nr)      !        35
TEXT2  0, h-2, STR ("%.4cm", nr)        ! 34.5678
TEXT2  0, h-3, STR ("%12.4cm", nr)      !    34.5678
TEXT2  0, h-4, STR ("%.6mm", nr)        !345.678000
TEXT2  0, h-5, STR ("%+15e", nr)       !+3.456780e-01
TEXT2  0, h-6, STR ("%ffi", nr)         !1'-2"
TEXT2  0, h-7, STR ("%0.16ffi", nr)     !1'-1 5/8"
TEXT2  0, h-8, STR ("% .3fdi", nr)      ! 1'-1.609"
TEXT2  0, h-9, STR ("% -10.4df", nr)    ! 1.1341'
TEXT2  0, h-10, STR ("%0.64fi", nr)     !13 39/64"
TEXT2  0, h-11, STR ("%+12.4di", nr)    !  +13.6094"
TEXT2  0, h-12, STR ("%#.3sqm", nr)     ! 346
TEXT2  0, h-13, STR ("%+sqcm", nr)      !+3,456.78
TEXT2  0, h-14, STR ("% .2sqmm", nr)    ! 345,678.00
TEXT2  0, h-15, STR ("%-12sqf", nr)     !3.72
TEXT2  0, h-16, STR ("%10sqi", nr)      !     535.80

alpha = 88.657

TEXT2  0, h-17, STR ("%+10.3dd", alpha) !  +88.657°
TEXT2  0, h-18, STR ("%.1dms", alpha)   !88°39'
TEXT2  0, h-19, STR ("%.2dms", alpha)   !88°39'25"
TEXT2  0, h-20, STR ("%10.4gr", alpha)  !  98.5078G
TEXT2  0, h-21, STR ("%rad", alpha)     !1.55R
TEXT2  0, h-22, STR ("%.2surv", alpha)  !N 1°20'35" E
```

SPLIT (string, format, var_1 [, var_2, ..., var_n])

Divise le paramètre chaîne conformément au format en une ou plusieurs parties numériques ou chaîne. Le procédure de division s'arrête en rencontrant la première partie qui ne soit pas conforme. Retourne le nombre de valeurs lues.

string : la chaîne à diviser

format : une combinaison quelconque de chaînes constantes, de %s et de %n -s. Les parties de la chaîne doivent être contendues dans les chaînes constantes, %s dénote une valeur de chaîne délimitée par des espaces ou des tabulations, %n dénote une valeur numérique.

var_i : les noms de variables où stocker les parties de chaînes divisées

Exemple:

```
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 (string_expression)

Retourne la largeur de la chaîne, affichée dans le style actuel. L'épaisseur en mètres est de STW (string_expression)/1000 * A_.

Exemple:


```
DEFINE STYLE "own" "Monaco", 180000 / A_, 0, 0
SET STYLE "own"
string = "abcd"
width = STW (string) / 1000 * A_
REQUEST ("Height_of_style", "own", height)
height = height / 1000 * A_
text2 0,0, string
rect2 0,0, width, -height
```

STRLEN (string_exp)

Retourne la longueur de la chaîne (le nombre de caractères)

STRSTR `(string_exp1, string_exp2)`

Retourne la position de la première apparition de la seconde chaîne dans la première chaîne. Si la première chaîne ne contient pas la seconde, la fonction retourne à 0.

STRSUB `(string_exp, begpos, numchars)`

Retourne une partie de chaîne du paramètre chaîne de caractères commençant à la position identifiée par le paramètre begpos, sa longueur est de numchars caractères.

Exemple:

```
ss = ""
REQUEST ("Linear_dimension", "", ss)
unit = ""
IF STRSTR (ss, "m")  > 0 THEN unit = "m"
IF STRSTR (ss, "mm") > 0 THEN unit = "mm"
IF STRSTR (ss, "cm") > 0 THEN unit = "cm"
TEXT2  0, 0, STR (ss, a) + " " + unit     ! 1.00 m
string = "Flowers.PICT"
len = STRLEN (string)
n = STRSTR (string, ".")
TEXT2  0, -1, STRSUB (string, 1, n - 1)   ! Flowers
TEXT2  0, -2, STRSUB (string, len - 4, 5) ! .PICT
```

Fonctions spéciales

En plus des variables globales, ce sont les fonctions spéciales qui servent dans le script à la communication avec ArchiCAD. Soit elles s'informent de l'état courant et des réglages de préférence du programme, soit elles font référence à l'environnement courant de l'élément de bibliothèque. Les appels de requête peuvent également servir à la communication avec les extensions **GDL**.

Il existe deux types de fonctions spéciales: les requêtes et la fonction `IND`:

REQ (parameter_string)

REQUEST (question_name, name | index, var1 [, var2,....])

IND (MATERIAL, name_string)
IND (FILL, name_string)
IND (LINE_TYPE, name_string)
IND (STYLE, name_string)

Pour détails, voir Annexe: Fonctions spéciales.

Chapitre 11
Instructions de contrôle

Ce chapitre passe en revue les commandes GDL disponibles pour le contrôle de boucles et de branchements dans le script et introduit le concept de la manipulation du tampon de paramètres permettant de stocker des valeurs de paramètres pour usage ultérieur. Il explique également comment utiliser des objets comme appels de macro et comment afficher les expressions calculées.

GDL peut également traiter des opérations sur des fichiers externes au moyen d'applications spéciales appelées Extensions. Les commandes utilisées sont présentées dans ce chapitre, et un exemple est donné en Annexe.

11.1 Instructions de contrôle de flux

FOR `varnam = initial_value TO end_value [ STEP step_value ]`

Première instruction d'une boucle FOR. Si le mot-clé STEP et la valeur_pas manquent, par défaut la valeur est égale à 1.

Les variables globales ne peuvent servir de variables de contrôle de boucle.

Exemple:

```
FOR I=1 TO 10 STEP 2
      PRINT I
NEXT I
```

NEXT `varnam`

Dernière instruction d'une boucle FOR.

La variable d'itération varie depuis la valeur_initiale jusqu'à la valeur_finale par incrément (ou décrément) de la valeur_pas à chaque exécution du corps de la boucle (instructions entre FOR et NEXT). Si la variable d'itération dépasse la valeur de valeur_finale, le programme exécute l'instruction à la suite de NEXT.

Les deux fragments de programme ci-dessous sont équivalents:

```
! 1st
  A = B
1:IF C > 0 AND A > D OR C < 0 AND A < D THEN 2
  PRINT A
  A = A + C
  GOTO 1
2:

! 2nd
  FOR A = B TO D STEP C
        PRINT A
  NEXT A
```

L'exemple ci-dessus montre que step_value = 0 cause une boucle infinie.

Une seule instruction NEXT est admise après FOR. Il est permis de quitter la boucle par GOTO (ou IF ... GOTO) et de revenir après l'avoir quittée, mais il n'est pas permis d'entrer dans la boucle en sautant l'instruction FOR.

```
DO
    [stmt1
    stmt2
    ...
    stmtn]
WHILE condition
```

Les instructions (stmt) entre les mots-clefs sont exécutées tant que la condition est vraie.

La condition est vérifiée après chaque exécution de l'instruction.

```
WHILE condition DO
    [stmt1
    stmt2
    ...
    stmtn]
ENDWHILE
```

Les instructions (stmt) entre les mots-clefs sont exécutées tant que la condition est vraie.

La condition est vérifiée avant chaque exécution de l'instruction.

```
REPEAT
    [stmt1
    stmt2
    ...
    stmtn]
UNTIL condition
```

Les instructions (stmt) entre les mots-clefs sont exécutées jusqu'à ce que la conditon devienne vraie.

La condition est vérifiée après chaque exécution de l'instruction.

Exemple:

Les 4 séquences de commandes GDL suivantes sont équivalentes:

```
! 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 condition **THEN** label
IF condition **GOTO** label
IF condition **GOSUB** label

Saut conditionnel. Si la valeur de cond est 0, la commande n'a pas d'effet, sinon l'exécution continue au label.

Exemples:

```
IF  A THEN 28
IF  I > J GOTO 200+I*J
IF  I > 0 GOSUB 9000
```

```
IF  condition THEN statement [ELSE statement]
```

or

```
IF  condition THEN
    [stmt1
    stmt2
    ...
    stmtn]
[ELSE
    stmtn+1
    stmtn2
    ...
    stmtn+m]
ENDIF
```

Si vous mettez une seule commande après les mots-clefs `THEN` et/ou `ELSE` dans la même ligne, `ENDIF` n'est pas nécessaire. Une commande après `THEN` ou `ELSE` dans la même ligne signifie un `ENDIF` définitif.

S'il y a une nouvelle ligne après `THEN`, alors les commandes suivantes (placées chacun après un mot-clef `ELSE` ou `ENDIF`) ne seront exécutées qu'au cas où l'expression de la condition est vraie (différente de zéro)/ Autrement, ce sont les commandes suivant `ELSE` qui seront exécutées. Si le mot-clef `ELSE` est absent, les commandes après `ENDIF` seront exécutées.

Exemple:

```
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 `label`

Saut inconditionnel. Le programme exécute un branchement à l'instruction repérée par la valeur du label.

Exemple:

```
GOTO K+2
```

GOSUB `label`

Appel de procédure interne où le label est le point d'entrée de la procédure.

Voir labels (étiquettes) sous Eléments de syntaxe de base.

RETURN

Retour de la procédure interne.

END
EXIT

Fin du **GDL** script courant. Le programme se termine ou retourne au niveau supérieur. Il est possible d'utiliser plusieurs `END`s ou `EXIT`s dans un même fichier **GDL**.

BREAKPOINT `expression`

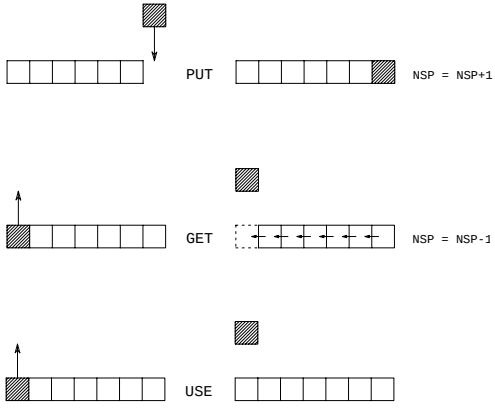
En utilisant cette commande, vous pouvez spécifier un point de rupture dans le script GDL.

Quand vous lancez le debugger, il s'arrêtera à la commande `BREAKPOINT` si la valeur de l'expression est vraie (1). En cas d'exécution "normale", l'interpréteur GDL passe cette commande sans aucune action.

11.2 Manipulation du tampon de paramètres

Le tampon de paramètres est une structure de données intégrée qui peut être utilisée si certaines valeurs (par ex. des coordonnées) changent selon une règle qui peut être décrite en utilisant une expression mathématique, si vous souhaitez, par exemple, stocker les valeurs courantes des variables.

Le tampon de paramètres, de longueur infinie, permet de stocker des valeurs en utilisant la commande PUT. Cette commande stocke les valeurs données à la fin du tampon. Les valeurs peuvent être utilisées ultérieurement (commandes GET, USE) dans le même ordre consécutif que celui dans lequel elles ont été définies. Une commande GET(n) ou USE(n) équivaut à n valeurs séparées par des virgules. Elle peut donc être utilisée dans une liste de paramètres GDL où un nombre n de valeurs sont requises.



PUT `expression [ , expression ] . . .`

Conserve les valeurs données dans l'ordre donné dans le tampon de paramètres interne.

GET `(n)`

Utilise les valeurs n suivantes du tampon de paramètres interne et les efface.

USE (n)

Utilise les valeurs n suivantes du tampon de paramètres interne sans les effacer. Des fonctions USE et GET consécutives peuvent utiliser la même séquence de paramètres.

NSP

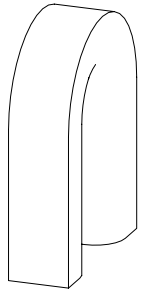
Retourne le nombre de paramètres conservés dans le tampon interne.

Exemple d'utilisation du tampon interne:

```
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
```



La description complète:

```
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
```

11.3 Objets macro

Bien que vos objets 3D puissent être toujours décomposés en éléments complexes ou primaires, parfois il est préférable de définir ces éléments complexes spécialement pour certains usages. Ces éléments personnalisés sont appelés MACROs.

