GDL开发基础 V2.0

AC二次开发学习交流QQ群： 391765618

夏 雨

本次GDL开发技术培训的目标

- 一部分同事对GDL有一个大致的了解，以便以后的工作中与GDL开发人员交流时更加顺畅
- 一部分对GDL开发有浓厚兴趣的同事，争取达到能够自己动手开发一些简单的GDL程序

ArchiCAD二次开发学习线路图

- 鉴于ARchiCAD的二次开发是以ARchiCAD为平台，以其自带的GDL语言、API以及C++语言为开发手段，ARchiCAD19与20都是以VS 2010为开发工具，难度较大，为了加快对ARchiCAD二次开发的掌握，提出以下学习线路图以供参考：
  - 学习掌握ARchiCAD的基本应用和操作。可以看一些相关的培训视频，也可以找适当的操作教程来学习。关键是要动手，一边学习一边动手操作。
  - 学习GDL的开发。可以参考学习一下《GDL基础培训V2.0》；曾旭东、谭洁编写的《GDL参数化程序语言设计》以及陈磊老师相关的GDL基础培训。“ARchiCAD二次开发与应用”QQ群391765618中有相关资料。
  - 学习C++相关的编程知识。建议学习《C++primer》。如果基础不是太好，可以学习其他更适合入门的教材，比如《C++ primer plus》。
  - 学习ARchiCAD API开发相关技术。可供学习的材料主要是ARchiCAD网站上的东西，包括开发文档。学习时可以先从一些案例开始学习。
- 大致的学习线路我认为这样比较合理，但是学习的过程中需要不断的实践，比如一些GDL的开发等等。可以按上面的线路图灵活运用，关键是要由易到难，循序渐进，边学边练，多沟通多交流,要有助人为乐的奉献和共享精神。
- 可能需要预备一些相关的知识，比如《线性代数》、《计算机图形学》、几何造型等方面的相关知识。
- 开发的过程中多向专业的工程师学习，自己有时间也可以补充一些与建筑相关的行业知识。
- 学习网络资源:
  - http://archicad-talk.graphisoft.com GDL和API开发碰到的问题这里很多都提到
  - http://archicadapi.graphisoft.com/
  - https://en.wikipedia.org/wiki/ArchiCAD

“Hello,GDL程序”(案例1)

- 在3D脚本界面窗口中输入如下代码

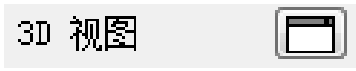
  Text 0.002,0, “Hello，AC的粉丝”

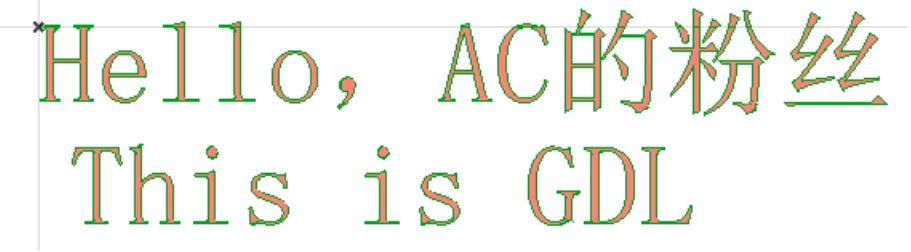
  ADDx 0.002

  ADDy -0.008

  Text 0.003,0, “This is GDL”

  DEL 2

- 点击“3D视图”窗口  结果如上图所示



TEXT

TEXT d, 0, expression

A 3D representation of the value of a string or numeric type expression in the current style.

See the [SET] STYLE command and the DEFINE STYLE command.

d: thickness of the characters in meters.

In the current version of GDL, the second parameter is always zero.

Note: For compatibility with the 2D GDL script, character heights are always interpreted in millimeters in DEFINE STYLE statements.

GDL程序开发需要具备的基本素质

- 对软件开发有浓厚的兴趣（这是最重要的第一条）
- 敢于挑战自我，能承受压力
- 具备比较强的自学能力，最好是英语比较好
- 有一些灵气或者写软件的天赋
- 基本的空间几何想象力和数学知识
- 最好具备基本的线性代数知识
- 学习一下图形学方面的知识会有帮助
- 具备一些基本的数据结构知识
- 掌握一些基本的算法

GDL概述

GDL是什么？

- GDL是Geometric Description Language的缩写（几何描述语言），1982年问世，其实际上就是一种用来描述几何造型的语言。
- GDL是一种参数化程序设计语言，是智能化参数驱动构件的技术基础。其语法与BASIC类似。
- GDL是一种脚本语言，运行时需要一个解析器进行解析后才能运行。
- GDL是 ArchiCAD核心的高级技术，目前是通过ArchiCAD的内置解析器解析后运行。
- 以界面友好的参数程序环境描述2D和3D的元素
- GDL对象保存在外部的图库中；对象在ArchiCAD中也称为库部件。每个库部件包含若干描述不同目的的文本脚本，包括2D符号，3D模型和其他可输出的描述。

GDL能做什么？

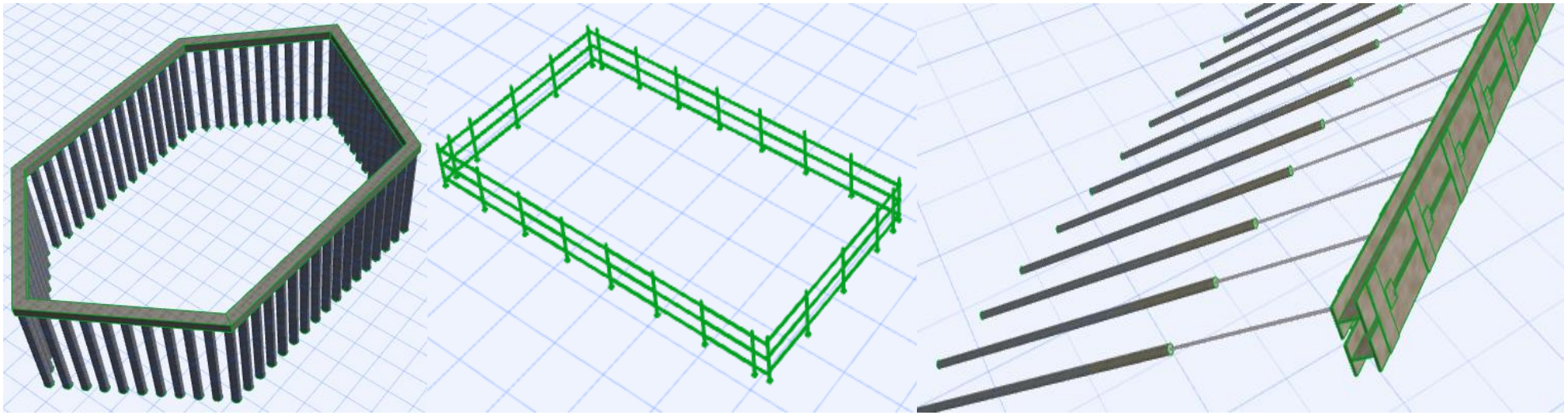
- GDL在ArchiCAD与用户之间打开了一扇交流对话的窗口。ArchiCAD没有提供的3D模型表达，用户都可以通过GDL来实现。
- 实际工程中我们遇到的2D/3D模型，几乎都可以用GDL来创建。
- ArchiCAD的智能部件，都是由GDL创建的。门，窗，天窗，家具，灯具，区域，楼梯，栏杆等均由GDL定义。还有：墙、屋顶、楼板的附件也可以由GDL定义。
- 参数化尺寸驱动的智能部件，在2D，3D状态可通过热点进行编辑，更加直观的进行部件设计。

GDL能做什么？----对于设计师

- 设计师通过对GDL的编写，训练逻辑思维，设计细节更趋于合理
- 模拟设计细部，对构造设计更加合理化。
- 提高设计精度，图纸质量，更加精准的预算投资。
- 随着部件的积累，提高设计、出图效率。

GDL能做什么？----对于开发商

- 内部管理有序、有实力的开发商都有自己标准的工程做法，如果将部品部件等做成GDL，将更加便于标准化、算量精准化。
- 近几年兴起的装配式钢结构、装配式预制件混凝土结构，模块化施工，GDL更加有利于装配式设计施工技术的应用。
- GDL是实现虚拟建造的有力工具之一。

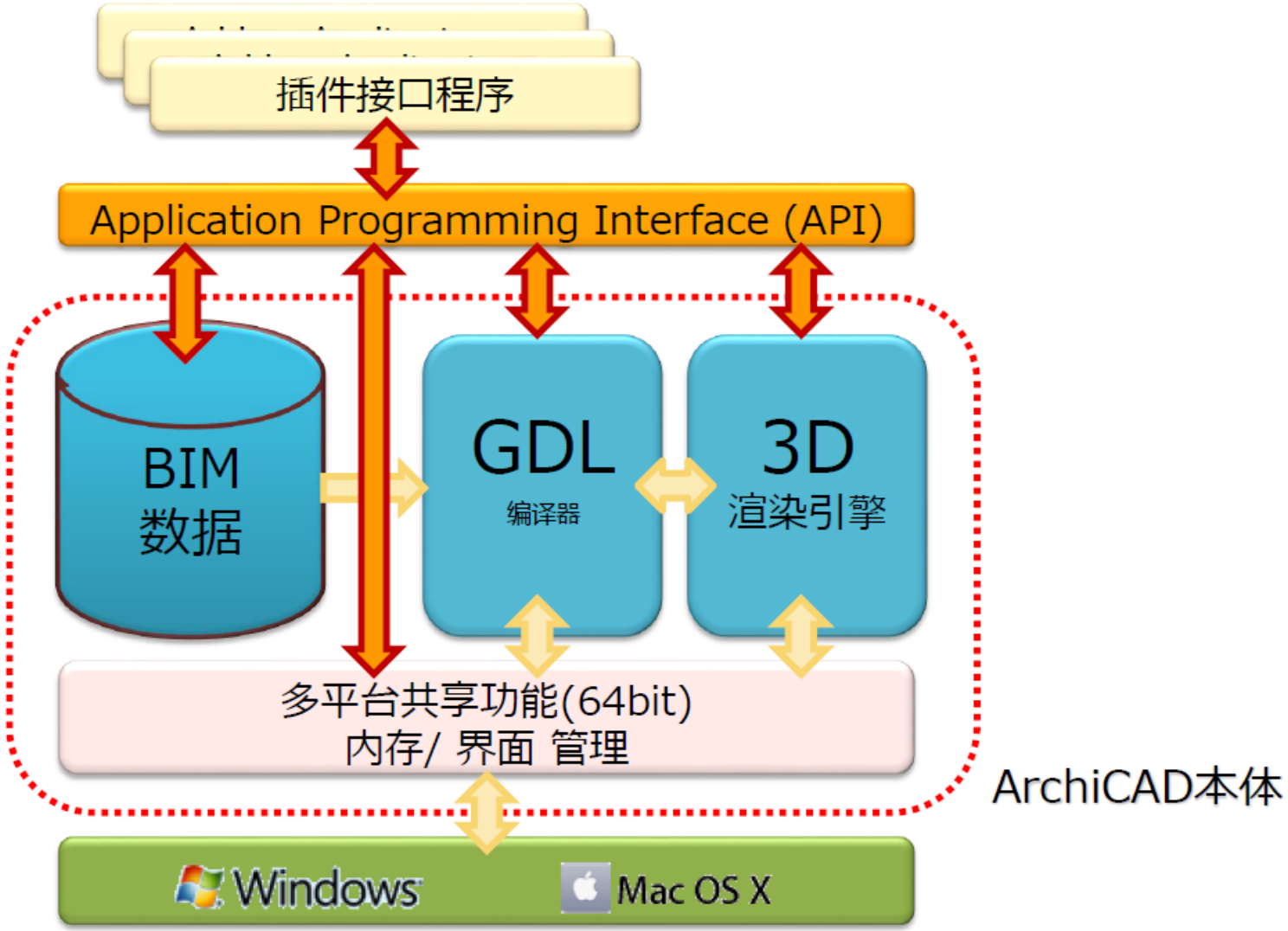


GDL能做什么？----对于设备厂商

- 把设备做成标准的GDL，发布在网上免费下载，有利于设计施工单位实现三维建模，用户多了就更加有利于其商品的推广。

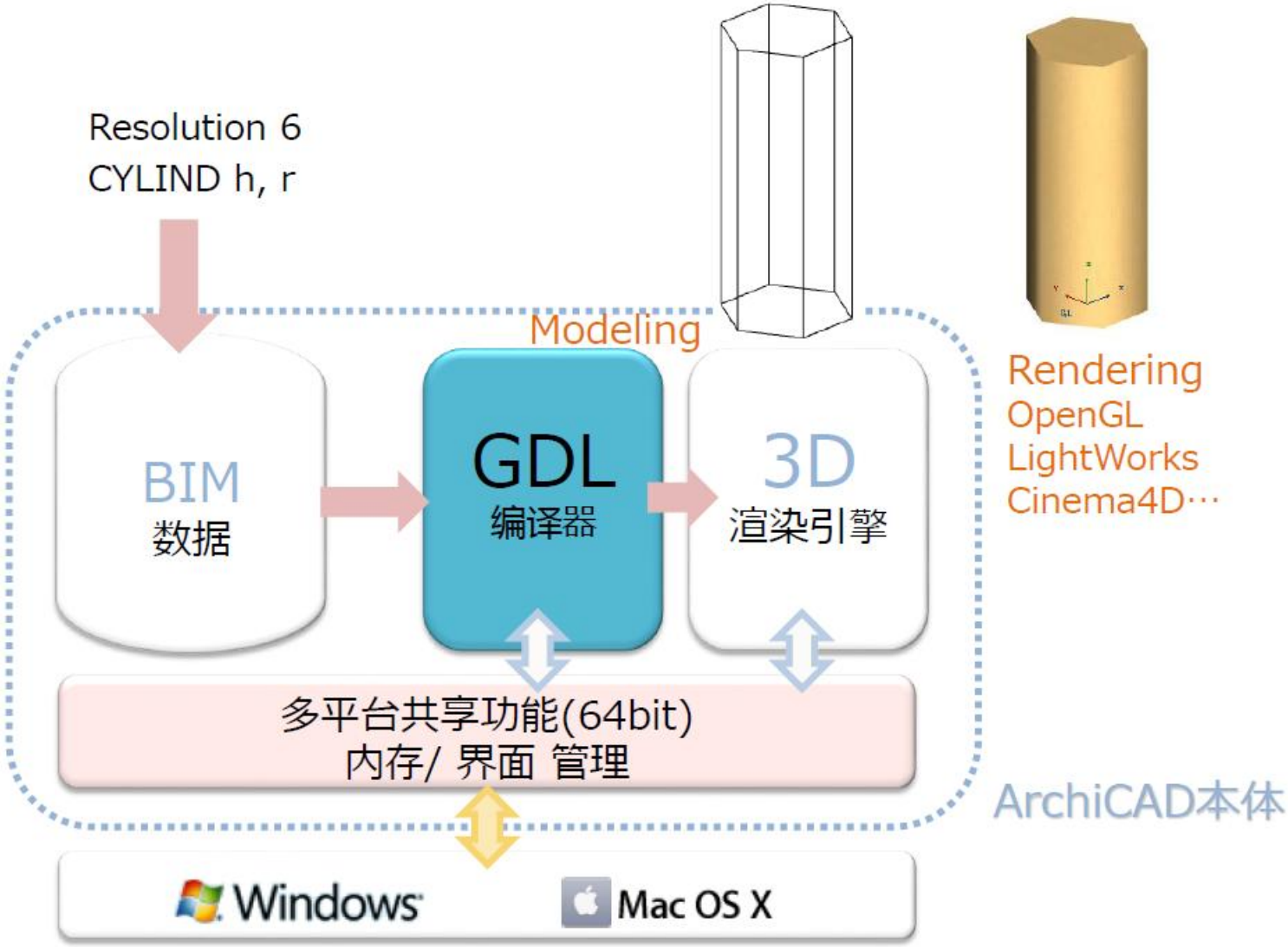
GDL与ArchiCAD的关系

- ArchiCAD的基本构造



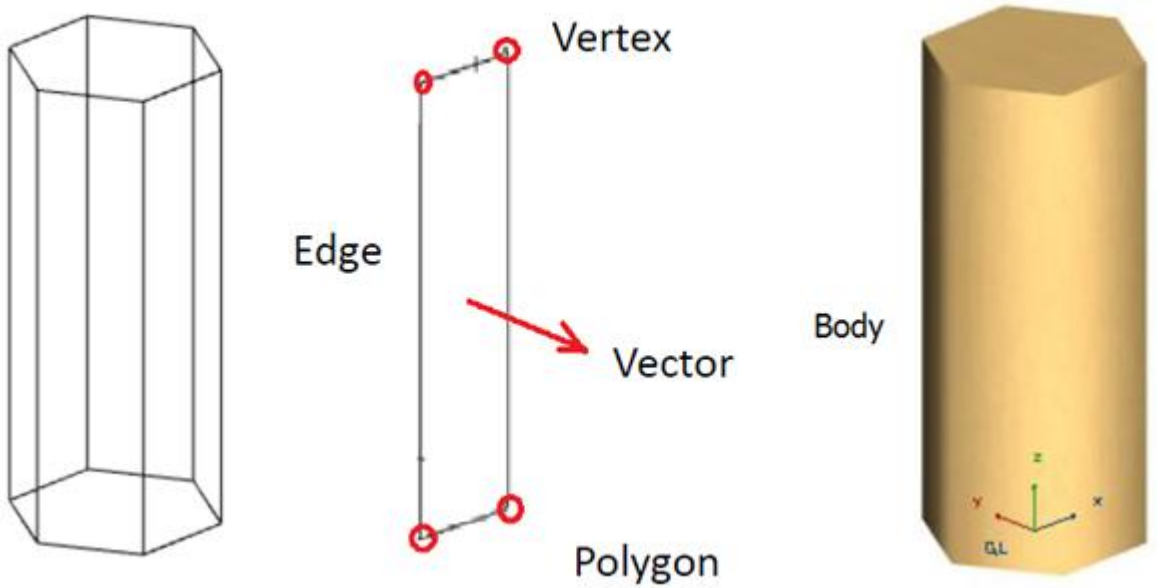
GDL与ArchiCAD的关系

- GDL Geometric Description Language –几何程序设计语言



边界表示法的基本概念

- 边界表示法（B-Rep）是实体造型中以物体的边界为基础来定义和描述几何形体，并能给出完整和显示的界面描述的造型方法。
  - 顶点(Vertex)
  - 法线(Normal Vector)
  - 边(Edge)
  - 面(Polygon)
  - 体(Body)



CYLIND指令程序(案例2)

- 在3D脚本界面窗口中输入如下代码

```
RESOL  7
h = 1.5 : r  = 0.8
Cylind h,r
```

- 点击“3D视图”窗口观看结果

RESOL

RESOL n

Sets smoothness for cylindrical elements and arcs in polylines. Circles are converted to regular polygons having n sides. Arc conversion is proportional to this.

After a RESOL statement, any previous RADIUS and TOLER statements lose their effect.

Restriction of parameters: n >= 3

Default: RESOL 36

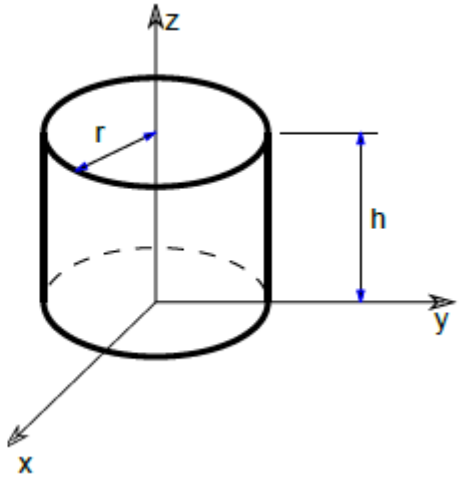
CYLIND

CYLIND h, r

Right cylinder, coaxial with the z-axis with a height of h and a radius of r.

If h=0, a circle is generated in the x-y plane.

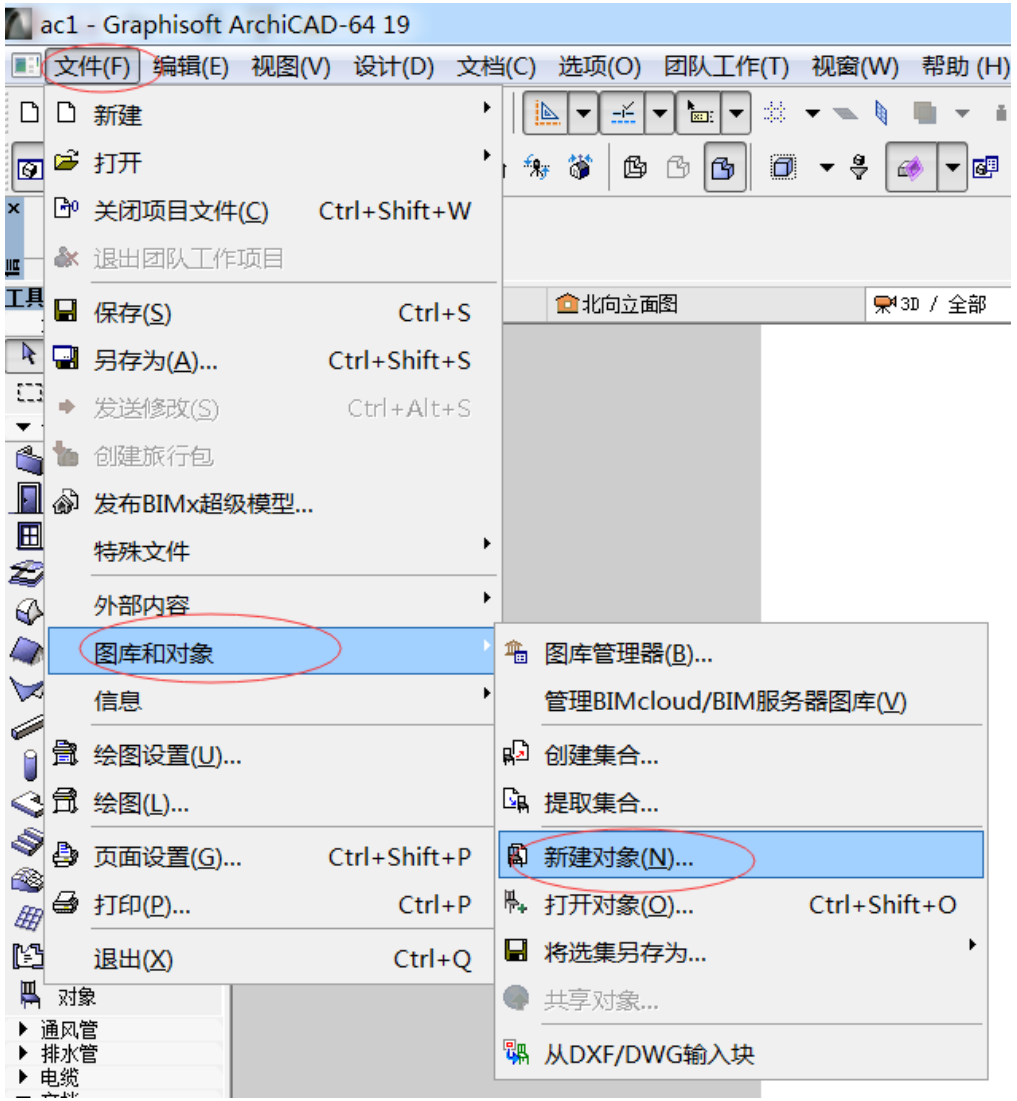
If r=0, a line is generated along the z axis.

