CHAPTER 13

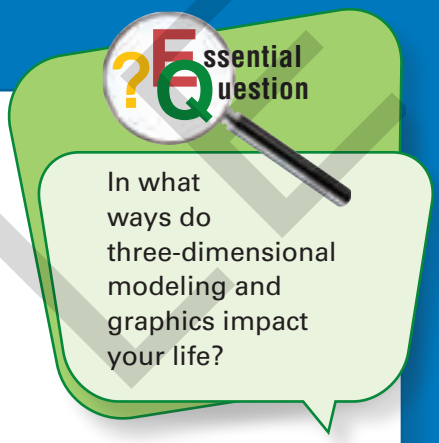
Working in Three Dimensions

Chapter Preview

Section 13.1 **Learning the Basics of Three-Dimensional Modeling**

Section 13.2 **Introducing OpenSCAD**

Section 13.3 **Exploring Careers in the 3D Field**



Learning Outcomes

- ☐ LO 13.1 Describe the basics of 3D modeling.
- ☐ LO 13.2 Use functions to create shapes in OpenSCAD.
- ☐ LO 13.3 Compose a career plan for employment in the 3D field.

Terms

extrude
model
rapid prototyping (RP)
rendering
wireframe

The CSTA icon notes correlations to the CSTA K–12 Computer Science Standards throughout this chapter. Please see Appendix C for a full listing.

Life exists in three dimensions. Real objects have volume and occupy space. Even something that seems to be two dimensional, such as a piece of paper, is actually defined by three dimensions. A piece of paper has thickness, even if very small. Think of the difference between a sheet of printer paper and a sheet of construction paper. Both sheets have a thickness, but the construction paper is much thicker.

Scratch and App Inventor create two-dimensional images and layouts. There are many software programs to create three-dimensional images and layouts. Some programs are used to create three-dimensional objects. Others are used to create images and layouts of these objects, while some programs can do it all. OpenSCAD is a free 3D modeling software. It uses mathematical functions to create models for solid objects. This chapter uses OpenSCAD to explore creating 3D models.

College and Career Readiness

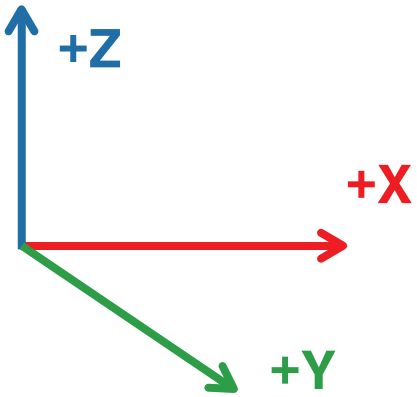
Reading Prep. **Before reading this chapter, look at the chapter title. Write a paragraph describing what you already know about the topic. After reading the chapter, write a paragraph to summarize what you have learned. How do the two paragraphs compare?**

LO 13.1

Describe the basics of 3D modeling.

Section 13.1 Learning the Basics of Three-Dimensional Modeling

A two-dimensional (2D) layout has two axes: X and Y. The point where these two axes intersect is called the origin, and has the coordinates (0,0). As you may have learned in math class, any two axes form a plane. The X and Y axes form the XY plane. However, the real world does not exist in two dimensions. Real objects have volume and fill space. This requires a third axis, the Z-axis. The X, Y, and Z axes intersect at the origin (0,0,0) and are at right angles to each other, as shown in **Figure 13-1**. The X and Y coordinates determine the length and width of the model, while the Z coordinate determines its height. A virtual object that has volume is said to be a three-dimensional (3D) ***model***. The three axes divide space into eight sections called octants. The first octant is the space where all three coordinates are positive. The origin is the point with coordinates (0,0,0).



