LEARNING OBJECTIVES

General
• Confidence writing basic code with simple parameters
• Understanding measurement and dimensions

3D Design (Parametric Modeling)
• Modifying parameters
• Basic OpenSCAD code
• Translation

TERMINOLOGY
• Variable: Symbol that signifies a value that can be fixed or changed depending on its definition
• Function: A body of code that returns a value or action
• Parameter: A variable within a set boundary
• Facets: Flat surfaces that make up the outside of an object. The more facets, the smoother the object.
• Debug: To search for and fix incorrect portions of code
• Customizer: Program built into Thingiverse.com that allows OpenSCAD files to be uploaded as user-editable models
• Render: To generate an output based on code written. In OpenSCAD, the render output is a solid object.

INVESTIGATE:
PARAMETRIC AND CUSTOMIZABLE MODELS

One of the advantages of designing models with code is that you can make them parametric. This means you can have elements in your model that are easy to change, like the space between two parts or the length of a lever. The design process is all about iteration: you design something, try to print it in real life, learn from your model, redesign your model, and repeat. Parametric models help shorten the design process.

Before you get started with your own code, take a look at some of the customizable models on Thingiverse. Each customizable model was designed with OpenSCAD code that was then put into MakerBot Customizer, which provides an easy user interface for modifying parameters in the model.

• Go to www.thingiverse.com. Select the Explore tab and then choose Customizable Things. You’ll see a wide variety of models that Thingiverse users have contributed.
• If desired, change the second category from All to another one, for example Toys and Games.
• Now choose something for the third category, for example, Dice. Many people have contributed models to Thingiverse for customizable dice! Choose one of the models, and once the model is open, press the button that says Open in Customizer.

• Within Customizer, you’ll be able to modify whatever parameters the designer has made parametric, or changeable, in their model. Each model will have different parameters that will vary based on what the designer has enabled. Play around with Customizer; every time you choose or change a value on the left interface, the picture on the right will update to reflect your changes. • You can also look at the code that the designer used to make their model by clicking on the View Source button that’s under the picture of the model in Customizer. Some models are made with very complicated code, and some are surprisingly simple.

• Investigate two or three other customizable models of different types, changing parameters and peeking at the code for each one. There is great power, flexibility, and variety in designing with code. What would you design with this type of tool?

EXPLORE:
MODELING WITH OPENSCLAD CODE

When you open the OpenSCAD software, you’ll see a window like the one shown. The basic workflow is to type code into the editor window on the left, press F5 on a PC (or Function-F5 on a Mac) to compile the code, and then look in the view area window on the right to see the result.

A small console log at the bottom right will display output notes and sometimes error messages. You can use the mouse to navigate in the view area. Follow the steps in this section to get comfortable with how OpenSCAD works.
INTERFACE:

1. Editor window – Type code here to define your model
2. System options – Save your file, undo/redo actions, and format your code
3. Model options – Preview, Render, and Export STL files from this menu
4. View area – Displays the model defined in the editor window
5. View menu – Change your viewing angle using your mouse or the buttons in this menu
6. Console – Displays output notes and error messages
STEP 1: TYPE AND COMPILE YOUR FIRST OPENSCAD COMMAND

- Type `cube([30,40,10]);` in the editor window. Be sure to type it exactly as written.
- Press F5 (or Function-F5) to see the resulting cuboid. As an alternative to using F5, you can use the Preview button in the Model Options menu.
- If your axes and/or scale markers are not visible, turn them on from the View menu.
- Notice that the cuboid is **30 mm** in the x-direction, **40 mm** in the y-direction, and **10 mm** the z-direction.
STEP 2: NAVIGATE AROUND THE OBJECT IN THE VIEW AREA.

• **Orbit** – *Left-click and drag* to orbit around the model.
• **Pan** – *Right-click and drag* to move the view around.
• **Zoom** – *Scroll the mouse wheel* up and down.

STEP 3: GET COMFORTABLE WITH BASIC OPENSCAD PRIMITIVES.

• Try typing the **primitives** commands in the table below into your OpenSCAD editor window and then press F5 (or Function-F5) to see what happens.
• Experiment with modifying the code to get a feel for how the commands work.