CALL `macro_name_string [,parameter_list]`

CALL `macro_name_string` **`PARAMETERS`** `[name`$_1$`=value`$_1$`,… name`$_n$`=value`$_n$`]`

Les noms de macro ne doivent pas être plus longs que 31 caractères.

Les noms de macro peuvent être des constantes our des variables de type caractère. Vous ne pouvez pas utiliser des opérations de chaîne de caractère comme nom de macro. Attention! Si des variables ou paramètres de type chaîne de caractères sont utilisés comme nom de macro, la macro appelée ne sera pas incluse dans le projet archive, même si vous avez choisi l'option "Inclure tous les éléments de bibliothèques actives".

Le nom de la macro doit être mis entre guillemets (",',`,´,”,’,“,‘), à moins qu'il ne corresonde à la définition des identificateurs, c'est-à-dire qu'il commence par une lettre ou le caractère '_' ou '~' et consiste uniquement de lettres, de nombres et des caractères '_' et '~'. Sinon, les guillemets utilisés dans l’instruction d’appel doivent être les mêmes, au début et à la fin, et doivent être différents de tout autre caractère du nom de macro.

Un nom de macro lui-même peut aussi être utilisé comme une commande, sans le mot-clef `CALL`:

macro_name `[parameter_list]`

macro_name **`PARAMETERS`** `[name`$_1$`=value`$_1$`,… name`$_n$`=value`$_n$`]`

Le premier type d'appel de macro peut être utilisé avec de simples fichiers texte GDL ainsi qu'avec n'importe quel élément de bibliothèque, à condition que la liste des paramètres contienne uniquement des paramètres d'une seule lettre (A…Z). Cette forme de l'appel de macro peut être utilisée pour des raisons de compatibilité avec les versions précédentes, mais nous recommandons le second type. L'interprétation de la liste de paramètres est la suivante: la valeur du paramètre A sera la première valeur de la liste, la valeur du paramètre B sera la seconde valeur, et ainsi de suite. Si la macro (élément de

bibliothèque) ne contient pas de paramètre d'une seule lettre correspondant à la valeur, l'interprétation continue en passant cette valeur, mais le programme vous en avertit. Les expressions de type caractère ne sont pas permises dans la liste des valeurs.

Le second type d'appel de macro peut être utilisé avec les éléments de bibliothèque, mais pas avec les simples fichiers texte GDL. Après le mot-clef PARAMETERS, vous devez lister (dans n'importe quel ordre) les noms des paramètres de la macro appelée, avec un signe '=' et une valeur pour chaque paramètre. Il est possible d'utiliser ici des expressions de type chaîne de caractères, mais faites bien attention à n'attribuer une valeur de caractère qu'aux paramètres de type caractères. Si le nom d'un paramètre de la liste de paramètres ne se trouve pas dans la macro appelée, un message d'alerte est affiché. Les paramètres de la macro appelée qui n'ont pas été listés dans l'appel de macro auront leurs valeurs originales par défaut telles qu\elles ont été définies dans l'élément de bibliothèque appelé comme macro.

La macro **GDL** a son propre environnement qui dépend de son ordre d'appel. Les valeurs actuelles des options `MODEL, RADIUS, RESOL, TOLER, PEN, LINE_TYPE, MATERIAL, FILL, STYLE, SHADOW` et la transformation courante sont valides dans la macro. Vous pouvez les utiliser ou les modifier, mais avec un effet local seulement, sans effet au niveau à partir duquel la macro a été appelée.

Donner des paramètres lors de l'appel d'une macro signifie une affectation implicite de valeur au niveau de la macro.

Les paramètres A et B sont généralement utilisés pour la redéfinition de la taille des objets.

Exemples:

```
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
                    ! use parameter default values
door-1 PARAMETERS
```

Pour résumer: si vous n'avez pas besoin de paramètres ayant un nom long ou de type caractères, le type de GDL texte simple peut être suffisant. Ce type de GDL ne peut être appelé qu'avec le premier type d'appel de macro, puisqu'il ne possède pas de liste de paramètres modifiables. Par contre, si vous ne voulez pas limiter les

noms de paramètres aux lettres de A à Z, ou si vous voulez inclure des chaînes de caractères dans la liste de paramètres, la macro doit être un élément de bibliothèque et elle doit être appelée selon la syntaxe définie pour le second type.

11.4 L'instruction de sortie

PRINT `expression [, expression ]. . .`

Affiche commentaires et valeurs des expressions dans une zone de dialogue. Les arguments peuvent être des chaînes de caractères ou des expressions numériques, séparés par des virgules. Le nombre possible d'expressions est illimité.

Exemples:

```
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
```

11.5 Opérations de fichiers

Les mots-clefs suivants permettent d'ouvrir des fichiers externes en lecture/écriture pour les manipuler en allant y lire/écrire des valeurs à partir de/à destination de scripts GDL. Cette procédure nécessite l'utilisation d'extensions spéciales. Les fichiers de type TEXT peuvent être traités avec l'extension 'TEXT GDL I/O Extension'. Des développeurs externes peuvent écrire des extensions pour d'autres types de fichiers.

OPEN `(filter, filename, paramstring)`

filter : chaîne, le nom d'une extension existante

filename : chaîne, le nom du fichier

paramstring: chaîne, contient les caractères de séparation spécifiques à l'extension opérationnelle ainsi que le mode d'ouverture. Son contenu est interprété par l'extension

Ouvre un fichier comme spécifié. La valeur en retour est un entier positif identifiant le fichier spécifique. Cette valeur sera le numéro de référence du fichier pour les instances suivantes.

INPUT `(channel, recordID, fieldID, var1 [, var2,...])`

recordID, fieldID :

la position de départ de la lecture de la chaîne ou de l'expression numérique, son contenu est interprété par l'extension

Le nombre de paramètres donné définit le nombre de valeurs à partir de la position de départ qui seront lus dans le fichier identifié par la valeur channel. Il doit y avoir au moins une valeur dans la liste des paramètres. Cette fonction met les valeurs lues dans les paramètres. Les valeurs peuvent être de type numérique ou chaîne de caractère, indépendamment du type de paramètre défini pour les stocker.

La valeur en retour est le nombre de valeurs lues. Si un caractère de fin de fichier est rencontré, la valeur sera de -1.

VARTYPE `(expr)`

Retourne 1 si le type de l'expression est numérique et 2 s'il s'agit d'une chaîne de caractères.

Cela se révèle utile lors de la lecture de valeurs dans les variables à l'aide de la commande `INPUT`, qui peut modifier le type de variables en fonction des valeurs actuelles.

Le type de ces variables n'est pas vérifié pendant le processus de compilation.

OUTPUT `channel, recordID, fieldID, expr1 [, expr2, ...]`

recordID, fieldID:

> la position de départ de l'écriture de la chaîne ou de l'expression numérique, son contenu est interprété par l'extension

Ecrit autant de valeurs dans le fichier identifié par la valeur channel à partir de la position donnée qu'il n'y a d'expressions définies. Il doit y avoir au moins une expression. Le type des valeurs est le même que celui des expressions.

CLOSE `channel`

Ferme le fichier identifié par la valeur channel.

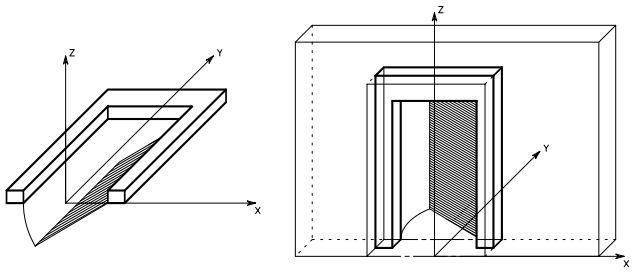
Chapitre 12

Instructions spéciales pour Portes et Fenêtres

Ce chapitre présente les options spéciales liées à la création d'éléments de bibliothèque de type Porte/Fenêtre.

12.1 Directives générales

Quand vous placez une porte ou une fenêtre dans un mur, la position par défaut de son système de coordonnées est orientée de manière à ce que le plan x-y soit vertical et que l'axe z pointe horizontalement dans le mur. L'origine est placée à bas et au centre de l'ouverture, à l'extérieur du mur. Ainsi, les portes/fenêtres peuvent être facilement modelées par des éléments sur le plan x-y. Voir illustrations ci-dessous.



En raison du comportement spécial de ces éléments de bibliothèque, leur symbole 2D est généré à partir d'une projection intégrée qui n'est pas accessible par l'utilisateur (une vue de côté retournée d'une direction de 90 degrés). Le symbole et la forme 3D sont ajustés à l'origine de la porte/fenêtre par le centre (x) inférieur (y) de la surface englobante, mais aucun ajustement n'est fait le long de l'axe z pour permettre à l'utilisateur de dessiner des portes/fenêtres sortant du mur vers le haut ou vers le bas.

Tenant compte des règles, suivez ces indications pour obtenir des portes/fenêtres qui fonctionnement correctement:

- En construisant la porte/fenêtre dans la fenêtre plan, pensez toujours que vous la regardez de l'intérieur du mur dans lequel elle sera implantée.
- Pensez au niveau Projet Zéro comme à la surface externe du mur.
- Les éléments se trouvant à l'intérieur du mur, comme le cadre de la fenêtre, sont à placer au-dessus du niveau Zéro.
- Les pans de porte s'ouvrant vers l'extérieur doivent se trouver en-dessous du niveau Zéro.

12.2 Créer un élément de bibliothèque de type Porte ou Fenêtre

Plusieurs possibilités existent pour créer des éléments de type Porte ou Fenêtre. Les problèmes qui se présent sont différents cas par cas:

- Création de porte/fenêtre rectanguaire dans un mur droit
- Création de porte/fenêtre non rectangulaire dans un mur droit
- Création de porte/fenêtre rectangulare dans un mur courbe
- Création de porte/fenêtre non rectangulaire dans un mur courbe

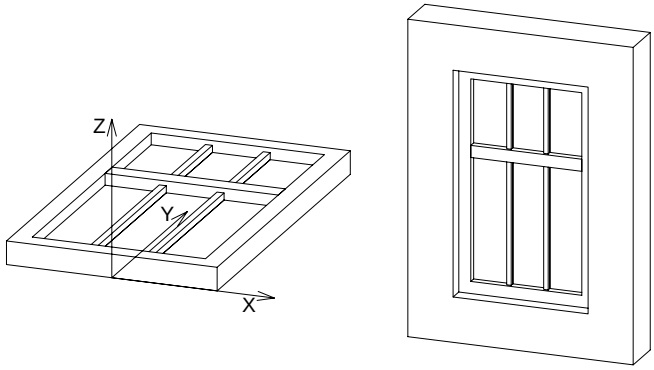
Porte ou Fenêtre rectangulaire dans un Mur droit

Ceci est la manière la plus simple de créer portes et fenêtres. L'utilisation de commandes GDL simples comme `PRISM_` ou `RECT` est recommandée.

Si vous voulez faire correspondre les matières de surface des éléments porte/fenêtre avec celles du mur, la surface inférieure des éléments doit correspondre à l'extérieur du mur et la surface supérieure des éléments à l'intérieur du mur. Pour cela, utilisez les variables globales G_, H_ et I_ dans vos scripts qui vous donnent les matières du mur dans lequel la porte/fenêtre est insérée. Dans le script 2D, les variables globales E_, F_ et A˜ peuvent être utiles, car elles retournent le numéro de stylo du contour et hachure de mur plus le numéro d'index de la hachure du mur sur le plan dans lequel la porte/fenêtre est insérée. Pour les murs composites, vous devez utiliser les variables globales correspondantes. Voir Annexe pour les détails.

La Bibliothèque ArchiCAD comprend un grand nombre de macros de porte/fenêtre. Ces scripts GDL contiennent des éléments de construction communs qui sont utilisés par de nombreuses portes/fenêtres. Il y a des macros pour générer des cadres, des panneaux et d'autres parties de porte/fenêtre fréquemment utilisées. Ouvrez quelques éléments de type porte et fenêtre pour voir quel type de macro ils appellent et quel est le type de parties que ces macros génèrent.

