LostInSpace.GSM

Macro object for position calculating for ArchiCAD 8.1 objects and above. This is a not placeable object and is for GDL programmers only.

1.0 Installation

Extract the archive into any folder of the library, your own object will be part of. The following objects and files will be extracted:

LostInSpace81.GSM
Macro file for ArchICAD 8.1 and above

LostInSpace.GSM
Macro file for ArchICAD 10 and above

LostInSpaceDemo.GSM
Example object with an tetrahedron or pyramid. (Needs LostInSpace81.GSM)

LostInSpaceDemoManual1_81.GSM
Example object of a cube script in this manual, chapter 7.0. For ArchiCAD 8.1 and above usage. (Needs LostInSpace81.GSM)

LostInSpaceDemoManual1_100.GSM
Example object of a cube script in this manual, chapter 7.0. For ArchiCAD 10 and above usage. (Needs LostInSpace.GSM)

LostInSpaceDemoManual2.GSM
Example object of a flexible desk lamp, which shows its inner axis and draws a line between shade and XY-plane in object origin. For ArchiCAD 8.1 and above usage. (Needs LostInSpace81.GSM)

LostInSpace – Manual.pdf
This manual file.

The manual file and the example files can be deleted, if not needed anymore. If the macro will not be used in ArchiCAD 8.1 or 9, the object *LostInSpace81.GSM* can be deleted too.

2.0 Functionality

The object calculates the position of any node in 3D after any transformation of the coordinate system.

The problem is known to all ambitioned programmers: Rotation, offset and scaling in 3D is done very easily by the transformation commands like ROTx, ADD or MULx. If you script the 2D appearance with small effort by a PROJECT2 command you are not able to position the exact hotspots at the corners of the 2D boundary of the projected model.

This macro can this calculate for you. The principle is easy. You list all done transformations into one multidimensional array and into a second one all the edges, generated after all these transformations and you want to calculate now. Call the macro in the following and it returns the coordinates of all assigned edges related to the objects origin without any transformations needed. You get the real positions in space. This information can not only be used by positioning of hotspots, but also e.g. for calculating the distance of any corner of the model to story elevation or any other basis. Or you could calculate a vector out of two edges to show a direction resulting out of a transformation. Helpful for positioning lights.

The macro can be called out of any script context. 2D-, 3D, parameter script: any is possible.

3.0 Syntax

The arrays need a strictly notation to transfer the transformation commands and the coordinates of the searched corners correctly and in the right order.

3.1 Transformation array

Define any array with 4 columns and min. 1 row.

The order of the transformations is essential. The first row of the array contains the first transformation, the second contains the second and so on. The first column of each row contains an integer notation of the transformation command (see shedule below) and the following up to three contain the arguments of the commands. Be aware, that in case of usage of variables and parameters for the arguments of the original transformation command in 3D script, they need to have now the same value in the array. This needs eventually double calculation of these arguments in each context the values are needed.

The array is transfered to macro by the appended PARAMETERS extension of the CALL command. See example below.

Notation-#	*Command*	*Arguments*		
0	no transformation			
1	ADD	X	Y	Z
1	ADDX	X	0	0
1	ADDY	0	Y	0
1	ADDZ	0	0	Z
2	MUL	X	Y	Z
2	MUL X	X	1	1
2	MUL Y	1	Y	1
2	MUL Z	1	1	Z
3	ROTX	angle	-	-
4	ROTY	angle	-	-
5	ROTZ	angle	-	-

If you call the macro more than once you need to clear the used array before you store new values into it.

3.1 Coordinates array

Define any array with 3 columns and min. 1 row.

The order of the appearance of the edges is irrelevant, but is identical with the order in the returned array.

Each row represent the three coordinates of one corner in the transformed coordinate system. The first column contains the X, the second the Y and the last the Z coordinate.

4.0 Returned coordinates

There are two ways the macro can return the calculated values:

A. By using the parameter buffer

B. By usage of the RETURNED PARAMETERS appendix possible with ArchiCAD 10 and above.

There are two macro objects included. The LostInSpace81.GSM objects allows only method A. LostInSpace100.GSM allows both methods selectable by an additional switch overgiven in the calling statement. See chapter 5.0.

5.0 Flags

There are two flags to select calculation options. See examples and GDL manual for help to GDL syntax and functionality.

only2D=0
Returns X, Y and Z coordinates.

only2D=1
Returns only X and Y coordinates.

buffer=0

buffer=1
Returns the values in the parameter buffer.

Returns the value using the RETURNED PARAMETERS command, allowing from ArchiCAD 10 on to return values from macros to the main object.