<table>
<thead>
<tr>
<th>CODE EXAMPLE</th>
<th>WHAT THE CODE DOES</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>cube(30);</td>
<td>Creates a cube centered at the origin with all sides of length 30mm.</td>
<td>The semicolon tells OpenSCAD to draw the object. You'll need one at the end of many of your lines of code.</td>
</tr>
<tr>
<td>cube([30,40,10]);</td>
<td>Creates a cuboid with one of its corners at the origin and side lengths of 30 mm, 40 mm, and 10 mm.</td>
<td>The square brackets denote [x,y,z] coordinates. The round brackets (parentheses) pass information to the cube command.</td>
</tr>
<tr>
<td>cube([30,40,10], center=true);</td>
<td>Creates the same cuboid as above but with the center of the cube at the origin.</td>
<td>Sometimes one type of centering is more convenient than another.</td>
</tr>
<tr>
<td>sphere(20, $fn=24);</td>
<td>Creates a sphere with radius 20mm, centered at the origin, and with 24 facets around the equator.</td>
<td>Increasing the number of facets makes a smoother sphere, but can make your code take longer to compile.</td>
</tr>
<tr>
<td>cylinder(h=20, r=10, $fn=40);</td>
<td>Creates a cylinder with height 20mm and radius 10mm, centered at the origin, and with 40 facets around the equator.</td>
<td>Try setting $fn=6 or $fn=4; with a low number of facets so your cylinder can be a hexagonal or square prism.</td>
</tr>
<tr>
<td>cylinder(h=20, r1=10, r2=5);</td>
<td>Creates a truncated cone with height 20mm, lower radius 10mm, and upper radius 5mm, centered at the origin.</td>
<td>Note that when you don't specify $fn, OpenSCAD uses a default value.</td>
</tr>
</tbody>
</table>
STEP 4: GET COMFORTABLE WITH BASIC OPENSCAD MODIFIERS.

OpenSCAD also has commands for moving, scaling, extruding, and combining primitives. These are powerful tools for turning primitives into complete designs.

- Try typing the code snippets below into the editor window, and press F5 (or Function-F5) to see what happens.
- Experiment with modifying the code to get a feel for what the commands can do.

<table>
<thead>
<tr>
<th>CODE EXAMPLE</th>
<th>WHAT THE CODE DOES</th>
<th>NOTES</th>
</tr>
</thead>
<tbody>
<tr>
<td>translate([-20,0,30])  cube(20, center=true);</td>
<td>Creates a 20mm cube centered at the origin, and then shifts the location of that cube -20 mm in the x-direction and 30 mm in the z-direction</td>
<td>The translate command modifies whatever primitive immediately follows it in the code. Note that translate itself doesn't get drawn, so it has no semicolon.</td>
</tr>
<tr>
<td>rotate(45, [0,1,0])  cube(20, center=true);</td>
<td>Creates a 20mm cube centered at the origin, and then rotates that cube 45 degrees around the y-axis (along the vector [0,1,0])</td>
<td>You may want to left-drag in the view area to look around the object and see how it was rotated.</td>
</tr>
<tr>
<td>linear_extrude(h=20)  circle(20, $fn=6);</td>
<td>Creates a “circle” of radius 20 mm with only six facets (in other words, a hexagon), and then extrudes that shape upward for 20 mm.</td>
<td>Notice the use of the 2D primitive circle. The command linear_extrude only works on 2D objects.</td>
</tr>
<tr>
<td>linear_extrude(h=20, twist=60)  circle(20, $fn=6);</td>
<td>Creates the same “hexagon-circle” with radius 20mm, and then extrudes that shape upwards for 20mm while rotating a total of 60 degrees.</td>
<td>The linear_extrude command is a modifier, not a primitive, so it does not need a semicolon.</td>
</tr>
<tr>
<td>difference(){  cube(20, center=true);  translate([0,0,-20])  cylinder(h=50, r=8);  }</td>
<td>Create a 20 mm cube and then remove a cylinder shape from it. The cylinder is translated downward so it penetrates through the cube.</td>
<td>The difference command draws the first object and then subtracts every other listed object from the first.</td>
</tr>
</tbody>
</table>

STEP 5: MAKE SIMPLE OBJECTS WITH OPENSCAD CODE.
Now use the code above to make some basic objects. See how many of the following objects you can make with OpenSCAD code. Remember to press F5 (or Function-F5) after each update to your code so you can see the result.

- A 30 mm sphere that’s shifted –10 mm in the x-direction, 40 mm in the y-direction, and 15 mm in the z-direction.
- A cuboid that measures 10 mm in the x-direction, 100 mm in the y-direction, and 5 mm in the z-direction.
- Two cylinders that intersect at right angles to each other.
- A pointy cone shape with another pointy cone shape underneath it in the opposite direction (like two ice-cream cones stuck together around their rims).
- A box with holes in each of its sides. You can use three cylinders to form the holes. Can you align the holes exactly to the center?
- An octagonal prism that twists 30 degrees from its base to its top.
- An open box with three small spheres inside it.
- A snowman!

STEP 6: EXPLORE MORE WITH ONLINE OPENScad TUTORIALS AND DOCUMENTATION.

- David Dobervich’s OpenSCAD Tutorial #1 video on YouTube
- Patrick Conner’s Welcome to OpenSCAD video on YouTube
- OpenSCAD documentation, Cheat Sheet, and list of Getting Started Tutorials
  http://www.openscad.org/documentation.html

CREATE:
CUSTOMIZE A NAME TAG USING OPENScad CODE

In this project, you’ll design and 3D print a wavy-bordered nametag by modifying parameters in OpenSCAD code. You’ll also dive into the code and learn how these parameters create the nametag design. Advanced students may want to change the code itself to create even more varied designs.
STEP 1: TRY TO FIGURE OUT WHAT THE CODE IS DOING.