Exemple:



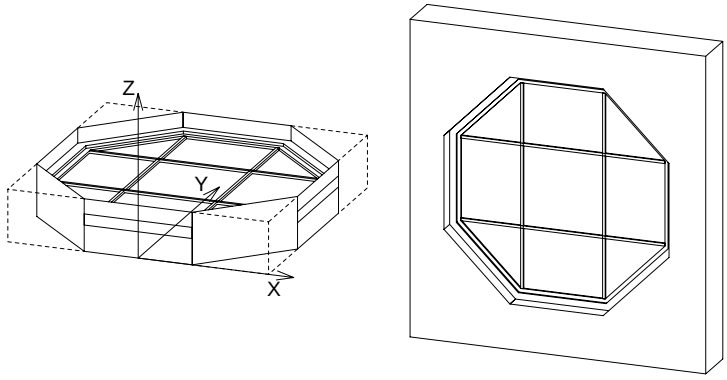

```
A=0.9: B=1.5: C=0.1: D=0.08
E=0.08: F=0.9: G=0.03: H=3
PRISM_ 10,C,
       -A/2, 0, 15, A/2, 0, 15,
       A/2, B, 15, -A/2, B, 15,
       -A/2, 0, -1,
       -A/2+D, D, 15, A/2-D, D, 15,
       A/2-D, B-D, 15, -A/2+D, B-D, 15,
       -A/2+D, D, -1
ADDX -A/2+D, F, 0
BRICK A-2*D, E, C
ADDX -G/2, -F+D, C/2
GOSUB 1
ADDZ -G
GOSUB 1
DEL 2
MATERIAL "Glass"
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
```

Porte ou Fenêtre non rectangulaire dans un Mur droit

En travaillant avec des portes et des fenêtres, il importe de savoir qu'ArchiCAD coupe toujours une ouverture rectangulaire dans le mur dans lequel la porte ou fenêtre sera placée. La taille de la baie est déterminée par les paramètres A et B de l'élément de bibliothèque. Toutefois, si la porte ou fenêtre n'est pas rectangulaire en façade, elle ne remplit pas entièrement l'ouverture rectangulaire. Il existe deux solutions:

1. Le script 3D doit contenir des parties qui génèrent des partie de mur qui rempliront les trous entre le corps de la porte/fenêtre et les arêtes du mur rectangulaire coupé. Dans ce cas, il faut faire particulièrement attention à la visibilité des arêtes.



2. Utiliser la commande `WALLHOLE` dans ArchiCAD 6.0 ou version postérieure. Ellle permet de définir une forme polygonale à couper dans le mur où sera placée la porte/fenêtre.

WALLHOLE `n, status,`
`x1, y1, mask1,`
`. . .`
`xn, yn, maskn`
`[, x, y, z]`

n : le nombre de nœuds du polygone

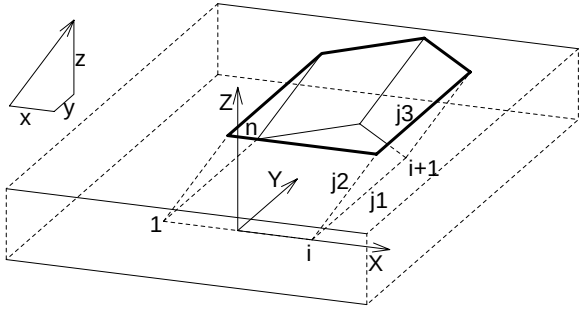
status :

1: utiliser les attributs propres du corps pour les polygones et arêtes générés
2: les polygones coupés générés seront traités comme des polygones normaux

x_i, y_i : coordonnées du polygone de coupe transversale

$mask_i$: similaire à l'instruction CUTPOLYA
$mask_i = j_1 + 2 * j_2 + 4 * j_3$

x, y, z : vecteur de direction optionnel (par défaut l'axe Z de la porte/fenêtre)



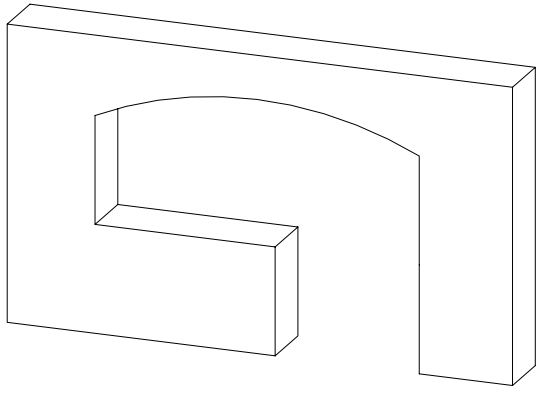
Cette commande peut être utilisée dans le script 3D des portes et fenêtres pour couper un trou personnalisé dans le mur. Pendant la génération en 3D du mur courant, les scripts 3D de tous ses portes et fenêtres seront interprétés dans générer de modèle pour collecter toutes les commandes WALLHOLE. Si de telles commandes existent, ArchiCAD va couper le mur courant en utilisant un tube infini avec la coupe transversale polygonale et la direction définie dans le script. Il peut y avoir un nombre illimité de WALLHOLEs pour toute porte/fenêtre, il est donc possible de couper plusieurs ouvertures pour la même porte/fenêtre et ces ouvertures peuvent même s'intersecter. S'il y a au moins une commande WALLHOLE interprétée dans le script 3D de la porte/fenêtre, ArchiCAD ne générera pas d'ouverture rectangulaire pour cette porte/fenêtre.

Remarque: L'ébrasure 3D ne sera pas automatiquement générée pour les ouvertures personnalisées, vous devez la générer à partir du script.

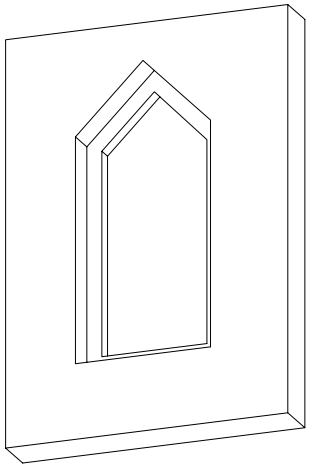
Le trou personnalisé de cette manière ne sera visible qu'en 3D, car les commandes WALLHOLE n'ont pas d'effet en 2D. La représentation 2D peut être scriptée si nécessaire (pour l'utiliser, débrancher encadrement sur le plan).

Il est recommandé d'utiliser des coupes transversales polygonales convexes, car l'utilisation de polygones concaves peut donner lieu à des rendus bizarres ou à des erreurs en coupe. Vous pouvez par ailleurs combiner des polygones convexes pour en obtenir de concaves.

Exemples:



```
RESOL 72
L1=2.7 : L2=1.2 : H1=2.1 : H2=0.3 : H3=0.9
R=((L1/2)^2+H2^2)/(2*H2)
A=ATN((L1/2)/(R-H2))
WALLHOLE 5,1,
       -L1/2,H3,15,
       L1/2,H3,15,
       L1/2,H1-H2,13,
       0,H1-R,915,
       0,2*A,4015
WALLHOLE 4,1,
       L1/2-L2,0,15,
       L1/2,0,15,
       L1/2,H3,15,
       L1/2-L2,H3,15
```

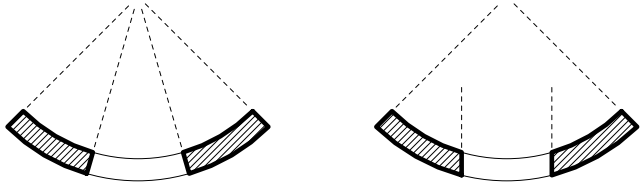


```
WALLHOLE     5,1,
       -0.45, 0,     15,
       0.45,  0,     15,
       0.45,  1.5,   15,
       0,     1.95,  15,
       -0.45, 1.5,   15

PRISM_ 12,    0.1,
       -0.45, 0,     15,
       0.45,  0,     15,
       0.45,  1.5,   15,
       0,     1.95,  15,
       -0.45, 1.5,   15,
       -0.45, 0,     -1,
       -0.35, 0.1,   15,
       0.35,  0.1,   15,
       0.35,  1.45,  15,
       0,     1.80,  15,
       -0.35, 1.44,  15,
       -0.35, 0.1,   -1
```

Porte ou Fenêtre rectangulaire dans un Mur courbe

En plaçant une porte ou fenêtre dans un mur courbe, les côtés de l'ouverture pratiquée dans le mur varie comme sur l'illustration qui suit.

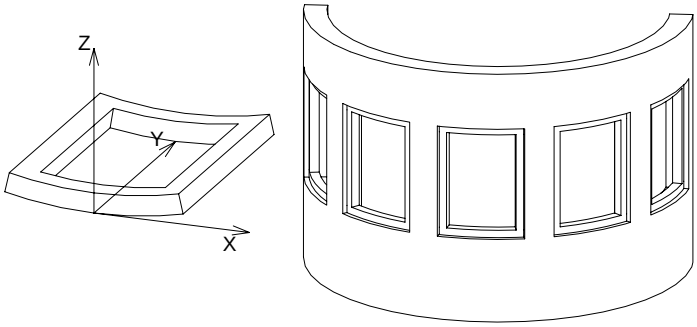


L'ouverture à gauche est créée lorsque ArchiCAD coupe automatiquement l'ouverture de porte/fenêtre. Dans ce cas, les côtés seront radiaux. A droite, l'ouverture est coupée en utilisant la commande `WALLHOLE` dans le script 3D de l'élément porte/fenêtre. L'objet lui-même doit être scripté en tenant compte de ces facteurs.

De plus, il importe de prendre en considération si la porte/fenêtre sera placée dans un mur droit ou courbe.

Pour les portes/fenêtres droites (à gauche), l'épaisseur et la largeur de l'élément sont étroitement liées à l'épaisseur du mur, puisque, au-delà d'une certaine dimension, l'objet ne serait plus dans le mur. En utilisant de véritables portes/fenêtres courbes. ce problème n'apparaît pas.

Exemple:

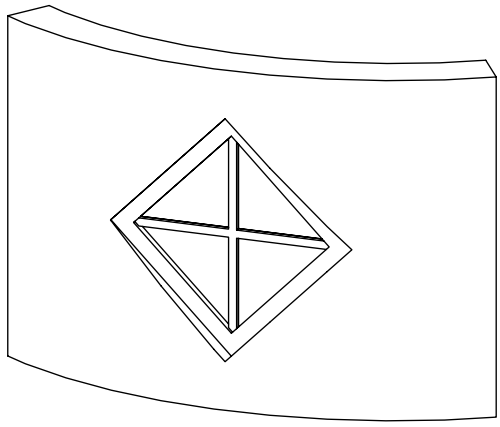



```
RESOL  72
ROTX   -90
MULY   -1
C= 0.12 : Z=(360*A)/(2*R_*PI)
Y= (360*C)/(2*R_*PI)
A1= 270+Z/2 : A2=270-Z/2
GOSUB  1
ADDZ   B
MULZ   -1
GOSUB  1
DEL    2
ADDZ   C
GOSUB  2
MULX   -1
GOSUB  2
END
1:
PRISM_        9,          C,
  COS(A2)*R_,           SIN(A2)*R_+R_,       11,
  COS(A2+Y)*R_,          SIN(A2+Y)*R_+R_,    13,
  0,                    R_,                  900,
  0,                    Z-2*Y,               4009,
  COS(A1)*R_,           SIN(A1)*R_+R_,       11,
  COS(A1)*(R_-0.1),     SIN(A1)*(R_-0.1)+R_, 11,
  COS(A1-Y)*(R_-0.1),   SIN(A1-Y)*(R_-0.1)+R_, 13,
  0,                    -(Z-2*Y),            4009,
  COS(A2)*(R_-0.1),     SIN(A2)*(R_-0.1)+R_, 11
RETURN
2:
PRISM_        4,          B-2*C,
  COS(A2)*R_,           SIN(A2)*R_+R_,       10,
  COS(A2+Y)*R_,          SIN(A2+Y)*R_+R_,    15,
  COS(A2+Y)*(R_-0.1),   SIN(A2+Y)*(R_-0.1)+R_,10,
  COS(A2)*(R_-0.1),     SIN(A2)*(R_-0.1)+R_, 10
RETURN
```

Porte ou Fenêtre non rectangulaire dans un Mur courbe

Les directives générales valables pour les portes et fenêtres rectangulaires s'appliquent ici aussi.