6.0 Returning values

buffer=1
The parameter buffer contains all coordinates in double or triple packages. If you overgave 4 edges the buffer contains 12 (only2D=0) or 8 (only2D=1) values. The order is X1,Y1,Z1,X2,Y2,Z2,...,XnYnZn. The Z values are missing, if the Z coordinate wasn't calculated.

buffer=0
You have to define an array for the returning values before the macro call in the calling object. It can be dinamical. As in the method of the buffer usage all values will be returned into this one array. Into one (!) dimension of the array. Regardless, if you declared a first dimension for

the edges and a second for the coordinates. The plain name of the array has to be added after a ‚RETURNED_PARAMETERS statement immediate after the CALL-command. See examp,le below. The array there is **xyz[]**.The order is as described above in buffer method.

7.0 Example

See also LostInSpaceDemoManual1_81.GSM or the other example objects, listed in chapter 1.0 to see how the macro works. LostInSpaceDemoManual1_100.GSM is the same demonstration object, but with an additional returning values technology possible with ArchiCAD 10 and above.

See following example script how to calculate the 8 edges of a cube after several transformations of the coordinate system. Example for ArchiCAD 10+.

3D-Skript Code:

```
! ### transformatons
ADD 0.2,0.8,-0.1
ROTx -235
ADDy -0.6
MULz 0.75
ROTy 15

! ### create cube
BLOCK a,b,zzyzx
```

2D-Skript Code:

```
HOTSPOT2 0,0

! ### project 3D model for surface

PROJECT2 3,270,2


! ### store transformaton into stack

DIM trans[][4]
t=0    ! reset counter

t=t+1   ! --> ADD
trans[t][1]=1
trans[t][2]=0.2 : trans[t][3]=0.8 : trans[t][4]=-0.1
t=t+1   !-> ROTX
trans[t][1]=3 : trans[t][2]=-235
t=t+1   ! --> ADDY
trans[t][1]=1 : trans[t][3]=-0.6
t=t+1   ! --> MULZ
trans[t][1]=2 : trans[t][2]=1 : trans[t][3]=1 : trans[t][4]=0.75
t=t+1   !-> ROTY
trans[t][1]=4 : trans[t][2]=15


! ### store coordinates into stack

DIM xyz[][3]
ci=0    ! reset counter
! bottom edges
ci=ci+1 : xyz[ci][1]=0 : xyz[ci][2]=0 : xyz[ci][3]=0
ci=ci+1 : xyz[ci][1]=0 : xyz[ci][2]=b : xyz[ci][3]=0
ci=ci+1 : xyz[ci][1]=a : xyz[ci][2]=0 : xyz[ci][3]=0
ci=ci+1 : xyz[ci][1]=a : xyz[ci][2]=b : xyz[ci][3]=0
! top edges
```

```
ci=ci+1 : xyz[ci][1]=0 : xyz[ci][2]=0 : xyz[ci][3]=zzyzx
ci=ci+1 : xyz[ci][1]=0 : xyz[ci][2]=b : xyz[ci][3]=zzyzx
ci=ci+1 : xyz[ci][1]=a : xyz[ci][2]=0 : xyz[ci][3]=zzyzx
ci=ci+1 : xyz[ci][1]=a : xyz[ci][2]=b : xyz[ci][3]=zzyzx


! ### output hotspots

! transform ccordinates
CALL "LostnSpace" PARAMETERS transformaton=trans,
   coordinates=xyz,
   buffer=0, only2D=1, RETURNED_PARAMETERS xyz


FOR i=1 to ci
  HOTSPOT2 xyz[i][1],xyz[i][2],i
  NEXT i
```

8.0 License agreement

This object was developed with all possible accuracy and was tested in several environments. But the author can't guarantee for any errrors containing the objects or resulting out of its usage. The author can't be hamed for any data loss or inproper working of the object itself or the context it is running in. Not for direct or indirect problems out of any problematic or wrong behaviour.

You may use this macro object office internal for you and with any of your objects. If you want to give away your own written objects together with this macro, you need to buy a developer license of the object. It is internal the same object, but with allowance to duplicate it. The developer license is needed independent of the commercial background of your decision to give away the main object.

Have fun with this object!

Preview:

See object homepage: http://www.opengdl.org/Default.aspx?tabid=2706.

The object is a quiet short piece of code, but it took me two years to think over and I started many efforts and other tracks with many discarded lines of GDL code before. The object code may not be recompiled, changed, unhidened or used outside of the license agreement. Respect this. Only the tiny license fees make it possible to develop and publish more amzing objects.

November 2007