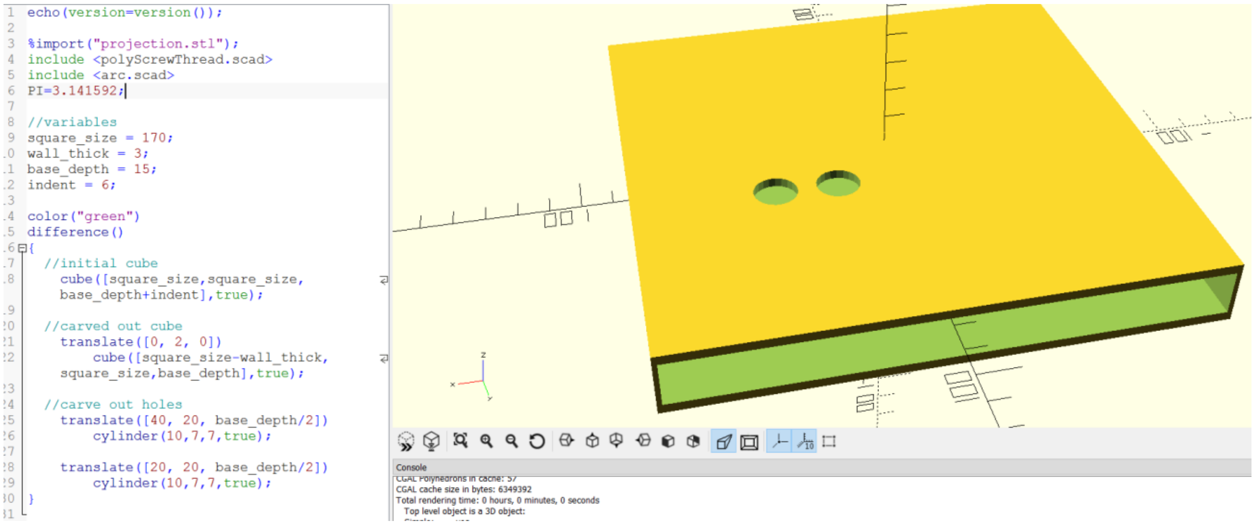
# Open SCAD example

OpenSCAD is a code based mechanical design program. The problem I have with Solidworks is that to truly master is you need to spend time daily with the interface and shortcuts. If not going back to it constantly it is easy to forget the small things that make the program stand out. What I enjoy about OpenSCAD is that it is code based and even after a year away from it, it is easy to jump right back in and understand exactly what is going on. For an example I made a simple case to demonstrate the basic principles.

You can see the interface and case in Figure 1. You insert your code on the left, then whenever you compile, the figure reflects on the right.



```
echo(version=version());

%import("projection.stl");
include <polyScrewThread.scad>
include <arc.scad>
PI=3.141592;

//variables
square_size = 170;
wall_thick = 3;
base_depth = 15;
indent = 6;

color("green")
difference()
{
  //initial cube
    cube([square_size,square_size,
    base_depth+indent],true);

  //carved out cube
    translate([0, 2, 0])
        cube([square_size-wall_thick,
    square_size,base_depth],true);

  //carve out holes
    translate([40, 20, base_depth/2])
        cylinder(10,7,7,true);

    translate([20, 20, base_depth/2])
        cylinder(10,7,7,true);
}
```

```
CGAL Polyhedrons in cache: 57
CGAL cache size in bytes: 6349392
Total rendering time: 0 hours, 0 minutes, 0 seconds
   Top level object is a 3D object:
```

Figure 1

At the top are the include files. These are libraries you can include to bring in additional constructs. Some of these are made by third parties, and some are provided by the OpenSCAD creators. These libraries can be very useful, and though I did not need any here, I put in the includes as an example. These libraries can be seen below…

%import("projection.stl");

include <polyScrewThread.scad>

include <arc.scad>

PI=3.141592;

Below the libraries are the variables used in the rest of the program. These variables are helpful to assign values to easily recognizable names. You can then use them throughout to place and move pieces of the design. The variables can be seen below…