Exemple :



```
C=0.1 : D=0.025
Z=A/2-SQR(2)*C : Y=A/2-SQR(2)*C-D
ADDY A/2
WALLHOLE      4,     1,
       0,     -A/2,  15,
       A/2,   0,     15,
       0,     A/2,   15,
       -A/2,  0,     15
PRISM_ 10,    0.1,
       0,     -A/2,  15,
       A/2,   0,     15,
       0,     A/2,   15,
       -A/2,  0,     15,
       0,     -A/2,  -1,
       0,     -Z,    15,
       Z,     0,     15,
       0,     Z,     15,
       -Z,    0,     15,
       0,     -Z,    -1
ADDZ   0.02
GOSUB  1
ADDZ   0.03
```

```
GOSUB  1
ADDY   -Z
SET MATERIAL "Glass"
ROTZ   45
RECT   SQR(2)*Z,    SQR(2)*Z
END
1:
PRISM_ 16,    0.03,
       0,     -Z,    15,
       D,     -Y,    15,
       D,     -D,    15,
       Y,     -D,    15,
       Z,     0,     15,
       Z,     D,     15,
       D,     D,     15,
       D,     Y,     15,
       0,     Z,     15,
       -D,    Y,     15,
       -D,    D,     15,
       -Y,    D,     15,
       -Z,    0,     15,
       -Y,    -D,    15,
       -D,    -D,    15,
       -D,    -Y,    15
RETURN
```

Annexe

A Variables globales

Les variables globales permettent de stocker des valeurs spéciales du modèle pour accéder à des informations géométriques de l'environnement de la macro. Vous pouvez, par exemple, accéder aux paramètres d'un mur en définissant une fenêtre qui doit cadrer dedans. Les variables globales ne sont pas empilées pendant les appels de macro.

Informations générales sur l'environnement

GLOB_SCRIPT_TYPE	T~	type du script courant
1-script de descriptif, 2-2D script, 3-3D script, 4-non implémenté, 5-script de liste de valeurs, 6-script maître		
GLOB_CONTEXT		contexte de l'apparition
1-éditeur élément, 2-plan d'étage, 3-vue 3D, 4-coupe/façade, 5-dialogue Optoins, 6-liste		
GLOB_SCALE	A_	échelle de dessin
selon la fenêtre active		
GLOB_NORTH_DIR	U~	direction Nord du projet
relativement au système de coordonnées par défaut du projet et selon les réglages faits dans le dialogue Héliodon		
GLOB_DRAWING_BGD_PEN		numéro de stylo pour la couleur de fond du dessin
le stylo le plus proche de la palette courante à la couleur de fond de la fenêtre courante		
GLOB_SUN_AZIMUTH		azimut soleil
en fonction des paramètres du dialogue Héliodon		
GLOB_SUN_ALTITUDE		altitude du soleil
en fonction des paramètres du dialogue Héliodon		
GLOB_MODPAR_NAME		nom du dernier paramètre modifié
dans le dialogue Options ou dans l'éditeur d'élément de bibliothèque		

Informations d'étage

GLOB_HSTORY_ELEV	B_	altitude de l'étage hôte
l'étage hôte est celui sur lequel l'objet a été placé		
GLOB_HSTORY_HEIGHT	Q_	hauteur de l'étage hôte
l'étage hôte est celui sur lequel l'objet a été placé		
GLOB_CSTORY_ELEV	Q~	altitude de l'étage courant
l'étage courant est celui qui apparaît actuellement dans la fenêtre Plan d'étage		
GLOB_CSTORY_HEIGHT	R~	hauteur de l'étage courant
l'étage courant est celui qui apparaît actuellement dans la fenêtre Plan d'étage		
GLOB_CH_STORY_DIST	S~	position relative de l'étage courant par rapport à l'étage hôte
l'étage courant est celui qui apparaît actuellement dans la fenêtre Plan d'étage		

Informations de séquence d'animation

GLOB_FRAME_NR	N_	numéro de l'image dans l'animation
pour animations seulement, 0 pour images fixes		
GLOB_FIRST_FRAME	O_	index de la première image d'une séquence
pour animations seulement, 0 pour images fixes		
GLOB_LAST_FRAME	P_	index de la dernière image d'une séquence
pour animations seulement, 0 pour images fixes		
GLOB_EYEPOS_X	K~	position courante de la caméra en x
seulement pour perspectives animées et fixes		
GLOB_EYEPOS_Y	L~	position courante de la caméra en y
seulement pour perspectives animées et fixes		
GLOB_EYEPOS_Z	M~	position courante de la caméra en z
seulement pour perspectives animées et fixes		
GLOB_TARGPOS_X	N~	position courante du point visé en x
seulement pour perspectives animées et fixes		
GLOB_TARGPOS_Y	O~	position courante du point visé en y
seulement pour perspectives animées et fixes		
GLOB_TARGPOS_Z	P~	position courante du point visé en z
seulement pour perspectives animées et fixes		

Paramètres d'élément généraux

GLOB_LAYER	calque de l'élément
nom du calque auquel l'élément est assigné	
GLOB_ID	ID de l'élément
ID défini dans le dialogue de paramétrage	
GLOB_INTID	ID interne de l'élément
l'ID interne unique généré par le programme (non disponible pour l'utilisateur)	
GLOB_ELEVATION	J_ altitude de base de l'élément
par rapport à son étage hôte (à l'exception de la hauteur de seuil de porte et de fenêtre, selon la configuration actuelle)	

Paramètres d'élément généraux - pour listes seulement

GLOB_ELEM_TYPE	type d'élément
1-objet, 2-lampe, 3-fenêtre, 4-porte, 5-mur, 6-poteau, 7-dalle, 8-toiture, 9-hachure, 10-maillage, 11-zone	

Paramètres d'Objet, Lampe, Porte, Fenêtre

SYMB_LINETYPE	type de ligne de l'élément de bibliothèque
appliqué par défaut au symbole 2D	
SYMB_FILL	motif de hachure de l'élément de bibliothèque
appliqué sur les surfaces coupées dans les fenêtres Coupe/Façade	

SYMB_FILL_PEN		couleur de hachure de l'élément de bibliothèque
appliqué sur les surfaces coupées dans les fenêtres Coupe/Façade		
SYMB_FBGD_PEN		couleur de fond de hachure de l'élément de bibliothèque
appliqué sur les surfaces coupées dans les fenêtres Coupe/Façade		
SYMB_SECT_PEN		couleur de l'élément de bibliothèque en coupe
appliqué sur les contours des surfaces coupées dans les fenêtres Coupe/Façade		
SYMB_VIEW_PEN	L_	couleur par défaut de l'élément de bibliothèque
appliqué sur toutes les arêtes en 3D et sur les arêtes visibles en Coupe/Façade		
SYMB_MAT	M_	matière par défaut de l'élément de bibliothèque
SYMB_POS_X	X~	position de l'élément de bibliothèque (x)
relativement à l'origine du projet (sauf porte/fenêtre: par rapport au premier point du mur hôte)		
SYMB_POS_Y	Y~	position de l'élément de bibliothèque (y)
relativement à l'origine du projet (sauf porte/fenêtre: par rapport au premier point du mur hôte)		
SYMB_POS_Z	Z~	position de l'élément de bibliothèque (z)
relativement à l'origine du projet (sauf porte/fenêtre: par rapport au premier point du mur hôte)		

Paramètres d'Objet et Lampe

SYMB_ROTANGLE	W~	angle de rotation de l'élément de bibliothèque
rotation numérique à partir du dialogue de paramétrage exécuté autour du point d'ancrage courant		
SYMB_MIRRORED	V~	élément de bibliothèque symétrique
0-non, 1-oui (la symétrie est exécutée sur le point d'ancrage courant)		

Paramètres d'Objet, Lampe, Porte et Fenêtre - pour listes seulement

SYMB_A_SIZE	longueur/largeur nominale de l'élément
longueur d'objet/lampe, largeur de fenêtre/porte (paramètre fixe)	
SYMB_B_SIZE	largeur/longueur nominale de l'élément
largeur d'objet/lampe, largeur de fenêtre/porte (paramètre fixe)	

Paramètres d'Objet et de Lampe - pour listes seulement

SYMB_Z_SIZE	hauteur nominale de l'élément
si le dernier paramètre utilisateur est nommé au format ZZYZX, utilisé pour la hauteur nominale, sinon 0	

Paramètres de Fenêtre et Porte

WIDO_REVEAL_ON		ébrasure fenêtre/porte active
0-non, 1-oui		
WIDO_SILL	K_	seuil de porte/fenêtre
murs courbes: en direction radiale à l'angle d'ouverture de taille nominale		
WIDO_RIGHT_JAMB	B~	feuillure de porte/fenêtre côté droit
comme dans dialogue Options Ebrasure		

WIDO_LEFT_JAMB		feuillure de porte/fenêtre côté gauche
comme dans dialogue Options Ebrasure		
WIDO_THRES_DEPTH	C~	profondeur seuil/appui fenêtre/porte
feuillure de porte/fenêtre côté gauche		*comme dans dialogue Options Ebrasure*
WIDO_HEAD_DEPTH	D~	hauteur de l'ébrasure
comme dans dialogue Options Ebrasure		
WIDO_REVEAL_SIDE	E~	côté ébrasure opposé à côté d'ouverture
1-oui, 0-non - en plaçant l'élément, la valeur par défaut est 0 pour fenêtres, 1 pour portes		
WIDO_FRAME_THICKNESS	F~	épaisseur du cadre de la fenêtre/porte
en basculant une porte/fenêtre, elle subira une symétrie et déplacée de cette valeur		
WIDO_POSITION	H~	décalage de la porte/fenêtre
angle ou distance entre l'axe de l'ouvertue et le vecteur normal au premier point du mur		
WIDO_ORIENTATION		orientation de l'ouverture porte/fenêtre
gauche/droite - ne fonctionne de manière normale que si la porte/fenêtre correspond aux normes locales		
WIDO_MARKER_TXT		texte marque fenêtre/porte
comme défini dans le sous-dialogue Cotation Porte/Fenêtre		
WIDO_SUBFL_THICKNESS		épaisseur chape (correction du seuil)
comme défini dans le sous-dialogue Cotation Porte/Fenêtre		
WIDO_PREFIX		préfixe hauteur de seuil porte/fenêtre
comme défini dans le sous-dialogue Cotation Porte/Fenêtre		
WIDO_CUSTOM_MARKER		marque personnalisée de porte/fenêtre
1-paramètres peuvent être utilisés dans le script 2D tant que la cotation automatique n'est pas présente		
WIDO_ORIG_DIST	R_	distance de l'origine locale à partir de l'extrémité du mur
distance de l'origine locale à partir du centre du mur courbe, 0 pour murs droits		
WIDO_PWALL_INSET		incrustation dans mur

Paramètres de Lampe - pour listes seulement

LIGHT_ON	lumière branchée
0-débranchée, 1-branchée: comme défini dans le dialogue Options Lampe (paramètre fixe)	
LIGHT_RED	composante rouge de la couleur de la lumière
comme défini dans le dialogue Options Lampe (paramètre fixe)	
LIGHT_GREEN	composante verte de la couleur de la lumière
comme défini dans le dialogue Options Lampe (paramètre fixe)	
LIGHT_BLUE	composante bleue de la couleur de la lumière
comme défini dans le dialogue Options Lampe (paramètre fixe)	
LIGHT_INTENSITY	intensité de la lumière
comme défini dans le dialogue Options Lampe (paramètre fixe)	

Paramètres d'étiquette

LABEL_POSITION	position de l'étiquette
tableau [3][2] contenant les coordonnées des 3 points définissant la position de l'étiquette	

LABEL_CUSTOM_ARROW	utilise flèche symbole
1 si la case Utiliser flèche symbole est cochée, 0 si non	
LABEL_ARROW_PEN	couleur de flèche dans le dialogue d'options
LABEL_ARROWHEAD_PEN	couleur de pointe de flèche dans le dialogue d'options
LABEL_FONT_NAME	nom de la police dans le dialogue d'options
LABEL_TEXT_SIZE	taille du texte dans le dialogue d'options
LABEL_TEXT_PEN	couleur du texte dans le dialogue d'options
LABEL_FONT_STYLE	style de police dans le dialogue d'options
0-normal, 1-gras, 2-italique	
LABEL_FRAME_ON	cadre d'étiquette oui/non
1 si la case Encadrer est cochée 0 si non	
LABEL_ANCHOR_POS	
1, 2 ou 3 selon la position du point d'ancrage dans le dialogue d'options	

