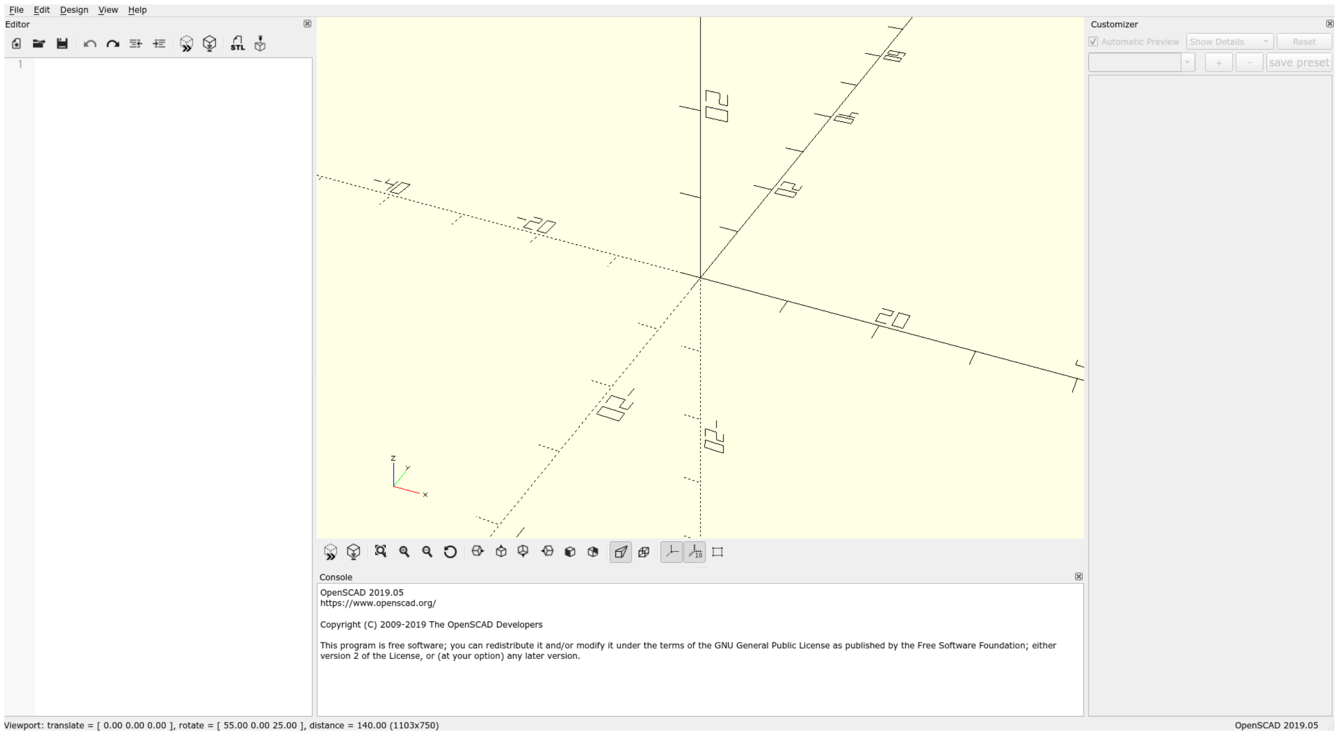
OpenSCAD
Workshop 1: Making a mast holder

Step 1: Download and install OpenSCAD.

Windows and Mac: http://www.openscad.org/downloads.html
Linux: There is probably a version in your distro's repositories, if there is one, it is probably best to get it from there. On the page linked for the Windows and Mac users, there are instructions on how to install it for some common distros, as well as links to snaps and flatpacks.

Step 2: Open OpenSCAD and click new.

It should take you to an empty project that looks something like this:



```
OpenSCAD 2019.05
https://www.openscad.org/

Copyright (C) 2009-2019 The OpenSCAD Developers

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.
```

```
Viewport: translate = [ 0.00 0.00 0.00 ], rotate = [ 55.00 0.00 25.00 ], distance = 140.00 (1103x750)
OpenSCAD 2019.05
```

On the left is the editor. This is where you'll be writing your code. The panel in the top middle is the viewport. It will give a visual representation of what you are doing. The bottom middle panel is the console. Error messages will be highlighted in yellow. The panel on the right is the Customizer. You can go ahead and close it using the x in its top right.

Take a moment to familiarize yourself with the viewport and its controls:

- Left-Click and drag is rotate.
- Right-Click and drag is pan.
- Scroll to zoom.
- You should also try out the buttons under the viewport.

Step 3: Make a cuboid

In OpenScad, the main way you make models is that you create shapes and then modify them. So in general the syntax looks like this:

```
...
modification2()
modification()
shape();
```

The modifications are applied in reverse order, so in this example, “modification” is applied before “modification2”.

Don’t worry if seems a little abstract for the moment, it probably will become clearer once you start using it.