Goodheart-Willcox Publisher

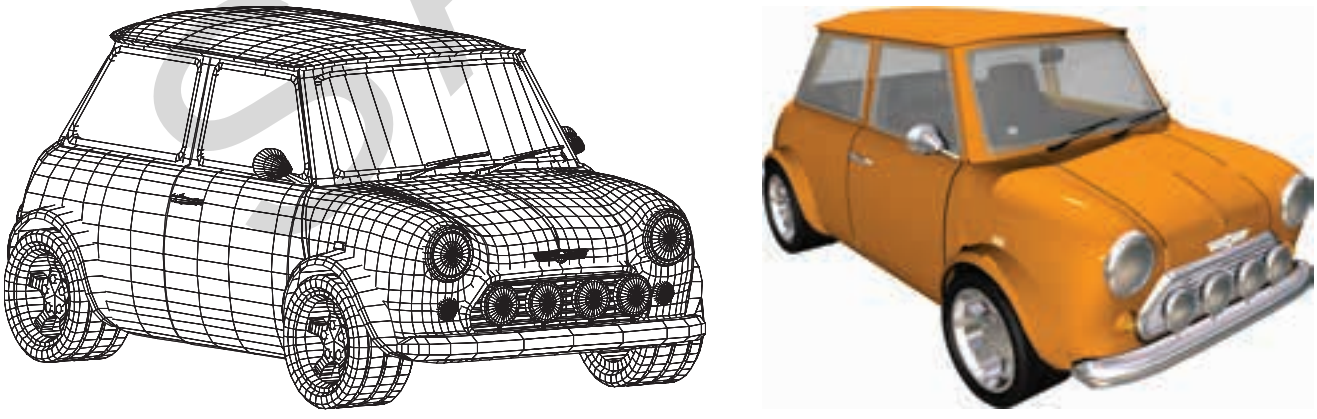
Figure 13-1.
Locating a point in space requires three coordinates. The three axes, X, Y, and Z, are at right angles to each other.

3D Software

The computer programs used to make 3D models are called 3D modeling software. Just as you would make a physical model out of clay, the software forms the shape of an object. Software does this using mathematical representation. There are two basic ways to create an electronic 3D model. The engineer or artist can draw the model directly in the software or write mathematical functions to describe the object. If the physical object already exists, it can be scanned with a 3D scanner. Either method creates a file with a 3D model consisting of a cluster of polygons, as shown in **Figure 13-2**. This cluster of polygons is referred to as the ***wireframe***. The polygons form a skin or surface over the wireframe and change shape as the object is resized or modified.

FYI

The first cruise ship designed entirely in 3D was launched by Royal Caribbean in spring of 2017.



Goodheart-Willcox Publisher

Figure 13-2.
On the left is a 3D model displayed as a wireframe. On the right is the same model with materials and textures applied, and the model is rendered.

There are many 3D modeling software programs, which can make the choices overwhelming. Also, there is no industry standard. Some for-purchase 3D modeling software includes 3ds Max, Maya, SolidWorks, Rhinoceros 3D, and Bryce. Some free 3D modeling software includes Alice, Tinkercad, SketchUp, and Blender. There are dozens of other programs, both free and for purchase.

Rendering

After a 3D model is created, there are several ways in which it can be used. It may be:

- included as part of a motion picture
- used in a video game engine
- placed in a virtual reality world
- saved to an image file and printed as a 2D graphic
- made into a physical object using a 3D printer

Rendering is the process of generating the 3D model from the instructions in the software package. If the 3D model will be used electronically, rendering may include applying colors, shading, and highlights. This is referred to as the skin for the model. The colors are based on the materials and textures applied by the designer. The highlights and shadows are based on the lighting the designer adds to the scene. When a model is used for a physical object, the colors are determined by the color of the filament used to print the object. It also may have colors applied after printing.

Many sources, including an article in *The Economist*, have argued that 3D printing signals the beginning of a third Industrial Revolution. The second Industrial Revolution is characterized by the assembly line that started to dominate manufacturing in the late 19th century. The first Industrial Revolution began in Britain in the late 18th century when the textile industry was mechanized.

3D Printing

Another way to use a 3D model is to output it as a physical object. In the past, models and prototypes for manufacturing were carved from wood or glued together from small pieces of plastic. These prototypes could take weeks to prepare. They also could be very expensive. Rapid prototyping was born in the 1980s as a way to create prototypes faster and for less money. ***Rapid prototyping (RP)*** is the process of quickly creating a physical object from a 3D model.

Part of the rapid prototyping process is using a 3D printer to create a physical part. Much of the same technology found in inkjet printers is found in 3D printers. The 3D printer builds the object by depositing material, layer by layer, in the shape of the model, as shown in **Figure 13-3**. The print head moves back and forth along the X and Y axes as it gradually moves upward along the Z-axis.