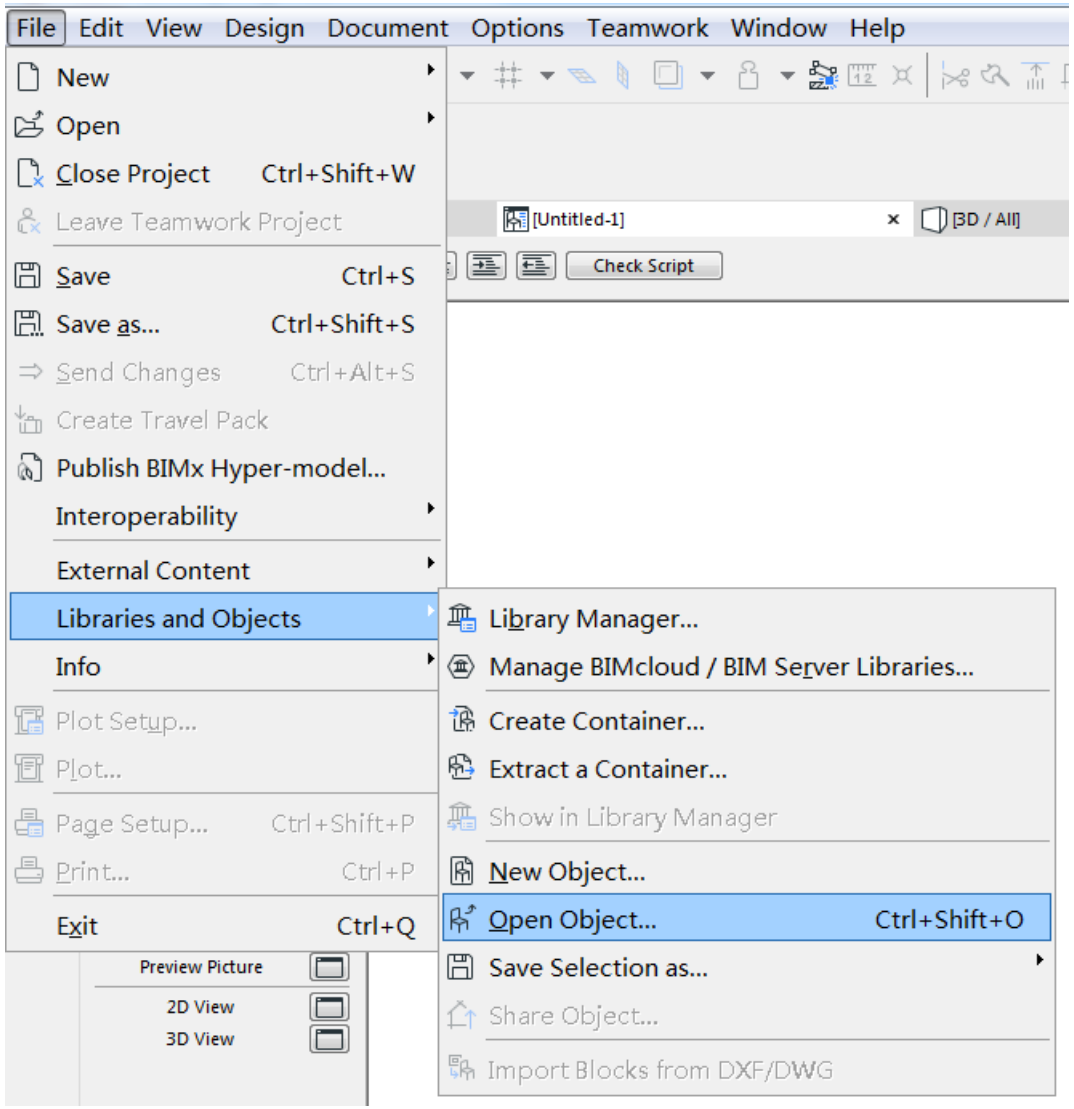


ArchiCAD中GDL的开发环境

（相关细节请参考AC帮助文档GDL Object Editor章节）

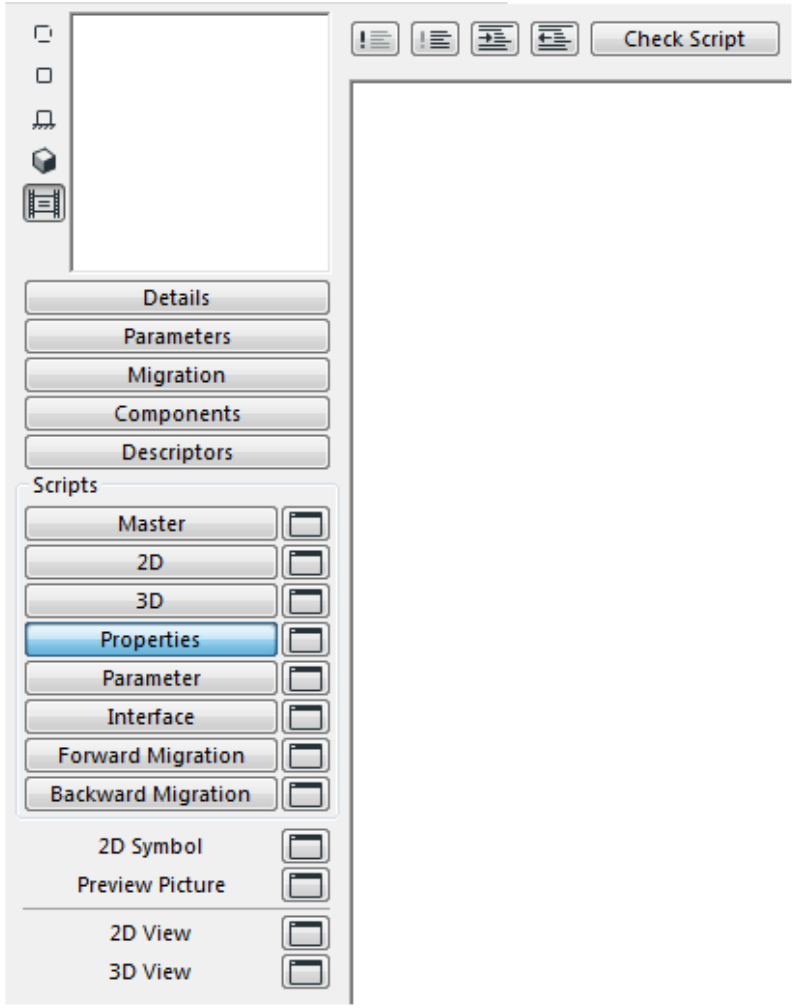
新建/打开一个gdl对象



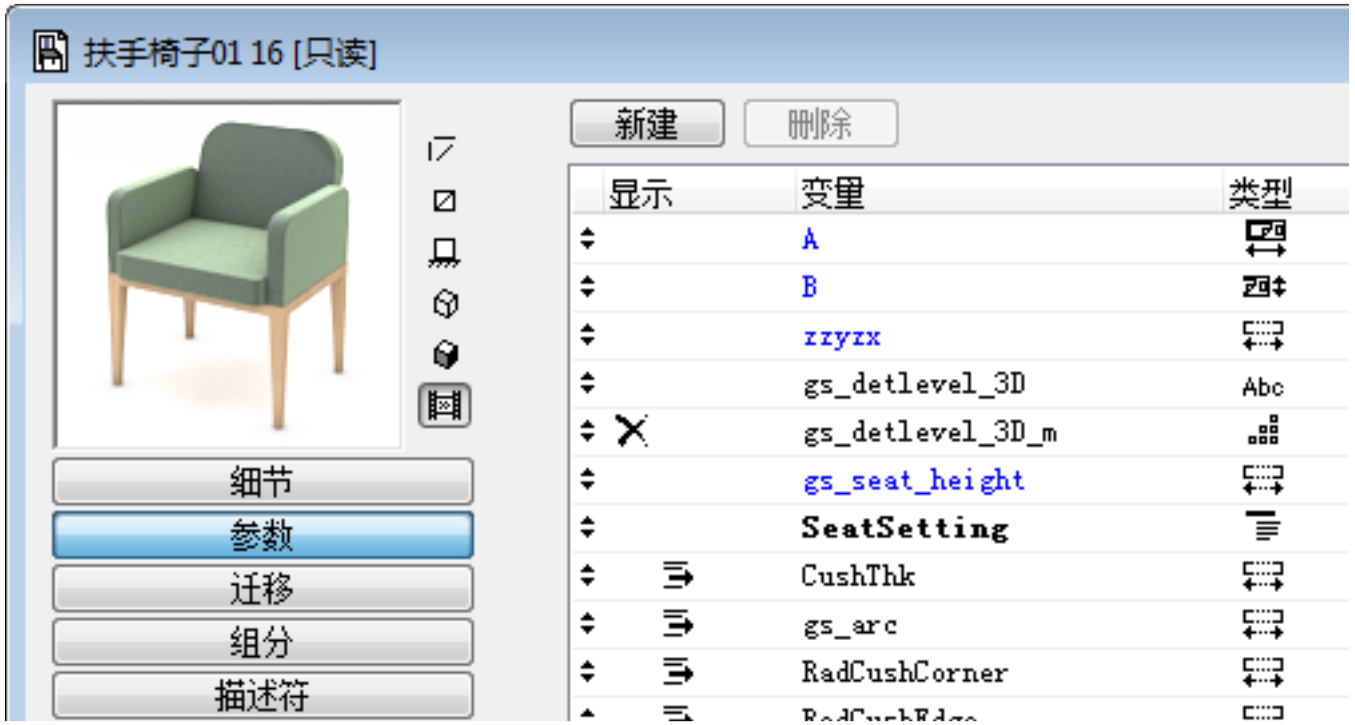


GDL编辑器英汉对照


检查脚本
细节
参数
迁移
组分
描述符
脚本
样板
2D
3D
属性
参数
界面
向前迁移
向后迁移
2D 符号
预览图片
2D视图
3D 视图


Check Script
Details
Parameters
Migration
Components
Descriptors
Scripts
Master
2D
3D
Properties
Parameter
Interface
Forward Migration
Backward Migration
2D Symbol
Preview Picture
2D View
3D View

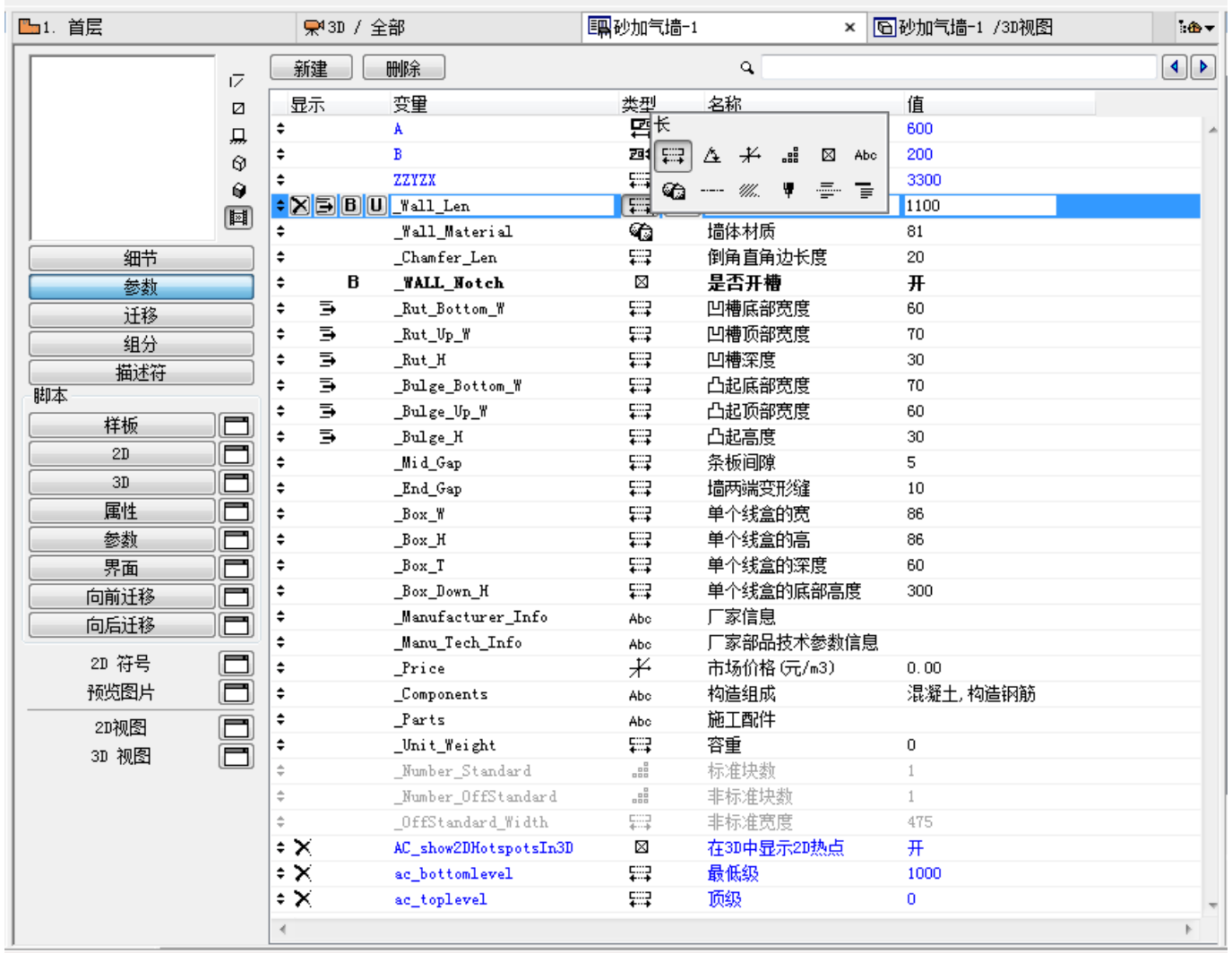
对象预览图片



预览图片窗口可以六种格式中的一种进行预览。

- 点击窗口下面按钮中的一个，选择预览图片的显示选项：2D 符号（图形）、2D 完全视图、立面图、消隐线、着色和渲染预览。
- 使用立面图、消隐线或阴影显示选项，您可点击预览窗口以 45 度角为单位旋转图像。

参数



1、代码中需要与用户进行交互的参数都在这里定义。

2、通过点击“新建”按钮新建参数。

3、输入“变量”名称（需要符合gdl的命名规则），变量名将在脚本中使用。

4、选择想要的参数类型，参数类型包括:长度，角度，实数（指定小数表示的非量纲值），整数，布尔，文本（最多255个字符）、材质，线型，填充，画笔，分割线和标题。这些类型都可以数组。

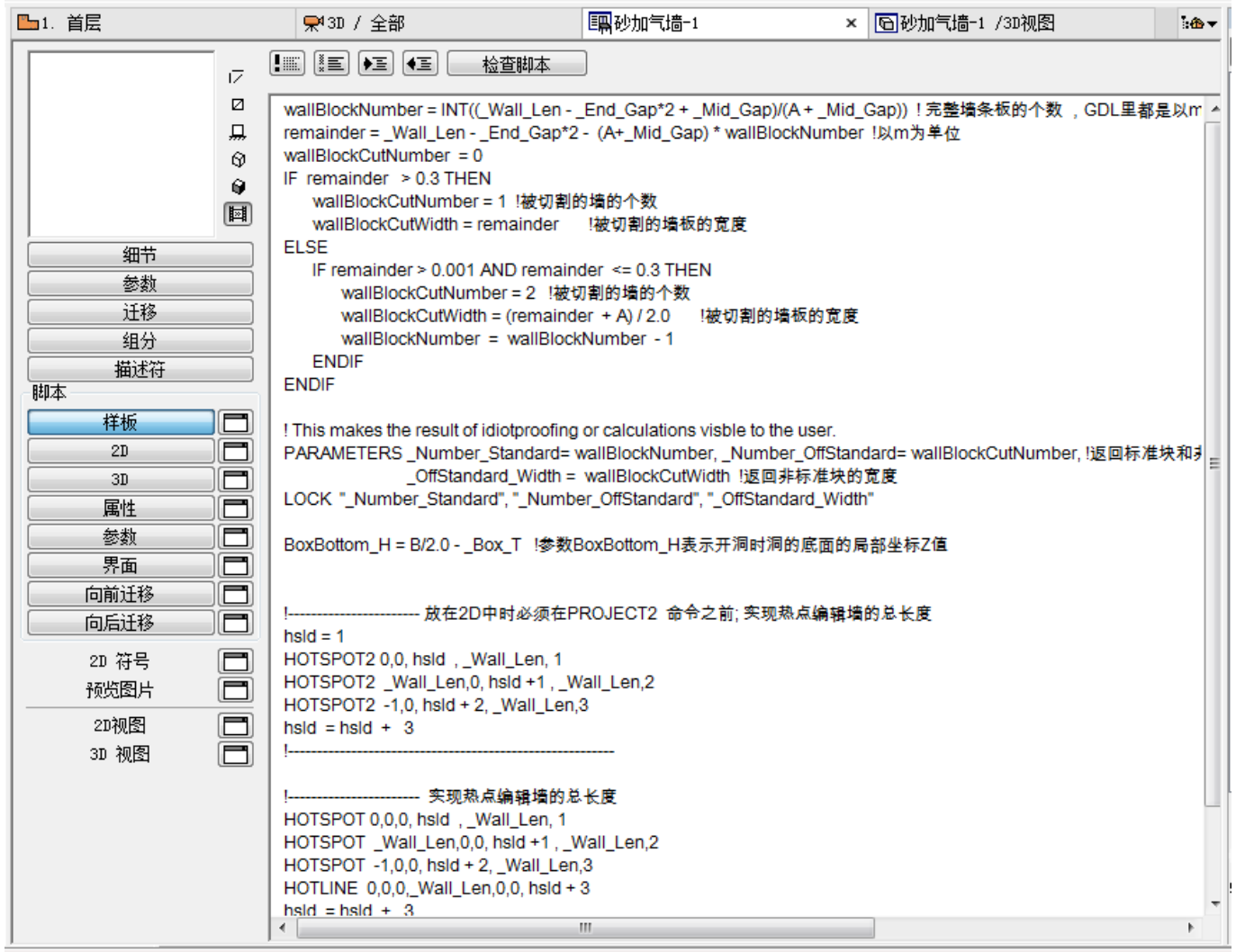
5、输入参数名称（将显示在界面上）

6、输入的“值”将作为默认值使用。

7、“显示”列中：

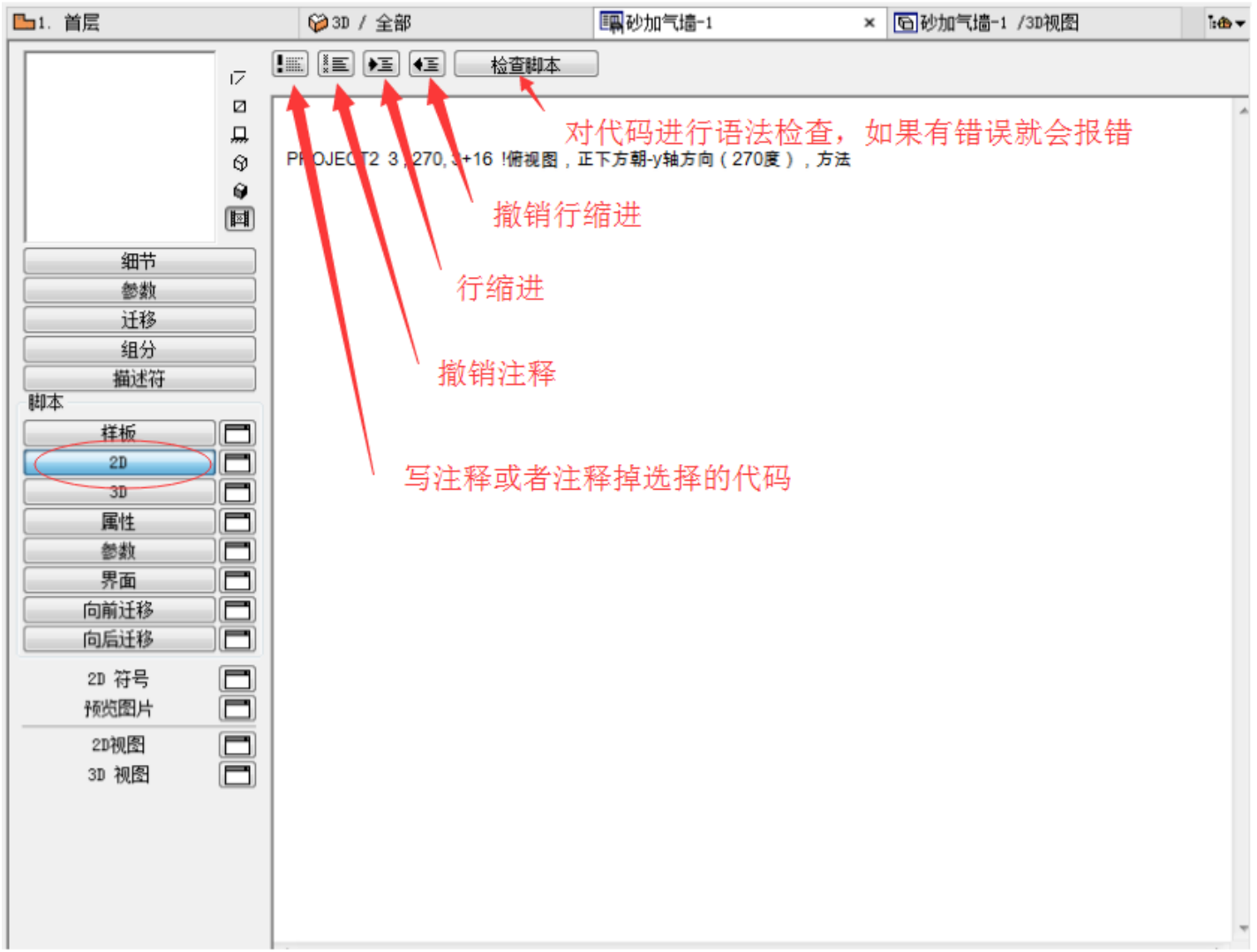
- “叉”表示界面中看不到该参数名称，隐藏了。
- “隶属”表示该参数属于上一级参数的次级参数，界面上会缩进，但标题类型参数不能是次级的。
- “B”（**bold**）表示粗体显示。
- “U”（**Unique**）选择U的参数在对象被“拾取参数”和“注射参数”时该参数不参与这个操作。