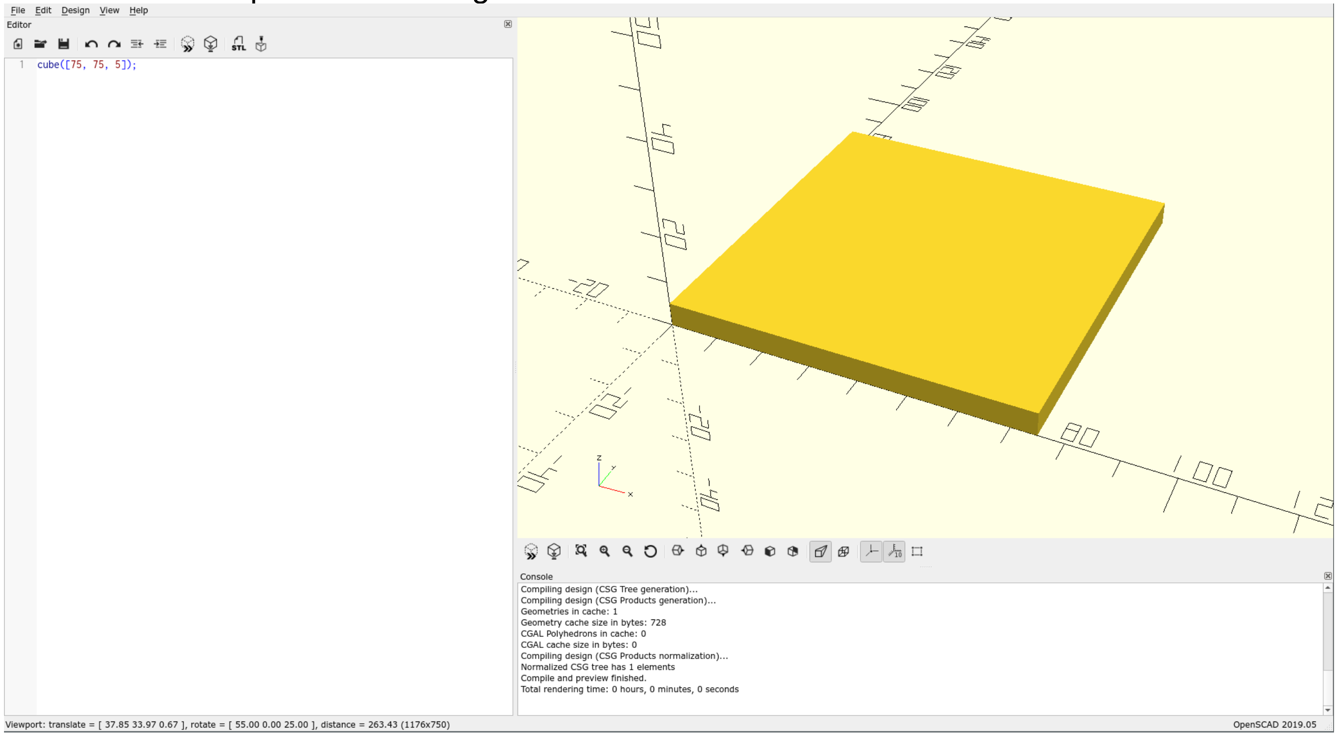
Now back onto the cuboid: To make a cuboid type this into the editor:

```
cube([75, 75, 5]);
```

The numbers in the array correspond to the x side-length, the y side-length, and z side-length (in that order) of the cuboid. The sizes are in millimeters (and could have been non-integers). Most things in OpenSCAD are measured in millimeters or degrees.

Next, you need to tell OpenSCAD to preview your code. Press F5 or the preview button, which is the box with two arrow-heads in-front of it (there are two preview buttons, one above the editor and one below the viewport).

You should end up with something that looks like this:



```
cube([75, 75, 5]);
```

```
Compiling design (CSG Tree generation)...
Compiling design (CSG Products generation)...
Geometries in cache: 1
Geometry cache size in bytes: 728
CGAL Polyhedrons in cache: 0
CGAL cache size in bytes: 0
Compiling design (CSG Products normalization)...
Normalized CSG tree has 1 elements
Compile and preview finished.
Total rendering time: 0 hours, 0 minutes, 0 seconds
```

Viewport: translate = [37.85 33.97 0.67], rotate = [55.00 0.00 25.00], distance = 263.43 (1176x750)

OpenSCAD 2019.05

Step 4: Use variables.

You sometimes want to re-use a value multiple times. You could just type the value every time you need it. But that can quickly get messy if you want to change the value later. You will end up having to change every single instance of that value, and that's time-consuming and error-prone, so instead, we define variables.

Variables are defined like this:

```
my_variable_name = 34.5;
```

After you have done this you can now use the name you gave your variable instead of the value.

As we want to use the values for the size of the cuboid again later, we want to make variables out of them.

Make variables representing the values, and then replace the values with their variable names. You should end up with code that looks something like this:

```
base_width = 75;
base_thickness = 5;
cube([base_width, base_width, base_thickness]);
```

When you press preview nothing should change.

Step 6: Make the outside of the mast shaft.

