Grid*Pro* API - CUBE

Tutorial - 1

Introduction

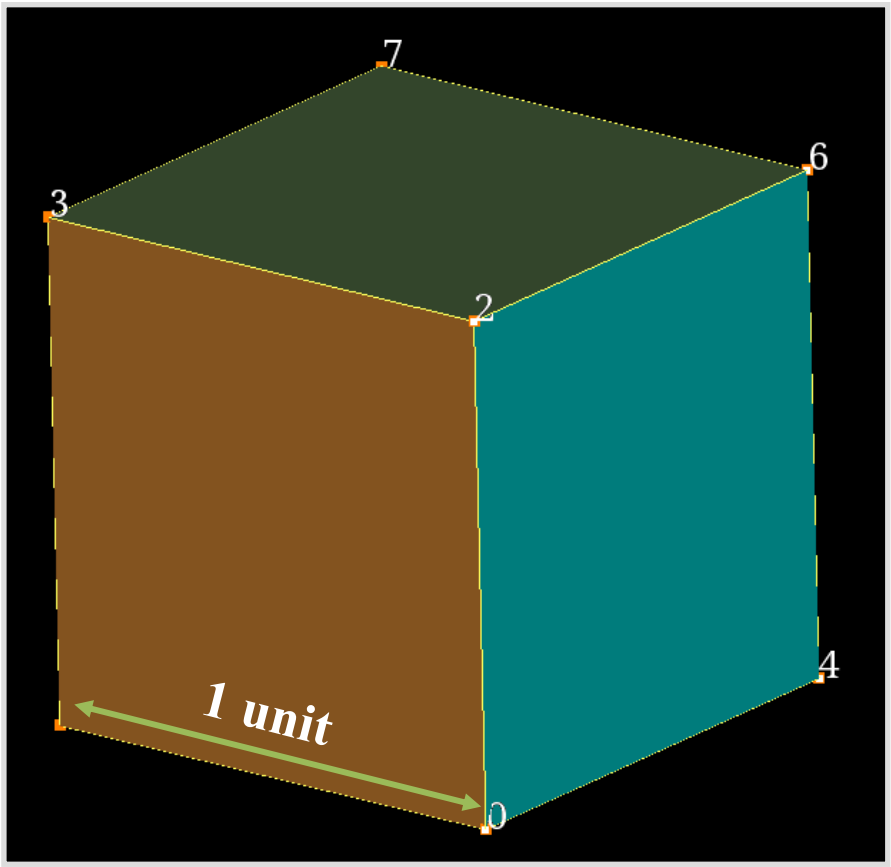
- The GridPro API is a python implementation of GridPro's Topology Input Language(TIL)
- The Topology Input Language(TIL) enables user to build a wireframe used to define a grid before allowing the grid generation engine to generate a grid.
- With the help of this API, users will have the flexibility to define the building process of a topology file from scratch without any GUI access.
- The API requires a valid license. Please reach out to support@gridpro.com if you need one.
- Currently the API works only with python 2.7 on all OS.
- Please check the GridPro/doc/Python_API folder for detailed documentation on each command

Problem Definition

The goal of this tutorial is to create a topology for a cube using the API.

- 6 surfaces
- create block by specifying corners
- Assign block faces to corresponding surface

The Corresponding tutorial using the GUI can be found in the GridPro basic tutorials.



- Cube dimensions:

Center: (0.5 0.5 0.5)

L*W*H: 1*1*1 units

Frame File Introduction

The frame file with extension *.fra is similar to a project file. It has the following Information.