Master script

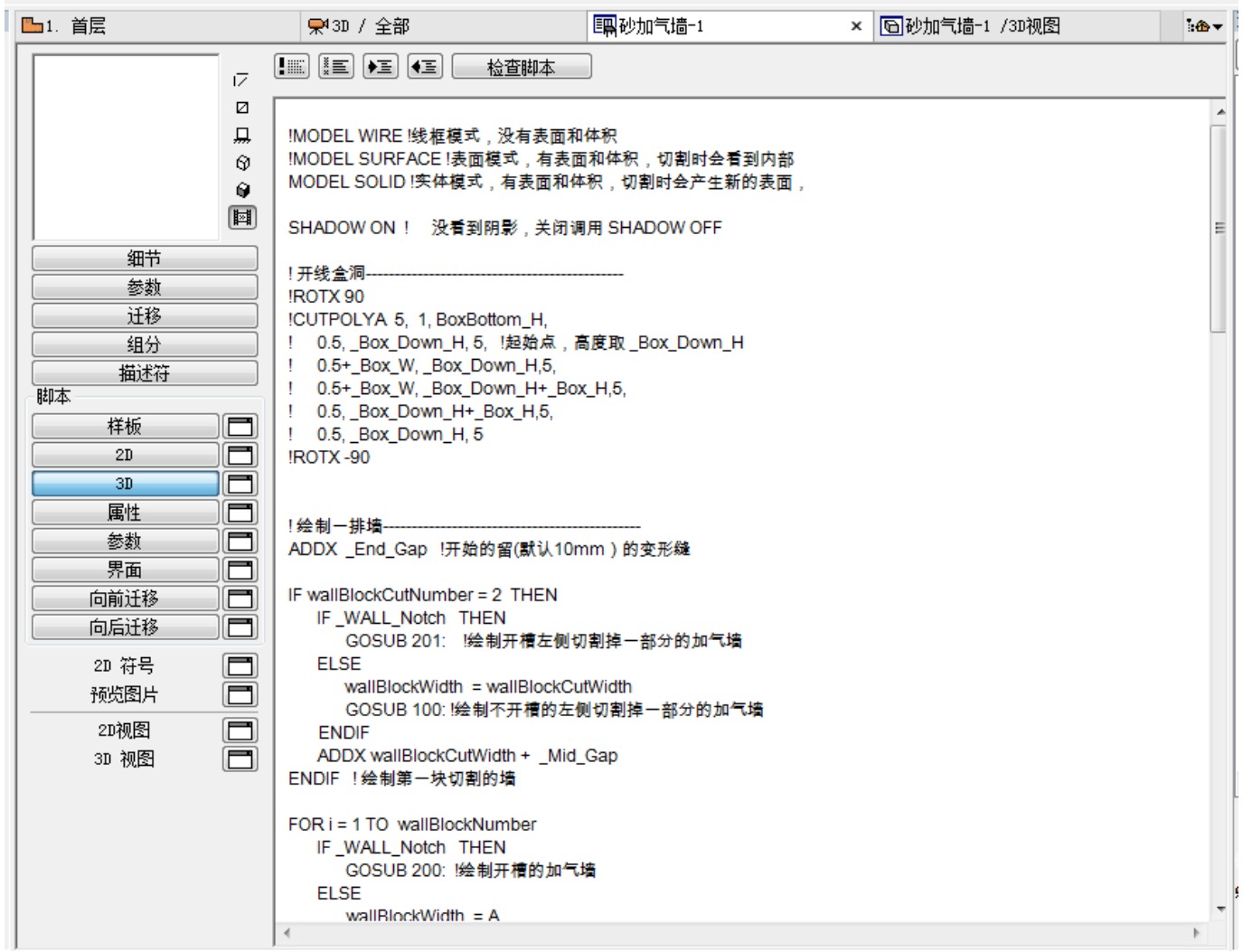


样板（master script）中的代码将在2D/3D中的代码之前被执行，所以一般的数据处理放在这里，处理后的数据将影响2D/3D中的模型显示结果。

2D script

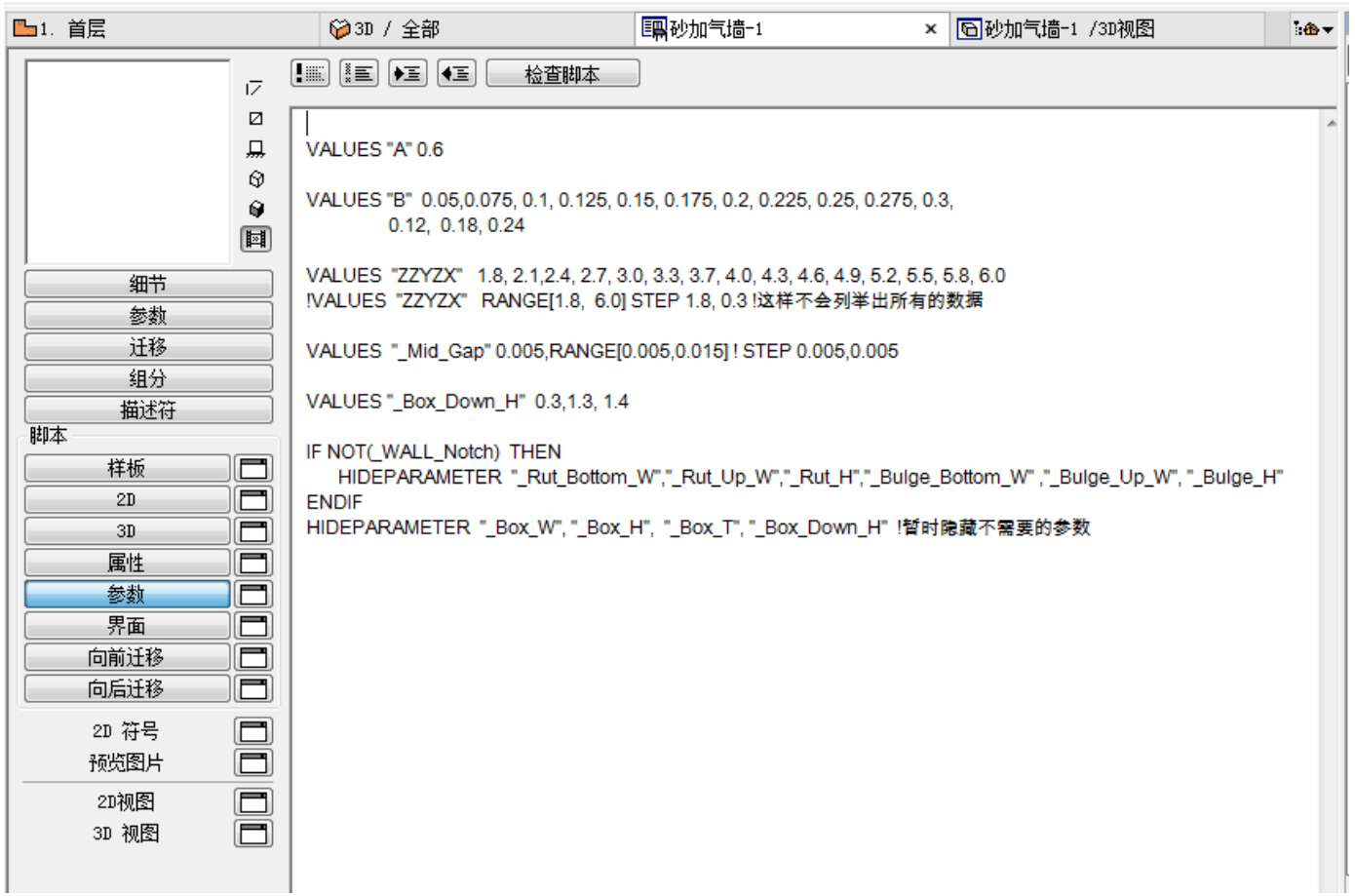

1. 首层
3D / 全部
砂加气墙-1
砂加气墙-1 /3D视图
检查脚本
对代码进行语法检查，如果有错误就会报错
PROJECT2 3 ,270, 3+16 !俯视图，正下方朝-y轴方向（270度），方法
撤销行缩进
行缩进
撤销注释
写注释或者注释掉选择的代码
细节
参数
迁移
组分
描述符
脚本
样板
2D
3D
属性
参数
界面
向前迁移
向后迁移
2D 符号
预览图片
2D视图
3D 视图

3D script



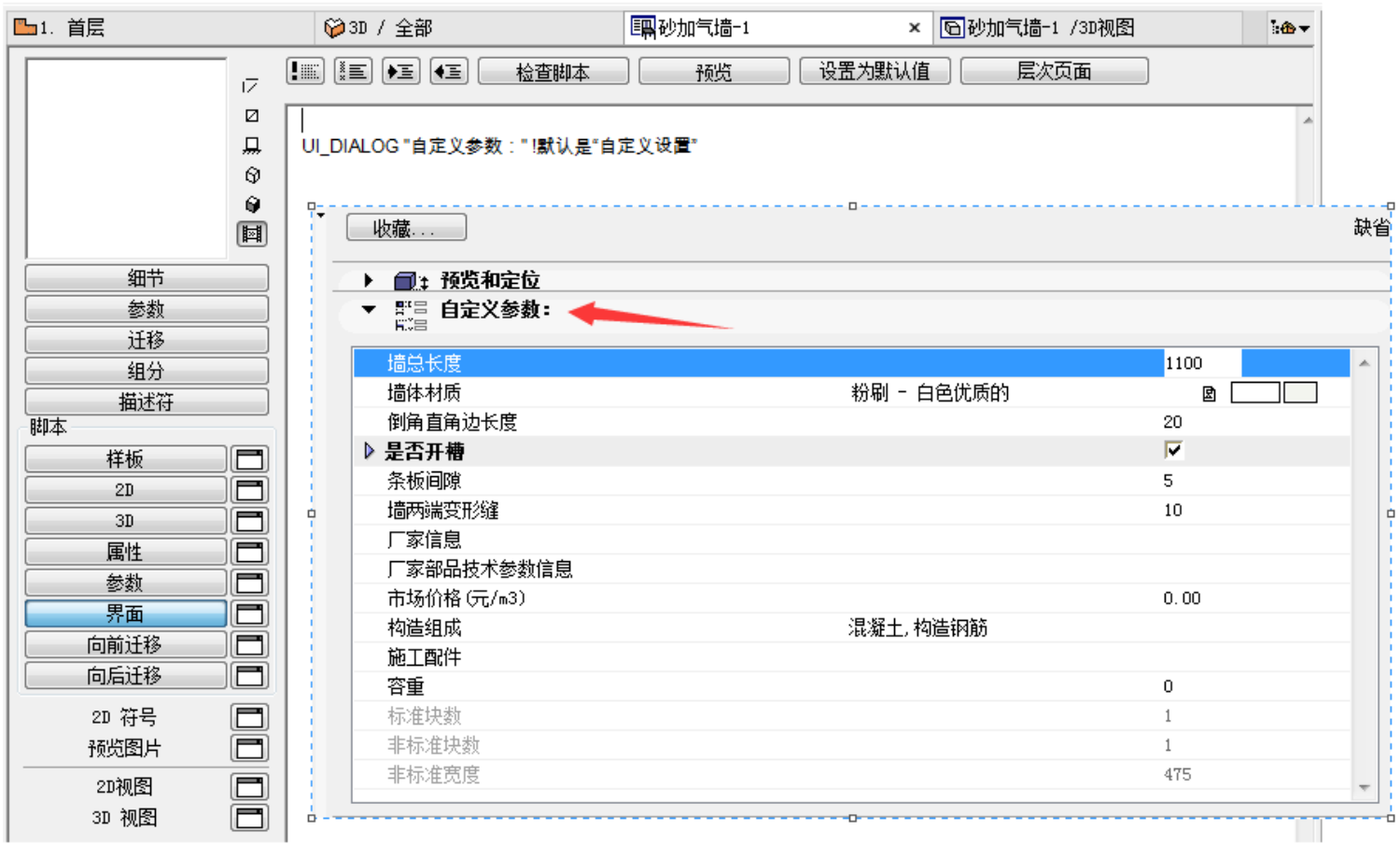
创建3D模型的代码就都写在这里了。

Parameter script



创建2D图形的代码就都写在这里

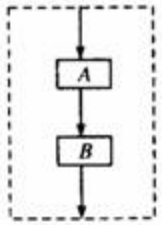
Interface script



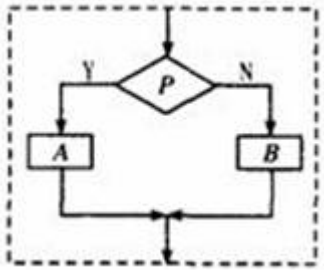
GDL基本语法

程序设计语言的三个基本结构

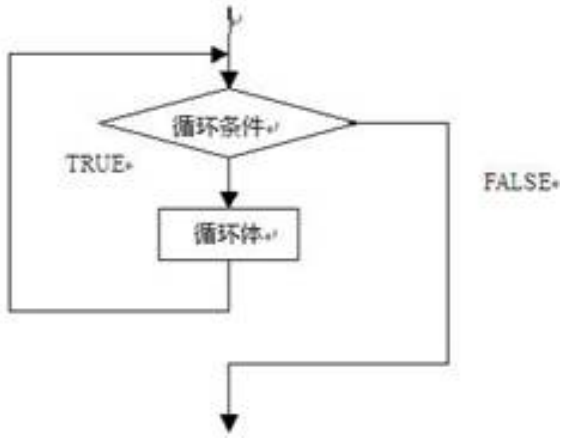
- 顺序结构：



- 分支结构（选择结构）：



- 循环结构：




```
!绘制一排墙----------------------------------------
ADDX _End_Gap  !开始的留(默认10mm ) 的变形缝

IF wallBlockCutNumber = 2  THEN
    IF _WALL_Notch  THEN
        GOSUB 201:  !绘制开槽左侧切割掉一部分的加气墙
    ELSE
        wallBlockWidth  = wallBlockCutWidth
        GOSUB 100: !绘制不开槽的左侧切割掉一部分的加气墙
    ENDIF
    ADDX wallBlockCutWidth + _Mid_Gap      分支结构
ENDIF  !绘制第一块切割的墙

FOR i = 1 TO  wallBlockNumber
    IF _WALL_Notch  THEN
        GOSUB 200:  !绘制开槽的加气墙
    ELSE
        wallBlockWidth  = A                循环结构
        GOSUB 100:  !绘制不开槽的加气墙
    ENDIF
    ADDX (A + _Mid_Gap)
NEXT i
```

循序结构

变量命名

- GDL中变量命名法则：以 字母、下划线_ 或者 ~打头，后面接字母、_、数字或者~组成一个标识符来定义一个变量。字母不分大小写。
  - 如：_radius, ~radius1,radius, Radius都是正确的命名，但是radius, Radius是同一个变量名。
  - 1_radius, 2~radius1,3radius都是错误的。
  - GDL中无需指定变量类型，直接写变量名即可
  - 一般的语言是_或者字母打头的
  - 命名的法则一般有三种：驼峰命名法、帕斯卡命名法和匈牙利命名法。
- 变量代表计算机内存中的一个地址，用来存储数据（GDL中的数据类型包含整数、实数、字符串）

变量赋值

- 使用等号“=”对变量进行赋值
  - radius = 0.5 !常量赋值
  - ~radius1 = radius !变量赋值
  - _radius = radius + 0.5 !表达式赋值

数组变量

- GDL中数组变量的定义
  - DIM myArray1[3] ！定义一维数组
  - myArray1[1] = 10 ！GDL中下标从1开始
  - myArray1[0] = 1 ！检查会报错
  - PRINT myArray1[1] ！结果打印出“10”这里的PRINT是gdl中的一个指令
  - PRINT myArray1[2] ！~~结果打印出“0”，没有赋值的变量默认值为0~~代码检查过不去，必须赋初值
- **VARDIM1(dim1_expr)/VARDIM2(dim2_expr)**
  - **分别返回数组的一维与二维的元素个数**
  - **VARDIM1(**myArray1**) !!!返回3，前提是**myArray1的3个元素必须赋初值
  - **dim myArr2[5][6] !!!....对数组各个元素赋值**
  - **VARDIM1(**myArr2**) !!!返回5**
  - **VARDIM2(myArr2) !!!返回数组二维的个数，这里是6**
  - **VARDIM2(**myArr1**) !!!返回0**

数组变量程序(案例3)

```
dim myArray1[3]
dim myArray2[5][6]

!!! 数组在使用之前必须赋初值，这也是很好的编程风格之一
for i =1 to 3
    myArray1[i]=i
next i
for i =1 to 5
    for j =1 to 6
        myArray2[i][j]=i*j
    next j
next i

print "VARDIM1(myArray1)=", VARDIM1(myArray1) !!!返回3
print "VARDIM1(myArray2)=", VARDIM1(myArray2) !!!返回5
print "VARDIM2(myArray2)=", VARDIM2(myArray2) !!!返回6
print "VARDIM2(myArray1)=", VARDIM2(myArray1) !!!返回0
```

全局变量

- 全局变量
  - 全局变量使得储存模型的特殊值成为可能，这样就允许您访问有关GDL宏环境的几何信息。例如，当您定义必须在墙中安装的窗口时，可以访问墙参数。全局变量在宏调用期间不堆栈。
- 局部变量与全局变量
  - 局部变量只是在当前的命令执行中可见，命令结束后就销毁掉了。我们自己定义的变量都是局部变量。
  - 全局变量是在ArchiCAD的整个进程中有效的，直到退出ArchiCAD时才会销毁。全局变量应该是ArchiCAD自己预定义好的。比如那些以Glob_打头的变量都是全局变量。

算术运算符

运算符	名称	优先级	备注
^(或**)	乘方	2	
*	乘	3	
/	除	3	/’（除）的结果始终是实数
MOD(或%)	模(余数)	3	X MOD Y 表达式的含义： X - Y * INT (X/Y)
+	加	4	＋（加）也可以应用于字符串表达式：结果是字符串并置。
-	减	4	

注：＋（加）也可以应用于字符串表达式：结果是字符串合并。‘/’(除)的结果始终是实数，而其它运算符的结果取决于运算对象的类型：如果所有运算对象都是整数，结果将是整数，否则是实数。

逻辑运算符

运算符	名称	优先级	备注
=	等于	5	同时也是赋值符号
<	小于	5	
>	大于	5	
<=	小于或等于	5	
>=	大于或等于	5	
<>(或者#)	不等于	5	

注：这些运算符也可以在任何两个字符串表达式之间使用（字符串比较是区分大小写）。结果是一个整数，1或0。不建议使用有真正运算对象‘=’（等于）、‘<=’（小于或等于）、‘>=’（大于或等于）、‘<>’（或#）（不等于）运算符，因为这些运算符可以导致精确度问题。

布尔运算符

运算符	名称	优先级	备注
AND（或&）	逻辑与	6	同时为true结果才为true
OR（或|）	逻辑或	7	只要有一个为true则结果为true
EXOR（或@）	逻辑异或	8	不一样则结果为true

注：布尔运算符对整数数值起作用。因此，0是指“false”，而任何其它数值是指“ture”。逻辑表达式的值也是整数，即1代表“ture”，0代表“false”。不建议使用有真正运算对象的布尔运算符，因为这些运算符可以导致精确度的问题。

基本语句

- 以换行符结束一个基本的语句
  - 比如： radius = 0.5 !注意，GDL脚本中的长度单位默认是以m为单位
- 对于简单的语句，可以把几个语句写在同一行，但是必须以“：”分开
  - 比如： radius = 0.5 ： diameter = radius * 2 !这里是两个语句
- 语句一般由基本的赋值语句和GDL指令语句构成，GDL语法比较简单，要掌握GDL，重点就是掌握各种GDL指令的用法

条件语句1

- **IF – GOTO语句语法：**
  - **IF** condition **THEN** label
  - **IF** condition **GOTO** label
  - **IF** condition **GOSUB** label
- 条件跳转语句，如果condition的值为0（逻辑值为false），这句指令就不起作用，否则跳转到label标识的语句继续执行，这里的Then/GOTO/THEN GOTO是等价的。

```
a=10
IF a=10 THEN 100  :  IF a=20  GOTO 200 :  IF a=30  GOSUB 300
100:
   STATEMENTS
200:
  STATEMENTS
END
300:
   STATEMENTS
RETURN
```

条件语句2

- **IF** condition **THEN** statement [**ELSE** statement] ! 写在同一行无需ENDIF
- **IF** condition **THEN**
  [statement1
  statement2
  …
  statementn]
  [**ELSE** ! 注意：没有ELSE IF 这样的语句，如果这样写会解析为else里嵌套一个if语句
  statementn+1
  statementn+2
  …
  statementn+m]
  ENDIF ! 有多行必须用ENDIF来结束

IF语句(案例4)

```
RESOL  30  !!! 取值3~36，默认值为36
h = 1.5
r  = 0.8
drawWhat = "300"  !!! "200"  "300" "400"

if drawWhat = "100" then CYLIND h,r     !!!圆柱体
if drawWhat = "200" then
     SPHERE r      !!! 球体
endif
if drawWhat = "300"  or drawWhat = "400" then
     ELLIPS h, r     !!! 半椭圆体
     ADDx r*3
     ELLIPS r, r      !!! 半球体
     del  1
endif
```

FOR循环语句

- **FOR** variable_name = initial_value **TO** end_value [**STEP** step_value]

  …… **!想要执行的语句，注意**step_value的值在这里改变是不会起作用的

  NEXT variable_name

注：如果省略STEP则其默认是为1

!!!求1~100之间奇数之和

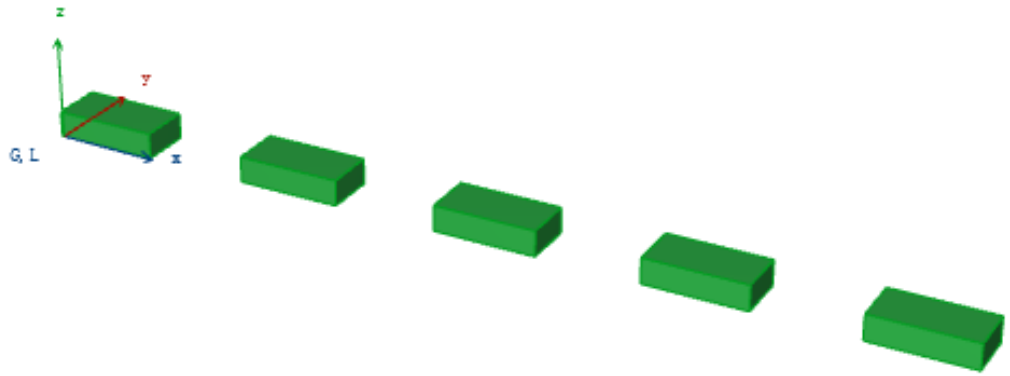
```
sum = 0
FOR i=1 TO 100 STEP 2
   sum = sum + i
NEXT i
PRINT “sum = ” + STR(sum,2,0)   ！打印的结果为”sum = 2500”
```

FOR循环语句(案例4-1)

```
!!!求1~100之间奇数之和
sum = 0
FOR i=1 TO 100 STEP 2
   sum = sum + i
NEXT i
PRINT “sum = ” + STR(sum,2,0)   ！打印的结果为”sum = 25”
```

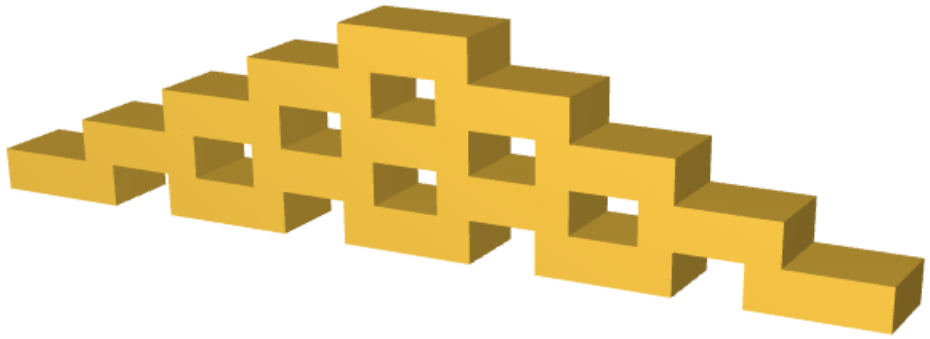
FOR循环语句(案例4-2)

```
a = 1 : b = 0.5 : c = 0.25
for i = 1 TO 5
        ADDx  (i-1)*2
        BLOCK  a,b,c
        DEL 1
next  i
```



FOR循环语句(案例4-3)

```
material  myMaterial
a = 0.8 : b = 0.5 : c = 0.25
for i = 0 to 4
    ADDz  i*c
    ADDx  i*a*0.75
    for j = 1 TO 5-i
        ADDx  (j-1)*1.5*a
        BLOCK  a,b,c
        DEL 1
    next  j
    DEL  2
next  i
```



DO/WHILE循环语句

DO **！不管什么情况总会进入循环，至少会执行一次**

[statment1

statement2

...

statementn]

WHILE condition **！一定要在循环中改变 condition 的值，否则可能进入死循环**

WHILE condition **DO** **！先判断条件，有可能一次也不执行**

[statement1

statement2

...

statementn]

ENDWHILE

REPEAT/UNTIL循环语句

REPEAT

[statement1

statement2

...