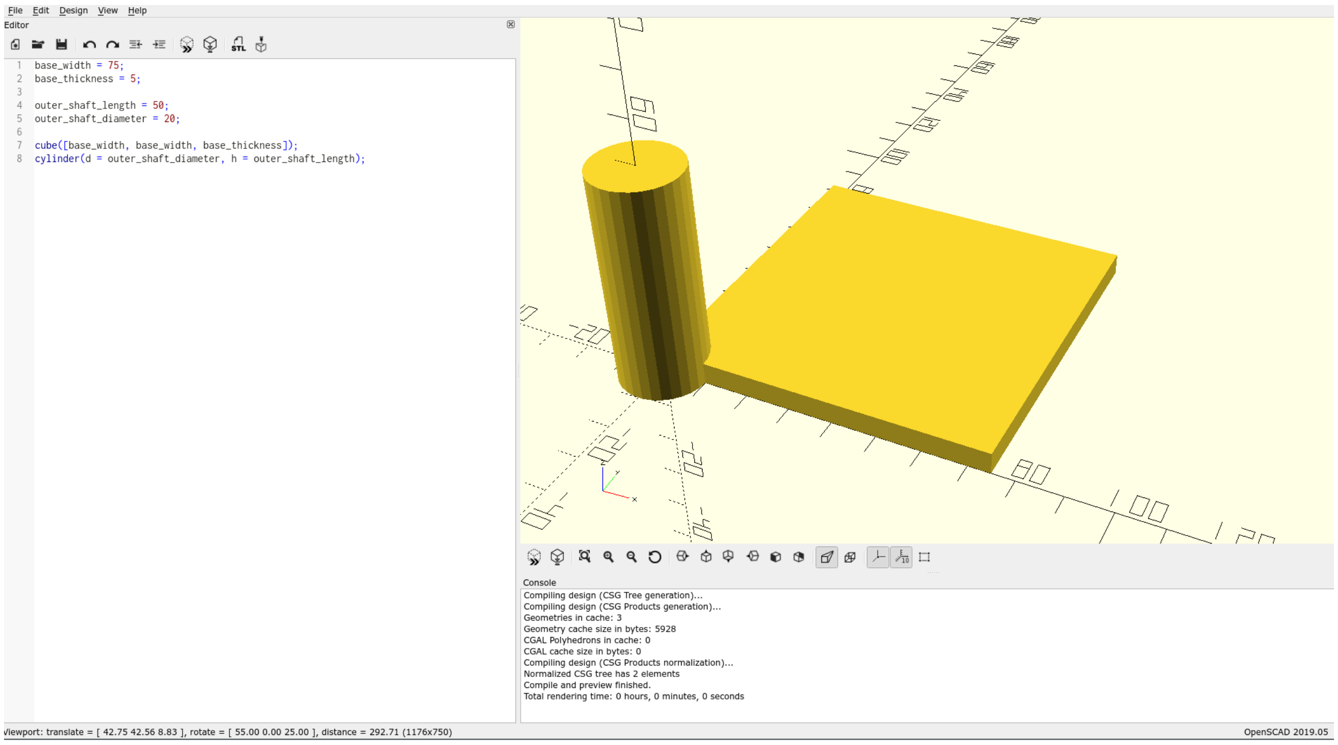
We are going to make the mast shaft out of a cylinder. We are going to need the values defining the cylinder multiple times so we need to turn them into variables.

```
outer_shaft_length = 50;
outer_shaft_diameter = 20;
```

To create the cylinder itself the syntax is:

```
cylinder(d = outer_shaft_diameter, h = outer_shaft_length);
```

Once you have added these lines to the project, press the preview button and you should end up with something that looks like this:



```
base_width = 75;
base_thickness = 5;

outer_shaft_length = 50;
outer_shaft_diameter = 20;

cube([base_width, base_width, base_thickness]);
cylinder(d = outer_shaft_diameter, h = outer_shaft_length);
```

```
Compiling design (CSG Tree generation)...
Compiling design (CSG Products generation)...
Geometries in cache: 3
Geometry cache size in bytes: 5928
CGAL Polyhedrons in cache: 0
CGAL cache size in bytes: 0
Compiling design (CSG Products normalization)...
Normalized CSG tree has 2 elements
Compile and preview finished.
Total rendering time: 0 hours, 0 minutes, 0 seconds
```

```
Viewport: translate = [ 42.75 42.56 8.83 ], rotate = [ 55.00 0.00 25.00 ], distance = 292.71 (1176x750)
```

OpenSCAD 2019.05

Step 7: Move the cylinder into place.

We are now going to use one of those modifiers I talked about earlier. We are going to use the translate modifier.

The syntax for translate looks like this:

```
translate([x, y, z])
```

Where x is the amount to move the object in the x-direction, and y is the amount to move the object in the y-direction, and z is the amount to move the object in the z-direction.

If you remember from before the syntax for applying modifiers looks like this:

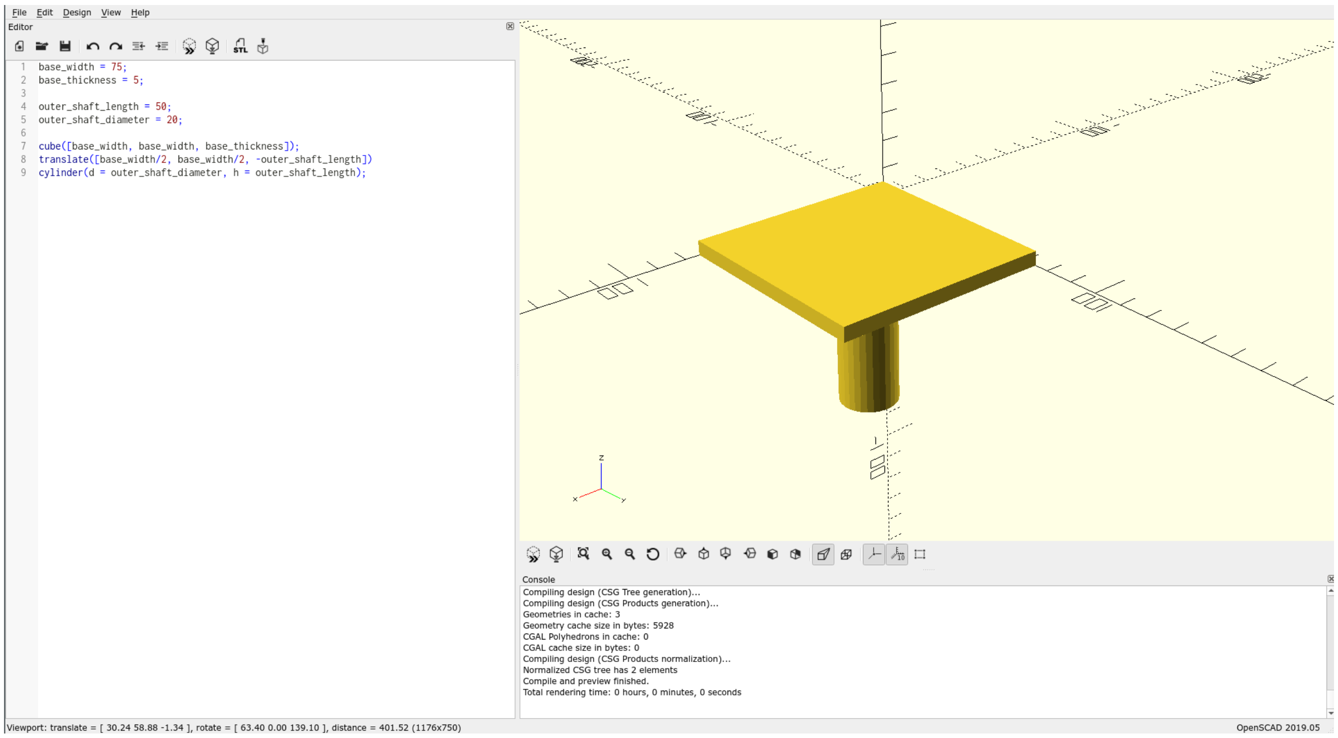
```
modification()
shape();
```

So we are going to apply this pattern to the cylinder:

```
translate([base_width/2, base_width/2, -outer_shaft_length])
cylinder(d = outer_shaft_diameter, h = outer_shaft_length);
```

Notice that the line with translate on it doesn't have ";" at the end.

After you press preview you should end up with something that looks like this:



```
base_width = 75;
base_thickness = 5;

outer_shaft_length = 50;
outer_shaft_diameter = 20;

cube([base_width, base_width, base_thickness]);
translate([base_width/2, base_width/2, -outer_shaft_length])
cylinder(d = outer_shaft_diameter, h = outer_shaft_length);
```

```
Compiling design (CSG Tree generation)...
Compiling design (CSG Products generation)...
Geometries in cache: 3
Geometry cache size in bytes: 5928
CGAL Polyhedrons in cache: 0
CGAL cache size in bytes: 0
Compiling design (CSG Products normalization)...
Normalized CSG tree has 2 elements
Compile and preview finished.
Total rendering time: 0 hours, 0 minutes, 0 seconds
```

```
Viewport: translate = [ 30.24 58.88 -1.34 ], rotate = [ 63.40 0.00 139.10 ], distance = 401.52 (1176x750)
```

Step 8: Make the cylinder smoother.

You have probably noticed that the cylinder is not very smooth. This is in-fact one of OpenSCAD's main limitations, it does not deal with curves. For the most part, this can be gotten around by increasing the number of line segments OpenSCAD will use. We can do this by changing one of the system variables.

Add this line to the beginning of the file:

```
$fn = 100;
```

You should make changes to the system variables at the top of the file, as changing them can be very resource-intensive.

Press preview again and the cylinder should become much smoother.

The performance effects can be best seen when you render the project. You can render the project by pressing F6 or the button with the cube with an hour-glass in-front (next to the preview buttons).

Try changing the value and then rendering to see the results and how much time it takes. "$fn = 1000" takes a few seconds for my laptop to render. Don't set the number too high (>5000ish) because this can cause OpenSCAD to hang.

Step 9: Cut a hole in the shaft.

When you want to cut a shape out of another shape you use a difference block. The syntax for this looks like this:

```
difference() {
     shape();
     shape2();
     shape3();
     ...
}
```

When OpenSCAD encounters this, it will create “shape”, and then it will see where “shape” intersects with any of the shapes that come after it in the block. Anywhere they do intersect will be removed from “shape”.

But there is a problem for us. Our shape so far is made from two shapes. For it to be the first shape in a difference block it needs to be one shape. Otherwise, we would cut one out of the other. The way we make them one shape is by using a union block. The syntax for this is:

```
union() {
     shape();
     shape2();
     shape3();
     ...
}
```

This tells OpenSCAD we want all of the shapes inside of the union to be treated as one shape.