Paramètres de Mur - disponibles pour Portes/Fenêtres

WALL_RESOL	J~	résolution en 3D d'un mur courbe
effective en 3D seulement		
WALL_THICKNESS	C_	épaisseur du mur
pour les murs inclinés: épaisseur du mur à l'axe d'ouverture (axe z local)		
WALL_INCL		inclinaison des surfaces de mur
l'angle entre deux surfaces de mur inclinées - 0 pour les murs droits simples		
WALL_HEIGHT	D_	hauteur du mur
WALL_MAT_A	G_	matière de mur du côté opposé de l'ouverture
peut varier d'ouverture en ouverture dans un même mur		
WALL_MAT_B	H_	matière de mur du côté de l'ouverture
peut varier d'ouverture en ouverture dans un même mur		
WALL_MAT_EDGE	I_	matière de mur des bords du mur
WALL_LINETYPE		type de ligne du mur
pour les contours seulement dans la fenêtre plan d'étage		
WALL_FILL	A~	motif de hachure du mur
index de hachure - valeur négative pour les composites		
WALL_FILL_PEN	F_	numéro de stylo de la hachure du mur
WALL_COMPS_NAME		structure composite du mur
nom de la structure composite		
WALL_SKINS_NUMBER		nombre des couches de mur composites
entre 1 et 8, 0 pour les hachures simples		
WALL_SKINS_PARAMS		paramètres des couches de mur composites
matrice de 6 colonnes: hachure, épaisseur, couleur, couleur de hachure, couleur de fond de hachure, état structure, et de 8 lignes max.		
WALL_SECT_PEN	E_	numéro de stylo du contour du mur en coupe
appliqué sur les contours des surfaces coupées des murs en plan et en coupe		
WALL_VIEW _PEN		numéro de stylo des contours de mur visibles
appliqué sur toutes les arêtes en 3D et sur les arêtes visibles en coupe		

WALL_FBGD_PEN	numéro de stylo du fond de hachure du murl
WALL_DIRECTION	direction du mur
murs droits: direction de la ligne de référence, murs courbes: direction de la corde de l'arc	
WALL_POSITION	coordonnées absolues du mur
la position du premier point du mur par rapport à l'origine du projet	

Paramètres de Mur - pour listes seulement

WALL_LENGTH_A	longueur du mur du côté ligne de référence
WALL_LENGTH_B	longueur du mur du côté opposé à la ligne de référence
WALL_SURFACE_A	surface du mur du côté ligne de référence
WALL_SURFACE_B	surface du mur du côté opposé à la ligne de référence
WALL_EDGE_SURF	surface de l'arête du mur
WALL_VOLUME	volume du mur
WALL_DOORS_NR	nombre de portes dans le mur
WALL_WINDS_NR	nombre de fenêtres dans le mur
WALL_HOLES_NR	nombre de baies libres dans le mur
WALL_DOORS_SURF	surface des portes dans le mur
WALL_WINDS_SURF	surface des fenêtres dans le mur
WALL_HOLES_SURF	surface des baies libres dans le mur
WALL_WINDS_WID	largeur totale des fenêtres dans le mur
WALL_DOORS_WID	largeur totale des portes dans le mur
WALL_COLUMNS_NR	nombre de poteaux dans le mur

Paramètres de Poteau - pour listes seulement

COLU_CORE	descriptif âme/revêtement
pour compatibilité: effectif seulement dans le script descriptif des fichiers .CPS (Descriptif Poteau)	
COLU_HEIGHT	hauteur du poteau
COLU_VENEER_WIDTH	épaisseur du revêtement du poteau
COLU_MAT	matière du poteau
Remarque: Le revêtement mur remplace la matière du poteau par celle des murs qui s'y connectent	
COLU_LINETYPE	type de ligne du poteau
appliqué sur les contours seulement en vue plan	
COLU_CORE_FILL	hachure de l'âme du poteau
COLU_VENEER_FILL	hachure du revêtement du poteau
COLU_SECT_PEN	numéro de stylo des contours du poteau en coupe
appliqué au contour des surfaces coupées en vue plan et coupe	
COLU_VIEW_PEN	numéro de stylo du poteau visible
appliqué à toutes les arêtes en 3D et sur les arêtes visibles en coupe	
COLU_CORE_FILL_PEN	numéro de stylo de la hachure de l'âme du poteau
COLU_CORE_FBGD_PEN	numéro de stylo du fond de hachure de l'âme du poteau
COLU_VENEER_FILL_PEN	numéro de stylo de la hachure du revêtement du poteau
COLU_VENEER_FBGD_PEN	numéro de stylo du fond de hachure du revêtement du poteau

COLU_CORE_SURF	surface de l'âme du poteau
COLU_CORE_VOL	volume de l'âme du poteau
COLU_VENEER_SURF	surface du revêtement du poteau
COLU_VENEER_VOL	volume du revêtement du poteau

Paramètres de Poutre – pour listes seulement

BEAM_THICKNESS	epaisseur de la poutre
BEAM_HEIGHT	hauteur de la poutre
BEAM_REFLINE_OFFSET	décalage de la ligne de référence par rapport aux axes de la poutre
BEAM_PRIORITY	numéro d'index prioritaire d'intersection 3D
BEAM_MAT_RIGHT	matière de la poutre, côté droit de la ligne de référence
BEAM_MAT_LEFT	matière de la poutre, côté gauche de la ligne de référence
BEAM_MAT_TOP	matière du haut de la poutre
BEAM_MAT_BOTTOM	matière du bas de la poutre
BEAM_MAT_END	matière de la poutre aux deux extrémités
BEAM_OUTLINE_LINETYPE	type de ligne du contour de la poutre
BEAM_AXES_LINETYPE	type de ligne des axes de poutre
BEAM_FILL	motif de hachures de la poutre
BEAM_FILL_PEN	couleur de hachure de la poutre
BEAM_SECT_PEN	couleur des contours de la poutre en coupe
BEAM_FBGD_PEN	couleur de la hachure du fond de la poutre
BEAM_DIRECTION	direction de la ligne de référence de la poutre
BEAM_POSITION	coordonnées absolues du point de départ des axes de poutre
BEAM_LENGTH_RIGHT	longueur de la poutre, côté droit de la ligne de référence
BEAM_LENGTH_LEFT	longueur de la poutre, côté gauche de la ligne de référence
BEAM_RIGHT_SURF	surface de la poutre, côté droit de la ligne de référence
BEAM_LEFT_SURF	surface de la poutre, côté gauche de la ligne de référence
BEAM_TOP_SURF	surface du haut de la poutre
BEAM_BOTTOM_SURF	surface du bas de la poutre
BEAM_END_SURF	surface des deux extrémités de la poutre
BEAM_VOLUME	volume de la poutre
BEAM_HOLES_NR	nombre de trous dans la poutre
BEAM_HOLES_SURF	surface totale des trous dans la poutre
BEAM_HOLE_EDGE_SURF	surface totale des arêtes de trous dans la poutre
BEAM_HOLES_VOLUME	volume total des trous dans la poutre

Paramètres de Dalle - pour listes seulement

SLAB_THICKNESS	épaisseur de la dalle
SLAB_MAT_TOP	matière de la surface supérieure de la dalle
SLAB_MAT_EDGE	matière des arêtes de la dalle
SLAB_MAT_BOTT	matière de la surface inférieure de la dalle
SLAB_LINETYPE	type de ligne de la dalle

SLAB_FILL	hachure de la dalle
index de hachure - valeur négative pour les composites	
SLAB_FILL_PEN	numéro de stylo de la hachure de la dalle
SLAB_FILLBGD_PEN	numéro de stylo du fond de hachure de la dalle
SLAB_COMPS_NAME	structure composite de la dalle
nom de la structure composite	
SLAB_SKINS_NUMBER	nombre des couches de dalle composites
entre 1 et 8, 0 pour les hachures simples	
SLAB_SKINS_PARAMS	paramètres des couches de dalle composites
matrice de 6 colonnes: hachure, épaisseur, couleur, couleur de hachure, couleur de fond de hachure, état structure, et de 8 lignes max.	
SLAB_SECT_PEN	numéro de stylo des contours de dalle en coupe
appliqué sur les contours des surfaces coupées en plan et en coupe	
SLAB_VIEW_PEN	numéro de stylo de la dalle visible
appliqué sur toutes les arêtes en 3D et sur les arêtes visibles en coupe	
SLAB_TOP_SURF	surface supérieure de la dalle
SLAB_BOT_SURF	surface inférieure de la dalle
SLAB_EDGE_SURF	surface des arêtes de la dalle
SLAB_PERIMETER	périmètre de la dalle
SLAB_VOLUME	volume de la dalle
SLAB_SEGMENTS_NR	number de segments de dalle
SLAB_HOLES_NR	number de trémies dans la dalle
SLAB_HOLES_AREA	surface des trémies dans la dalle
SLAB_HOLES_PRM	périmètre des trémies dans la dalle

Paramètres de Toiture - pour listes seulement

ROOF_THICKNESS	épaisseur de la toiture
ROOF_ANGLE	pente de la toiture
ROOF_MAT_TOP	matière de la surface supérieure de la toiture
ROOF_MAT_EDGE	matière des arêtes de la toiture
ROOF_MAT_BOTT	matière de la surface inférieure de la toiture
ROOF_LINETYPE	type de ligne de la toiture
appliqué sur les contours en vue plan seulement	
ROOF_FILL	hachure de la toiture
index de hachure - valeur négative pour les structures composites	
ROOF_FILL_PEN	numéro de stylo de la hachure de toiture
ROOF_FBGD_PEN	numéro de stylo du fond de hachure de toiture
ROOF_COMPS_NAME	structure composite de la toiture
nom de la structure composite	
ROOF_SKINS_NUMBER	nombre des couches de toiture composites
entre 1 et 8, 0 pour les hachures simples	
ROOF_SKINS_PARAMS	paramètres des couches de toiture composites
matrice de 6 colonnes: hachure, épaisseur, couleur, couleur de hachure, couleur de fond de hachure, état structure, et de 8 lignes max.	

ROOF_SECT_PEN	numéro de stylo des contours de toiture en coupe
appliqué sur les contours des surfaces coupées en plan et en coupe	
ROOF_VIEW_PEN	numéro de stylo de la toiture visible
appliqué sur toutes les arêtes en 3D et sur les arêtes visibles en coupe	
ROOF_BOTTOM_SURF	surface inférieure de la toiture
ROOF_TOP_SURF	surface supérieure de la toiture
ROOF_EDGE_SURF	surface des arêtes de la toiture
ROOF_PERIMETER	périmètre de la toiture
ROOF_VOLUME	volume de la toiture
ROOF_SEGMENTS_NR	nombre de segments de la toiture
ROOF_HOLES_NR	nombre d'ouverture dans la toiture
ROOF_HOLES_AREA	surface des ouvertures dans la toiture
ROOF_HOLES_PRM	périmètre des ouvertures dans la toiture

Paramètres de Hachure - pour listes seulement

FILL_LINETYPE	type de ligne de la hachure
FILL_FILL	motif de hachure de la hachure
FILL_FILL_PEN	numéro de stylo de la hachure de la hachure
FILL_PEN	numéro de stylo de la hachure
FILL_FBGD_PEN	numéro de stylo du fond de hachure de la hachure
FILL_SURF	surface de la hachure
FILL_PERIMETER	périmètre de la hachure
FILL_SEGMENT_NR	nombre des segments de la hachure
FILL_HOLES_NR	nombre des trous dans la hachure
FILL_HOLES_PRM	périmètre des trous dans la hachure
FILL_HOLES_AREA	surface des trous dans la hachure

Paramètres de Maillage - pour listes seulement

MESH_TYPE	type du maillage
1- corps solide, 2 - surface et arêtes, 3 - surface seulement	
MESH_BASE_OFFSET	décalage de la surface inférieure par rapport au niveau de base
MESH_USEREDGE_PEN	numéro de stylo des arêtes définies manuellement
MESH_TRIEDGE_PEN	numéro de stylo des arêtes de maillage par triangulation
MESH_SECT_PEN	numéro de stylo des contours du maillage en coupe
appliqué sur les contours des surfaces coupées des murs en plan et en coupe	
MESH_VIEW_PEN	numéro de stylo des contours des maillage visibles
appliqué à toutes les arêtes en 3D et sur les arêtes visibles en coupe	
MESH_MAT_TOP	matière de la surface supérieure du maillage
MESH_MAT_EDGE	matière des arêtes du maillage
MESH_MAT_BOTT	matière de la surface inférieure du maillage
MESH_LINETYPE	type de ligne du maillage
appliqué aux contours seulement en plan	
MESH_FILL	motif de hachure du maillage

MESH_FILL_PEN	numéro de stylo de la hachure du maillage
MESH_FBGD_PEN	numéro de stylo du fond de hachure du maillage
MESH_BOTTOM_SURF	surface inférieure du maillage
MESH_TOP_SURF	surface supérieure du maillage
MESH_EDGE_SURF	surface des arêtes du maillage
MESH_PERIMETER	périmètre du maillage
MESH_VOLUME	volume du maillage
MESH_SEGMENTS_NR	nombre de segments du maillage
MESH_HOLES_NR	nombre de trémies dans le maillage
MESH_HOLES_AREA	surface des trémies dans le maillage
MESH_HOLES_PRM	périmètres des trémies dans le maillage

Variables globales utilisables librement

GLOB_USER_1	S_	
GLOB_USER_2	T_	
GLOB_USER_3	U_	
GLOB_USER_4	V_	
GLOB_USER_5	W_	
GLOB_USER_6	X_	
GLOB_USER_7	Y_	
GLOB_USER_8	Z_	
GLOB_USER_9	G~	
GLOB_USER_10	I~	les variables 1 à 10 sont initialisées comme numériques par défaut
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		les variables 11 à 20 sont initialisées comme chaînes par défaut

B

Fonctions spéciales

REQ `(parameter_string)`

Cette fonction demande l'état courant du programme. Son paramètre - la question - est une chaîne de caractères. L'interpréteur **GDL** donne en réponse une valeur numérique. Si la question ne peut pas être interprétée, la réponse est négative.