The objects created with a 3D printer are most commonly made from a plastic filament. However, there are many other materials that can be used. For example, liquid food can be squeezed out layer by layer to form decorative and edible items. Some 3D printers are designed to create parts from metal. Almost any material that can be made to flow has the potential to be adapted to 3D printing.

Alexander Tolstykh/Shutterstock.com

Figure 13-3.
A 3D printer can create a physical object from a 3D model designed in 3D modeling software.

There are many examples where 3D printing is already in use in manufacturing. Athletic shoe companies are using 3D printing to create custom-fit shoes. Eyewear companies are producing on-demand customized frames for glasses. In May 2015, Airbus announced that its new jet included over 1,000 components manufactured by 3D printing. Some military air forces are also using 3D printing to print spare parts for planes. Many consumers probably already own something that was made with the help of a 3D printer, such as a piece of jewelry or a customized cover for a smartphone.

Examples of the uses of 3D printing in the medical field are becoming more common. Surgery using 3D-printed, personalized instruments has been done in many areas. Such applications include total joint replacement and craniomaxillofacial reconstruction. Parts that have been 3D printed can be used to make robotic hands, as shown in **Figure 13-4**. The hearing aid and dental industries are expected to be the biggest areas of future development using 3D printing technology. Many modern dental offices have a 3D printer that can build a dental crown from a block of porcelain in less than 10 minutes. A 3D model is made using a 3D scan of the patient's mouth. Instead of setting down layers of material, the porcelain is carved by two diamond drills. With this in-office practice, the dentist does not need to send work to a lab. The crown can also be made and installed in a single patient visit.

Veterinarian Dr. Michelle Oblak from the Ontario Veterinary College at the University of Guelph treated a dog named Patches with a 3D-printed version of the dog's skull. Patches had a likely fatal tumor on her head. Dr. Oblak led a team to print a titanium version of Patches' skull to replace the part of her skull that was removed along with the tumor. Using this technology shaved months off Patches' recovery.

Ozgur Guvenc/Shutterstock.com

Figure 13-4.
The parts of this robotic hand were created with a 3D printer.

Biotechnology firms are making progress in creating organs and body parts using 3D printing. In this process, layers of living cells are deposited onto a gel medium one layer at a time—slowly building a three-dimensional organ such as a kidney. In 2016, researchers successfully implanted 3D-printed tissue and bone fragments into test animals. This achievement is an important step. However, being able to 3D print tissue for humans is still a long way away.

Museums are using 3D printing to help preserve cultural heritage. European and North American museums are using 3D printing to recreate missing pieces of relics. They are also able to offer 3D-printed copies of museum objects to the public. The Metropolitan Museum of Art and the British Museum create 3D-printed souvenirs to sell in their gift shops.

While these labor- or money-saving applications of 3D printing are nice to have, there is one application that is impossible to perform in any other way. The International Space Station (ISS) has a 3D printer. It is used to make parts that are not available on the ISS. Scientists on Earth create 3D models and send those files to the ISS. The 3D printer then makes the part. These applications are being tested for interplanetary flights that would not have routine supply runs from Earth.

Currently, 3D printing technology is used to build houses by making foam models and filling them with concrete for the foundation. This technology allows architects to design structures with curves and more elaborate forms than just straight lines. This technology has other applications. It could be used to create habitats on moons and other planets.

A 3D model that will be printed does not need materials and lighting. The viewing angle is also not important, and it will not be animated. What is important is where in virtual space the model is positioned.

FYI

The NASA website has many 3D models that can be downloaded and printed using a 3D printer.

Hands-On Example 13.1

Researching 3D Software

There are many 3D modeling programs. Some of these are free, while others are for-purchase software. Many of the for-purchase software programs offer a trial or demo version.

1. Using the Internet or other sources, identify five 3D modeling programs.
2. Create a table that lists the name of each program, its price, its primary use, and whether a demo or trial version is available.
3. Write a brief paragraph describing each program. Cite the website where you located the information.
4. Save the document as HOE13-01 in your working folder.
5. Discuss your findings with your class.

Try It!

There are many different 3D printers on the market. There are printers aimed at the average consumer that can be purchased for several hundred dollars. There are also high-end 3D printers that cost many thousands of dollars. These are generally used by industry. Identify four 3D printers aimed at consumers and one aimed at industry. Create a table comparing the features of the printers, including price. Save the document as TI13-01 in your working folder.