Wrap the cuboid and cylinder in a union block and then difference block. You’re code should end up looking something like this:

```
difference() {
    union() {
        cube([base_width, base_width, base_thickness]);
        translate([base_width/2, base_width/2, -outer_shaft_length])
        cylinder(d = outer_shaft_diameter, h = outer_shaft_length);
    }
}
```

When you press preview nothing should change.

Now we are going to cut a cylinder out of the shaft. We are going to be using the same values several times, so we need to make two more variables defining the shape of the cylinder we want to cut out:

```
inner_shaft_diameter = 15;
inner_shaft_length = outer_shaft_length - 5;
```

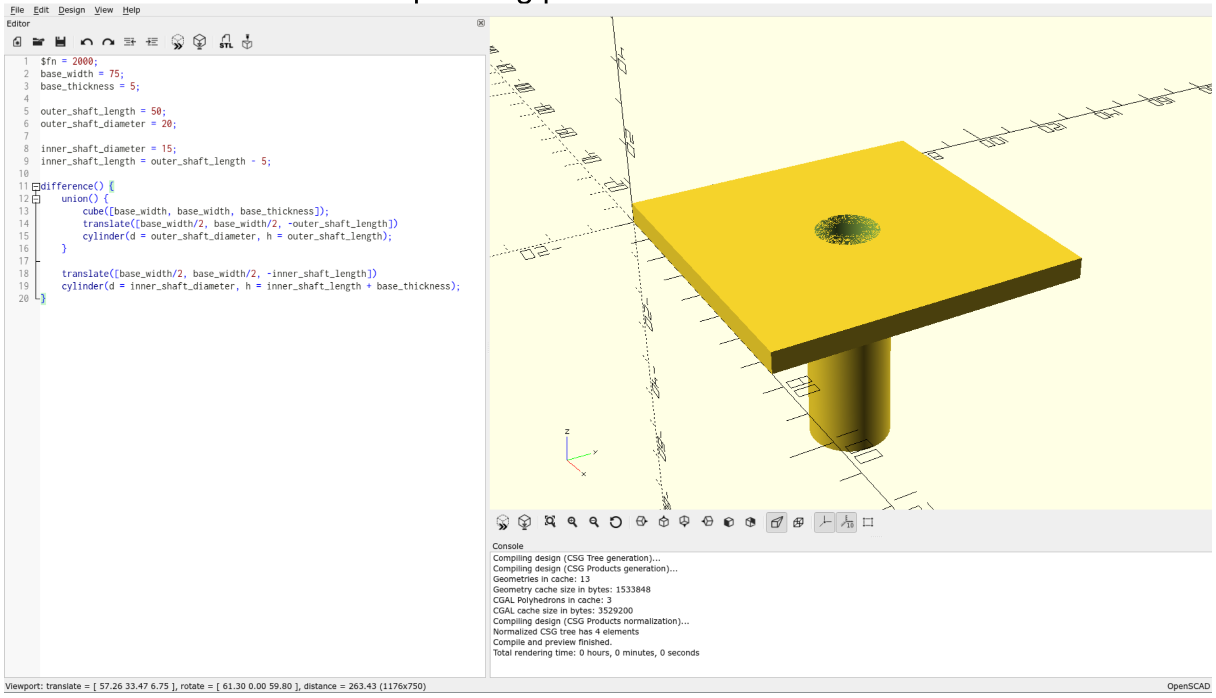
Notice how “inner_shaft_length” is defined in terms of “outer_shaft_length”.

To cut the cylinder out of the shaft we need to create it after the union but still inside of the difference. We are going to create the cylinder in the same way we did before. Firstly creating the correct size cylinder and the moving it to the correct location: The code should look something like this:

```
difference() {
    union() {
     ...
    }
    translate([base_width/2, base_width/2, -inner_shaft_length])
    cylinder(d = inner_shaft_diameter, h = inner_shaft_length +
                                          base_thickness);
}
```

Press preview. If this has gone correctly you may see some visual artifacts on the top of the model. When you move the viewport around they will shimmer. This is normal. When you use difference and a face of the base shape and a face of the cutouts are exactly on top of each other this happens. You can get rid of this by rendering instead.

What it should look like after pressing preview:

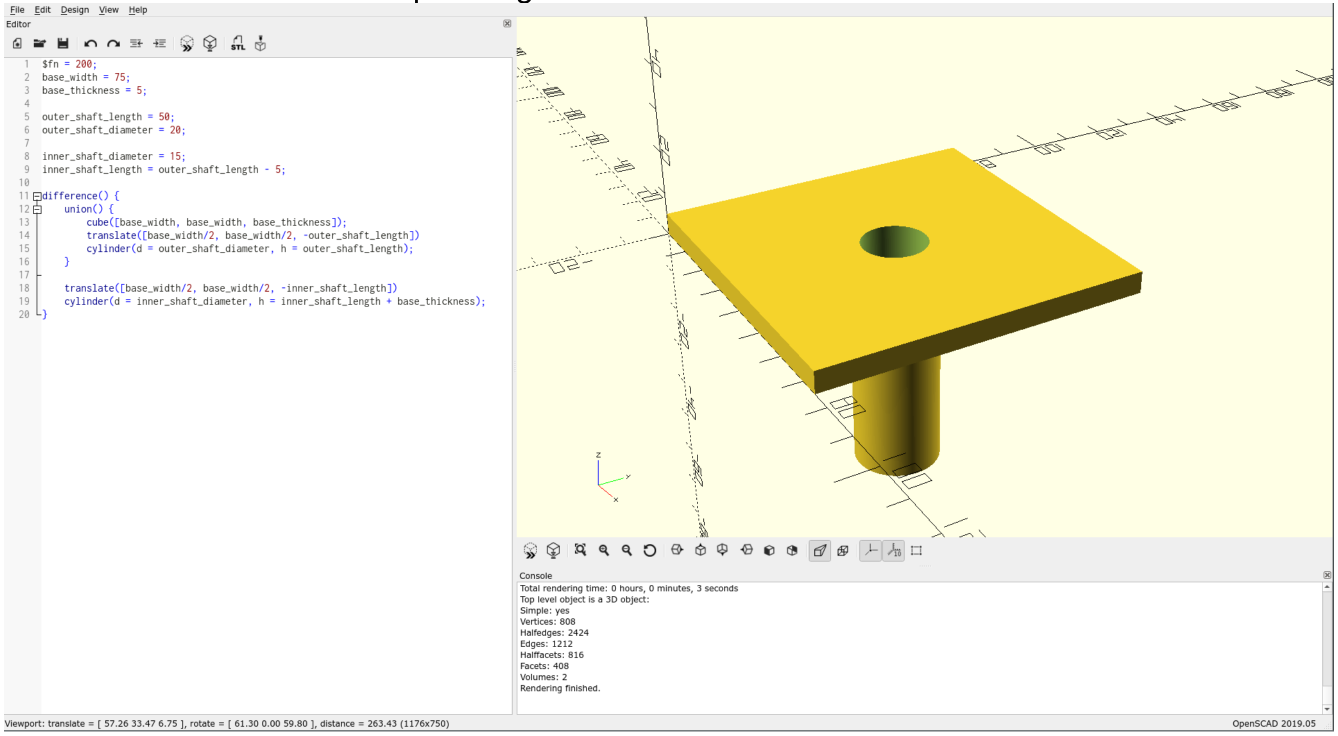


```
$fn = 2000;
base_width = 75;
base_thickness = 5;

outer_shaft_length = 50;
outer_shaft_diameter = 20;

inner_shaft_diameter = 15;
inner_shaft_length = outer_shaft_length - 5;

difference() {
    union() {
        cube([base_width, base_width, base_thickness]);
        translate([base_width/2, base_width/2, -outer_shaft_length])
        cylinder(d = outer_shaft_diameter, h = outer_shaft_length);
    }

    translate([base_width/2, base_width/2, -inner_shaft_length])
    cylinder(d = inner_shaft_diameter, h = inner_shaft_length + base_thickness);
}
```

```
Compiling design (CSG Tree generation)...
Compiling design (CSG Products generation)...
Geometries in cache: 13
Geometry cache size in bytes: 1533848
CGAL Polyhedrons in cache: 3
CGAL cache size in bytes: 3529200
Compiling design (CSG Products normalization)...
Normalized CSG tree has 4 elements
Compile and preview finished.
Total rendering time: 0 hours, 0 minutes, 0 seconds
```

Viewport: translate = [57.26 33.47 6.75], rotate = [61.30 0.00 59.80], distance = 263.43 (1176x750)

What it should look like after pressing render:



```
$fn = 200;
base_width = 75;
base_thickness = 5;

outer_shaft_length = 50;
outer_shaft_diameter = 20;

inner_shaft_diameter = 15;
inner_shaft_length = outer_shaft_length - 5;

difference() {
    union() {
        cube([base_width, base_width, base_thickness]);
        translate([base_width/2, base_width/2, -outer_shaft_length])
        cylinder(d = outer_shaft_diameter, h = outer_shaft_length);
    }

    translate([base_width/2, base_width/2, -inner_shaft_length])
    cylinder(d = inner_shaft_diameter, h = inner_shaft_length + base_thickness);
}
```

```
Total rendering time: 0 hours, 0 minutes, 3 seconds
Top level object is a 3D object:
Simple: yes
Vertices: 808
Halfedges: 2424
Edges: 1212
Halffacets: 816
Facets: 408
Volumes: 2
Rendering finished.
```

```
Viewport: translate = [ 57.26 33.47 6.75 ], rotate = [ 61.30 0.00 59.80 ], distance = 263.43 (1176x750)
OpenSCAD 2019.05
```

Step 10: Add mounting holes.

OpenSCAD has for loops that allow you to generate do the same thing multiple times but with changing values.

The syntax for a for loop looks like this:

```
for (i = [start:step:end]) {
    ...
}
```

What this will do is that it will first run the code in between the brackets with “i” set to the value of “start”. Then it will add “step” to “i”. And run the code between the brackets again. It will keep adding “step” to “i” and running the code between the brackets until “i” is greater than “end”.

While we are only going to use this to make four holes, and so this is a little overkill, this method is very useful for making lots of the same object so it’s worth seeing.

