



Starting design with OpenSCAD

Adrian Johnstone, January 3, 2020

OpenSCAD is a free text-based 3D modelling tool which runs on Windows, OS X and Linux. It is easy to install and use, but complicated things sometimes need a lot of thought.

The learning curve – visual vs text-based tools

Most 3D design tools are visual: objects are drawn directly by the engineer and manipulated using a point-and-click interface. Skilled users can achieve excellent results but learning such a tool can take a long time.

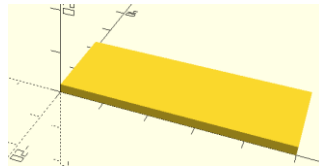
There is an alternative: write a script describing how to build the 3D model out of a few geometrical primitives such as cylinders and cuboids, and then have the computer draw it for you. The OpenSCAD tool uses this approach and runs on all common current personal computers. There is a related tool called OpenJSCAD which runs purely in the browser and thus can even be used on tablets and phones.

These tools are easy to learn because they only provide a few very simple facilities. On the other hand, operations such as adding a fillet to a joint can probably be done with a single command in most visual packages, whereas in OpenSCAD some ingenuity is sometimes required.

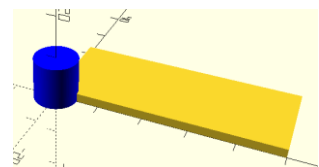
The basic approach

Text based tools use *computational solid geometry* to build up complex objects from simple geometrical shapes and operations such as union, intersection and difference.

For instance, a 2mm thick plate can be specified as `cube([50,20,2]);` which when rendered looks like this:



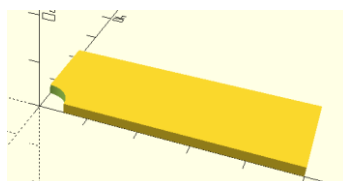
We can add a blue 5mm diameter cylinder by writing: `color("blue") cylinder(10,5,5);`



Objects can be 'subtracted' from one another using the `difference()` operation. If we write

```
difference() { cube([50,20,2]); cylinder(10,5,5); }
```

we get

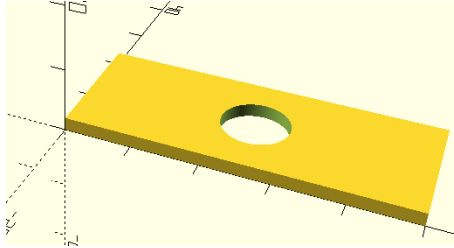


Objects can be moved about in space using the `translate()` operation. In OpenSCAD, points in space are specified using a *vector* which is written as a coordinate triple in square brackets: `[x, y, z]`.

If the cylinder's position is moved to the centre of the plate, then we get the effect of a drilled hole:

```
difference() {  
    cube([50,20,2]);  
    translate([25,10,0]) cylinder(10,5,5);  
}
```

renders as



Advantages and disadvantages

The main advantages of these script-based systems are

Precision: there is no need for a visual snap grid or complicated snap operators – object locations and dimensions are specified numerically and exactly.

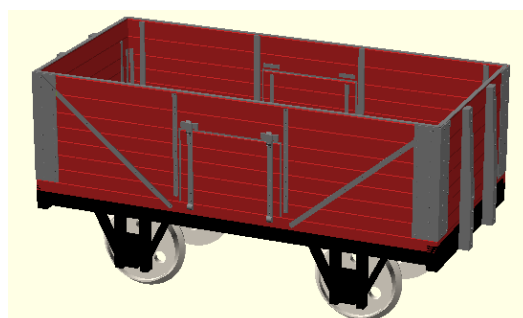
Interoperability: scripts are simple text files which can be edited with normal text editors, emailed and displayed without needing to install special software.

Modularity: the language encourages packaging of designs into small named modules that can be shared and reused

Paramaterisability: instead of using explicit numbers everywhere, we can used named values that are specified once and used many times: modifying the single named value allows global changes to the properties of a 3D model.

Free, mostly bug free and lots of free examples: these tools are simple compared to visual editors and have few untested dark corners. OpenSCAD has a vast number of users, so there is a lot of online help available.

The main disadvantage is that rather than just picking up the corner of an object on the screen and stretching it so that it looks right, the engineer has to work out numerically where it should go, and that often requires some trig and geometry. Specifications can also become quite large and that requires discipline when organising modules. For instance the script for this RCH wagon, with its multitude of plates, nuts and bolts, is over a thousand lines long.

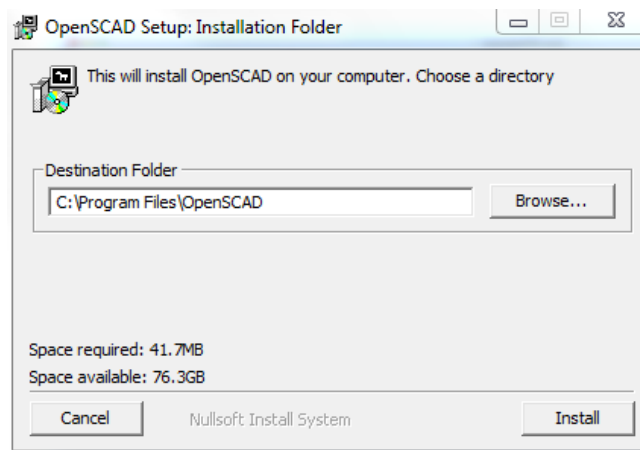


Installing OpenSCAD on Windows and generating 3D prints

1. Use a web browser to visit <https://www.openscad.org/> which in January 2020 looked like this:



2. Click the **Download OpenSCAD** button and follow your browser's procedure to save and run the file **OpenSCAD-2019.05-x86-64-Installer.exe** which will take you to this dialog box:

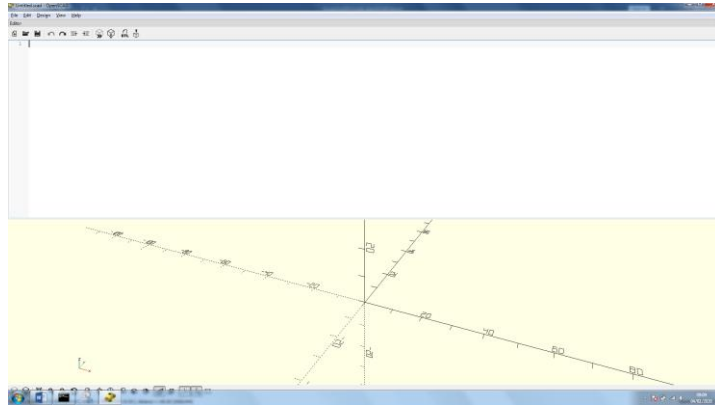


3. The default installation directory is fine, so just click the **Install** button. OpenSCAD is small, so installation should take little time.

4. Start OpenSCAD via the Windows start menu. The welcome dialog looks like this (but your list of recent files would initially be empty).



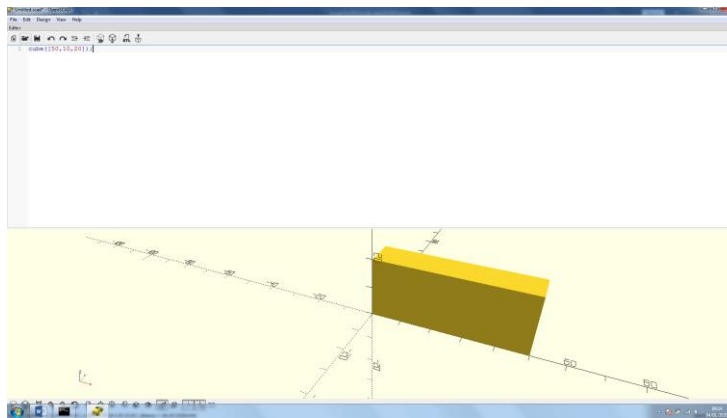
5. Click on the **New** button. A window will open showing an empty text editor and an empty 3D preview.




6. Click on the white editor window, and type into it this line

```
cube ([ 50 , 10 , 20 ] ) ;
```

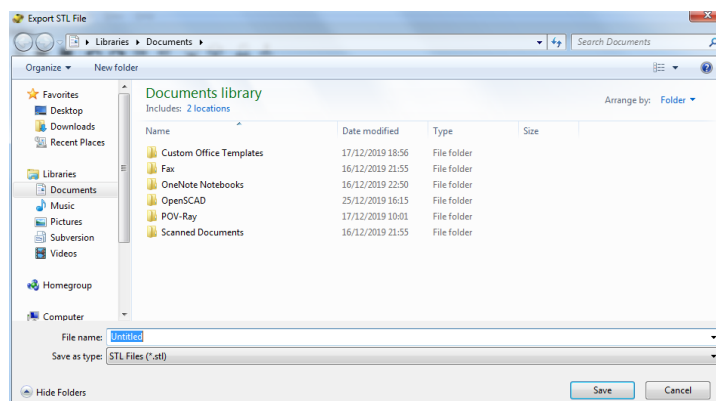
7. Press function key 5 (F5) and the script will be rendered



8. Click on the 3D preview window and try rotating and panning by click-and-dragging the left and right buttons. You can zoom in and out using your mouse wheel, or by using the  buttons at the bottom of the window.

9. The F5 preview function is fast for viewing but does not produce the detail needed for a 3D print. To prepare for printing, press F6 to do a full render. For simple objects, the appearance does not change. For complex objects, especially those with curves, there may be an appreciable delay.

10. Then press F7 to get the save STL dialog, save the STL file to your preferred location and try a test print.



Example 1 – a door wedge

Here is a 3D printed door wedge which we can describe in OpenSCAD as the difference between two cuboids



The wedge is 12cm by 3cm by 2.5cm at its highest point. OpenSCAD uses millimetres as the basic unit, so we make a cuboid of the correct size, centred at the origin by writing:

```
cube([120, 30, 25], center = true);
```

By default, everything in OpenSCAD renders as a sort-of golden yellow. When building up a design, it is useful to add colours to show the parts that are being currently manipulated: an object can be prefixed by a colour specification (written the American way):

```
color ("cyan") cube([120, 30, 25] , center = true);
```

We now add a second cuboid, rotated by seven degrees around the Y axis to form the top surface of the wedge:

```
color ("cyan") cube([120, 30, 25], center = true);  
rotate([0,-7,0]) cube([125, 35, 25], center=true);
```

The slanted cuboid needs to be raised by 10mm: a translation of 10 units along the Z axis:

```
color ("cyan") cube([120, 30, 25], center = true);  
translate([0,0,10])  
rotate([0,-7,0]) cube([125, 35, 25], center=true);
```

Finally, we 'subtract' the slanted cuboid from the original one to form the wedge:

```
difference() {  
  color ("cyan") cube([120, 30, 25], center = true);  
  translate([0,0,10])  
  rotate([0,-7,0]) cube([125, 35, 25], center=true);  
}
```

