

Running SolidPython from FreeCAD

Prerequisites:

FreeCAD, OpenSCAD and SolidPython must be installed on the computer.

SolidPython is available here: <https://github.com/SolidCode/SolidPython>

A first test:


- Enable the report panel: Menu View – Panels - Report
- Create new macro and enter SolidPython code e.g.

```
from solid import *
from solid.utils import *
d=cube(10) - up(5)(right(15)(sphere(12)))
print (scad_render(d))
```

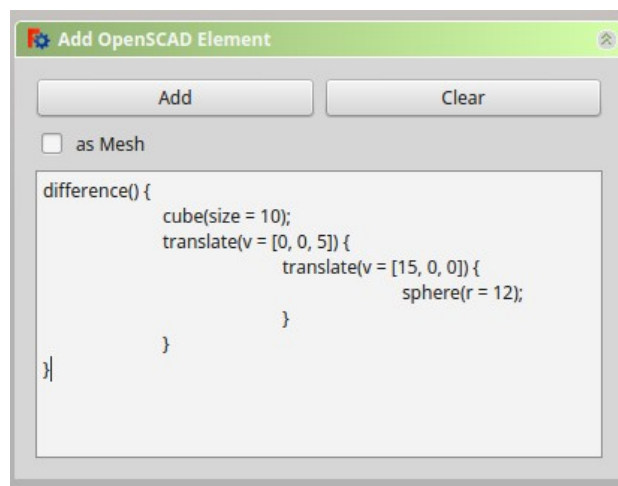
Now the OpenSCAD source code is displayed in the report panel:

```
Report view
difference() {
  cube(size = 10);
  translate(v = [0, 0, 5]) {
    translate(v = [15, 0, 0]) {
      sphere(r = 12);
    }
  }
}
```

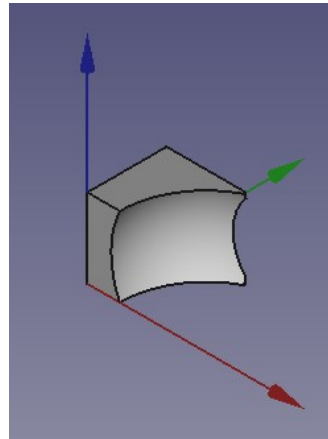
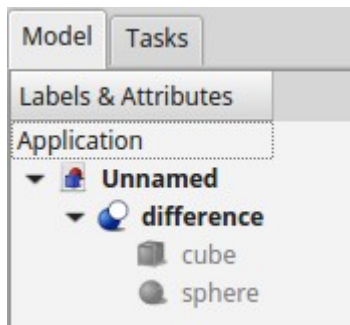
Copy it.

- Open the OpenSCAD window: Menu OpenSCAD – Add OpenSCAD element 

Paste the openSCAD code to the window that opens (eventually clear existing code before pasting):



Now the result is visible in the object tree and in as a drawing:



Automatic rendering of the drawing

When I began to experiment with SolidPython, I found the described procedure a little bit clumsy. Wouldn't it be possible to automate the process, so that it would be enough to write the code and let FreeCAD do the rest? Surely this must be possible, due to FreeCAD's excellent Python scripting possibilities!

So I had a look into the source code of the OpenSCAD workbench of FreeCAD. For each button there is a class, in our case **AddOpenSCADElement**:

```
class AddOpenSCADElement:
    ...
    def Activated(self):
        panel = AddSCADTask()
        FreeCADGui.Control.showDialog(panel)
    ....
```

The interesting class is **AddSCADTask**, and here the method **addelement**:

```
class AddSCADTask:
    ...

    def addelement(self):
        scadstr=unicode(self.form.textEdit.toPlainText()).encode('utf8')
        asmesh=self.form.checkboxmesh.checkState()
        import OpenSCADUtils, os
        extension= 'stl' if asmesh else 'csg'
        try:
            tmpfilename=OpenSCADUtils.callopenscadstring(scadstr,extension)
            doc=FreeCAD.activeDocument() or FreeCAD.newDocument()
            if asmesh:
                import Mesh
                Mesh.insert(tmpfilename,doc.Name)
            else:
                import importCSG
                importCSG.insert(tmpfilename,doc.Name)
        try:
            os.unlink(tmpfilename)
```

```

except OSError:
    pass
...

```

Not all of this is necessary for our task.

The essential lines are

```

tmpfilename=OpenSCADUtils.callopenscadstring(scadstr, 'csg')
doc=FreeCAD.activeDocument() or FreeCAD.newDocument()
importCSG.insert(tmpfilename, doc.Name)

```

where scadstr is the string with the OpenSCAD commands that is generated from SolidPython.

To test this, I wrote a small macro and started it from FreeCAD:

```

1 # -*- coding: utf-8 -*-
2 # imports for FreeCAD
3 import FreeCAD
4 import OpenSCADUtils
5 import importCSG
6
7 # imports for SolidPython
8 from solid import *
9 from solid.utils import *
10
11
12 # drawing with SolidPython
13 c=cube(20)-right(15)(sphere(10))
14 scadstr = scad_render(c)
15 print scadstr
16
17 # automatic rendering of scadstr in FreeCAD
18 tmpfilename=OpenSCADUtils.callopenscadstring(scadstr, 'csg')
19 doc=FreeCAD.activeDocument() or FreeCAD.newDocument()
20 importCSG.insert(tmpfilename, doc.Name)

```

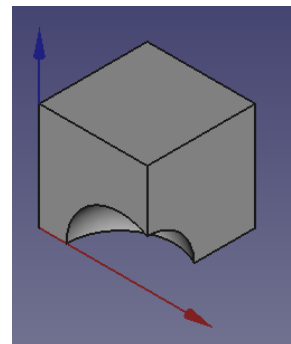
This threw some error messages, but the drawing was automatically generated, as expected:

```

(openscad:2991): Gtk-WARNING **: Unable to locate theme engine in
module_path: "oxygen-gtk",

WARNING: Token 'DOT' defined, but not used
WARNING: Token 'WORD' defined, but not used
WARNING: There are 2 unused tokens
Unable to create 'parsetab.py'
[Errno 13] Permission denied: 'parsetab.py'
End processing CSG file

```



By the way, the parsetab.py error showed also with the other method