First, we need to use some values several times so we are going to need to make some variables.

```
mounting_hole_radius = 2.5;
distance_from_edge = 10;
```

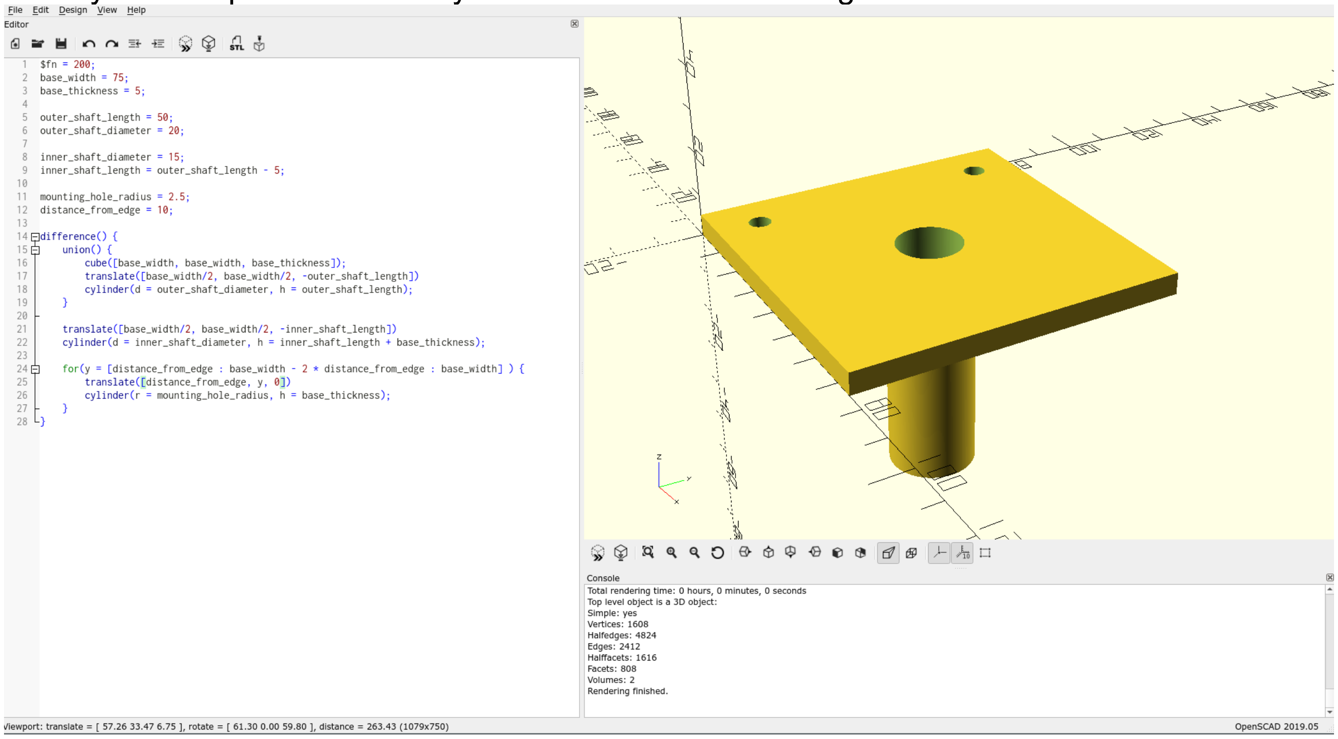
First, we are going to make only two of the holes, so it's a little easier to see what is happening.

Inside of the difference but after the union, we are going to make a for loop that creates cylinders of the correct size, then moves it to the correct place. The code should look like this:

```
difference() {
    union() {
          ...
    }

     ...

    for(y = [distance_from_edge : base_width - 2 * distance_from_edge
: base_width] ) {
        translate([distance_from_edge, y, 0])
        cylinder(r = mounting_hole_radius, h = base_thickness);
    }
}
```

This might look a little bit complicated, it's not helped by the fact that the for is going over two lines. You can expand your editor by dragging on the edge between it and the viewport to change its size.

Once you have pressed render you it should look something like this:



```
$fn = 200;
base_width = 75;
base_thickness = 5;

outer_shaft_length = 50;
outer_shaft_diameter = 20;

inner_shaft_diameter = 15;
inner_shaft_length = outer_shaft_length - 5;

mounting_hole_radius = 2.5;
distance_from_edge = 10;

difference() {
    union() {
        cube([base_width, base_width, base_thickness]);
        translate([base_width/2, base_width/2, -outer_shaft_length])
        cylinder(d = outer_shaft_diameter, h = outer_shaft_length);
    }

    translate([base_width/2, base_width/2, -inner_shaft_length])
    cylinder(d = inner_shaft_diameter, h = inner_shaft_length + base_thickness);

    for(y = [distance_from_edge : base_width - 2 * distance_from_edge : base_width] ) {
        translate([distance_from_edge, y, 0])
        cylinder(r = mounting_hole_radius, h = base_thickness);
    }
}
```

```
Total rendering time: 0 hours, 0 minutes, 0 seconds
Top level object is a 3D object:
Simple: yes
Vertices: 1608
Halfedges: 4824
Edges: 2412
Halffacets: 1616
Facets: 808
Volumes: 2
Rendering finished.
```