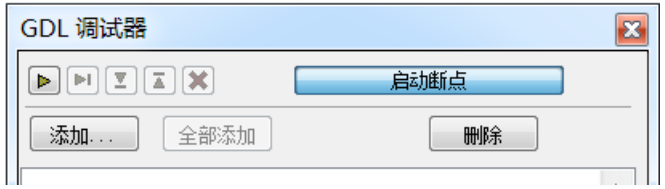
statementn]

UNTIL *condition*

这些在关键字之间的指令将被执行，直到condition变为true。每次运行完这些指令后将测试condition的值，判断是否需要再次重复执行。所以必须在这些指令中添加改变condition值的语句，使得condition的值在适当的时候变为true，否则就会出现死循环。

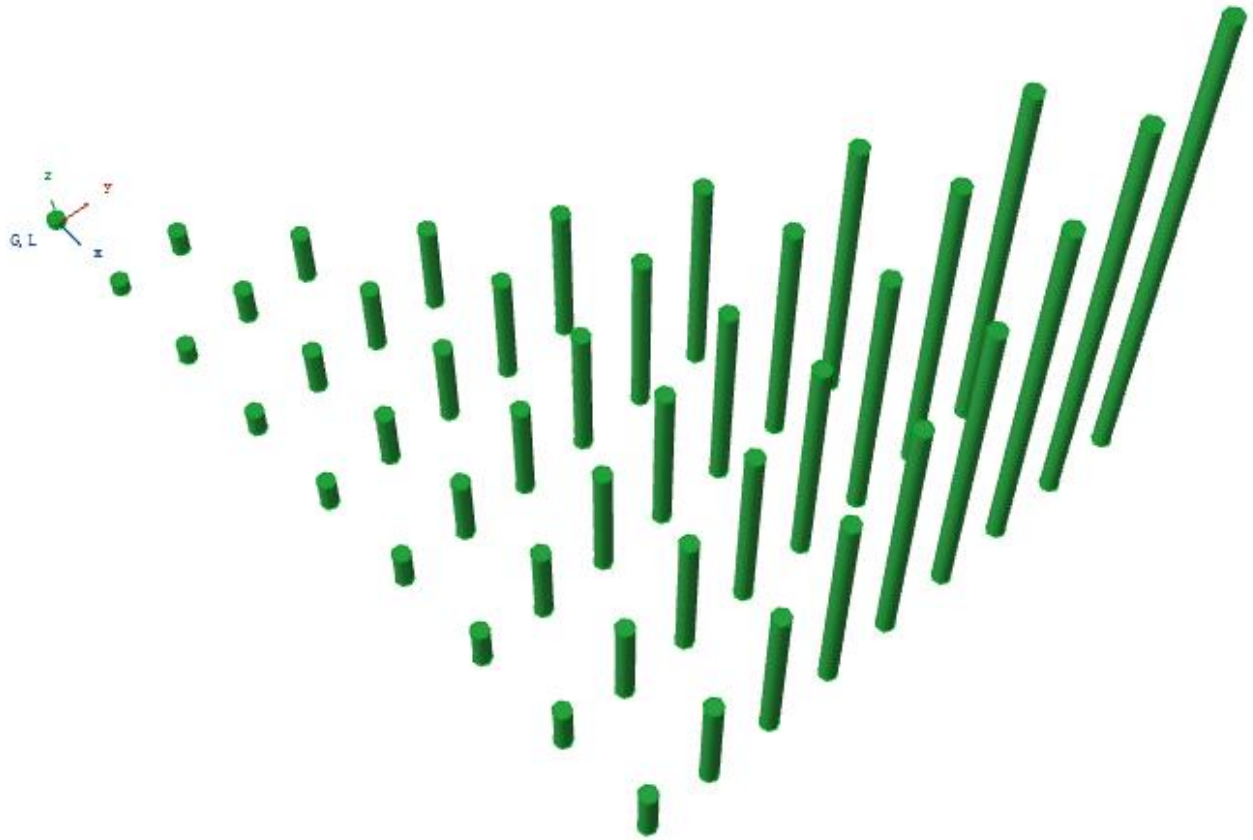
跳转/断点语句

- GOTO指令
  - GOTO *myLable* 无条件转移指令，程序将执行由*myLable*标签指定的一个分支
- GOSUB指令
  - GOSUB *myFunction* 内部子程序的调用， *myFunction*标签是子程序的入口
- RETURN指令
  - RETURN由一个内部子程序返回到调用子程序的代码处，继续执行后续的代码
- BREAKPOINT
  - BREAKPOINT *expression* 使用这个命令，可以在GDL脚本中指定一个断点。如果参数（数值表达式）的值为ture（1），并且勾选了调试器的**启动断点**选项，GDL调试器将停在此命令的位置。在“正常”执行模式中，GDL仅仅是跳过此命令。



循环语句(案例4-4)

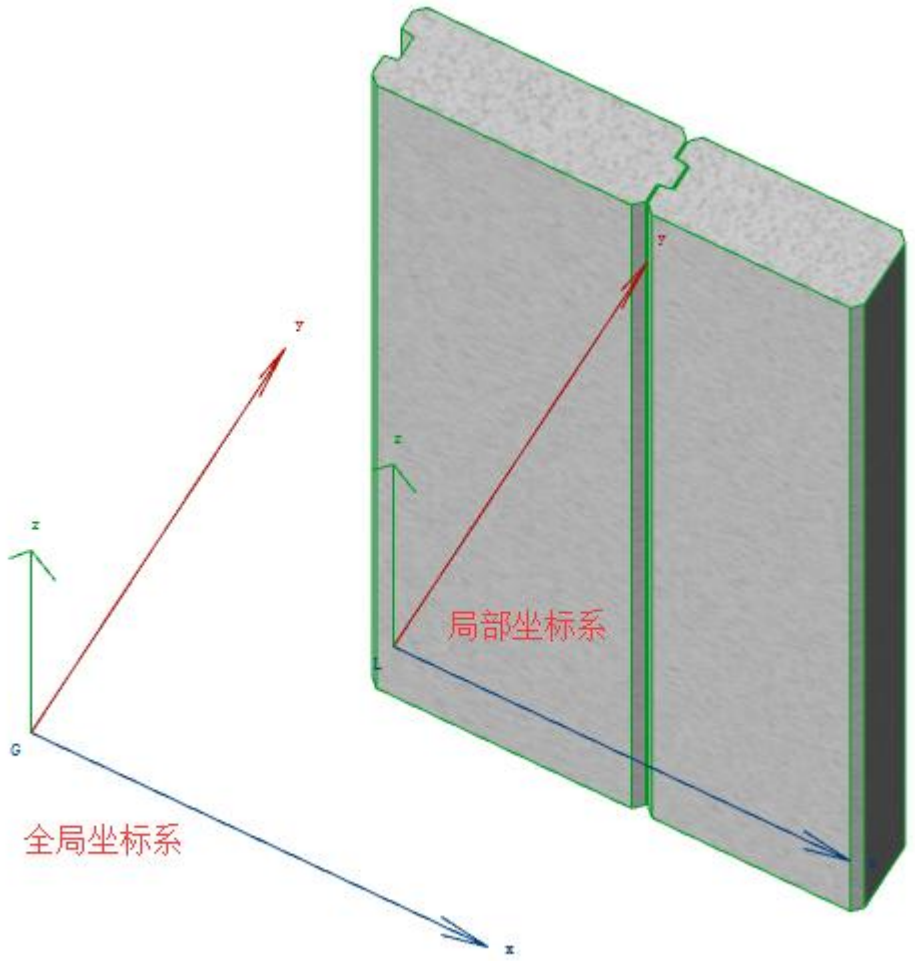
- 绘制一个圆柱体的三角形阵列，布局就像一个乘法口诀表一样，每个圆柱体的半径r=0.25m，高为乘法口诀乘积的1/5，X轴方向间距为3m，Y轴方向间距为2m。如下图所示：



ArchiCAD中GDL的坐标系及坐标变换

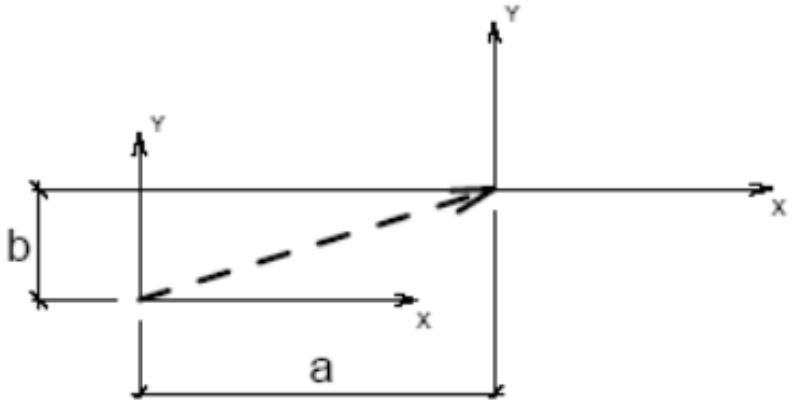
GDL中的坐标系

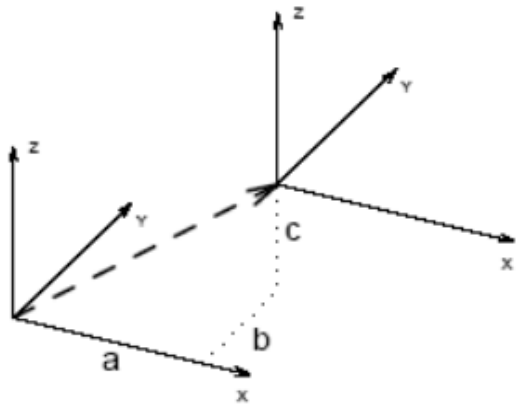
- GDL中使用的是笛卡尔坐标系，右手系
- 有一个全局坐标系和一个局部坐标系
- 可以通过ROT,ROTx, ROTy,ROTz,ROT2，ADD，ADDx,ADDy,ADD等命令来进行坐标转换。

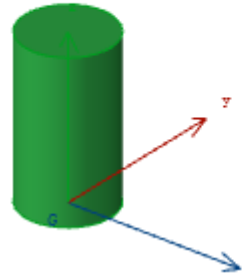


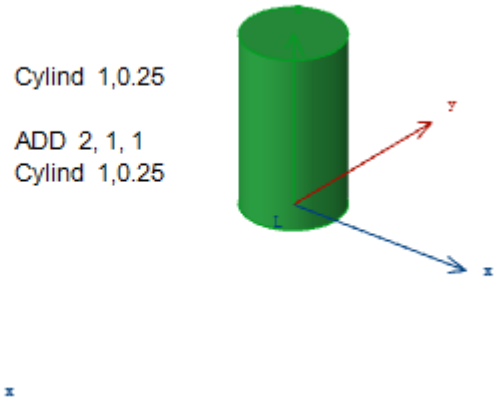
GDL中的坐标系

- 在执行2d,3d形状命令之前，先将坐标系移动到正确的位置。再来执行形状命令，使得物件在你想要的位置。所有的图形命令都是在自己的局部坐标系中建模的。







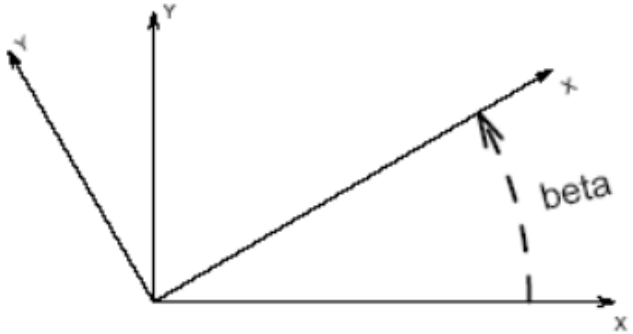


ADD a, b, c ！将创建一个局部坐标系，将坐标原点移多了点(a,b,c)

- 使用坐标转换命令，可以预定义它的位置及绕轴旋转。这些转换对前面已生成的形状不起作用，它们只对后来生成的形状生效。

2D坐标转换

- **ADD2** x, y　　!!!坐标位置平移。
- **MUL2** x, y　　!!!比例关系转换，当x或y为负数则镜像。
- **ROT2** beta　　!!!坐标旋转变换。

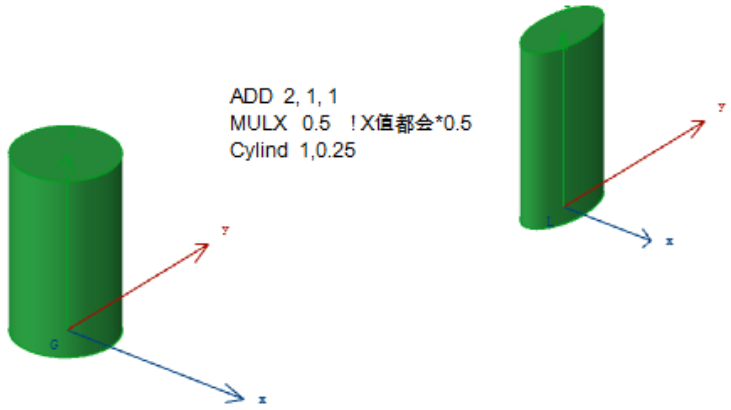


- （它们相对应的3D转换是：ADD,MUL,ROTX,ROTY,ROTZ, ROT）
- 每一个坐标转换命令在堆栈中对应1个条目（可以用DEL 1删除），如果不删除的话，他们就会组成一个链表，对后续的建模会产生一个叠加的影响。如果不再需要最好使用DEL指令来删除,使坐标复原。

3D坐标转换

- **ADDX dx** !!!沿着X轴移动dx的量。
- **ADDY dy** !!!沿着Y轴移动dy的量。
- **ADDZ dz** !!!沿着Z轴移动dz的量。
- **ADD** dx, dy, dz !!!等价于 ADDX dx: ADDY dy: ADDZ dz !但是前者是1个条目，用DEL 1来删除，后者是3个条目，要用DEL 3来删除
- 举例： ADD 2.0,3.0,1.0 !局部坐标原点将在X/Y/Z轴上分别平移2.0m，3.0m和1.0m

3D坐标转换



- **MULX** mx
- **MULY** my
- **MULZ** mz !Scales the local coordinate system along the given axis. Negative mx, my, mz means simultaneous mirroring.
- **MUL** mx, my, mz !!!等价于MULX mx: MULY my: MULZ mz。但是前者在堆栈中只有一个条目，因此它可以用DEL 1来删除。
- **ROTX** alphax
- **ROTY** alphay
- **ROTZ** alphaz !围绕给定的轴，分别按逆时针旋转alphax、alphay、alphaz度（类似右手螺旋定则）

XFORM（矩阵转换）

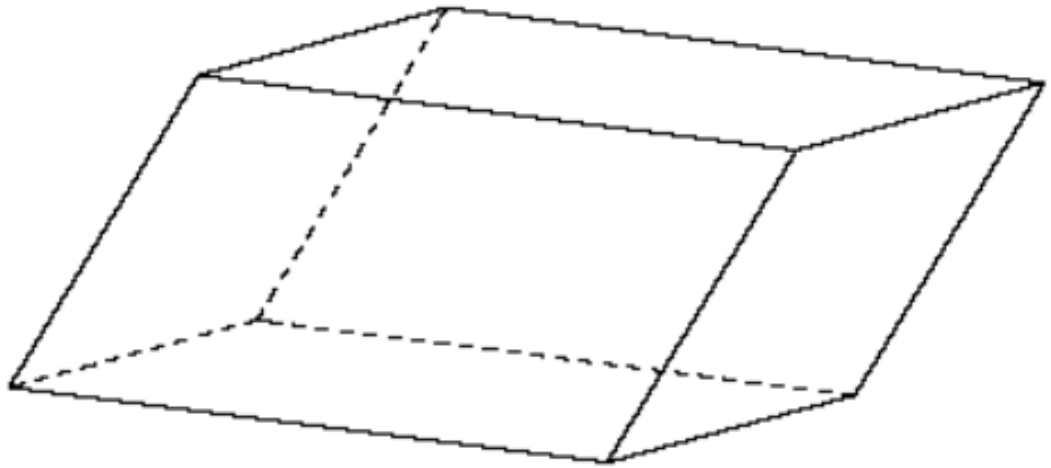
XFORM a11, a12, a13, a14,
a21, a22, a23, a24,
a31, a32, a33, a34

- 定义完整的转换矩阵。它主要用于GDL代码的自动生成。它在堆栈中只有一个条目。

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

- 举例：

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



坐标转换栈管理

- 在GDL脚本中每一条坐标变换指令(如右图所示)都将按其出现先后顺序压入栈中，形成一个总的转换矩阵比如T。
  当执行每一个二维/三维指令时，都将首先按矩阵T进行坐标变换，所以当使用完某个坐标变换后最好及时的清除掉它。
- **DEL** n [, begin_i]
  - 从转换堆栈中删除n个条目。
  - 如果未定义begin_i，则删除转换堆栈中前面的n个条目。局部坐标系后退到删除后的位置。
  - 如果指定了begin_i转换，则从begin_i指示的一个条目开始，向后删除n个条目(含begin_i)。 begin_i的编号从1开始，如果指定了begin_i参数,并且n是负值,则向前删除n条(不含begin_i)。
  - 如果在当前脚本发布的转换少于给出的n个参数所指示的转换，删除的只是发布的转换。
- **DEL TOP:** 删除当前脚本的所有当前转换(谨慎使用)。
- NTR():返回当前坐标转换的条目数量。

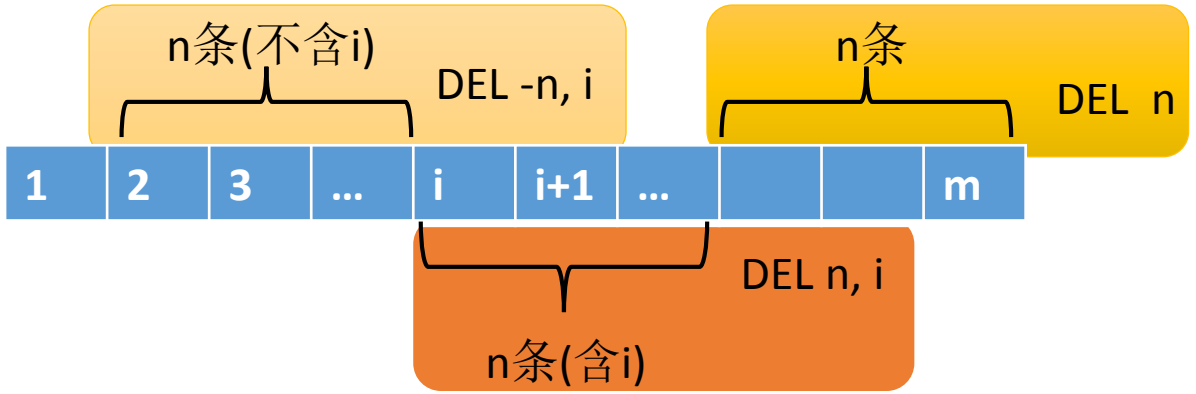
2D Transformations
- ADD2
- MUL2
- ROT2

3D Transformations
- ADDX
- ADDY
- ADDZ
- ADD
- MULX
- MULY
- MULZ
- MUL
- ROTX
- ROTY
- ROTZ
- ROT
- XFORM

坐标转换栈管理(案例5)

BLOCK 0.5, 0.5, 0.5　!!!第1个块

ADDX 2
ADDY 2
ADDZ 2
DEL 1　　!!!依次演示del 1 / del 2 / del 3 / del 2,1 / del 2,1 / del -1,3
BLOCK 1, 0.3, 1　　!!!第2个块



常用3D图形


常用3D形体
基本的3D形体
BLOCK, Brick a,b,c
Cylind h,r
Sphere
Ellips
Cone
Prism
Prism_
Bprism_,CPRISM_,CPRISM_{2},CPRISM_{3}
复杂的3D
Slab_ Cslab_
Cwall_ , BWALL_ , XWALL_
BEAM
CROOF_ , CROOF_{2} , CROOF_{3} , CROOF_{4} ,
MESH
Armc
Arme
Elbow
由单一多段线生产
Extrude
Pyramid
Revolve
由两个多段线生成
Ruled
Sweep
Tube
COONS
MASS
POLYROOF

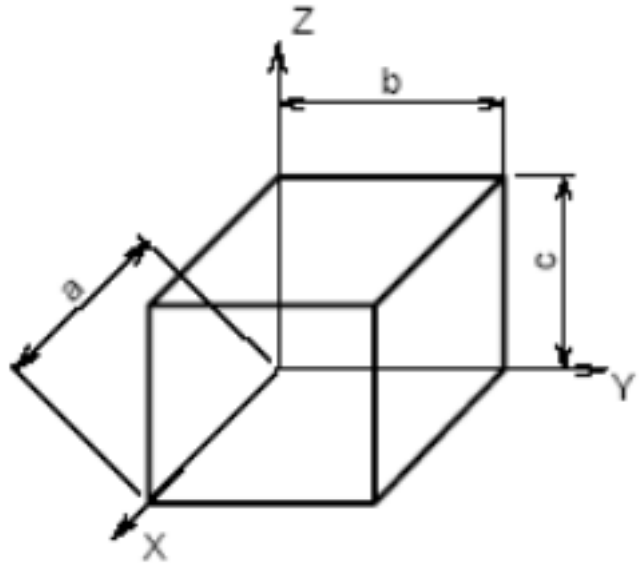
基本3D形体

BLOCK 『块』
BRICK 『砖』
CYLIND 『圆柱体』
SPHERE 『球体』
ELLIPS 『椭圆体』
CONE 『圆锥体』
PRISM 『棱柱体』
PRISM_
CPRISM_
BPRISM_
FPRISM_
HPRISM_
SPRISM_

SLAB
SLAB_
CSLAB_
CWALL_
BWALL_
XWALL_
XWALL_{2}
BEAM 『梁』
CROOF_
MESH
ARMC
ARME
ELBOW『弯管』

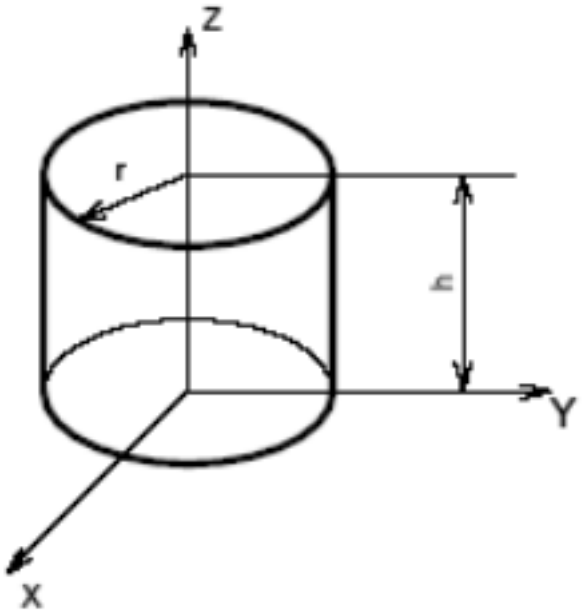
常用的基本3D形:

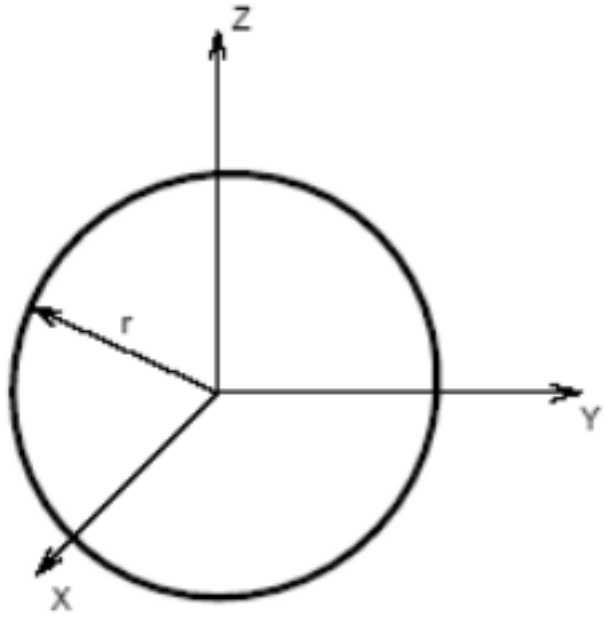
- **BLOCK**『块』BLOCK a, b, c
- **BRICK** 『砖』BRICK a, b, c
- 块的第一个角在局部原点，长度a、b及c的边分别是沿着x、y及z轴。零值创建退化的图块（矩形或线）。

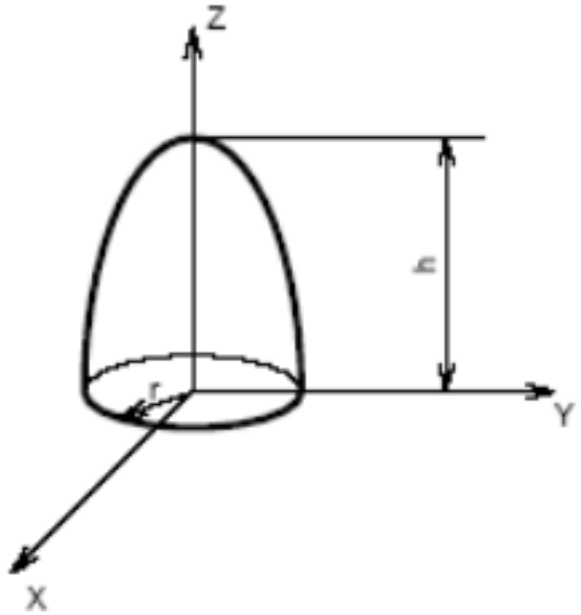


常用的基本3D形:

- **CYLIND**『圆柱体』CYLIND h, r
- **SPHERE**『球体』SPHERE r
- **ELLIPS**『椭圆体』ELLIPS h, r半椭圆体。其x-y平面中的横截面是一个中心在原点，半径为r的圆。沿着z轴的半轴长度是h。
- ELLIPS r, r ! 半球体



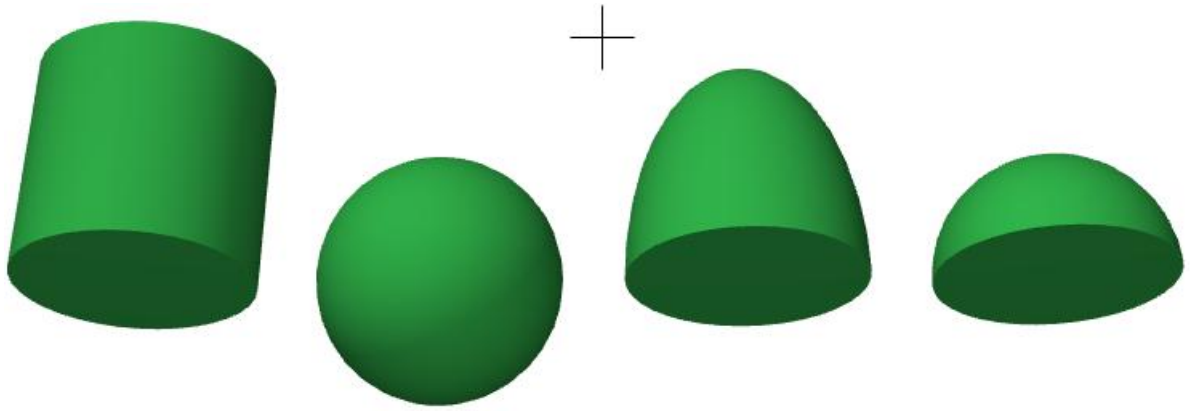




CYLIND / SPHERE / ELLIPS (案例6):

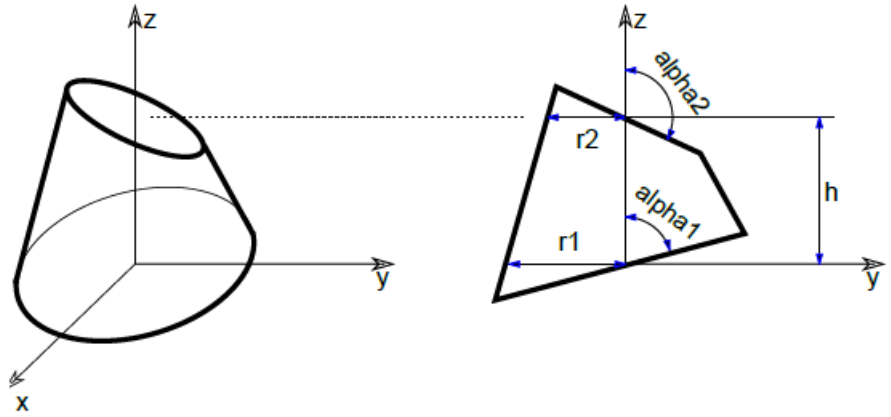
- **CYLIND**『圆柱体』CYLIND h, r
- **SPHERE**『球体』SPHERE r
- **ELLIPS**『椭圆体』ELLIPS h, r半椭圆体。其x-y平面中的横截面是一个中心在原点，半径为r的圆。沿着z轴的半轴长度是h。
- ELLIPS r, r ! 半球体

```
RESOL  20  !!! 取值3~36，默认值为36
h = 1.5
r  = 0.8
CYLIND h,r     !!!圆柱体
ADDx  2.0      !!! X轴方向平移2m
SPHERE r       !!! 球体
ADDx  2.0      !!! X轴方向平移2m
ELLIPS h, r    !!! 半椭圆体
ADDx  2.0      !!! X轴方向平移2m
ELLIPS r, r    !!! 半球体
DEL  3
```



常用的基本3D形:

- **CONE h, r1, r2, alpha1, alpha2** 截头圆锥体，那里的alpha1和alpha2是端面到z轴的倾斜角度，r1和r2是两端圆形的半径，h是沿着z轴的高度。如果h＝0，alpha1和alpha2的值则忽略不计，圆环在x-y平面生成。alpha1和alpha2用度表示。



参数限制：

0 < alpha1 < 180 and 0 < alpha2 < 180°

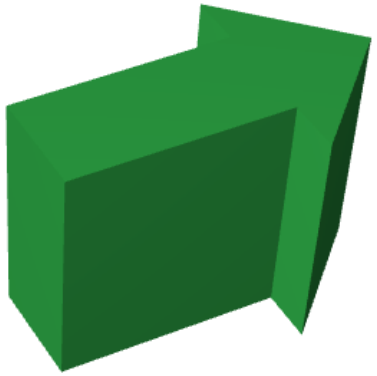
- 举例：CONE h, r, 0, 90, 90！正则锥(自己练习一下)

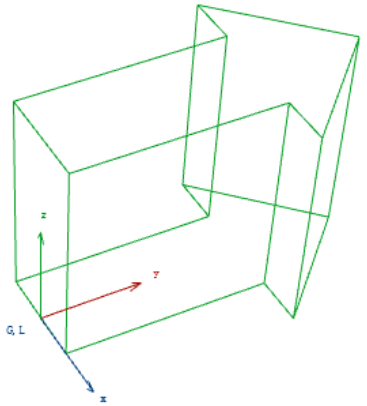
常用的基本3D形(案例6-1):

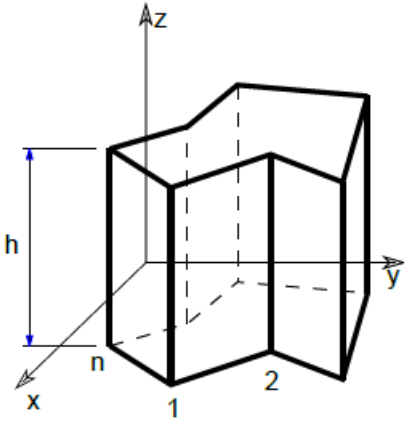
- **PRISM**『棱柱体』**PRISM** n, h, x1, y1, ... xn, yn

直立棱柱，其底部多边形在x-y平面（见POLY及POLY_的参数）。沿着z轴的高度是绝对值（h）。也可以使用负h值，该情况的第二个底部多边形在x-y平面下。参数限制：n >= 3

```
a = 1 : h = 2
PRISM  7, h,
       0.5*a, 0,
       0.5*a, 2*a,
       a, 2*a,
       0, 3*a,
       -a, 2*a,
       -0.5*a, 2*a,
       -0.5*a, 0
```







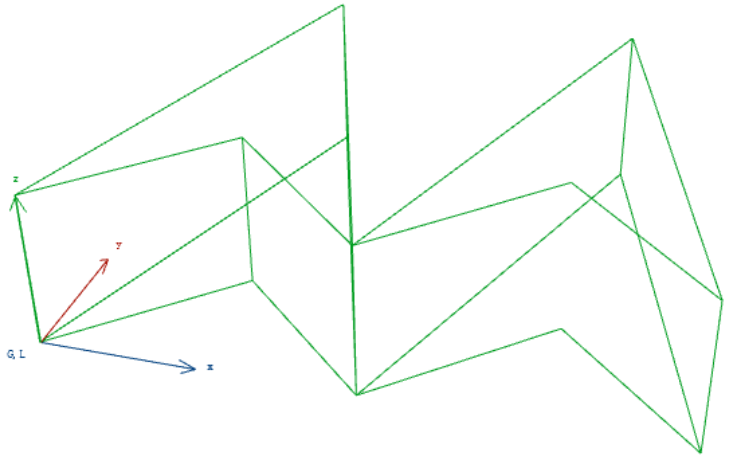
常用的基本3D形(案例6-2):

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

  类似于PRISM语句，但可以忽略任何的水平边及侧面。参数限制： n >= 3

- si：用来控制多边形的边及侧面可见性的状态码。也可以定义孔洞及用特殊的限制条件创建多义线的线段和弧。

```
PRISM_ 4,1,
        0,0,15,
        1,1,15,
        2,0,15,
        1,3,15
ADDx 2
PRISM_ 4,1,
        0,0,7,  !!! 7=1+2+4
        1,1,5,  !!! 5=1+4
        2,0,15,
        1,3,15
del 1
```



常用的基本3D形:

- **CPRISM_** top_material, bottom_material, side_material,

  n, h,x1, y1, s1, ..., xn, yn, sn
- PRISM_ 命令的延展，前三个参数为材质的名称或者索引，分别表示顶部、底部和侧面。
- 其它参数与命令PRISM_ 相同。
- 参考案例6-2
- **CPRISM_其他版本的命令**
  - **CPRISM_{2}**
  - **CPRISM_{3}**
  - **CPRISM_{4}**

三维圆管建模程序(案例6-3)

```
!!! CPRISM_ 指令案例
r2 = 0.2
r1 = r2 - 0.01
h = 2.7
CPRISM_  Coll_Material,Coll_Material,Coll_Material, 4, h,
      0, 0, 900+15,
      r2 , 360, 4000+15,
      0,0, 900+15,
      r1 , 360, 4000+15
```

常用的基本3D形(CPRISM_{2}):

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

CPRISM_{2} is an extension of the CPRISM_ command with the possibility of defining different angles and materials for each side of the prism. The side angle definition is similar to the one of the CROOF_ command.

- **alphai:** the angle between the face belonging to the edge i of the prism and the plane perpendicular to the base.面F_i与面F_v之间的夹角；其中F_i为边i所在的侧面，F_v为过边i且与底面垂直的面。
- **mati:** material reference that allows you to control the material of the side surfaces.用来控制侧面的材料。

常用的基本3D形(CPRISM_{3}):

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

CPRISM_{3} is an extension of the CPRISM_{2} command with the possibility of controlling the global behavior of the generated prism.

- **mask:** controls the global behavior of the generated prism.

  mask = j1 + 2*j2 + 4*j3, where each j can be 0 or 1.

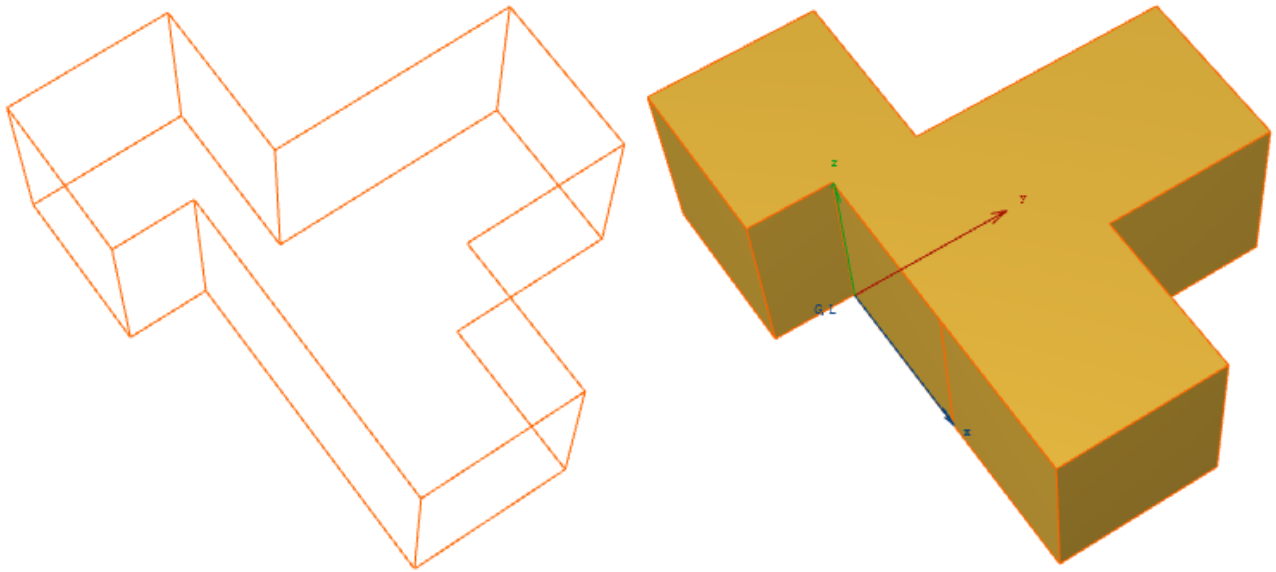
  j1: top edge in line elimination. 移除顶部重叠线(前提是线所在侧面重叠)

  j2: bottom edge in line elimination.移除顶部重叠线(前提是线所在侧面重叠)

  j3: side edge in line elimination. 移除顶部重叠线(前提是线所在侧面重叠)

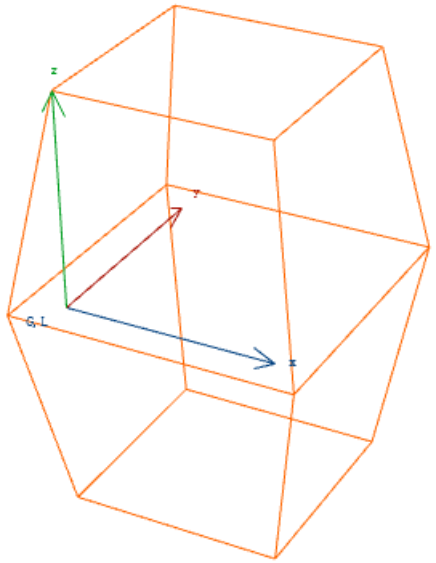
常用的基本3D形(CPRISM_{3}案例6-4):

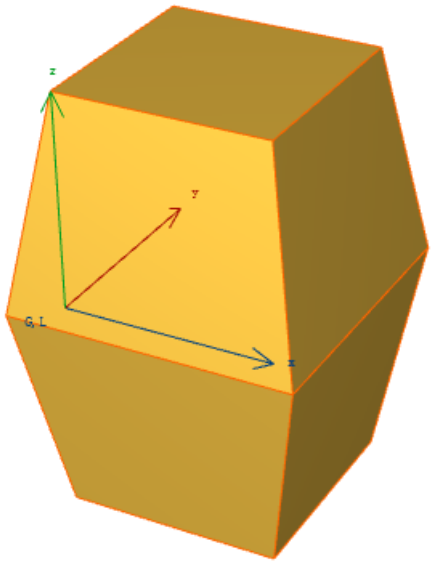
```
PEN 3
mat = IND (MATERIAL, "金属 - 金")
 mask = 1+2+4  !!! 移除重叠侧面上的重叠线(顶部、侧面、底部)
FOR i=1 TO 4 STEP 1
!       IF i = 1 THEN mask = 1+2+4
!       IF i = 2 THEN mask = 1
!       IF i = 3 THEN mask = 2
!       IF i = 4 THEN mask = 4
        CPRISM_{3} mat, mat, mat, mask,
                5, 1,
                0, 0, 0, 15, mat,
                1, 0, 0, 15, mat,
                1, 1, 0, 15, mat,
                0, 1, 0, 15, mat,
                0, 0, 0, -1, mat
        BODY -1
        DEL TOP
        IF i = 1 THEN ADDY 1
        IF i = 2 THEN
            ADDX -1
            addy -0.5
         ENDIF
        IF i = 3 THEN ADDX 1
NEXT i
```



常用的基本3D形(CPRISM_{3}案例6-5):

```
PEN 3
mat = IND (MATERIAL, "金属 - 金")
mask = 1+2+4  !!! 移除重叠侧面上的重叠线(顶部、侧面、底部)
CPRISM_{3} mat, mat, mat, mask,
        5, 1,
        0, 0, 10, 15, mat,
        1, 0, 10, 15, mat,
        1, 1, 10, 15, mat,
        0, 1, 10, 15, mat,
        0, 0, 10, -1, mat
MULz  -1
CPRISM_{3} mat, mat, mat, mask,
   5, 1,
   0, 0, 10, 15, mat,
   1, 0, 10, 15, mat,
   1, 1, 10, 15, mat,
   0, 1, 10, 15, mat,
   0, 0, 10, -1, mat
DEL  1
BODY -1
```





从这个案例证明：非侧面重叠时，重叠线不会被移除

常用的基本3D形(CPRISM_{4}):

- **CPRISM_{4}** top_material, bottom_material, side_material, mask,

  n, h,

  x1, y1, alpha1, s1, mat1,

  ...

  xn, yn, alphan, sn, matn

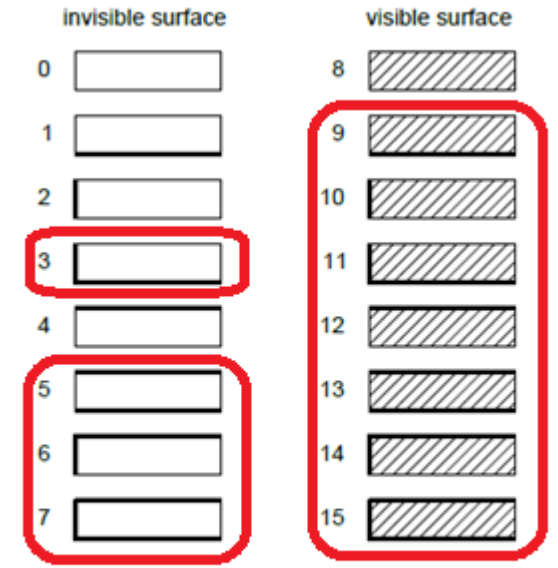
CPRISM_{4} is an extension of the CPRISM_{3} command with the possibility of using inline material definition, that means materials defined in GDL script locally also can be used next to materials defined in global material definitions.

状态码 (STATUS CODES)的意义

- 在平面中创建多义线时，可以通过状态码来设置线段或圆弧特性：
  - 控制平面多义线边的可见性
  - 定义多义线中的孔洞
  - 控制侧边及表面的可见性
  - 创建多义线中的线段和弧
- 下列许多GDL元素在节点处需要设置有状态码：
  - POLY_, PLANE_, PRISM_, CPRISM_, BPRISM_, FPRISM_, HPRISM_, SPRISM_, SLAB_, CSLAB_, CROOF_,EXTRUDE, PYRAMID, REVOLVE, SWEEP, TUBE, TUBEA

状态码 (STATUS CODES)的语法

- si = 1*j1 + 2*j2 + 4*j3 + 8*j4 + 64*j7[+ a_code] , 这里的j1、j2、j3、j4、j7 取值 0 或 1. **si:**是一个二进制整数 (在 0 到 127之间) 或者 -1.
  - j1, j2, j3, j4 表示相关的顶点或者边是否显示(1显示，0不显示):
    - j1: 控制下水平边
    - j2: 控制垂直边
    - j3:控制上水平边
    - j4:控制侧面
  - j7: 仅当j2=1 并且j7=1时才有实际意义
    - 此时只有从当前的视图方向看，当垂直线是轮廓线时垂直线才显示出来。

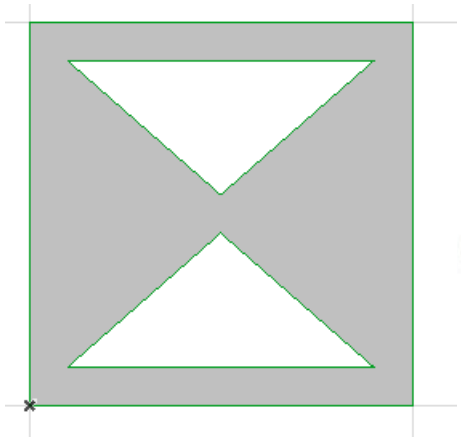


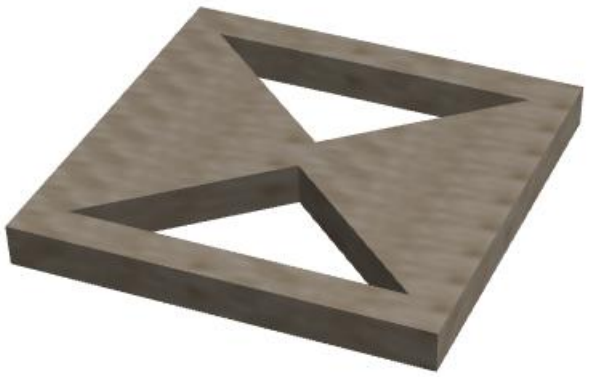
红色框中为1、2、4、8组合而成

状态码 si为-1时(案例7)

- si 为 -1时，用来直接在棱柱体中定义孔洞。
  - 它标记轮廓的结束及轮廓内孔洞的开始
  - 它还用来指示一个孔洞轮廓的结束及另一个孔洞轮廓的开始。该值前的坐标(x,y,si)必须和轮廓/孔洞的第一点的坐标一致。
  - 如果您使用了掩码值-1，参数列表中的最后一个掩码值则必须是-1，以标记最后一个孔洞的结束。
  - 孔洞必须是分离的，禁止内部相交，否则不能正确得进行着色/渲染。

```
model wire
CPRISM_  myMaterial, myMaterial, myMaterial,
        13, 0.2,
        0, 0, 15 -1,  !!!第1条下边没了 （打开model wire）
        2, 0, 15 -2,  !!!第2条垂直边没了（打开model wire）
        2, 2, 15 -4,  !!!第3条上边没了（打开model wire）
        0, 2, 15 -8,  !!!第4个侧面没了 （注释掉model wire）
        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
```





状态码 si为-1时(案例7-1)

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

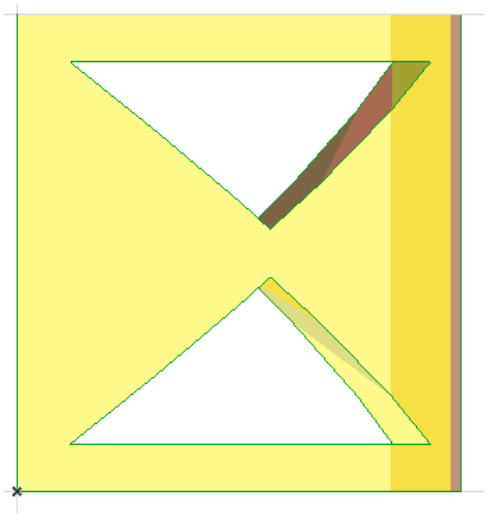
A smooth curved prism, based on the same data structure as the straight CPRISM_ element. The only additional parameter is radius.