Liste des questions existantes:

"**GDL_version**"

numéro de version du compilateur/interpréteur GDL compiler/interpreter. (Attention: il n'est pas égal à celui de la version d'ArchiCAD).

"**Program**"

code du programme (1: ArchiCAD, 2: T.CAD, etc.).

"**Serial_number**"

le numéro de série de la clé électronique.

"**Model_size**"

taille en octets de la structure de données 3D courantes.

"**Red_of_material** name"
"**Green_of_material** name"
"**Blue_of_material** name"

Définissent les composantes couleur de la matière donnée en valeurs RVB entre 0 et 1.

"**Red_of_pen** index"
"**Green_of_pen** index"
"**Blue_of_pen** index"

Définissent les composantes couleur du stylo donnée en valeurs RVB entre 0 et 1.

"**Pen_of_RGB** r g b"

Définit l'index du stylo le plus proche de la couleur donnée. Les constantes r, g et b figurent dans la chaîne avec des valeurs entre 0 et 1.

REQUEST `(question_name, name | index, var1 [, var2,....])`

Le premier paramètre représente la chaîne de la question tandis que le second représente l'objet de la question (s'il y en a un) et peut être de type caractère ou numérique. Par exemple, la question peut être "Rgb_of_material" et son objet le nom de la matière, ou la question peut être "Rgb_of_pen" et son objet l'index du stylo. Les autres paramètres sont les noms de variables dans lesquelles les valeurs de retour (les réponses) sont stoquées. La valeur en retour de la fonction est la réponse (en cas de mauvaise question ou de nom non existant, la valeur est de 0).

REQUEST `("Name_of_program", "", programname)`

Retourne dans la variable donnée le nom du programme, par exemple: "ArchiCAD", "topCAD" , etc.

REQUEST `("Name_of_macro", "", myname)`

REQUEST `("Name_of_main", "", mainname)`

Après avoir exécuté ces appels de fonction, la variable 'myname' contiendra le nom de la macro, tandis que 'mainname' contiendra le nom de la macro principale (s'il n'y en a pas, chaîne nulle).

REQUEST `("ID_of_main", "", idstring)`

Retourne dans la variable 'idstring', pour les éléments de bibliothèque placés sur le plan, l'ID défini dans le dialogue Type outil (sinon, chaîne nulle)..

REQUEST `("Name_of_plan", "", name)`

Retouene dans la vaiable le nom du projet courant.

REQUEST `("Story", "", index, storyname)`

Retourne dans les variables 'index' et 'storyname' le numéro et le nom de l'étage courant.

REQUEST `("Home_story", "", index, storyname)`

Retourne dans les variables 'index' et 'storyname' le numéro et le nom de l'étage original.

REQUEST `("Internal_id", "", id)`

Retourne dans la variable 'id' l'ID iterne de l'élément de bibliothèque.

```
REQUEST  ("Linear_dimension", "", formatstr)

REQUEST  ("Angular_dimension", "", formatstr)

REQUEST  ("Angular_length_dimension", "", formatstr)

REQUEST  ("Radial_dimension", "", formatstr)

REQUEST  ("Level_dimension", "", formatstr)

REQUEST  ("Elevation_dimension", "", formatstr)

REQUEST  ("Window_door_dimension", "", formatstr)

REQUEST  ("Sill_height_dimension", "", formatstr)

REQUEST  ("Area_dimension", "", formatstr)
REQUEST  ("CALC_LENGTH_UNIT", "", formatstr)
REQUEST  ("CALC_AREA_UNIT", "", formatstr)
REQUEST  ("CALC_VOLUME_UNIT", "", formatstr)
REQUEST  ("CALC_ANGLE_UNIT", "", formatstr)
```

Ces requêtes vous permettent de prendre vérifier les jeux de format de cote dans les dialogues Options/Facilités/Cotations et Unités de calcul d'ArchiCAD. Elles retournent une chaîne de format qui peut être utilisée comme le premier paramètre de la fonction `STR ()`.

Exemple:

```
format = ""
num = 60.55
REQUEST ("Angular_dimension", "", format)
!"%.2dd"
TEXT2 0, 0, STR (format, num)    !60.55°
```

```
REQUEST  ("Clean_intersections", "", state)
```

Retourne l'état de l'option Raccord murs du menu Options d'ArchiCAD (1=oui, 0=non)

```
REQUEST  ("Zone_category", "", name, code)
```

Retourne, pour les Zones, le nom et la chaîne de code de la catégorie de zone courante.

```
REQUEST  ("Zone_relations", "", catname, code, name, nr
[, catname2, code2, name2, nr2])
```

Retourne dans les variables le nom et le code de la catégorie de zone, le nom et le numéro de la zone dans laquelle se trouve l'élément de bibliothèque qui contient cette requête. Pour les

portes et les fenêtres, le nombre maximum de zones est de 2. La valeur en retour de la requête est le nombre de valeurs obtenues (0 si l'élément de bibliothèque ne se trouve dans aucune zone).

REQUEST **("Zone_colus_area", "", area)**

Retourne dans la variable la surface totale des poteaux placés dans la zone courante. N'a d'effet que sur les marques de zone.

REQUEST **("Custom_auto_label", "", name)**

Retourne dans la variable le nom de l'étiquette automatique personnalisée de l'élément de bibliothèque, ou une chaîne vide s'il n'y en a pas.

REQUEST **("Rgb_of_material", name, r, g, b)**

REQUEST **("Rgb_of_pen", penindex, r, g, b)**

REQUEST **("Pen_of_RGB", "r g b", penindex)**

Comme la fonction REQ(), retourne (en un seul appel) dans les variables spécifiés les valeurs des composantse r, g, b de la matière et du stylo, ou l'index du stylo correspondant aux valeurs r, g, b données.

REQUEST **("Height_of_style", name, height)**

Retourne dans la variable donnée la hauteur du nom de stylo exprimé en mètres.

REQUEST **("Name_of_material", index, name)**

Retourne dans la variable 'name' le nom de la matière identifiée par un index.

REQUEST **("Name_of_fill", index, name)**

Retourne dans la variable 'name' le nom de la hachure identifée par un index.

REQUEST **("Name_of_line_type", index, name)**

Retourne dans la variable 'name' le nom de la ligne identifée par un index..

REQUEST **("Name_of_style", index, name)**

Retourne dans la variable 'name' le nom du style identifé par un index.

Si l'index est inférieur à zéro, il se réfère à une matière, hachure, ligne ou style défini dans le script GDL ou dans le fichier

MASTER_GDL. Un appel de requête avec index = 0 retourne dans la variable le nom de la matière ou de la ligne par défaut. (Chaîne vide pour la hachure et le style.)
La valeur en retour de la requête est le nombre de valeurs obtenues (1 s'il n'y a pas eu d'erreur, 0 pour une erreur d'index non valable).

REQUEST `("window_door_show_dim", " ", show)`

Retourne 1 dans la variable "show" si l'option sélectionnée dans le pop-up Options/Options d'affichage/Portes et Fenêtres est est "Afficher avec cotations", sinon 0. S'utilise pour masquer/afficher les cotations personnalisées selon les Options d'affichage courantes.

REQUEST `("name_of_listed", " ", name)`

Retourne dans la variable "name" le nom de l'élément de bibliothèque associé à l'élément de bibliothèque de type propriété qui contient la requête. Pour des éléments (Murs, Dalles, etc), "name" sera une chaîne vide.

REQUEST `("window_door_zone_relev", " ", out_direction)`

N'a d'effet que pour les Portes et Fenêtres. S'utilise pour compléter une requête `"ZONE_RELATIONS"`. Retourne 1 dans la variable `"out_direction"` si la direction de l'ouverture de Porte/Fenêtre est celle de la première pièce identifiée par la requête `"ZONE_RELATIONS",` retourne 2 si la direction de l'ouverture est celle de la seconde pièce. Retourne également 2 s'il n'y a qu'une pièce et la direction de l'ouverture est vers l'extérieur.

REQUEST `("MATCHING_PROPERTIES", type, name1, name2, ...)`

Si type = 1, retourne dans les variables données les noms d'éléments de bibliothèque associés à des propriétés individuellement, dans les autres cas, les noms d'éléments de bibliothèques associés à des propriétés par critères. Dans le cas d'une étiquette associée, la fonction retourne les propriétés de l'élément auquel l'étiquette est associée.

REQUEST `(extension_name, parameter_string, v1, v2, ...)`

Si la question n'est pas l'une de celles énumérées plus haut, la fonction `REQUEST ()` essaie de l'utiliser comme un nom spécifique à une extension. Si cette extension se trouve dans le dossier Add-Ons, elle sera utilisée pour repérer autant de valeurs qu'il n'y a de noms de variables spéficiés. La chaîne de paramètres est interprétée par l'extension.

IND (MATERIAL, name_string)

IND (FILL, name_string)

IND (LINE_TYPE, name_string)

IND (STYLE, name_string)

Cette fonction retourne l'index courant des attributs de matière, de hachure, de type de ligne et de style. L'usage principal du nombre obtenu est de le transmettre à une macro qui requiert le même attribut que la macro qui l'appelle. Le résultat est négatif pour une définition temporaire et positive pour les définitions globales (voir les commandes *Matières, Motifs de hachures* et *Types de ligne* d'ArchiCAD).

Voir aussi MATERIAL, FILL, LINE_TYPE, STYLE dans le chapitre "Attributs".

C

GDL créé à partir du plan

Enregistrer le plan comme GDL ou Elément de bibliothèque donne les éléments GDL suivants. Vous pouvez utiliser ces documents GDL comme modèles pour vos Eléments de bibliothèque personnalisés. Voir aussi le chapitre "Formes 3D.