- The header of the file contains the settings for the topology file. i.e. density and dimension of the topology.
- Information about surfaces transformation details if there are any.
- The wireframe is written using their low level components. i.e. as corners along with their relevant edge and block creation details.

```
SET GRIDDEN 8
SET DIMENSION 3
SET DISPLAY.SURF ON
```

Settings for the fra file. i.e default density, dimension of the topology and surface information ON/OFF

```
COMPONENT main()
BEGIN
INPUT 1 surf(sOUT (0..5));
INPUT 2 corn(sIN (1:1..6));
END
```

Components in the fra file. One SURFACE and One TOPOLOGY(corner) component

```
COMPONENT surf()
BEGIN
s 0  -plane @({-0.5,0,0},{0,0,0}) -1   -t 0 0.5 0.5  ;
s 1  -plane @({0.5,0,0},{0,0,0}) -1   -t 1 0.5 0.5  ;
s 2  -plane @({0.5,0,0},{0,0,0}) -1   -R 0 1 0 -1 0 0 0 0 1  -t 0.5 0 0.5  ;
s 3  -plane @({0.5,0,0},{0,0,0}) -1   -R 0 -1 0 1 0 0 0 0 1  -t 0.5 1 0.5  ;
s 4  -plane @({0.5,0,0},{0,0,0}) -1   -R 0 0 1 0 1 0 -1 0 0  -t 0.5 0.5 0  ;
s 5  -plane @({0.5,0,0},{0,0,0}) -1   -R 0 0 -1 0 1 0 1 0 0  -t 0.5 0.5 1  ;
END
```

Surface list created

```
COMPONENT corn(sIN s[0..5],cIN c[0..7])
BEGIN
c 0    0.0000000000    0.0000000000    0.0000000000 -L c:0 -g 0 ;
c 1    1.0000000000    0.0000000000    0.0000000000 -L c:1 0 -g 0 ;
c 2    0.0000000000    1.0000000000    0.0000000000 -L c:2 0 -g 0 ;
c 3    1.0000000000    1.0000000000    0.0000000000 -L c:3 2 1 -g 0 ;
c 4    0.0000000000    0.0000000000    1.0000000000 -L c:4 0 -g 0 ;
c 5    1.0000000000    0.0000000000    1.0000000000 -L c:5 4 1 -g 0 ;
c 6    0.0000000000    1.0000000000    1.0000000000 -L c:6 4 2 -g 0 ;
c 7    1.0000000000    1.0000000000    1.0000000000 -L c:7 6 5 3 -g 0 ;
END
```

Corners list created

API Module

- “gp_utilities” is the python module of GridPro API. To make use of the functions in a module, you'll need to import the module with an import statement.

```
import gp_utilities
#An import statement is made up of the import keyword along with the name of  the module. In a Python file, this will be declared at the top of the code.
```

- 3D vectors are used in multiple functions. Here we are creating an alias to class. vector3d is the class name.

```
from gp_utilities import vector3d as vec
# An alias for gp_utilities.vector3d to represent a vector in 3d space
```

API Module

- Next, an object of the topology class has to be created. This object keeps track of the surfaces, corners , groups and the topology operations along with other state variables.

```
#Main Function
if(__name__ == '__main__'):
    # This is executed only when the python interpreter is running on the current script as the main program.
    topo = gp_utilities.Topology()
```

Import the API | Call the topology class | Create surfaces and corners | Block Creation | Assignments | Check validity of the topology | Write the fra file

Surface Creation

- A cube has 6 bounding surfaces, each of which is represented by a plane . A plane is defined by a centre and normal. For visualisation , we specify a size for each of them.

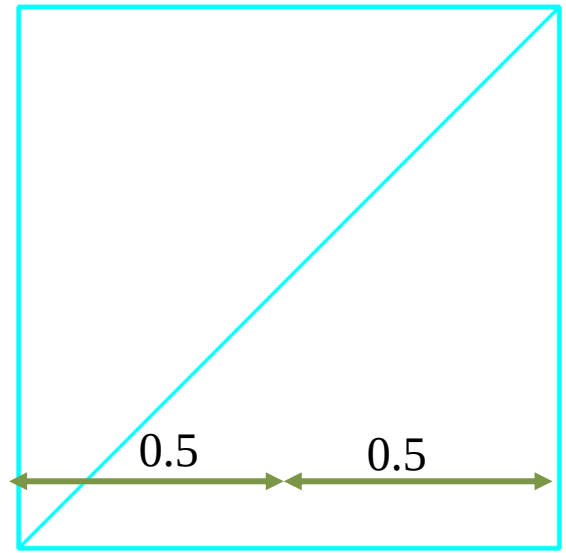
s0 = topo.add_plane(vec(0, 0.5, 0.5), vec(-0.5, 0, 0))

Syntax: add_plane(vec(0, 0.5, 0.5), vec(-0.5, 0, 0))

Where,

vec(0, 0.5, 0.5) – Defines the center of the plane

vec(-0.5, 0, 0) – Defines the normal & size of plane. i.e. X axis in this case, 0.5 defines the size of the plane from the center on both sides of the axis. So the entire length of the plane would be 1 unit.