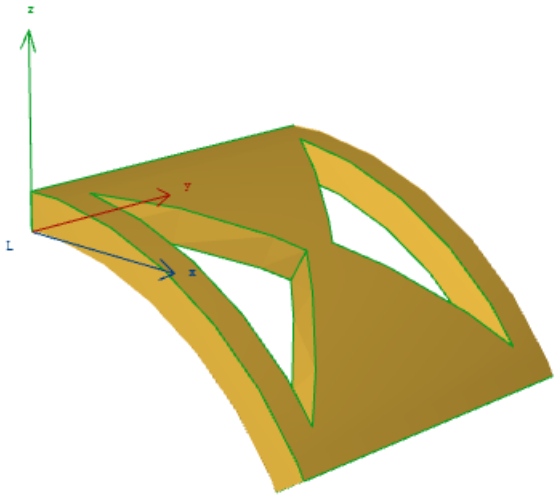
Derived from the corresponding CPRISM_ by bending the x-y plane onto a cylinder tangential to that plane. Edges along the x axis are transformed to circular arcs; edges along the y axis remain horizontal; edges along the z axis will be radial in direction.

See the BWALL_ command for details.

si: status code that allows you to control the visibility of polygon edges and side surfaces. You can also define holes and create segments and arcs in the polyline using special constraints.

```
model wire
BPRISM_  myMaterial, myMaterial, myMaterial,13,
        0.2, 2,         !!! h, r,  !!! 比CPRISM_命令多了一个参数R
        0, 0, 15 -1,  !!!第1条下边没了  （打开model wire）
        2, 0, 15 -2,  !!!第2条垂直边没了（打开model wire）
        2, 2, 15 -4,  !!!第3条上边没了（打开model wire）
        0, 2, 15 -8,  !!!第4个侧面没了  （注释掉model wire）
        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
```



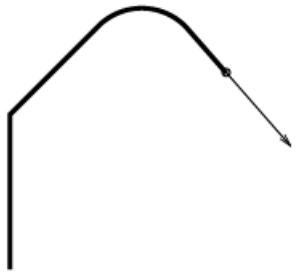


附加状态码 a_code(1)

- 附加状态码允许您使用特殊约束在折线中创建段和弧。 他们指的是下一个段或弧，原始状态码仅在指定的情况下才有效（在附加状态码后加上原始状态码：“a_code+ s”）。
- **注意：当**控制弧的“属性”指令RESOL的值大于8时，在POLY2_语句才生成真正的弧；否则，所有生成的弧将是分段的。

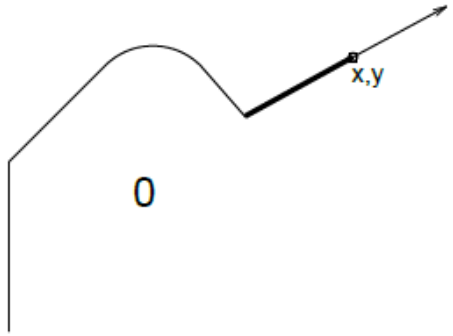
附加状态码 a_code(2)

多义线的前面部分：当前坐标与切线已经被定义



用绝对端点定义线段

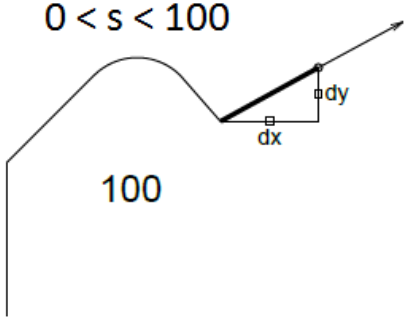
x, y, s !0 < s < 100



用相对端点定义线段

dx, dy, 100 + s

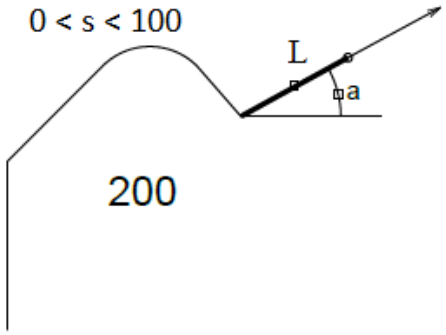
0 < s < 100



用长度和方向角定义线段

L, a, 200+s,

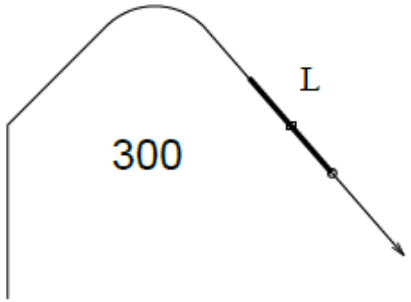
0 < s < 100



用长度和正切方向定义线段

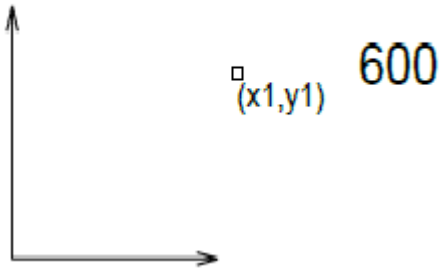
L, 0, 300+s,

0 < s < 100



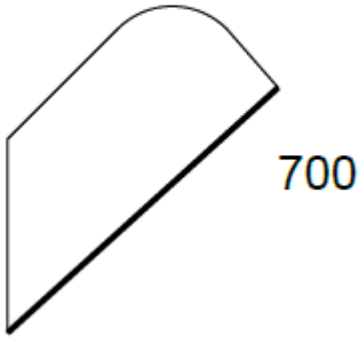
指定起始点

x1, y1, 600,



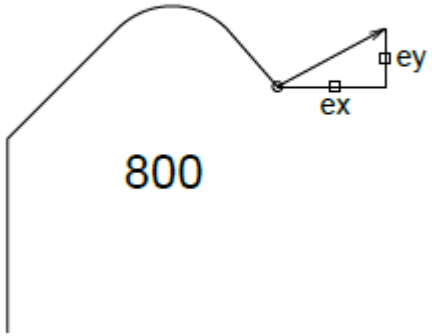
使多段线封闭

0, 0, 700,



设置切线

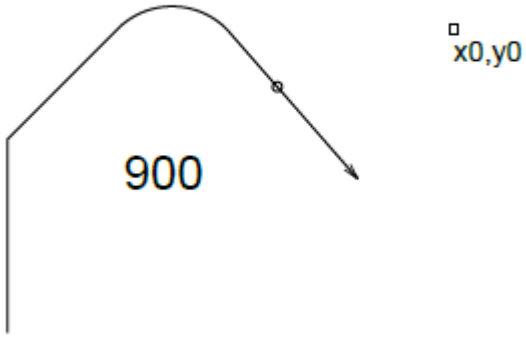
ex, ey, 800,



附加状态码 a_code(3)

设置中心点

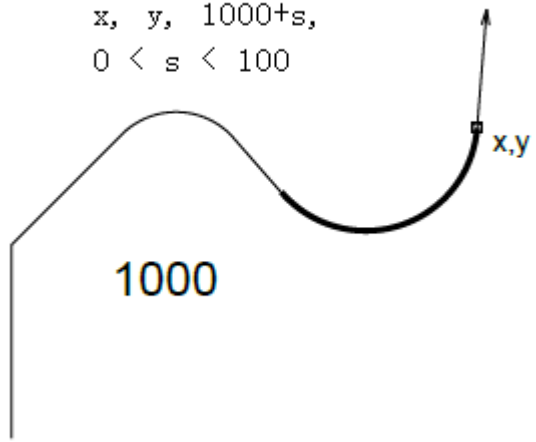
x0, y0, 900,



定义切线弧到端点

x, y, 1000+s,

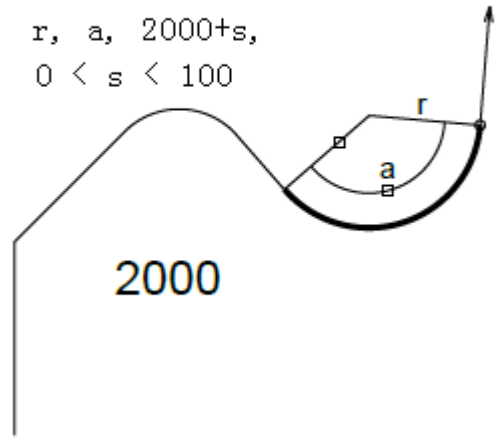
0 < s < 100



用半径和角度定义切线弧

r, a, 2000+s,

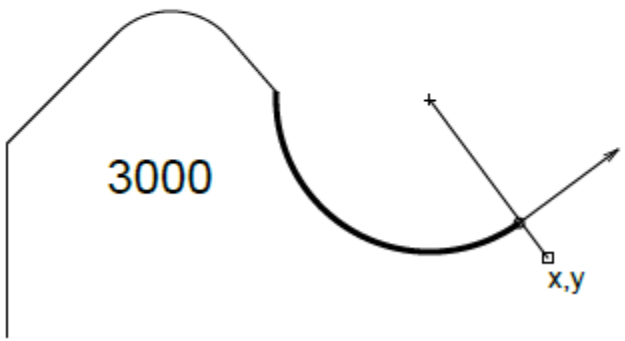
0 < s < 100



用中心点与圆弧上最后的端点定义圆弧

x, y, 3000+s,

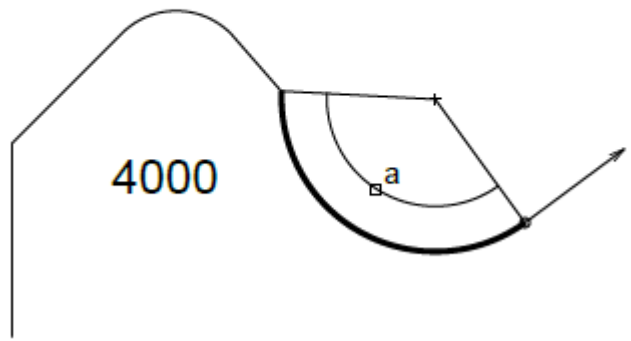
0 < s < 100



使用中心点和角度定义弧

0, a, 4000+s,

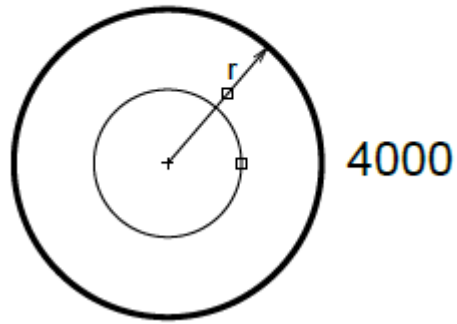
0 < s < 100



用中心点和半径定义一个圆

r, 360, 4000+s,

0 < s < 100

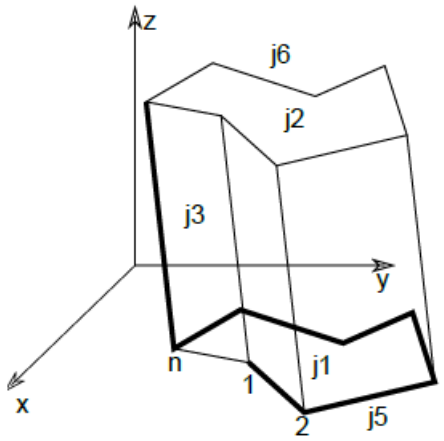


由一条多义线生成的3D实体

- EXTRUDE 『拉伸体』
- PYRAMID 『角椎体』
- REVOLVE 『旋转体』
- REVOLVE{2} 『旋转体』
- REVOLVE{3} 『旋转体』
- REVOLVE{4} 『旋转体』
- REVOLVE{5} 『旋转体』

EXTRUDE『拉伸体』命令

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



一般来说PRISM命令使用基于x-y平面的多义线。

基础之间的位移矢量是(dx,dy,dz)。EXTRUDE命令是PRISM及SLAB命令的一般化（也就是说PRISM及SLAB命令是EXTRUDE命令的特例）。底部多边形不一定要闭合，侧面的边并非始终与x-y平面垂直。底部多边形可能像PRISM_一样包括孔洞，能控制轮廓边的可见性。

- n：多义线的节点数。
- mask：控制底面、顶面及（在开放多义线情况）侧面多边形的存在。

  mask = j1 + 2*j2 + 4*j3 + 16*j5 + 32*j6 + 64*j7 + 128*j8, 这里的j 取值0 或者 1.

  j1: 控制底面是否存在,（base surface is present,）

  j2:控制顶面是否存在,（top surface is present,）

  j3: 侧面（闭合）是否存在，（side (closing) surface is present,）

  j5: 底边是否可见（base edges are visible,）

  j6: 顶边是否可见（top edges are visible.）

  j7: cross-section edges are visible, surface is articulated,

  j8: cross-section edges are sharp, the surface smoothing will stop here in OpenGL and rendering.
- si：棱边的状态，或标记多边形或孔洞的末端。您也可以用附加的状态码值在多义线中定义弧及线段。

  0: lateral edge starting from the node is visible,从节点开始的棱边可见

  1: lateral edges starting from the node are used for showing the contour,从节点开始的棱边用作显示轮廓

  -1: marks the end of the enclosing polygon or a hole, and means that the next node will be the first vertex of another hole.洞口的开始与结束

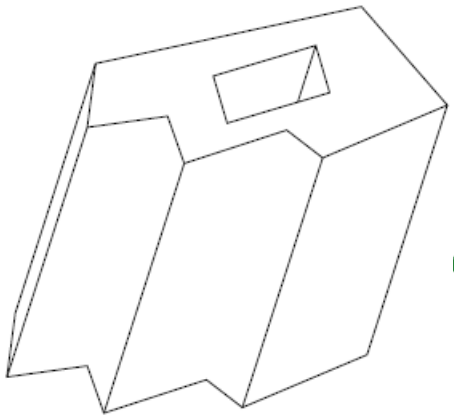
Additional status codes allow you to create segments and arcs in the planar polyline using special constraints

- 参数限制：n > 2

EXTRUDE命令（案例7-2）

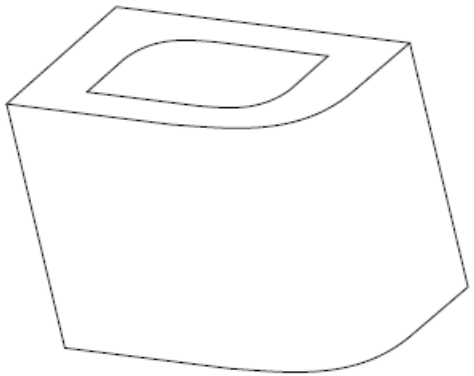
- **EXTRUDE** n, dx, dy, dz, mask, x1, y1, s1, ..., xn, yn, sn

```
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
```

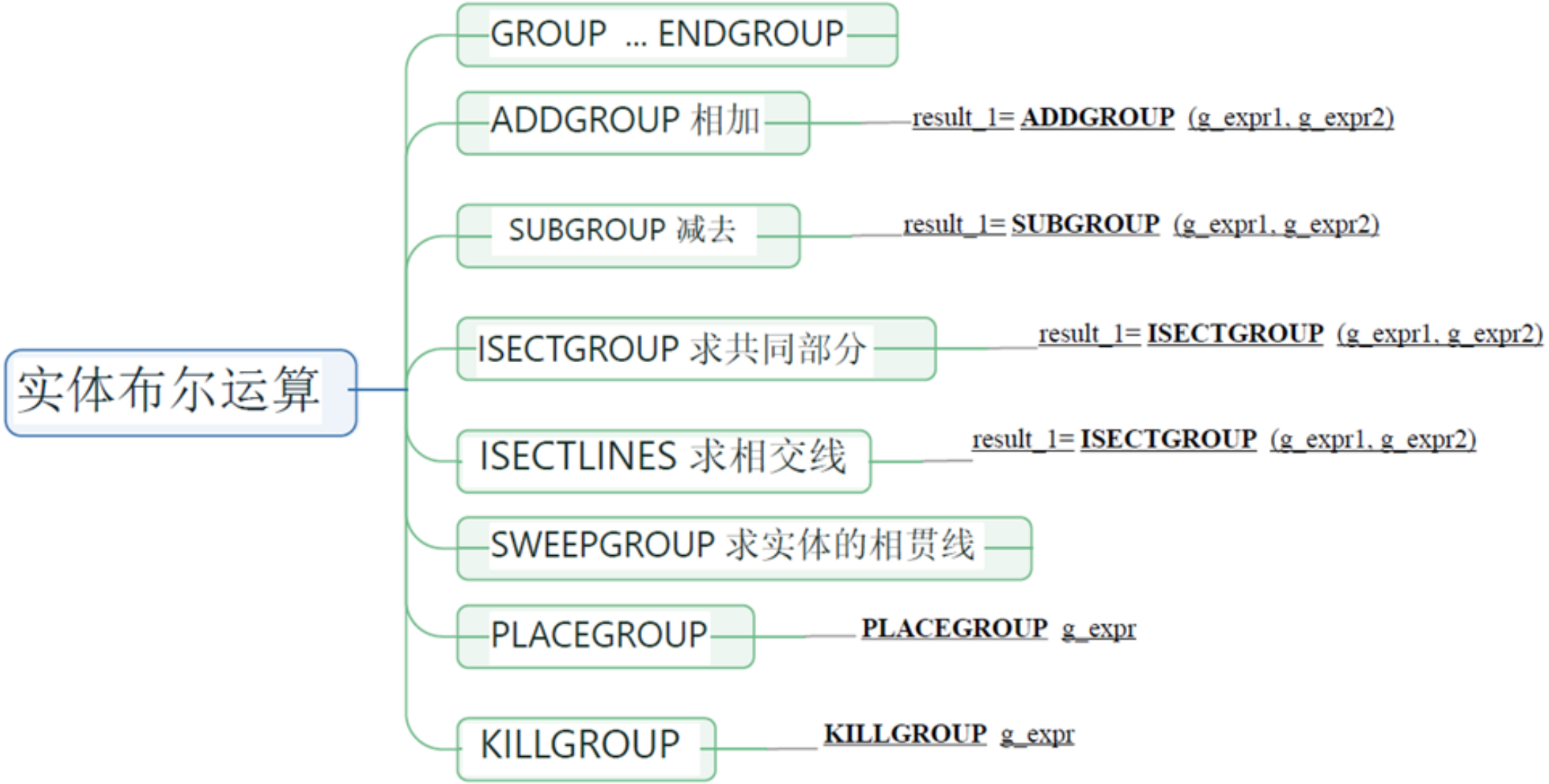


j2=0时顶面没有了

```
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
```



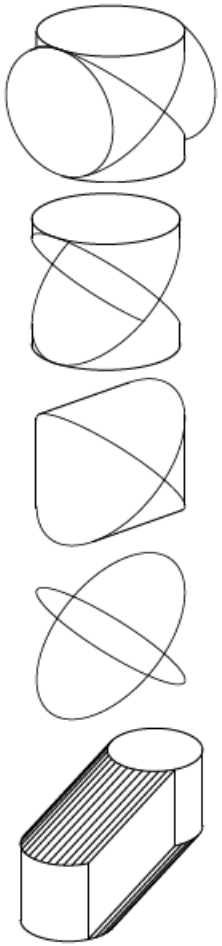
实体布尔运算


实体布尔运算
GROUP ... ENDGROUP
ADDGROUP 相加
result_1= ADDGROUP (g_expr1, g_expr2)
SUBGROUP 减去
result_1= SUBGROUP (g_expr1, g_expr2)
ISECTGROUP 求共同部分
result_1= ISECTGROUP (g_expr1, g_expr2)
ISECTLINES 求相交线
result_1= ISECTGROUP (g_expr1, g_expr2)
SWEEPGROUP 求实体的相贯线
PLACEGROUP
PLACEGROUP g_expr
KILLGROUP
KILLGROUP g_expr

SOLID GEOMETRY COMMANDS

- GDL is capable of performing specialized 3D operations between solids represented by groups. These operations can be one of the following

- ADDGROUP forming the Boolean union of two solids
- SUBGROUP forming the Boolean difference of two solids
- ISECTGROUP forming the Boolean intersection of two solids
- ISECTLINES calculating the intersection lines of two solids
- SWEEPGROUP sweeping a solid along a vector



GROUP - ENDGROUP

- **GROUP** “groupName"
  [statement1 ... statementn]
  ENDGROUP
- 在匹配的GROUP - ENDGROUP语句之间的所有实体将成为“groupName”组的一部分。 组实际上没有真正的生成(被放置)，创建的“组”可以在组操作中使用，也可以显式的调用PLACEGROUP命令来放置。
- 组定义无法嵌套，但可以调用包含组定义的宏，或者通过PLACEGROUP命令使用其他定义好的组。
- 组名在当前脚本中必须是唯一的。 在组定义之外的transformations、 cutplanes等对组内“部件”没有影响; 在组定义内部的transformations,、cutplanes对定义之外的“实体”没有影响。
- 组定义对于属性DEFINEs和SETs（笔，材料，填充）是透明的; 在组定义之前定义/设置的属性和在组定义之中定义/设置的属性对组都是有效的。

ADDGROUP/ SUBGROUP/ ISECTGROUP

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

ADDGROUP/ SUBGROUP/ ISECTGROUP参数说明

- **g_expr1:** 基础组标识符（组名）。
- **g_expr2:** 工具组标识符（组名）。
- **edgeColor:** the color of the new edge when it differs from 0.
- **materialId:** the material of the new face when it differs from 0.
- **materialColor:** the color of the new face when the materialId is 0 and it differs from 0.
- **operationStatus:** status control of the operation.

  operationStatus = j1 + 2*j2, where each j can be 0 or 1.

  j1: generated new edges will be invisible.

  j2: cut polygons of the result inherit material and texture projection from the corresponding polygons of the tool group.

ISECTLINES

- **ISECTLINES** (g_expr1, g_expr2)

  Group operations: addition, subtraction, intersection, intersection lines. The return value is a new group, which can be placed using the PLACEGROUP command, stored in a variable or used as a parameter in another group operation. Group operations can be performed between previously defined groups or groups result from any other group operation. g_expr1, g_expr2 are group type expressions. Group type expressions are either group names (string expressions) or group type variables or any combination of these in operations which result in groups. Note that the operations ADDGROUP, ISECTGROUP and ISECTLINES are symmetric in their parameterization while the order of parameter matters for SUBGROUP.