Script 3D:

Murs:	`LIN_, XWALL_`
Poteaux:	`CPRISM_`
Poutres:	BEAM
Dalles:	`CPRISM_`
Toitures:	`CSLAB_, CROOF_`
Maillages:	`MASS`
Fenêtres, Portes, Objets, Lampes:	`CALL`

Plans de coupe définis dans le dialogue Options Coupe 3D (activés par la commande Coupe 3D):

`CUTPLANE`

Plans de coupe et polygones générés par les Toitures:

`CUTPOLY` and `CUTPLANE`

Script 2D:

`HOTSPOT2, POLY2_B, LINE2`
`CALL`

D

Mots-clefs pour la vue 3D seulement

ADDX, ADDY, ADDZ
ADD
MULX, MULY, MULZ
MUL
ROTX, ROTY, ROTZ
ROT
XFORM

LIN_
RECT
POLY, POLY_
PLANE, PLANE_
CIRCLE
ARC

BLOCK, BRICK
CYLIND
SPHERE
ELLIPS
CONE
PRISM, PRISM_, CPRISM_, BPRISM_, FPRISM_, SPRISM_
SLAB, SLAB_, CSLAB_
CWALL_, BWALL_, XWALL_
WALLHOLE
CROOF_
ARMC
ARME
ELBOW

```
EXTRUDE
PYRAMID
REVOLVE
RULED
SWEEP
TUBE, TUBEA
COONS
MESH
MASS

LIGHT
PICTURE
TEXT

VERT, TEVE
VECT
EDGE
PGON, PIPG
COOR
BODY
BASE
BINARY

CUTPLANE
CUTSHAPE
CUTPOLY
CUTPOLYA
CUTEND

DEFINE MATERIAL
DEFINE TEXTURE
[SET] MATERIAL
SHADOW
MODEL
```

E

Mots-clefs pour le symbole 2D seulement

ADD2
MUL2
ROT2

HOTSPOT2
LINE2
RECT2
POLY2, POLY2_, POLY2_A, POLY2_B
ARC2
CIRCLE2
SPLINE2, SPLINE2A

PICTURE2
TEXT2
FRAGMENT2

PROJECT2

DEFINE FILL
DEFINE FILLA
DEFINE LINE_TYPE

[SET] FILL
[SET] LINE_TYPE

DRAWING2
DRAWING3

F

Mots-clefs pour l'usage 2D et 3D

DEL
[LET]
RADIUS
RESOL
TOLER
PEN
DEFINE STYLE
[SET] STYLE

G

Mots-clefs pour les scripts non géométriques

Script de Descriptif

DESCRIPTOR
COMPONENT
REF
SURFACE3D
VOLUME3D

POSITION

WALLS
COLUMNS
DOORS
WINDOWS
OBJECTS
PITCHED_ROOFS
HIP_ROOFS
LIGHTS
HATCHES
ROOMS
MESHES

DRAWING
BINARYPROP

Script de liste de valeurs

VALUES

PARAMETERS

LOCK

Script Interface

UI_DIALOG

UI_PAGE

UI_BUTTON

UI_PREV

UI_NEXT

UI_GROUPBOX

UI_SEPARATOR

UI_PICT

UI_STYLE

UI_OUTFIELD

UI_INFIELD

H Mots-clefs communs

Opérateurs

Fonctions

FOR, NEXT

DO, WHILE, ENDWHILE

REPEAT, UNTIL

IF, THEN, ELSE, ENDIF

GOTO

GOSUB

RETURN

END

EXIT

PUT

GET

USE

NSP

CALL, PARAMETERS

PRINT

OPEN

INPUT

OUTPUT

CLOSE

DIM

BREAKPOINT

I Mots-clefs réservés

Les mots-clefs suivants sont périmés, ils existent pour des raisons de compatibilité seulement, ou ne sont pas publiés.

BAS
BOX
FILTER
GDLBIN
LIN
LINE
NOD
NODE
ORIGO
PARS
RECT_
SFLINE
TET
TETRA
TRI
WALL_
VOCA
UI_OK
UI_CANCEL

J

Liste de conversion des anciens noms de variables globales

Il est possible d'utiliser les anciens noms de variables globales, mais nous recommandons l'utilisation des nouveaux noms. Chaque ancienne variable globale correspond à une nouvelle variable globale ayant un nom plus long.

```
A_    GLOB_SCALE
B_    GLOB_HSTORY_ELEV
C_    WALL_THICKNESS
D_    WALL_HEIGHT
E_    WALL_OUTLINE_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
```

K

Liste alphabétique des mots-clefs courants de GDL

ABS (x)

ACS (x)

ADD dx, dy, dz

ADD2 x, y

ADDX dx

ADDY dy

ADDZ dz

AND

ARC r, alpha, beta

ARC2 x, y, r, alpha, beta

ARMC r_1, r_2, l, h, d, alpha

ARME l, r_1, r_2, h, d

ASN (x)

ATN (x)

BASE

BEAM leftmat, rightmat, sidevmat, topmat, bottommat,
height, x_1, x_2, x_3, x_4,
y_1, y_2, y_3, y_4, t,
$mask_1$, $mask_2$, $mask_3$, $mask_4$

BINARY mode [, section]

BINARYPROP

BLOCK a, b, c

BODY status

BPRISM_ topmat, botmat, sidemat,
n, h, radius,
x_1, y_1, $mask_1$, . . . x_n, y_n, $mask_n$

BREAKPOINT expression

BRICK a, b, c

BWALL_ leftmat, rightmat, sidemat,
height, x_1, x_2, x_3, x_4, t, radius,
$mask_1$, $mask_2$, $mask_3$, $mask_4$,
n,
$xbeg_1$, $lower_1$, $xend_1$, $upper_1$,
$framevis_1$,
. . .
$xbeg_n$, $lower_n$, $xend_n$, $upper_n$,
$framevis_n$,
m,
a_1, b_1, c_1, d_1,
. . .
a_m, b_m, c_m, d_m

CALL macro_name_string [parameter_list]

CALL macro_name_string **PARAMETERS**
[$name_1$=$value_1$,
. . . $name_n$=$value_n$]

CEIL (x)

CIRCLE r

CIRCLE2 x, y, r

CLOSE channel

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

CONE h, r_1, r_2, $alpha_1$, $alpha_2$

COONS n, m, mask,
x_{11}, y_{11}, z_{11}, . . . x_{1n}, y_{1n}, z_{1n},
x_{21}, y_{21}, z_{21}, . . . x_{2n}, y_{2n}, z_{2n},
x_{31}, y_{31}, z_{31}, . . . x_{3m}, y_{3m}, z_{3m},
x_{41}, y_{41}, z_{41}, . . . x_{4m}, y_{4m}, z_{4m}

COOR wrap, $vert_1$, $vert_2$, $vert_3$, $vert_4$

COS (x)

CPRISM_ topmat, botmat, sidemat,
n, h, x_1, y_1, $mask_1$, . . . x_n, y_n,
$mask_n$

CROOF_ topmat, botmat, sidemat,
n,
x_b, y_b, x_e, y_e, height, angle,
thickness,
x_1, y_1, $alpha_1$, $mask_1$, . . . x_n, y_n,
$alpha_n$, $mask_n$

CSLAB_ topmat, botmat, sidemat,
n, h, x_1, y_1, z_1, $mask_1$, . . . x_n, y_n,
z_n, $mask_n$

CUTPLANE [x, y, z [, side]]
statements
CUTEND

CUTPLANE angle
statements
CUTEND

CUTPOLY n, status, d,
x_1, y_1, . . . x_n, y_n,
[, x, y, z]
statements
CUTEND

CUTPOLYA n, status, d,
x_1, y_1, $mask_1$, . . . x_n, y_n, $mask_n$
[, x, y, z]
statements
CUTEND

CUTSHAPE d
statements
CUTEND

CWALL_ leftmat, rightmat, sidemat,
height, x_1, x_2, x_3, x_4, t,
$mask_1$, $mask_2$, $mask_3$, $mask_4$,
n,
$xbeg_1$, $lower_1$, $xend_1$, $upper_1$,
$framevis_1$,
. . .
$xbeg_n$, $lower_n$, $xend_n$, $upper_n$,
$framevis_n$,
m,
a_1, b_1, c_1, d_1,
. . .
a_m, b_m, c_m, d_m

CYLIND h, r

DATABASE_SET set_name [, desc_name, comp_name, unit_name, key_name, crit_name, listset_name]

DEFINE FILL name pat_1, pat_2, pat_3, pat_4, pat_5, pat_6, pat_7, pat_8, spacing, angle, n, $freq_1$, dir_1, $offsetx_1$, $offsety_1$, m_1, len_{11}, . . . len_{m1}, . . . $freq_n$, dir_n, $offsetx_n$, $offsety_n$, m_n, len_{n1}, . . . len_{nm}

DEFINE FILLA name pat_1, pat_2, pat_3, pat_4, pat_5, pat_6, pat_7, pat_8, spacingx, spacingy, angle, n, $freq_1$, $doffset_1$, dir_1, $offsetx_1$, $offsety_1$, m_1, len_{11}, . . . len_{m1}, . . . $freq_n$, $doffset_n$,dir_n, $offsetx_n$, $offsety_n$, m_n, len_{n1}, . . . len_{nm}

DEFINE LINE_TYPE name spacing, n, len_1, . . . len_n

DEFINE MATERIAL name type, m_1, m_2, . . . m_n

DEFINE STYLE name font_family, size, anchor, facecode

DEFINE STYLE name PLOTMAKER, size, anchor, slant

DEFINE STYLE name PLOTTER, size, anchor, slant

DEFINE TEXTURE name, filename, x, y, mask, angle

DEL n

DEL TOP

DESCRIPTOR string

DIM var1 **[**dim_1**]**, var2 **[**dim_1**][**dim_2**]**, . . .

DO
statements
WHILE condition

DRAWINDEX nr

DRAWING

DRAWING2

DRAWING3

EDGE vert1, vert2, pgon1, pgon2, status

ELBOW r1, alpha, r2

ELLIPS h, r

END

EXIT

EXOR

EXP (x)

EXTRUDE n, dx, dy, dz, mask, x_1, y_1, s_1, . . . x_n, y_n, s_n

FILL index

FILL name_string

FOR varnam = initial_value **TO** end_value [**STEP** step_value]

FPRISM_ topmat, botmat, sidemat, hillmat, n, thickness, angle, hill_height, x_1, y_1 , $mask_1$, . . . x_n, y_n, $mask_n$

FRA (x)

FRAGMENT2 ALL, use_current_attributes_flag

FRAGMENT2 plane_index, use_current_attributes_flag

GET (n)

GOSUB label

GOTO label

HOTSPOT2 x, y

IF cond **GOSUB** label

IF cond **GOTO** label

IF cond **THEN** label

IF condition **THEN** statement [**ELSE** statement]

IF condition **THEN**

statements

[**ELSE**

statements]

ENDIF

IND (FILL, name_string)

IND (LINE_TYPE, name_string)

IND (MATERIAL, name_string)

IND (STYLE, name_string)

IND (TEXTURE, name_string)

INPUT (channel, recordID, fieldID, var1 [, var2, . . .]

INT (x)

LET

LGT (x)

LIGHT red, green, blue, shadow, radius, alpha, beta, angfalloff, $dist_1$, $dist_2$, distfalloff

LIN_ x_1, y_1, z_1, x_2, y_2, z_2

LINE_TYPE index

LINE_TYPE name_string

LINE2 x_1, y_1, x_2, y_2

LOCK $name_1$ [, $name_2$, ..., $name_n$]

LOG (x)

MASS topmat, botmat, sidemat, n, m, mask, h,
x_1, y_1, z_1, s_1,
. . .
x_n, y_n, z_n, s_n,
x_{n+1}, y_{n+1}, z_{n+1}, s_{n+1},
. . .
x_{n+m}, y_{n+m}, z_{n+m}, s_{n+m}

MATERIAL index

MATERIAL name_string

MAX (x_1,x_2, . . . x_n)

MESH a, b, m, n, mask,
z_{11}, z_{12}, . . . z_{1m},
z_{21}, z_{22}, . . . z_{2m},
. . .
z_{n1}, z_{n2}, . . . z_{nm}

MIN (x_1,x_2, . . . x_n)

MOD

MODEL SOLID

MODEL SURFACE

MODEL WIRE

MUL mx, my, mz

MUL2 x, y

MULX mx

MULY my

MULZ mz

NEXT varnam

NOT (x)

NSP

NTR ( )

OPEN (filter, filename, paramstring)

OR

OUTPUT channel, recordID, fieldID, expr1 [, expr2, . . .]

PARAMETERS $name_1$ = val_1 [, $name_2$ = val_2, …, $name_n$ = val_n]

PEN n

PGON n, ivect, status, $edge_1$, $edge_2$, . . . $edge_n$

PI

PICTURE filename, a, b, mask

PICTURE2 expression, a, b, mask

PIPG filename, a, b, mask, n, ivect, status, $edge_1$, $edge_2$, . . . $edge_n$

PLANE n, x_1, y_1, z_1, . . . x_n, y_n, z_n

PLANE_ n, x_1, y_1, z_1, $mask_1$, . . . x_n, y_n, z_n, $mask_n$

POLY n, x_1, y_1, . . . xn, yn

POLY_ n, x_1, y_1, $mask_1$, . . . x_n, y_n, $mask_n$

POLY2 n, framefill, x_1, y_1, . . . x_n, y_n

POLY2_ n, framefill, x_1, y_1, $mask_1$, . . . x_n, y_n, $mask_n$

POLY2_A n, framefill, fillpen, x_1, y_1, $mask_1$, . . . x_n, y_n, $mask_n$

POLY2_B n, framefill, fillpen, fillbkgdpen, x_1, y_1, $mask_1$, . . . x_n, y_n, $mask_n$

POSITION position_keyword

PRINT [expression] [, expression] . . .