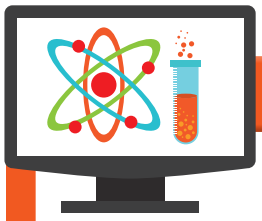
LO 13.2

Use functions to create shapes in OpenSCAD.

Section 13.2 Introducing OpenSCAD

Open SCAD is software for creating solid 3D CAD (computer aided design) objects. The user combines basic shapes and manipulates them by using the OpenSCAD scripting language. The basic shapes are from geometry. For example, cubes, spheres, circles, cylinders, and others are basic shapes in the scripting language. The user writes code to combine them into complex objects that can be saved as a 3D model suitable for printing or including in a virtual environment. The makers of OpenSCAD call this technique *constructive solid geometry*.

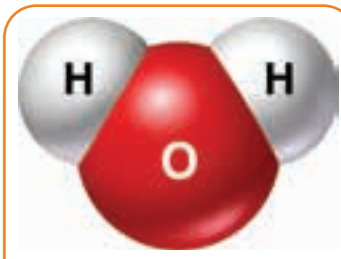
OpenSCAD is a free open-source program. This means that anyone can download it and use it. Experienced coders may also modify how OpenSCAD works for them because they have access to the source code for OpenSCAD. You may download a copy for use at home. Visit the OpenSCAD Homepage at https://openscad.org/ for download instructions.



Science and Coding

Models of Molecules

Modeling is very useful in creating an object that is otherwise difficult to see. Models of molecules are such an example. Making models, from simple molecules like water to very complex organic molecules such as the DNA molecule, is a challenging project. Relative sizes and locations of the individual atoms are critical to the accuracy of the model. In reality, the size of any atom varies with its state and proximity to other atoms. For this discussion, we will use a fixed size.



Water Molecule

Lightspring/Shutterstock.com

Models, such as this water molecule, are useful in helping scientists visualize objects that are difficult to see.

In a water molecule, the atoms are two hydrogen and one oxygen. An oxygen atom is 20 to 30 percent larger than a hydrogen atom. Notice that the hydrogen atoms are attached to the oxygen model at about a 104-degree angle. Once the model is designed with relative sizes and locations correct, it can be enlarged or shrunk to any size required.

Launch OpenSCAD and draw three spheres representing one oxygen atom and two hydrogen atoms. Set the radius of the oxygen sphere to 16 and the radius of the two hydrogens to 12. Translate the hydrogen atoms to either side of the oxygen at the proper angle. Use the information in the table to perform the translations. Color the oxygen atom red and the hydrogen atoms white.

Atom	Radius	Center of atom location	Color
Hydrogen1	12	(0, 16,12)	White
Hydrogen2	12	(0, -16,12)	White
Oxygen	16	(0, 0, 0)	Red

Preview the model. The geometry is obvious on the surface of the spheres. The default number of fragments drawn on a sphere is 30. Change the fragment number to 50 to smooth the surface. Add $fn=50 to the spheres' parameters.

Preview the model. Save your file as ScienceAndCode13.scad.