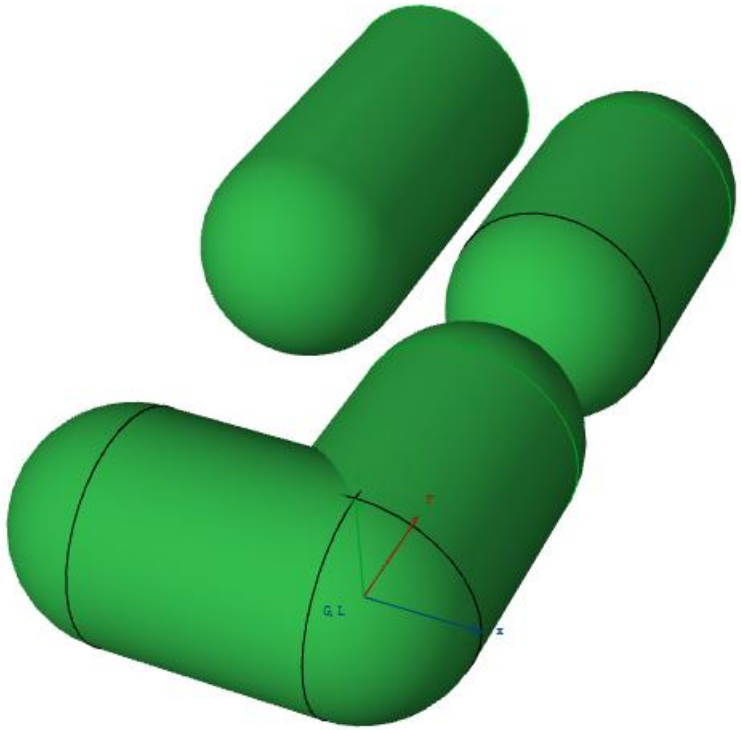
SWEEPGROUP

- **SWEEPGROUP** (g_expr, x, y, z)
  Returns a group that is created by sweeping the group parameter along the given direction. The command works for solid models only.
- **SWEEPGROUP{2}** (g_expr, x, y, z)
  The difference between SWEEPGROUP and SWEEPGROUP{2} is that in the former case the actual transformation matrix is applied again to the direction vector of the sweeping operation with respect to the current coordinate system. (In the case of SWEEPGROUP, the current transformation is applied to the direction vector twice with respect to the global coordinate system.)
  SWEEPGROUP：当前的变换矩阵不但对局部坐标系产生影响，同时对方向矢量参数还会产生一次影响。（平移对方向矢量是不起作用的）
  SWEEPGROUP{2}：当前的变换矩阵只对局部坐标系产生影响。

- **SWEEPGROUP{3}** (g_expr, x, y, z, edgeColor, materialId, materialColor, method)
  This version adds a new method selection to SWEEPGROUP{2} and works for surface models also.
  edgeColor: the color of the new edge when it differs from 0.
  materialId: the material of the new face when it differs from 0.
  materialColor: the color of the new face when the materialId is 0 and it differs from 0.
  method: controls the ending shape of the resulting body.
    0: same as SWEEPGROUP{2}, both ends come from the originating body,
    1: the start comes from the originating body, the sweep end is flat
- **SWEEPGROUP{4}** (g_expr, x, y, z, edgeColor, materialId, materialColor, method, status)
  This version adds a new status parameter to SWEEPGROUP{3}.
  status: Controls attributes of the result.
    status = 2*j2, where each j can be 0 or 1.
    j2: Keep per-polygon texture mapping parameters on the sweeped result (see the PGON command for details).

SWEEPGROUP(案例8)

```
GROUP "the_sphere" !!! 定义组
   SPHERE 1
ENDGROUP
ROTz  90
!!! 当前的矩阵变化对方向矢量产生了2次影响
PLACEGROUP  SWEEPGROUP("the_sphere", 2, 0, 0)
!!! 当前的矩阵变化对方向矢量产生了1次影响
PLACEGROUP  SWEEPGROUP{2} ("the_sphere", 2, 0, 0)
ADDX 4
!!! SWEEPGROUP{3} (g_expr, x, y, z, edgeColor, materialId, materialColor, method)
PLACEGROUP SWEEPGROUP{3} ("the_sphere", 2, 0, 0, 4, 0, 4, 0)
ADDY 2.5
!!!method取值1时终端为平面
PLACEGROUP SWEEPGROUP{3} ("the_sphere", 2, 0, 0, 4, 0, 4, 1)
del 3
```



CREATEGROUPWITHMATERIAL

- **CREATEGROUPWITHMATERIAL** (g_expr, repl_directive, pen, material)

  Returns a group that is created by replacing all pens and/or materials in group g_expr.

  g_expr:

  group expression identifying the base group.

  repl_directive:

  repl_directive = j1 + 2*j2 + 4*j3 + 8*j4, where each j can be 0 or 1.

  j1: replace pen,

  j2: replace material,

  j4: make edges invisible.

  pen:

  replacement pen index.

  material:

  replacement material index.

PLACEGROUP / KILLGROUP

- **PLACEGROUP** g_expr

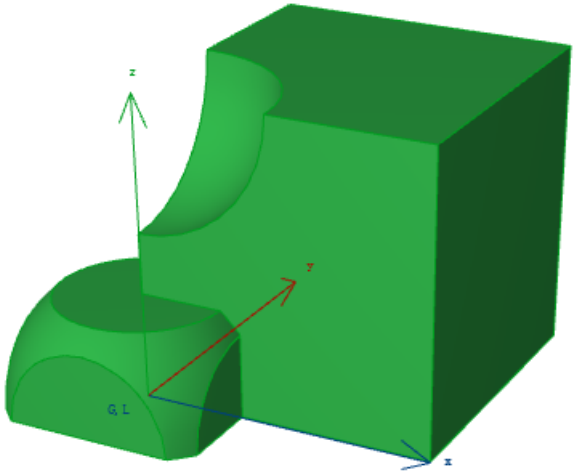
  Placing a group is the operation in which bodies are actually generated. Cutplanes and transformations are effective, the group expression is evaluated and the resulting bodies are stored in the 3D data structure.

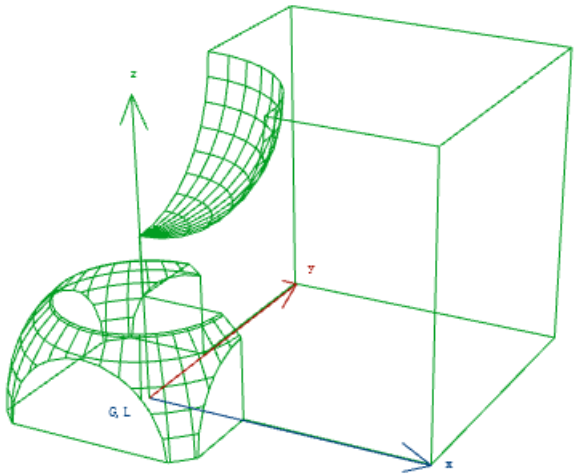
- **KILLGROUP** g_expr

  Clears the bodies of the specified group from the memory. After a KILLGROUP operation the group becomes empty. The names of killed groups cannot be reused in the same script. Clearing is executed automatically at the end of the interpretation or when returning from macro calls. For performance reasons this command should be used when a group is no longer needed.

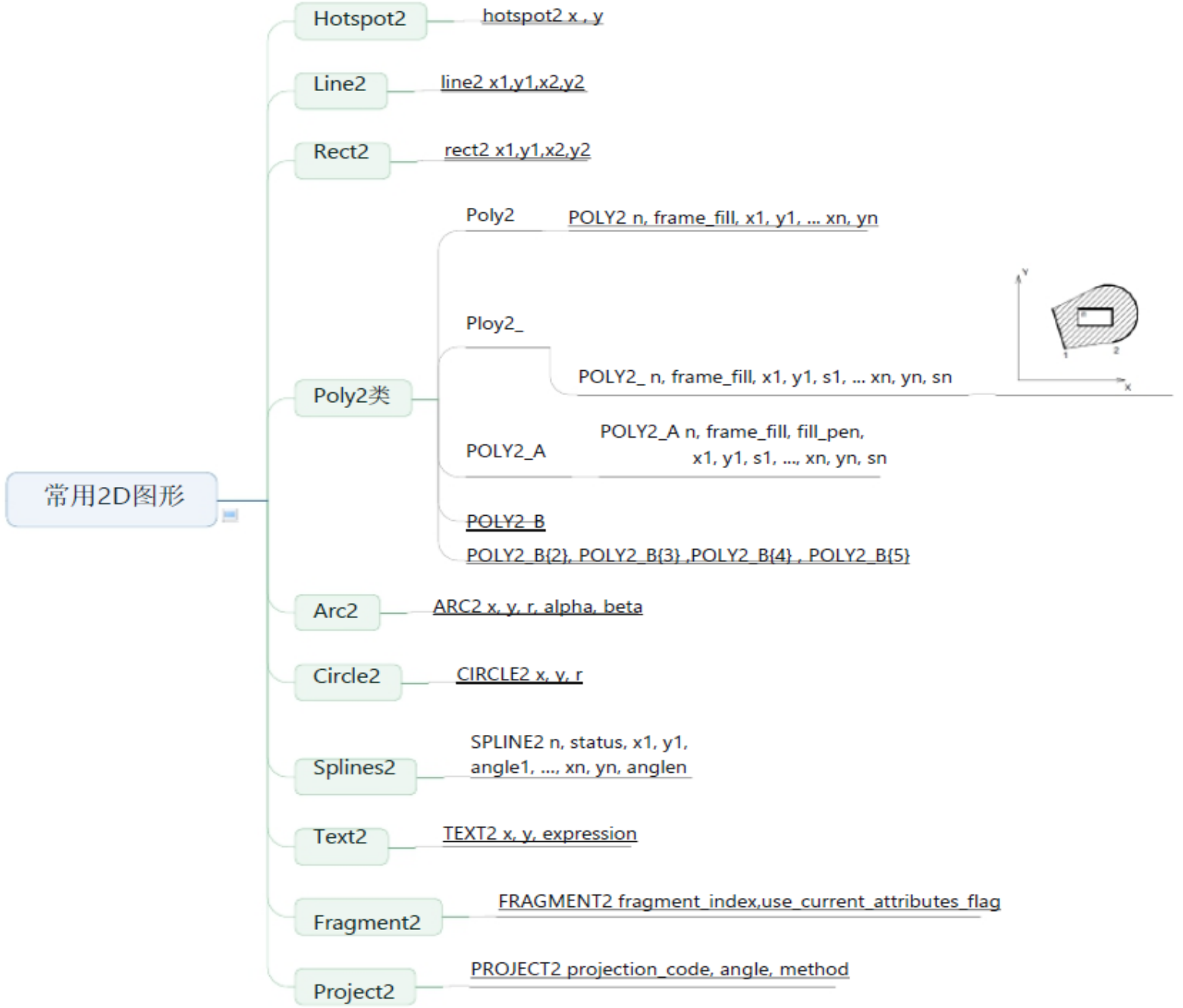
组的创建与操作(案例8-1)

```
GROUP “box”     !!! 创建组
   BRICK 1, 1, 1
ENDGROUP
GROUP "sphere"
   ADDZ 1
   SPHERE 0.45
   DEL 1
ENDGROUP
GROUP "semisphere"
   ELLIPS 0.45, 0.45
ENDGROUP
GROUP "brick"
   ADD -0.35, -0.35, 0
   BRICK 0.70, 0.70, 0.35
   DEL 1
ENDGROUP
! Subtracting the "sphere" from the "box"
result_1=SUBGROUP("box", "sphere")
! Intersecting the "semisphere" and the "brick"
result_2=ISECTGROUP("semisphere", "brick")
! Adding the generated bodies
result_3=ADDGROUP(result_1, result_2)
PLACEGROUP result_3   !!! 放置组，模型数据添加到BIM数据库中
KILLGROUP “box“         !!! 从内存中删除掉（释放占有的内存）
KILLGROUP "sphere"
KILLGROUP "semisphere"
KILLGROUP "brick"
```





常用2D图形


常用2D图形
Hotspot2
hotspot2 x , y
Line2
line2 x1,y1,x2,y2
Rect2
rect2 x1,y1,x2,y2
Poly2类
Poly2
POLY2 n, frame_fill, x1, y1, ... xn, yn
Ploy2_
POLY2_ n, frame_fill, x1, y1, s1, ... xn, yn, sn
POLY2_A
POLY2_A n, frame_fill, fill_pen,
x1, y1, s1, ..., xn, yn, sn
POLY2_B
POLY2_B{2}, POLY2_B{3} ,POLY2_B{4} , POLY2_B{5}
Arc2
ARC2 x, y, r, alpha, beta
Circle2
CIRCLE2 x, y, r
Splines2
SPLINE2 n, status, x1, y1,
angle1, ..., xn, yn, anglen
Text2
TEXT2 x, y, expression
Fragment2
FRAGMENT2 fragment_index,use_current_attributes_flag
Project2
PROJECT2 projection_code, angle, method

2D图形---HOTSPOT2

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

  unID: the unique identifier of the hotspot in the 2D Script. Useful if you have a variable number of hotspots.在2d脚本中热点的唯一标识符，当有多个热点时有用。

  paramReference:parameter that can be edited by this hotspot using the graphical hotspot based parameter editing method.使用基于图形热点的参数编辑方法，由该热点进行编辑的对应参数。

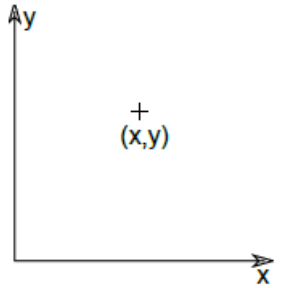
  displayParam: parameter to display in the information palette when editing the paramRefrence parameter. Members of arrays can be passed as well.当编辑paramRefrence所指参数时，显示在信息框中的参数，数组成员也可被传递。

  customDescription: custom description string of the displayed parameter in the information palette. When using this option,displayParam must be set as well (use paramReference for default).

  flags: hotspot's type + hotspot's attribute:

  type:

  1: length type editing, base hotspot,
  2: length type editing, moving hotspot,
  3: length type editing, reference hotspot (always hidden),
  4: angle type editing, base hotspot,
  5: angle type editing, moving hotspot,
  6: angle type editing, center of angle (always hidden),
  7: angle type editing, reference hotspot (always hidden).



  attribute: Can be zero or:

  attribute = 128*j_8 + 256*j_9 + 512*j_{10} + 1024*j_{11}, where each j can be 0 or 1.
  j_8: hide hotspot (meaningful for types: 1,2,4,5),
  j_9: editable base hotspot (for types: 1,4),
  j_{10}: reverse the angle in 2D (for type 6),
  j_{11}: use paramReference value as meters in paper space.

2D图形--- HOTLINE2、 HOTARC2

- **HOTLINE2** x1, y1, x2, y2, unID

  Status line definition between two points. Status line is a line which is recognized by the intelligent cursor but it is not visible in itself. Can have a unique ID for associative dimensioning purpose.

  状态线是通过两个点定义的。状态线可以被智能光标识别，但是他本身是不可见的。unID用于尺寸标注。

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

  Status arc definition with its centerpoint at (x, y) from the angle startangle to endangle, with a radius of r. Status arc is an arc which is recognized by the intelligent cursor but it is not visible in itself. Can have a unique ID for associative dimensioning purpose.

  状态弧是通过其圆心点、起始角、终止角和半径来定义的。状态弧可以被智能光标识别，但是他本身是不可见的。unID用于尺寸标注。

2D图形

- **LINE2**『线2D』

  LINE2 x1, y1, x2, y2

- **RECT2**『矩形**2D**』

  RECT2 x1, y1, x2, y2通过两个对角点来定义矩形。矩形的两条边平行X、Y轴。

- POLY2 多边形 2D』

  POLY2 n, frame_fill, x1, y1, ... , x1, y1, ... , x1, y1, ... xn , yn 一个有 n个节点的开放或者闭合多边形。

  参数限制： n >= 2

  frame_fill = j1 + 2*j2 + 4*j3

  这里的j1、j2、j3可以是0或1。

  j1（1）：仅轮廓

  j2（2）：仅填充

  j3（4）：闭合一个开放的多边形

2D图形

- **POLY2_**

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

  类似于普通的POLY2语句，但任何边都可以忽略不计。如果si= 0，从（xi,yi）顶点开始的边将被忽略。如果si= 1，则应该显示顶点。si= -1是用来直接定义孔洞的。您也可以用附加的状态码值在多义线中定义弧及线段。

  frame_fill = $j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 32*j_6 + 64*j_7$,这里的j 取值0 或者 1.

  j_1: 绘制轮廓线draw contour,

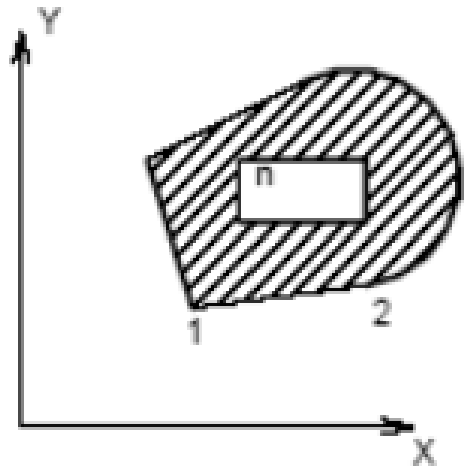
  j_2: 绘制填充draw fill,

  j_3:闭合一个开放的多边形close an open polygon,

  j_4: local fill orientation,

  j_6: fill is cut fill (default is drafting fill),

  j_7: fill is cover fill (only if j6 = 0, default is drafting fill).



2D图形----POLY2_的升级版本

- **POLY2_A** n, frame_fill, fill_pen,x1, y1, s1, ..., xn, yn, sn
- **POLY2_B** n, frame_fill,fill_pen, fill_background_pen,x1, y1, s1, ..., xn, yn, sn
- **POLY2_B{2}** n, frame_fill,fill_pen, fill_background_pen,fillOrigoX, fillOrigoY, fillAngle,x1, y1, s1, ..., xn, yn, sn
- **POLY2_B{3}** n, frame_fill,fill_pen, fill_background_pen,fillOrigoX, fillOrigoY,mxx, mxy, myx, myy, x1, y1, s1, ..., xn, yn, sn
- **POLY2_B{4}** n, frame_fill,fill_pen, fill_background_pen,fillOrigoX, fillOrigoY,mxx, mxy, myx, myy,gradientInnerRadius,x1, y1, s1, ..., xn, yn, sn
- **POLY2_B{5}** n, frame_fill, fillcategory, distortion_flags,fill_pen, fill_background_pen,fillOrigoX, fillOrigoY,mxx, mxy, myx, myy,gradientInnerRadius,x1, y1, s1, ..., xn, yn, sn

2D图形

- **ARC2** x, y, r, alpha, beta

  An arc with its centerpoint at (x, y) from the angle alpha to beta, with a radius of r.Alpha and beta are in degrees.

- **CIRCLE2** x, y, r

  A circle with its center at (x, y), with a radius of r.

- **SPLINE2** n, status, x1, y1,angle1, ..., xn, yn, anglen
- **SPLINE2A** n, status, x1, y1, angle1, length_previous1, length_next1,

  ...

  xn, yn, anglen, length_previousn,length_nextn

- **PICTURE2** expression, a, b, mask
- **PICTURE2{2}** expression, a, b, mask
- **TEXT2** x, y, expression
- **RICHTEXT2** x, y, textblock_name

2D图形---- PROJECT2、PROJECT2{2}

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

  Creates a projection of the 3D script in the same library part and adds the generated lines to the 2D parametric symbol. The 2nd version PROJECT2{2}, together with a previous [SET] FILL command, allows the user to control the fill background, origin and direction of the resulting drawing from the 2D script. The SET FILL 0 shortcut to get an empty fill does not work in this case, you need to reference an actual empty fill.

2D图形---- PROJECT2、 PROJECT2{2}

projection_code: the type of projection.

- 3: Top view,
- 4: Side view,
- 6: Frontal axonometry,
- 7: Isometric axonometry,
- 8: Monometric axonometry,
- 9: Dimetric axonometry,
- -3: Bottom view,
- -6: Frontal bottom view,
- -7: Isometric bottom view,
- -8: Monometric bottom view,
- -9: Dimetric bottom view.

angle: the azimuth angle set in the 3D Projection Settings dialog box.

method: the chosen imaging method.

- 1: wireframe,
- 2: hidden lines (analytic),
- 3: shading,
- 16: addition modifier: draws vectorial hatches (effective only in hidden line and shaded mode),
- 32: addition modifier: use current attributes instead of attributes from 3D (effective only in shading mode),
- 64: addition modifier: local fill orientation (effective only in shading mode),
- 128: addition modifier: lines are all inner lines (effective only together with 32). Default is generic,
- 256: addition modifier: lines are all contour lines (effective only together with 32, if 128 is not set). Default is generic,
- 512: addition modifier: fills are all cut (effective only together with 32). Default is drafting fills,
- 1024: addition modifier: fills are all cover (effective only together with 32, if 512 is not set). Default is drafting fills.

2D图形---- PROJECT2、 PROJECT2{2}

BackgroundColor: background color of the fill.

fillOrigoX: X coordinate of the fill origin.

fillOrigoY: Y coordinate of the fill origin.

filldirection: direction angle of fill.

Note: the [SET] FILL command is effective for PROJECT2{2}

Compatibility note: using PROJECT2 with method bit 32 not set and method bit 3 set (shading), the model being cut with the CUTPOLYA command without status bit 2 set (generating cut polygons) resulting cut polygon attributes can be different. Cut polygons will be generated with attributes defined by the SECT_FILL command in the 3D script.

2D图形---- PROJECT2参数说明

投影_代码：投影的类型

3：俯视图

4：侧视图

6：主轴测

7：等轴测

8：单一轴测

9：四边轴测

-3：仰视图

-6：正面仰视图

-7：等仰视图

-8：单一仰视图

-9：四边仰视图

角度：在3D投影设置对话框设置的方位角。

方法：所选的成像方法

1：线框

2：消隐线（解析）

3：着色

16：加法修改量，绘制矢量图案填充（只在消隐线及着色模式中有效）

32：加法修改量，使用当前属性，而不使用来自3D的属性（只在着色模式中有效）

64：加法修改量，局部填充方向（只在着色模式中有效）

128：加法修改量：线都是内线（只与32一起有效），默认是普通线。

256：加法修改量：线都是轮廓线（如果不设置128，则只与32一起有效），默认是普通线。

512：加法修改量：填充都是剪切填充（只与32一起有效），默认是绘制填充。

1024：加法修改量：填充都是覆盖填充（如果不设置512，则只与32一起有效），默认是绘制填充。

背景颜色：填充的背景颜色

填充原点X：填充原点的X坐标

填充原点Y：填充原点的Y坐标

填充方向：填充的方向角

- 注释：SET FILL对PROJECT2{2}是有效的

2D图形---- PROJECT2{3}

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

  Creates a projection of the 3D script in the same library part and adds the generated lines to the 2D parametric symbol. The third version,PROJECT2{3}, adds the possibility to define which parts of the projected model are required and to control separately the attributes of the cut and view part, including the line type. You can also generate the projection with actual parameters set in the command.

2D图形---- PROJECT2{3}

method: the chosen imaging method.

1: wireframe,
2: hidden lines (analytic),
3: shading,
16: addition modifier: draws vectorial hatches (effective only in hidden line and shaded mode),
32: addition modifier: use current attributes instead of attributes from 3D (effective only in shading mode),
64: addition modifier: local fill orientation (effective only in shading mode),
128: addition modifier: lines are all inner lines (effective only together with 32). Default is generic.
256: addition modifier: lines are all contour lines (effective only together with 32, if 128 is not set). Default is generic.
512: addition modifier: fills are all cut (effective only together with 32). Default is drafting fills.
1024: addition modifier: fills are all cover (effective only together with 32, if 512 is not set). Default is drafting fills.
2048: addition modifier: modifiers 16, 32, 64, 128, 256, 512, 1024 and fill attribute parameters are effective only for the view part of the projection. By default they are effective for all parts.
4096: addition modifier: modifiers 16, 32, 64, 128, 256, 512, 1024 and fill attribute parameters are effective only for the cut part of the projection. By default they are effective for all parts.
8192: addition modifier: cut fills are slanted.
16384: addition modifier: enables transparency for transparent surfaces. Note that transparency in this case means full transparency for surfaces with transmittance greater than 50, everything else is non-transparent.