Plane

Surface Creation

- All 6 surfaces from s0 to s5 are to be defined with their centre and normal in the following statements.

```
#Creating surface for each face of the cube
s0 = topo.add_plane(vec(0, 0.5, 0.5), vec(-0.5, 0, 0))
s1 = topo.add_plane(vec(1, 0.5, 0.5), vec( 0.5, 0, 0))
s2 = topo.add_plane(vec(0.5, 0, 0.5), vec(0, -0.5, 0))
s3 = topo.add_plane(vec(0.5, 1, 0.5), vec(0, 0.5,0))
s4 = topo.add_plane(vec(0.5, 0.5, 0), vec(0, 0, -0.5))
s5 = topo.add_plane(vec(0.5, 0.5, 1), vec(0, 0, 0.5))
```

Wireframe Creation

- To create the wireframe , in this case a box, we need to create 8 corners and 12 edges. Each wireframe corner can be created by specifying its x,y,z coordinates.

c0 = topo.add_corner(0, 0, 0)

Syntax: add_corner(0, 0, 0)

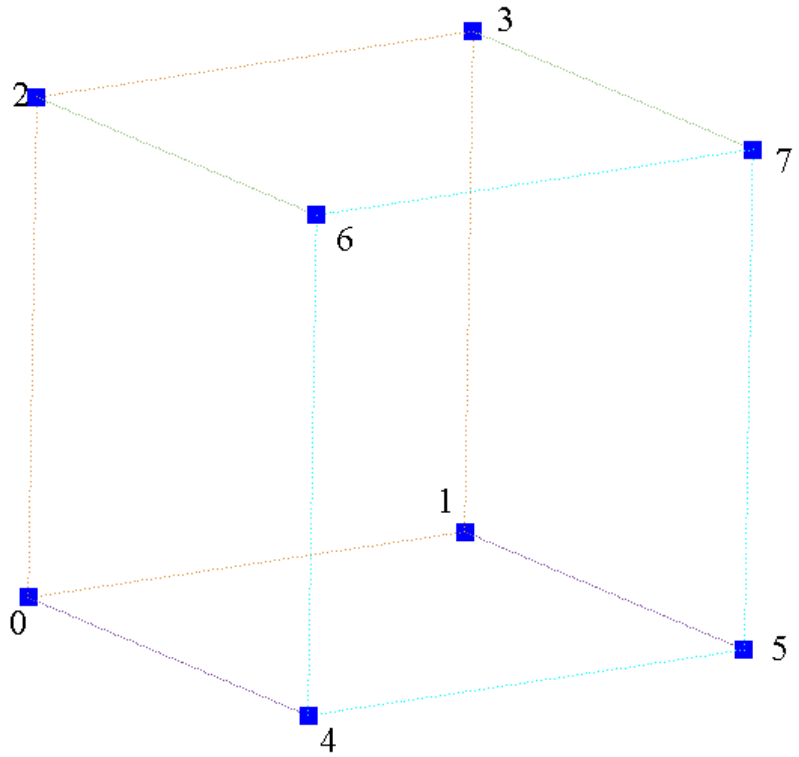
Where,

(0, 0, 0) – Defines the position of the corner.

Wireframe Creation

- All 8 corners are created by specifying the respective (x, y, z) coordinates)

```
#Corner Creation
c0 = topo.add_corner(0, 0, 0)
c1 = topo.add_corner(1, 0, 0)
c2 = topo.add_corner(0, 1, 0)
c3 = topo.add_corner(1, 1, 0)
c4 = topo.add_corner(0, 0, 1)
c5 = topo.add_corner(1, 0, 1)
c6 = topo.add_corner(0, 1, 1)
c7 = topo.add_corner(1, 1, 1)
```



Import the API | Call the topology class | Create surfaces and corners | Block Creation | Assignments | Check validity of the topology | Write the fra file

Wireframe Creation

- The edges can be created by using the member function “link “. They can be linked sequentially , however in this case they are linked simultaneously to form the 6 faces of a block.