//variables

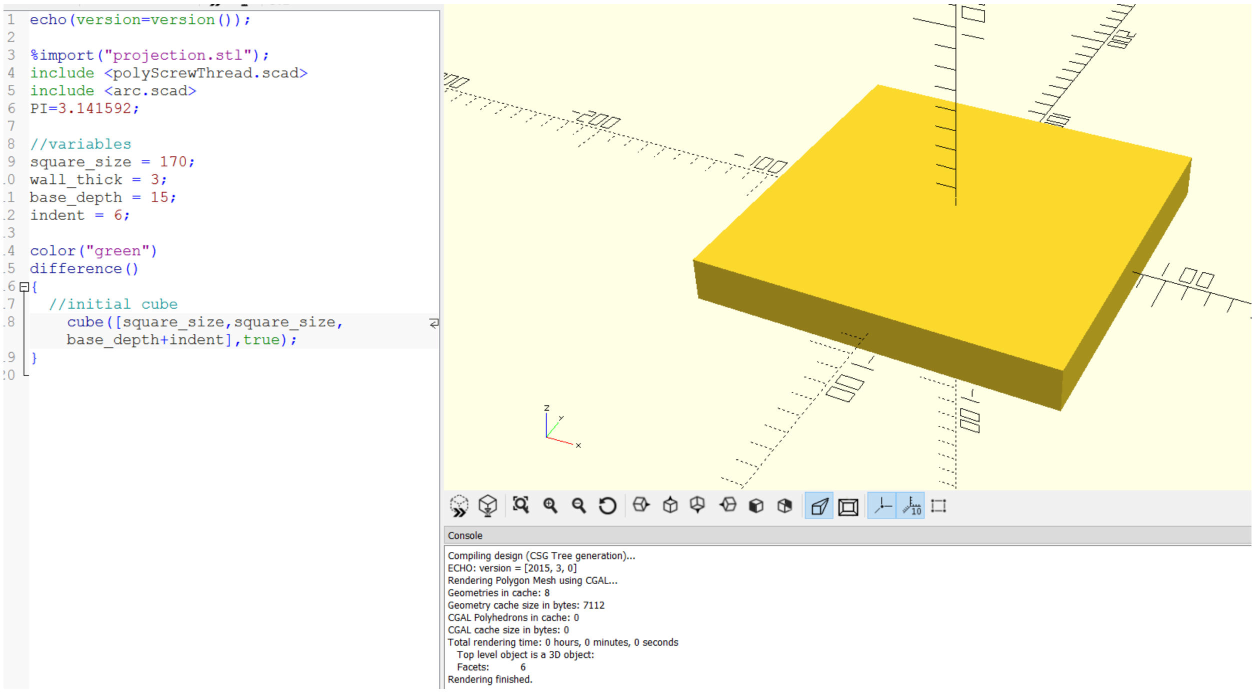
square_size = 170;

wall_thick = 3;

base_depth = 15;

indent = 6;

You can then begin the heart of the program, which is to create your shapes. First pay attention to where the code calls out the initial cube. This is the main block of the design and can be seen in figure 2.



```
echo(version=version());

%import("projection.stl");
include <polyScrewThread.scad>
include <arc.scad>
PI=3.141592;

//variables
square_size = 170;
wall_thick = 3;
base_depth = 15;
indent = 6;

color("green")
difference()
{
  //initial cube
    cube([square_size,square_size,
    base_depth+indent],true);
}
```

```
Compiling design (CSG Tree generation)...
ECHO: version = [2015, 3, 0]
Rendering Polygon Mesh using CGAL...
Geometries in cache: 8
Geometry cache size in bytes: 7112
CGAL Polyhedrons in cache: 0
CGAL cache size in bytes: 0
Total rendering time: 0 hours, 0 minutes, 0 seconds
   Top level object is a 3D object:
   Facets:         6
Rendering finished.
```

Figure 2

This is the block created by our cube command with nothing else. The parameters passed to it dictate the shape, and the parameters allowed can be looked up in the OpenSCAD documentation. The documentation contains all possible functions and their associated parameters to pass in and return. You will also notice that the cube is called inside the difference wrapper. There are many wrappers such as this, but difference can be rather handy, so I decided to show it off. Using this, the first shape called in the wrapper is the main block, and everything called after that is subtracted off the original block. Think of it as everything else is an anti-shape. You will see we carve out another smaller square and two circles from the original shape to leave our final box.

The other thing of note on some of the functions is the translate wrapper, which allows you to move shapes around to specific locations. This movement can be done with precise parameters and can be done relative to other parameters as well, which is in my opinion a much simpler way to work than other mechanical CAD programs.

That's all there is to it. You should know all the shapes that exist and however you can think to combine them in creating your masterpiece can in fact be coded. This is a program that really thrives off your imagination and the coding is a secondary aspect to it.