Known limitation: lines of the cut part cannot be treated separately, only all lines together can be set to be inner or contour.

parts: defines the parts to generate. The 1+2+4+8+16+32 value means all parts.

parts = j_1 + 2*j_2 + 4*j_3 + 8*j_4 + 16*j_5 + 32*j_6, where each j can be 0 or 1.

The j1, j2, j3, j4, j5, j6 numbers represent whether the corresponding parts of the projected model are present (1) or omitted (0):

j_1: cut polygons (with default fill attributes defined by SECT_FILL) (effective only in shading mode),
j_2: cut polygon edges,
j_3: view polygons,
j_4: view polygon edges,
j_5: project 3D hotspots as static 2D hotspots,
j_6: project 3D hotlines and hotarcs (including related 3D hotspots converted to static 2D hotspots).

2D图形---- FRAGMENT2

- **FRAGMENT2** fragment_index, use_current_attributes_flag
- **FRAGMENT2** ALL, use_current_attributes_flag

  fragment_index: The fragment with the given index is inserted into the 2D Full View with the current transformations. 使用当前转换矩阵插入给定片断索引的片断。

  ALL: If ALL is specified, all fragments are inserted.如果指定了ALL，则插入所有片断。

  use_current_attributes_flag: defines whether or not the current attributes will be used. 定义当前属性是否被使用。

    0: the fragment appears with the color, line type and fill type defined for it,使用片断的原来的颜色，线型和填充类型。

    1: the current settings of the script are used instead of the color, line type and fill type of the fragment.使用当前脚本代码的设置值代替片断的原来的颜色，线型和填充类型。

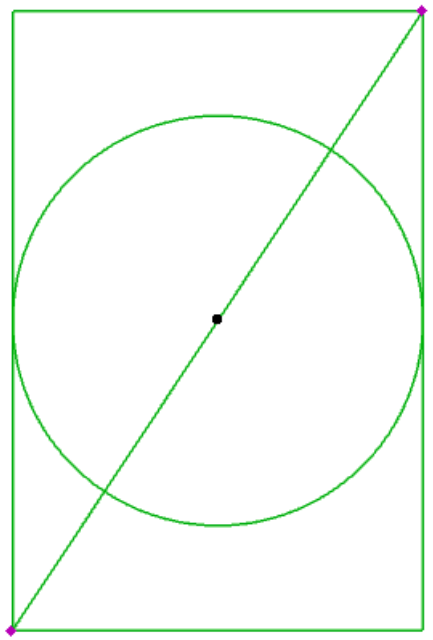
2D指令程序(案例9)

```
LINE2  x1, y1, x2, y2
RECT2  x1, y1, x2, y2
CIRCLE2   (x1+ x2)/2,  (y1+y2)/2, MIN(ABS(x2-x1),ABS(y2-y1))/2

unID = 1
hotspot2  (x1+ x2)/2,  (y1+y2)/2, unID : unID = unID +1

!!! x1
HOTSPOT2  0,     y1, unid , x1, 1+128 : unid = unid +1
HOTSPOT2  x1 ,  y1, unid , x1, 2 : unid = unid +1
HOTSPOT2 -1,     y1, unid , x1, 3 : unid = unid +1
!!! y1
HOTSPOT2  x1, 0,    unid , y1, 1+128 : unid = unid +1
HOTSPOT2  x1, y1,  unid , y1, 2 : unid = unid +1
HOTSPOT2  x1,-1,    unid , y1, 3 : unid = unid +1

!!! x2
HOTSPOT2  0,     y2, unid , x2, 1+128 : unid = unid +1
HOTSPOT2  x2 ,  y2, unid , x2, 2 : unid = unid +1
HOTSPOT2 -1,     y2, unid , x2, 3 : unid = unid +1
!!! y2
HOTSPOT2  x2, 0,    unid , y2, 1+128 : unid = unid +1
HOTSPOT2  x2, y2,  unid , y2, 2 : unid = unid +1
HOTSPOT2  x2,-1,    unid , y2, 3 : unid = unid +1
```

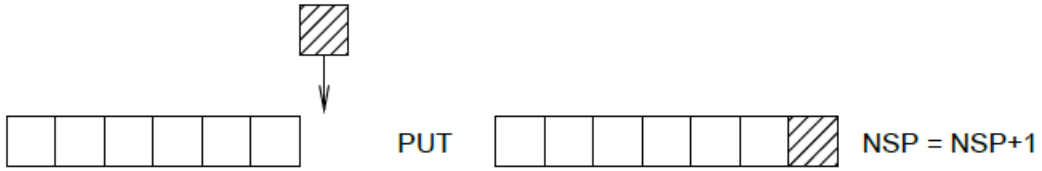


参数缓存器

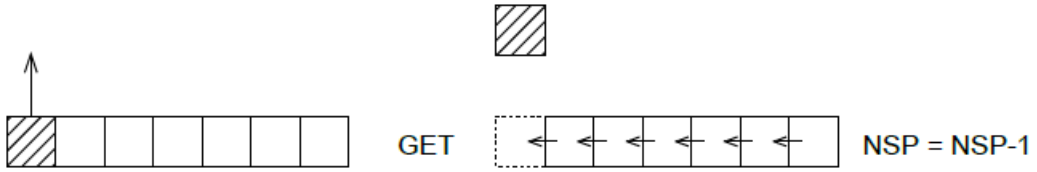
- 参数缓存器是一个内置的数据结构，如果某些值（如坐标）可以通过规则明确的数学表达式进行描述，则可能用到这个内置的数据结构。例如，如果您想存储某些变量的一组当前值，就可以把这组值写入到参数缓存器中，后面就可以直接从缓存器中读取这些值，非常方便。
- 参数缓存器是一个无限长的数组，使用PUT命令，您可以在当中存储数值。PUT命令在缓冲器的末尾存储指定的值。这些值以后可按照它们存入的顺序（用GET和USE命令）来使用（即先使用第一个存储的值）。GET(n)或USE(n)命令相当于用逗号隔开的n值。这样，它们可以用在任何需要n值的GDL参数列表。

参数缓存器

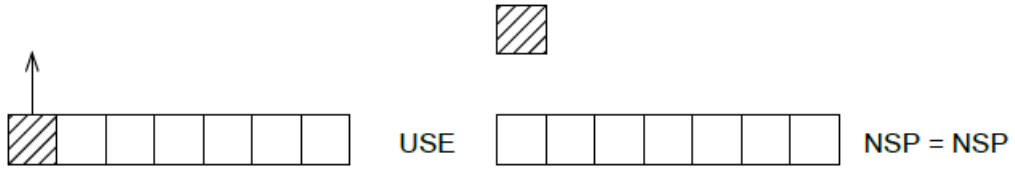
- PUT *expression [, expression, ...]*



- GET(n)



- USE(n)



- NSP： *返回保存在内部缓存器中的参数的个数。*

参数缓存器程序(案例10)

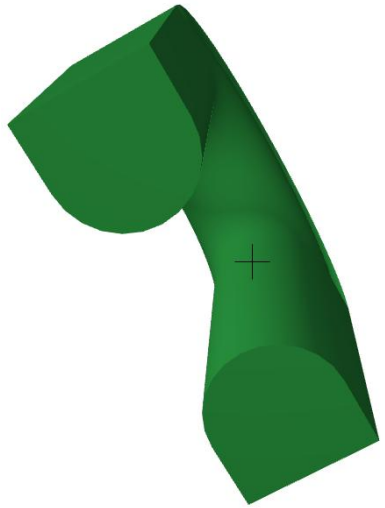
```
!!!Example: Using the parameter buffer:
r=2: b=6: c=4: d=10
n=12
s=180/n
FOR t=0 TO 180 STEP s
   PUT r+r*COS(T), c-r*SIN(t), 1
NEXT t
FOR i=1 TO 2
   EXTRUDE 3+NSP/3, 0,0,d, 1+16,
      0, b, 0,
      2*r, b, 0,
      USE(NSP),
      0, b, 0
   MULY -1
NEXT i
DEL 1
ADDZ d
REVOLVE 3+NSP/3, 180, 0,
   0, b, 0,
   2*r, b, 0,
   GET(NSP),
   0, b, 0
```

未使用参数缓存器(案例10)

```
!!!The full description:
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

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
```

```
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
```



GDL中的属性指令(2D/3D)

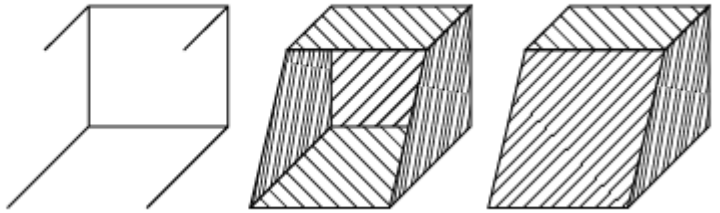
- 2D、3D中都可用的指令：
  - [LET] varName = n　　!赋值
  - RADIUS radius_min, radius_max ！ RESOL 取值： 当r< radius_min 时取6；
    当r> radius_max时取36， 否则取 6+30*(r- radius_min) /(radius_max- radius_min)
  - RESOL n　　！ 设置精度 3<=n<=36
  - TOLER d　　！ 设置公差
  - PEN n　　！ 画笔的颜色 0 < n <= 255
  - LINE_PROPERTY　expr　　! Expr取值0/1/2,依次表示普通线/在内部/轮廓线
  - [SET] STYLE name_string　!设置字体
  - [SET] STYLE index　　!设置字体

GDL中的属性指令(3D)

- 只用于3D中的指令：
  - MODEL WIRE ！线框模式
  - MODEL SURFACE ！表面模式
  - MODEL SOLID ！实体模式

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

下面是用一个面切割后的情况：



  - [SET] MATERIAL name_string / [SET] MATERIAL index ！指定表面材质， Surfaces in the BPRISM_, CPRISM_,FPRISM_, HPRISM_, SPRISM_, CSLAB_, CWALL_, BWALL_, XWALL_, CROOF_, MASS, bodies are exceptions to this rule. if there is no MATERIAL statement in the script, Default: MATERIAL 0
  - **SECT_FILL** fill, fill_background_pen,fill_pen, contour_pen
  - **SECT_ATTRS** fill, fill_background_pen,fill_pen, contour_pen [, line_type]
  - **SHADOW** casting [, catching] ! casting :Controls the shadow casting of the elements in PhotoRendering and in vectorial shadow casting. Catching:This optional parameter controls the appearance of shadows (from other bodies) on surfaces. ! If shadow casting isn't specified, the default will be AUTO

GDL中的属性指令(2D)

- 只用于2D中的指令：
  - **DRAWINDEX** number !Defines the drawing order of 2D Script elements. Elements with a smaller drawindex will be drawn first. *Restriction of parameters:* 0 < number <= 50; (In the current version of GDL only the 10, 20, 30, 40 and 50 DRAWINDEX values are valid. Other values will be rounded to these.) If no DRAWINDEX directive is present, the default drawing order is the following:1 Figures 2 Fills 3 Lines 4 Text elements.
  - [**SET**] **FILL** name_string / [**SET**] **FILL** index !All the 2D polygons generated afterwards will represent that fill until the next SET FILL statement. The index is a constant referring to a fill stack in the internal data structure. This stack is modified during GDL analysis and can also be modified from within the program. The use of the index instead of the fill name is only recommended with the prior use of the IND function.*Default:* SET FILL 0 i.e., empty fill, if there is no SET FILL statement in the script.
  - [**SET**] **LINE_TYPE** name_string / [**SET**] **LINE_TYPE** index !All the 2D lines generated afterwards will represent that line type (in lines, arcs, polylines) until the next SET LINE_TYPE statement. The index is a constant that refers to a line type stack in the internal data structure. This stack is modified during GDL analysis and can also be modified from the program. The use of the index instead of the line type name is only recommended with the prior use of the IND function. Default: SET LINE_TYPE 1 i.e., solid line, if there is no SET LINE_TYPE statement in the script.

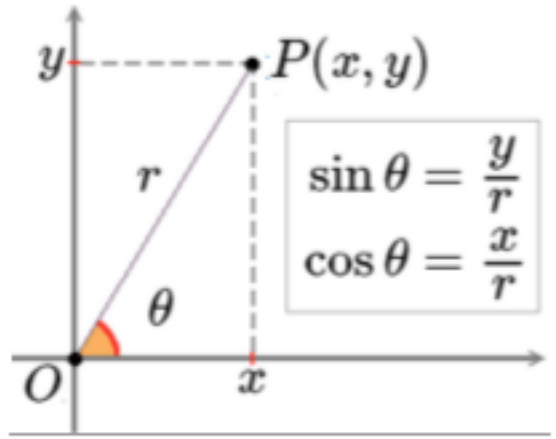
GDL函数

常用的算术函数

- **ABS**『绝对值』返回x的绝对值（如果x是整数，则返回整数，否则返回实数）
- **CEIL**『升限（整数）』返回不小于x的最小整数值（总是整数）（例如：CEIL(1.23) = 2； CEIL (-1.9) = -1）。
- **INT**『整数』返回x的整数部分（总是整数）。（例如：INT(1.23) = 1，INT (-1.23) = -2）。
- **FRA**『小数』返回x的小数部分（如果x是整数，则返回整数，否则返回实数）。（例如：FRA(1.23) = 0.23，FRA (-1.23) = 0.77）。
- **ROUND_INT**『取整』返回x四舍五入的整数部分。 ‘i= ROUND_INT (x)’表达式相当于下列的脚本：IF x < 0.0 THEN i= INT (x -0.5) ELSE i= INT (x + 0.5)
- **SGN**『符号』如果x是整数，则返回+1整数，如果x是负数，则返回-1整数，否则返回0整数
- **SQR**『平方根』返回x的平方根（总是实数）。

三角函数

- **ACS**『反余弦』返回x的反余弦。（-1.0 <= x <= 1.0；0°<= ACS(x) <= 180°）。
- **ASN**『反正弦』返回x的反正弦。（-1.0 <= x <= 1.0；-90°<= ASN (x) <= 90°）。
- **ATN**『反正切』返回x的反正切。（-90°<= ATN (x) <= 90°）。
- **COS**『余弦』返回x的余弦。
- **SIN**『正弦』返回x的正弦。
- **TAN**『正切』返回x的正切。
- **PI**『圆周率』**PI** = 3.1415926...。
- 以上所有返回值都是实数。



关于“宏”(1)

- 尽管您可能需要的3D对象总是可以分解为复杂或原始元素，但有时候，特定于某些应用程序需要定义这些复杂元素。对这些复杂的元素进行单独定义的GDL称为宏。
- GDL宏有自己的环境（环境变量），这取决于宏的调用顺序。 当前的MODEL，RADIUS，RESOL，TOLER，PEN，LINE_TYPE，MATERIAL，FILL，STYLE，SHADOW选项和当前转换矩阵都在定义的宏中有效。 您可以使用或修改这些选项，但修改只会在本地(宏内部)产生影响。 它们不会在宏被调用的级别上其作用。
- 给宏调用的参数意味着宏级别上的隐式值赋值。 参数A和B通常用于调整对象大小。

关于“宏”(2)

- 宏定义语法：

  CALL macro_name_string [,]

  PARAMETERS [**ALL**][name1=value1, ..., namen=valuen][[,]

  RETURNED_PARAMETERS r1, r2, ...]

  - **macro_name_string:** 字符串，已有库部件的名称
    - 宏名称不能超过31个字符。
    - 宏名称可以是字符串常量，字符串变量或参数。 字符串操作不能与宏调用一起用作宏名称。警告：如果将字符串变量或参数用作宏名称，则调用的宏可能不包括在归档项目中。
    - 要让GDL知道依赖关系，请对每个可能的宏名称使用FILE_DEPENDENCE命令。 除非符合标识符的定义，否则宏名称必须放在引号（单引号/双引号）之间，即以字母或‘_’或‘～’字符打头，并且仅包含字母，数字和‘_’和‘～’字符，CALL命令中使用的引号时必须在开头和结尾使用相同的引号，并且不能与宏名称中的任何字符相同。
    - 宏名称本身也可以用作命令，而不使用CALL关键字。

关于“宏”(3)

- **PARAMETERS:** the actual parameter list of the macro can follow
  - 被调用的宏的参数名称可以以任何顺序列出，同时使用“=”符号和实际值。
  - 您可以在此处使用字符串类型表达式，但只能为被调用宏的字符串类型参数赋予一个字符串值。
  - 必须为数组参数提供完整的数组值。
  - 如果在调用宏中找不到参数列表中的参数名称，将收到错误消息。
  - 在宏调用中未列出的被调用宏的参数将被赋予其原始默认值，如在库中称为宏的部分所定义的。

GDL中的常用命令汇总

3D Use Only

ADDX
ADDY
ADDZ
ADD
MULX
MULY
MULZ
MUL
ROTX
ROTY
ROTZ
ROT
XFORM

BLOCK
BRICK
CYLIND
SPHERE
ELLIPS
CONE
PRISM
PRISM_
CPRISM_
CPRISM_{2}
CPRISM_{3}
CPRISM_{4}
BPRISM_
FPRISM_
HPRISM_
SPRISM_
SPRISM_{2}
SPRISM_{3}
SPRISM_{4}
SLAB
SLAB_
CSLAB_
CWALL_
BWALL_
XWALL_

XWALL_{2}
XWALL_{3}
BEAM
CROOF_
CROOF_{2}
CROOF_{3}
CROOF_{4}
MESH
ARMC
ARME
ELBOW
EXTRUDE
PYRAMID
REVOLVE
REVOLVE{2}
REVOLVE{3}
REVOLVE{4}
REVOLVE{5}
RULED
RULED{2}
SWEEP
TUBE
TUBEA
COONS
MASS
MASS{2}

POLYROOF
POLYROOF{2}
POLYROOF{3}
POLYROOF{4}
EXTRUDEDSHELL
EXTRUDEDSHELL{2}
EXTRUDEDSHELL{3}
REVOLVEDSHELL
REVOLVEDSHELL{2}
REVOLVEDSHELL{3}
REVOLVEDSHELLANGULAR
REVOLVEDSHELLANGULAR{2}
REVOLVEDSHELLANGULAR{3}
RULEDSHELL
RULEDSHELL{2}
RULEDSHELL{3}
TEXT
BODY
BASE
POINTCLOUD
CUTPLANE
CUTEND (at UTPLANE, at CUTPLANE{2}, at CUTPLANE{3}, at CUTPOLY, at CUTPOLYA, at CUTSHAPE)
CUTPLANE{2}
CUTPLANE{3}

CUTPOLY
CUTPOLYA
CUTSHAPE
CUTFORM
CUTFORM{2}
GROUP
ENDGROUP
ADDGROUP
ADDGROUP{2}
ADDGROUP{3}
SUBGROUP
SUBGROUP{2}
SUBGROUP{3}
ISECTGROUP
ISECTGROUP{2}
ISECTGROUP{3}
ISECTLINES
PLACEGROUP
KILLGROUP
SWEEPGROUP
SWEEPGROUP{2}
SWEEPGROUP{3}
SWEEPGROUP{4}
CREATEGROUPWITHMATERIAL
BINARY
WALLNICHE

HOTSPOT
HOTLINE
HOTARC
LIN_
RECT
POLY
POLY_
PLANE
PLANE_
CIRCLE
ARC
LIGHT
PICTURE
RICHTEXT
VERT (at VERT, at VERT{2})
TEVE
VECT
EDGE
PGON
PGON{2}
PGON{3}
PIPG

COOR
COOR{2}
MODEL
WIRE
SURFACE
SOLID
MATERIAL
SECT_FILL
SECT_ATTRS
SHADOW
ON
OFF
AUTO

DEFINE MATERIAL (at DEFINE MATERIAL, at DEFINE MATERIAL BASED_ON)
BASED_ON
DEFINE TEXTURE
WALLHOLE

GDL中的常用命令汇总

2D Use Only

ADD2
MUL2
ROT2
LINE2
RECT2
POLY2
POLY2_
POLY2_A
POLY2_B
POLY2_B{2}
POLY2_B{3}
POLY2_B{4}
POLY2_B{5}
ARC2
CIRCLE2
SPLINE2
SPLINE2A
TEXT2
RICHTEXT2
FRAGMENT2
PROJECT2
PROJECT2{2}
PROJECT2{3}
DRAWING2
DRAWING3
DRAWING3{2}
DRAWING3{3}
WALLHOLE2
WALLHOLE2{2}
WALLBLOCK2
WALLBLOCK2{2}
WALLLINE2
WALLARC2

HOTSPOT2
HOTLINE2
HOTARC2
PICTURE2
PICTURE2{2}
LINE_PROPERTY
DRAWINDEX
FILL
LINE_TYPE
DEFINE FILL
DEFINE FILLA
DEFINE SYMBOL FILL
DEFINE SOLID_FILL
DEFINE EMPTY_FILL
DEFINE LINEAR_GRADIENT_FILL
DEFINE RADIAL_GRADIENT_FILL
DEFINE TRANSLUCENT_FILL
DEFINE IMAGE_FILL
DEFINE LINE_TYPE
DEFINE SYMBOL_LINE

2D and 3D Use

DEL (at DEL, at DEL TOP)
TOP
NTR
ADDITIONAL_DATA (at LIGHT, at DEFINE MATERIAL BASED_ON)
LET
RADIUS
RESOL
TOLER
PEN
SET (at [SET] STYLE, at [SET] MATERIAL, at [SET] FILL, at [SET] LINE_TYPE)
STYLE
DEFINE STYLE
DEFINE STYLE{2}
PARAGRAPH
ENDPARAGRAPH
TEXTBLOCK
TEXTBLOCK_

GDL中的常用命令汇总

Non-Geometric Scripts

Properties Script

```
DATABASE SET
DESCRIPTOR
REF DESCRIPTOR
COMPONENT
REF COMPONENT
BINARYPROP
SURFACE3D
VOLUME3D
POSITION
WALLS
COLUMNS
BEAMS
DOORS
WINDOWS
OBJECTS
CEILS
PITCHED_ROOFS
LIGHTS
HATCHES
ROOMS
MESHES
DRAWING
```

Interface Script

```
UI_DIALOG
UI_PAGE
UI_CURRENT_PAGE
UI_BUTTON (at UI_BUTTON, at UI_TOOLTIP)
UI_PREV
UI_NEXT
UI_FUNCTION
UI_LINK
UI_PICT_BUTTON (at UI_PICT_BUTTON, at UI_TOOLTIP)
UI_SEPARATOR
UI_GROUPBOX
UI_PICT (at UI_PICT, at UI_TOOLTIP)
UI_STYLE
UI_OUTFIELD (at UI_OUTFIELD, at UI_TOOLTIP)
UI_INFIELD (at UI_INFIELD, at UI_TOOLTIP)
UI_INFIELD{2} (at UI_INFIELD{2}, at UI_TOOLTIP)
UI_INFIELD{3} (at UI_INFIELD{3}, at UI_TOOLTIP)
UI_INFIELD{4} (at UI_INFIELD{4}, at UI_TOOLTIP)
UI_RADIOBUTTON (at UI_RADIOBUTTON, at UI_TOOLTIP)
UI_LISTFIELD (at UI_LISTFIELD, at UI_TOOLTIP)
UI_LISTITEM (at UI_LISTITEM, at UI_TOOLTIP)
UI_LISTITEM{2} (at UI_LISTITEM{2}, at UI_TOOLTIP)
UI_TOOLTIP
UI_COLORPICKER
UI_COLORPICKER{2}
UI_SLIDER
UI_SLIDER{2}
```

Parameter Script

```
VALUES
CUSTOM (at VALUES, at UI_INFIELD{4})
RANGE
VALUES{2}
PARAMETERS (at PARAMETERS, at CALL)
LOCK
ALL (at LOCK, at HIDEPARAMETER, at CALL)
HIDEPARAMETER
```

Forward and Backward Migration Scripts

```
SETMIGRATIONGUID
STORED PAR VALUE
DELETED_PAR_VALUE
NEWPARAMETER
```