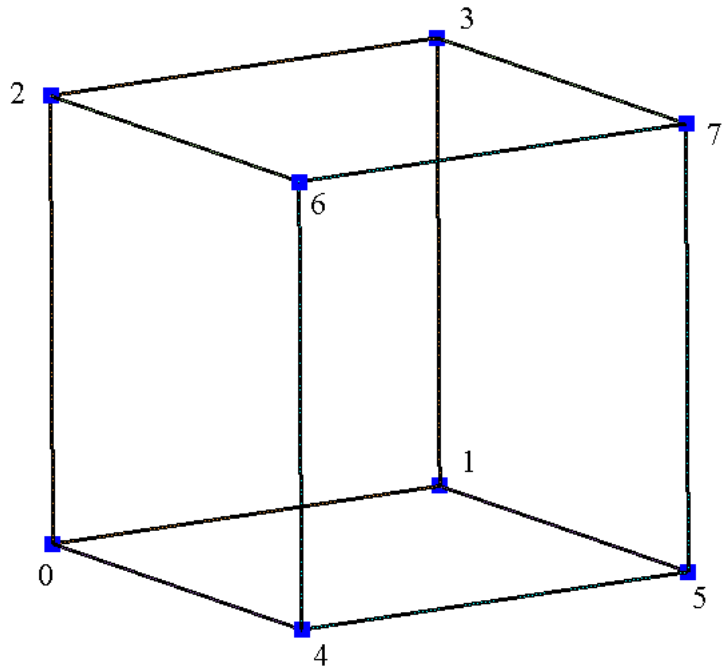
Syntax: link([(corner 1, corner 2)])

#Linking the corners

topo.link([(c0,c1), (c0,c2), (c2,c3), (c1,c3),

(c4,c5), (c4,c6), (c6,c7), (c5,c7),

(c0,c4), (c1,c5), (c2,c6), (c3,c7)])



Wireframe Creation

Once the wireframe and the surfaces are created the faces of the wireframe need to be associated with the surfaces, this association is called as Assignments. The assignments give a directive for projection to the block faces.

- Each face of the block has to be assigned to the corresponding surface of the cube.
- Currently, each corner of the face has to be assigned to the respective surfaces.

Syntax: <corner>.assign(<surface>)

```
#Assigning the corner c0 to the surfaces

c0.assign(s4)

c0.assign(s2)

c0.assign(s0)
```

Wireframe Creation

The corners can also be assigned to multiple surfaces with the member function ” assign_multiple”. The corners c4,c5,c6 and c7 are assigned using the assign_multiple function.

Syntax: (corner).assign_multiple([surf 1, surf 2, surf 3])

c1.assign(s4)

c1.assign(s2)

c1.assign(s1)

c2.assign(s4)

c2.assign(s3)

c2.assign(s0)

c3.assign(s4)

c3.assign(s3)

c3.assign(s1)

c4.assign_multiple([s5,s2,s0])

c5.assign_multiple([s5,s2,s1])

c6.assign_multiple([s5,s3,s0])

c7.assign_multiple([s5,s3,s1])

Topology Validity

In this case the assignment is the last step. At this point the topology can be checked whether it is valid (in other words if it abides by all the topology rules).The validity can be checked by using the member function “is_valid”.

Syntax: is_valid( )

#Checking Validity of the Topology

topo.is_valid( )

- If the topology is not valid, an error message will be printed while compiling.