Fully commented code for this lesson is downloadable from the MakerBot Learning account on Thingiverse.

Open the file **MakerBot_NametagCode_Project.scad** from MakerBot Learning on Thingiverse and look through the code in the editor window.

Don’t type or change anything yet; just scroll through, look at the text, and try to figure out what might be going on in the code. The code should look something like the image below.
Discussion

- In groups, discuss the code among yourselves, sharing ideas about what each part of the code might be for.
- Each group should write down at least three ideas to share with the class.
- Discuss each group’s ideas as a class, trying to decode the code together.
STEP 2: MODIFY TEXT AND FONT PARAMETERS
Near the top of the code are three sets of parameters that students can use to customize their nametag. The first set of parameters concerns the text, font, and font size of the nametag:

```plaintext
// TEXT AND FONT myword = “Caroline”;
myfont=”Phosphate:style=Inline”; fontsize = 8;
```

You can choose to use your name or word in `myword`, but keep in mind that longer names and words will take longer to 3D print. The `myfont` parameter uses the font names of the fonts installed on your local system. Use the Help / Font List pull-down menu to obtain a list of your available fonts. You can drag and drop font names directly from this list into your OpenSCAD code or copy/paste into the code. You should not change `fontsize` unless the student chooses a particularly large or small font shape.

Remember that after each change, you must press F5 (or Function-F5) or the preview icon to see the updated result in the view area. Compiling errors at this stage are usually due to lost quotation marks, missing semicolons, and incorrectly spelled font names.

STEP 3: ADJUST DIMENSIONS TO MATCH THE TEXT
The next set of parameters sets the overall size of the nametag:

```plaintext
// TOTAL DIMENSIONS
length = 56; width = 15;
height = 8;
```

The `length` and `width` parameters will have to be adjusted depending on the size and length of your word choices in the previous step. The `height` parameter determines the height of the tallest part of the border wave. Compiling errors at this stage are usually due to lost semicolons.

STEP 4: STYLE THE WAVE OF THE NAMETAG BORDER
The final set of parameters determines the style of the wavy border around the outside of the nametag:

```plaintext
// BORDER CURVE STYLE
frequency = 1; steps = 8; border = 2; inset = 2; base = 2;
```

The wavy border’s shape is determined by a trigonometric function, but you do not need to know any trigonometry to modify these parameters. The `frequency` parameter changes the number of waves. The `steps` parameter controls the number of times the wave height changes; lower values make stair-step shapes and higher values make smoother curves.
You’ll likely not need to change the border, inset, or base parameters, but feel free to explore adjusting them to see what happens.

**STEP 5: CREATE A NEW PARAMETER FOR TEXT HEIGHT**

So far, you’ve just been modifying existing parameters in the OpenSCAD code. In this step, you’ll create a new modifiable parameter. Look at the second section of the code, below the //PRINT THE TEXT comment on line 47. Notice that the code is read in reverse order: First the 2D text is created, then the letters are extruded upward, then the extruded letters are shifted up and to the right so they do not intersect the border.

- Determine the part of the code that controls the height of the letters, and figure out what to change to make the letters shorter or taller. (Hint: The text height starts out set to 6 mm.)
- Replace the numerical value on line 55 with a named parameter called textheight.
- Add a parameter definition to the top of the document on line 23 by typing `textheight = 6;`
- Now you can modify the text height by changing the value of this parameter on line 23.

**STEP 6: CREATE A NEW PARAMETER FOR CHARACTER SPACING.**

Now repeat the same process to add a new parameter called myspacing that controls the amount of space between the characters of the printed name.

- Try to find the place where a myspacing parameter would go in the code. (Hint: it’s part of the `text` command and it starts out set to 1.)
- Once this parameter is set up, modify its value to adjust the character spacing to your liking. The default value of 1 represents normal spacing. Note that even small changes like modifying to 1.2 or .9 can change the spacing considerably.
• You may have to readjust the **width** of the nametag if the student makes substantial changes to the **myspacing** parameter.

**STEP 7: OUTPUT AN STL MESH FOR 3D PRINTING.**

Once you’ve done designing your nametag, export the designs to STL files for 3D printing:

• First, to be safe, save your OpenSCAD code file and/or write down notes about the values of the parameters you chose in your design. The exported STL file will not save or record those numerical values, and if there’s a problem with an STL file, it’s good to have a way to reconstruct the basic design again.

• To generate a 3D mesh suitable for printing, press **F6** (or **Function-F6**) or use the **Render** button. This process can take a lot longer than **Preview**, especially if you’ve chosen high values for the **step** parameter. Watch the progress bar at the bottom right of the screen.

  *Note: The **F5** action only generates a preview of the 3D object in the view area, not a complete render.*

• When the **F6** render has completed, use the pull-down menu or the **Export STL** button to export the model as an STL file.

• The nametag STL files can now be imported into **MakerBot Desktop** and 3D printed, either individually or with multiple nametags on one build plate.