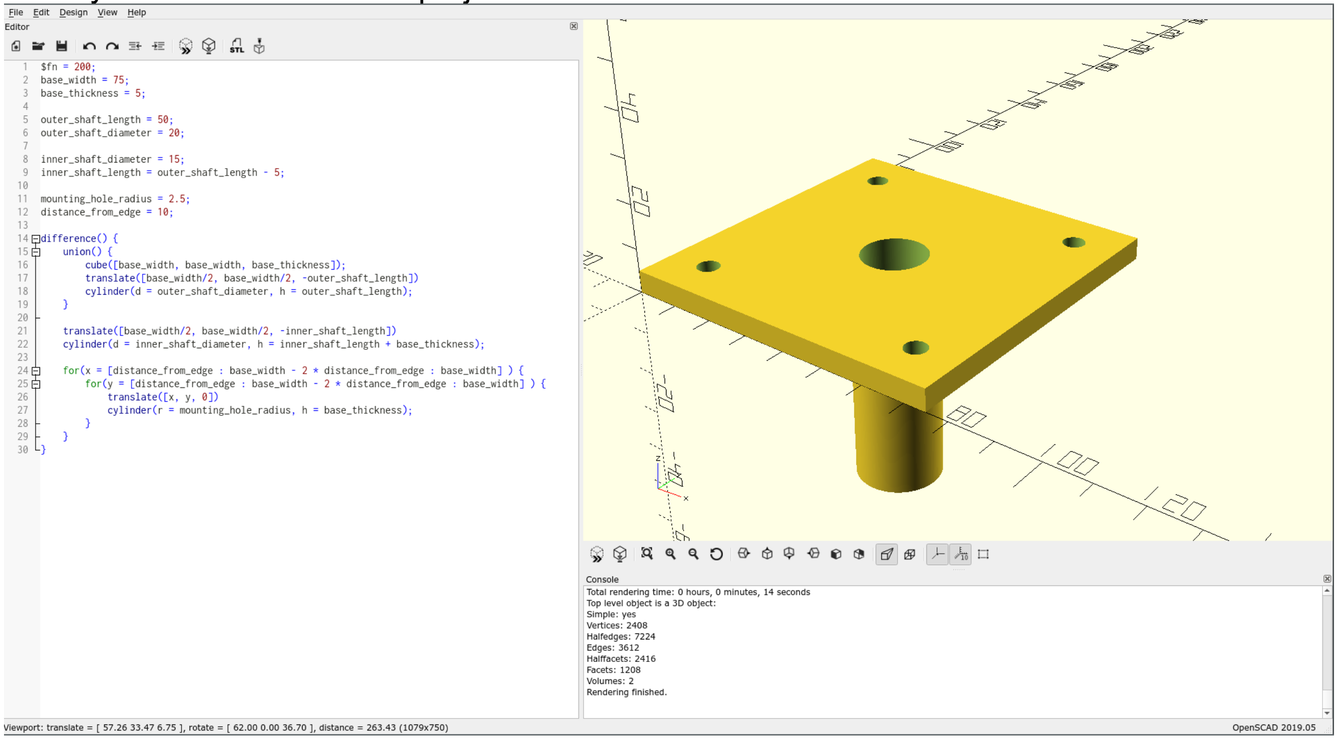
Viewport: translate = [57.26 33.47 6.75], rotate = [61.30 0.00 59.80], distance = 263.43 (1079x750)   OpenSCAD 2019.05

Now we're going to surround this is another for loop that is going to change the values in the x-direction in the same way:

```
difference() {
   union() {
     ...
   }
   ...
   for(x = [distance_from_edge : base_width - 2 * 
distance_from_edge : base_width] ) {
        for(y = [distance_from_edge : base_width - 2 * 
distance_from_edge : base_width] ) {
            translate([x, y, 0])
            cylinder(r = mounting_hole_radius, h = base_thickness);
        }
    }
}
```

Be careful not to miss that the "distance_from_edge" in the translate changes to an "x".

Once you have rendered the project it should look like this:



```
$fn = 200;
base_width = 75;
base_thickness = 5;

outer_shaft_length = 50;
outer_shaft_diameter = 20;

inner_shaft_diameter = 15;
inner_shaft_length = outer_shaft_length - 5;

mounting_hole_radius = 2.5;
distance_from_edge = 10;

difference() {
    union() {
        cube([base_width, base_width, base_thickness]);
        translate([base_width/2, base_width/2, -outer_shaft_length])
        cylinder(d = outer_shaft_diameter, h = outer_shaft_length);
    }

    translate([base_width/2, base_width/2, -inner_shaft_length])
    cylinder(d = inner_shaft_diameter, h = inner_shaft_length + base_thickness);

    for(x = [distance_from_edge : base_width - 2 * distance_from_edge : base_width] ) {
        for(y = [distance_from_edge : base_width - 2 * distance_from_edge : base_width] ) {
            translate([x, y, 0])
            cylinder(r = mounting_hole_radius, h = base_thickness);
        }
    }
}
```

```
Total rendering time: 0 hours, 0 minutes, 14 seconds
Top level object is a 3D object:
Simple: yes
Vertices: 2408
Halfedges: 7224
Edges: 3612
Halffacets: 2416
Facets: 1208
Volumes: 2
Rendering finished.
```

Viewport: translate = [57.26 33.47 6.75], rotate = [62.00 0.00 36.70], distance = 263.43 (1079x750)

OpenSCAD 2019.05

Step 11: Export

To export your model to a format that other programs can use, first you need to render the project. Then you need to press F7 or press the button with a piece of paper that is folded over and has “STL” written in front of it (above the editor). Then you can save the file where you like.