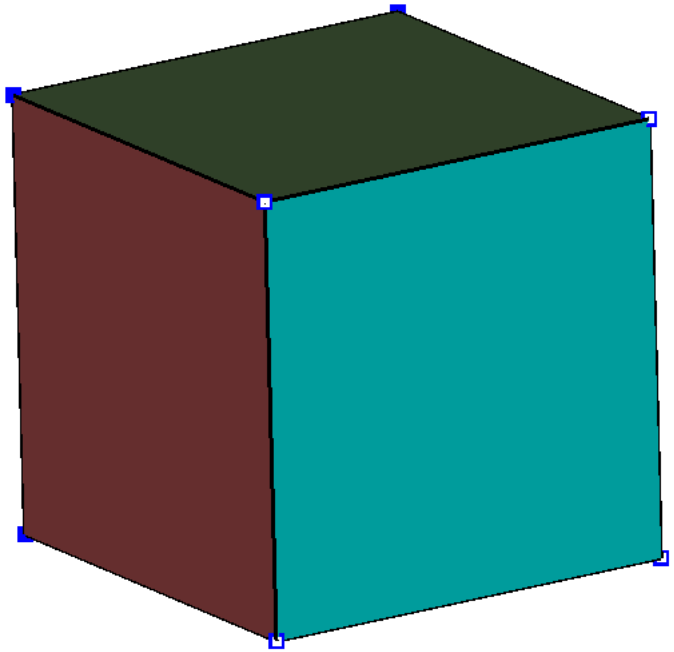
ERROR: Cannot build the topology.

Topology Output

- The wireframe along the the surface and its attributes are To verify whether our program is right so far, we will have to write out the fra file.

- Syntax: write_topology(“file_name")

```
#Writing the fra file
topo.write_topology("cube.fra")
```



Import the API	Call the topology class	Create 6 surfaces Create 8 corners	Link all 8 corners to form 6 faces which in turn forms a block	Assign the corners(face) to the surfaces.	Check validity of the topology	Write the fra file

Code Snippet

```
# IMPORT OPERATIONS
import gp_utilities
from gp_utilities import vector3d as vec

#Main Function
if(__name__ == '__main__'):
    topo = gp_utilities.Topology()      #Assign Topology
Class

#Creating surface for each face of the cube
    s0 = topo.add_plane(vec(0, 0.5, 0.5), vec(-0.5,0,0))
    s1 = topo.add_plane(vec(1, 0.5, 0.5), vec( 0.5,0,0))
    s2 = topo.add_plane(vec(0.5, 0, 0.5), vec(0,-0.5,0))
    s3 = topo.add_plane(vec(0.5, 1, 0.5), vec(0, 0.5,0))
    s4 = topo.add_plane(vec(0.5, 0.5, 0), vec(0,0,-0.5))
    s5 = topo.add_plane(vec(0.5, 0.5, 1), vec(0,0, 0.5))

#Corner Creation
    c0 = topo.add_corner(0, 0, 0)      #Corner id 0
    c1 = topo.add_corner(1, 0, 0)      #Corner id 1
    c2 = topo.add_corner(0, 1, 0)      #Corner id 2
    c3 = topo.add_corner(1, 1, 0)      #Corner id 3
    c4 = topo.add_corner(0, 0, 1)      #Corner id 4
    c5 = topo.add_corner(1, 0, 1)      #Corner id 5
    c6 = topo.add_corner(0, 1, 1)      #Corner id 6
    c7 = topo.add_corner(1, 1, 1)      #Corner id 7

#Linking the corners
    topo.link([(c0,c1), (c0,c2), (c2,c3), (c1,c3),
               (c4,c5), (c4,c6), (c6,c7), (c5,c7),
               (c0,c4), (c1,c5), (c2,c6), (c3,c7)])
```

```
#Assigning the corners to each of the surface
    c0.assign(s4)
    c0.assign(s2)    #c0.assign_multiple([s4,s2,s0])
    c0.assign(s0)

    c1.assign(s4)
    c1.assign(s2)
    c1.assign(s1)

    c2.assign(s4)
    c2.assign(s3)
    c2.assign(s0)

    c3.assign(s4)
    c3.assign(s3)
    c3.assign(s1)

    c4.assign_multiple([s5,s2,s0])

    c5.assign_multiple([s5,s2,s1])

    c6.assign_multiple([s5,s3,s0])

    c7.assign_multiple([s5,s3,s1])

#Checking validity of the Topology
    topo.is_valid( )

#Writing the fra file
    topo.write_topology("cube.fra")
```

Thank You

For any questions, contact:

support@gridpro.com