PRISM n, h, x_1, y_1, . . . x_n, y_n

PRISM_ n, h, x_1, y_1, $mask_1$, . . . x_n, y_n, $mask_n$

PROJECT2 projcode, angle, method

PUT expression [, expression] . . .

PYRAMID n, h, mask, x_1, y_1, s_1, . . . x_n, y_n, s_n

RADIUS rmin, rmax

RECT a, b

RECT2 x_1, y_1, x_2, y_2

REF COMPONENT code [, num_expr [, keycode]]

REF DESCRIPTOR code [, keycode]

REPEAT
statements
UNTIL condition

REQ (parameter_string)

REQUEST (question_name, name | index, var1 [, var2, . . .])

RESOL n

RETURN

REVOLVE n, alpha, mask, x_1, y_1, s_1, . . . x_n, y_n, s_n

RND (x)

ROT x, y, z, alpha

ROT2 alphaz

ROTX alphax

ROTY alphay

ROTZ alphaz

RULED n, mask,
u_1, v_1, s_1, . . . u_n, v_n, s_n,
x_1, y_1, z_1, . . . x_n, y_n, z_n

SET FILL index

SET FILL name_string

SET LINE_TYPE index

SET LINE_TYPE name_string

SET MATERIAL ndex

SET MATERIAL name_string

SET STYLE index

SET STYLE name_string

SGN (x)

SHADOW AUTO

SHADOW OFF

SHADOW ON

SIN (x)

SLAB n, h, x_1, y_1, z_1, . . . x_n, y_n, z_n

SLAB_ n, h, x_1, y_1, z_1, $mask_1$, . . . x_n, y_n, z_n, $mask_n$

SPHERE r

SPLINE2 n, status, x_1, y_1, $angle_1$, . . . x_n, y_n, $angle_n$

SPLINE2_A n, status,
x_1, y_1, $angle_1$, $lenPrev_1$, $lenNext_1$,
. . .
x_n, y_n, $angle_n$, $lenPrev_n$, $lenNext_n$

SPLIT (string, format, var_1 [, var_2, . . . var_n])

SPRISM_ topmat, botmat, sidemat,
n, x_b,y_b, x_e, y_e, h, angle,
x_1, y_1, $mask_1$, . . . x_n, y_n, $mask_n$

SQR (x)

STR (numeric_expression, len, frac)

STR (formatstring, numeric_expression)

STRLEN (string_exp)

STRSTR (string_exp1, string_exp2)

STRSUB (string_exp, begpos, numchars)

STW (string_expression)

STYLE index

STYLE name_string

SURFACE3D ( )

SWEEP n, m, alpha, scale, mask,
u_1, v_1, s_1, . . . u_n, v_n, s_n,
x_1, y_1, z_1, . . . x_m, y_m, z_m

TAN (x)

TEVE x, y, z, u, v

TEXT d, 0, expression

TEXT2 x, y, string

TOLER d

TUBE n, m, mask,
u_1, w_1, s_1, . . . u_n, w_n, s_n,
x_1, y_1, z_1, $angle_1$, . . . x_m, y_m, z_m,
$angle_m$

TUBEA n, m, mask,
u_1, w_1, s_1, . . . u_n, w_n, s_n,
x_1, y_1, z_1, . . . x_m, y_m, z_m

UI_BUTTON type, text, x, y, width, height

UI_DIALOG title [, sizex, sizey]

UI_GROUPBOX text, x, y, width, height

UI_INFIELD "name", x, y, width, height [,
versionFlag, pictName, nrImages,
nrRows, cellX, cellY, imageX, imageY,
$imageExp_1$, $text_1$,
…,
$imageExp_n$, $text_n$]

UI_NEXT

UI_OUTFIELD text, x, y, width, height

UI_PAGE pagenum

UI_PICT expression, x, y [, width, height]

UI_PREV

UI_SEPARATOR x_1, y_1, x_2, y_2

UI_STYLE fontsize, facecode

USE (n)

VALUES name, val_1 [, val_2, . . . val_n]

VECT x, y, z

VERT x, y, z

VOLUME3D ( )

WALLHOLE n, status,
x_1, y_1, $mask_1$, . . . x_n, y_n, $mask_n$
[, x, y, z]

WHILE condition **DO**
statements
ENDWHILE

XFORM a_{11}, a_{12}, a_{13}, a_{14},
a_{21}, a_{22}, a_{23}, a_{24},
a_{31}, a_{32}, a_{33}, a_{34}

XWALL_ leftmat, rightmat, sidevmat,
sidehmat,
height, x_1, x_2, x_3, x_4,
y_1, y_2, y_3, y_4,
t, radius,
reserved1, reserved2,
$mask_1$, $mask_2$, $mask_3$, $mask_4$,
n,
$xbeg_1$, $lower_1$, $xend_1$, $upper_1$,
$framevis_1$,
. . .
$xbeg_n$, $lower_n$, $xend_n$, $upper_n$,
$framevis_n$,
m,
a_1, b_1, c_1, d_1,
. . .
a_m, b_m, c_m, d_m

L

L'extension TEXT

Ceci est un échantillon d'extension démontrant les possibilités des opérations d'écriture et de lecture de fichiers. D'autres extensions sont disponibles auprès de Graphisoft ou d'autres éditeurs.

L'extension interprète les chaînes de la liste de paramètres des commandes **OPEN**, **INPUT**, **OUTPUT**.

L'extension texte considère soit qu'il existe un dossier intitulé " ArchiCAD Data Folder " auprès de l'application ArchiCAD soit qu'un chemin complet a été spécifié pour le fichier. Si ce dossier n'existe pas, l'extension va le créer, Le dossier peut contenir d'autres dossiers dans lesquels l'extension va rechercher les fichiers existants. Cette extension permet de lire et d'écrire des fichiers de type TEXT.

1. channel = **OPEN** (filter, filename, paramstring)

filter : spécifique à l'extension, dans le cas présent "TEXT"

filename: le nom du fichier à ouvrir

paramstring: contains the specific contient les caractères de séparation spécifiques à l'extension, ainsi que le mode d'ouverture

Ouvre le fichier. Si le fichier dans lequel vous souhaitez écrire n'existe pas, l'extension crée le fichier. Si le fichier à lire n'existe pas, un message d'erreur est affiché.

La valeur en retour est un entier positif qui va identifier le fichier spécifique. Cette valeur servira de numéro de référence du fichier.

Le paramstring peut contenir:

- **SEPARATOR** = après le mot-clef entre guillemets simples (') vous pouvez affecter un caractère à utiliser dans le fichier texte (en écriture et en lecture) pour séparer les colonnes.

Les cas spéciaux sont la tabulation ('\t') et la nouvelle ligne('\n').

- **MODE** = après le mot-clef le mode d'ouverture doit être défini. Il n'y a que trois modes possibles:

  RO (lecture seulement)

  WA (écriture seulement, ajouter à la fin du fichier)

WO écriture seulement, remplacer) toutes les données précédemment conservées dans le fichier seront perdues!

Un fichier ne peut être ouvert à la fois en lecture et en écriture.

- **DIALOG**
  En présence de ce mot-clef, un dialogue s'affiche à l'écran. Vous pouvez y introduire un nom de fichier.
- **FULLPATH**
  En présence de ce mot-clef, le nom du fichier sera interprété comme un chemin complet.

Mettez toujours une virgule (,) entre les arguments de SEPARATOR et de MODE.

Si vous utilisez des mots-clefs inexistants, si les caractères de séparations définis ne sont pas corrects ou s'il n'y a rien dans la chaîne de caractères, l'extension utilise par défaut SEPARATOR = '\t', MODE = RO.

Exemple:

```
ch1 = OPEN ("TEXT", "file1", "SEPARATOR=';',
            MODE=RO")
ch2 = OPEN ("TEXT", "file2", "")
ch3 = OPEN ("TEXT", "file3", "SEPARATOR='\n',
            MODE=WO")
```

2. n = **INPUT** (channel, recordID, fieldID, var1 [, var2, ...])

recordID: le nombre de lignes (numérique ou chaîne)

fieldID: le nombre de colonnes dans la ligne donnée

La commande lit à partir de la position de départ donnée dans le fichier identifié par 'channel', autant de valeurs qu'il n'y a de paramètres affectés. Il doit y avoir au moins une valeur dans la liste de paramètre. La fonction met les valeurs lues dans les paramètres de la séquence. Les valeurs peuvent être de type numérique ou caractère quel que soit le type de paramètre défini. A la place des valeurs dans le fichier, vous pouvez également avoir des expressions de type numérique ou caractère, chacune pouvant contenir des constantes numériques ou de type caractère, ainsi que des appels de fonction mathématique de GDL. Ces expressions sont évaluées pendant la procédure d'entrée.

Exemple: dans le fichier texte (le séparateur est ','):

```
1.23,1'2","aaa",2 * SIN(45) + 1,SQR (3)
```

La valeur en retour est le nombre de valeurs lues, en cas de fin de fichier rencontré, c'est -1.

Les nombres de lignes et de colonnes doivent être des entiers positifs, sinon un message d'erreur est affiché.

Si les nombres de lignes ou de colonnes sont incorrects, la saisie des données ne sera pas effectuée. (n = 0)

Si la ligne et la colonne peuvent être identifiées, il sera affecté à partir de la position de départ donnée autant de valeurs qu'il n'y a de paramètres, ou, s'il y a plus de paramètres que de valeurs, les paramètres auxquels aucune valeur ne correspond sera mis à zéro.

Pour les colonnes vides (par exemple, s'il n'y a rien entre les caractères de séparation), les paramètres seront mis à zéro.

Exemple:

```
 nr = INPUT (ch1, 1, 1, v1, v2, v3)
             ! input of three values from the first
column              ! of the first row
 PRINT nr, v1, v2, v3
```

3. **OUTPUT** channel, recordID, fieldID, expr1 [, expr2, ...]

recordID : s'il est positif, les valeurs de sortie seront suivies par une nouvelle ligne

fieldID : sans rôle, valeur non utilisée

Ecrit, dans le fichier défini par la valeur 'channel' et à partir de la position donnée, autant de valeurs qu'il n'y a d'expressions définies. Il doit y avoir au moins une expression. Les types des valeurs écrites sont les mêmes que ceux des expressions.

Pour l'extension texte, `OUTPUT` va placer (ajouter ou remplacer, en fonction du mode d'ouverture) les expressions données dans des positions consécutives en utilisant les caractères de séparation définis en ouvrant le fichier. Dans ce cas, la position donnée n'est pas interprétée.

La valeur `recordID` est utilisée pour définir les nouvelles lignes dans la sortie.

Si `recordID` est positif, les valeurs écrites seront suivies par une nouvelle ligne, sinon la dernière valeur sera suivie par un caractère de séparation.

Exemple:

```
string = "Date: 19.01.1996"
a = 1.5
OUTPUT ch2, 1, 0, string
  ! string followed by a new row
OUTPUT ch2, 0, 0, a, a + 1, a + 2
  ! separator character after a + 2 without new row
```

4. **CLOSE** channel

Ferme le fichier identifié par la valeur 'channel'.

Exemple:

Un objet GDL qui va simplement copier le contenu du fichier "f1" dams les fichiers "f2" et "f3" et mettre toutes les valeurs tabulées dans "f1" dans une ligne nouvelle dans "f2" et "f3".

```
ch1 = OPEN ("TEXT", "f1", "MODE=ro")
ch2 = OPEN ("TEXT", "f2", "SEPARATOR='\n', MODE=wo")
ch3 = OPEN ("TEXT", "f3", "SEPARATOR='\n', MODE=wo")
i = 1
1:
    n = INPUT (ch1, i, 1, var1, var2, var3, var4)
    IF n <> -1 THEN
         OUTPUT ch2, 1, 0, var1, var2, var3, var4
         OUTPUT ch3, 1, 0, var1, var2, var3, var4
         i = i + 1
         GOTO 1
    ELSE
         GOTO 2
    ENDIF
2:
    CLOSE ch1
    CLOSE ch2
    CLOSE ch3
END
```

Index

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W

X

Table des matières