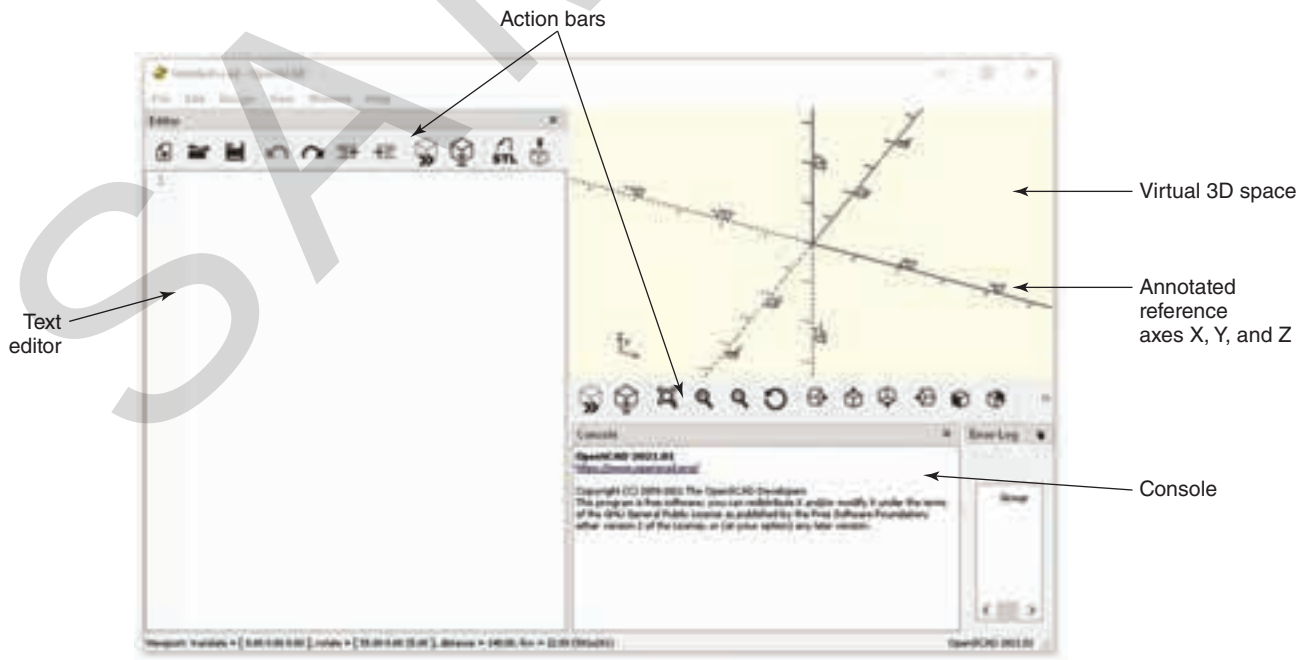
Marius Kintel and Claire Wolf created OpenSCAD and made its initial release in 2010. Updates and modifications have been made since then. The creators' intent was to provide a tool for programmers to use to create models suitable for CAD. This is not an artist's tool for making animated 3D movies.

Preparation

As you learned in Chapter 3, the first step in every project is planning. Think of an object to model. Use a storyboard to show the key points in the model. Determine the primitive shapes you can combine to make your model. Show a sketch of the final model. Make sketches of various views of the model, such as top view and side views. After the plan is created, you can begin assembling and coding the model.

Opening Screen

When OpenSCAD is launched, it looks very similar to Scratch. **Figure 13-5** shows the user interface for OpenSCAD. On the left is a text editor. This is where you write the scripts. On the right is the virtual 3D space where models are displayed. The console area provides feedback from OpenSCAD while you are building your model. The difference is that the preset blocks and sprites are missing. In this program you type in the code and the model is displayed in the 3D space. Two action bars provide quick navigation during development. You will find that the use of the mouse is limited to moving the cursor between these windows and action bars. You have no interactive mouse movements with the model, except to change the view. The friendly drag and drop of blocks in Scratch is not available in OpenSCAD.



Goodheart-Willcox Publisher

Figure 13-5.
The user interface for OpenSCAD when the software is launched.

Buttons and Keys	Action	Description
Left mouse	Rotate view	Dragging while holding down the left mouse button rotates the view around the axes.
Right mouse	Translate view	Dragging while holding down the right mouse button translates the view left and right.
Shift-Right mouse	Zoom view	Dragging the mouse while holding down the combination of Shift key and right mouse button zooms the view in and out.
F5	Reset view	Pressing F5 or clicking the circular arrow icon resets the view.

Goodheart-Willcox Publisher

Figure 13-6.
Mouse actions to change the view in OpenSCAD.

The view can be changed by the mouse movements shown in **Figure 13-6**.

Scripting in OpenSCAD

An instruction in OpenSCAD consists of a keyword, parameters, and a semicolon. As in Scratch, instructions can be imbedded inside other instructions. OpenSCAD uses a combination of parentheses ( ), curly braces { }, and square brackets [ ] to group items in an expression or instruction.

A useful tool provided to users is the cheat sheet. It's not really cheating to use it. *Cheat sheet* is a term used by programmers to indicate a summary of syntax. It is similar to the **Code** palette in Scratch. It lists all the basic instructions that you can use in OpenSCAD. The major difference between coding in Scratch and coding in OpenSCAD is that you must type the instructions into the text editor in OpenSCAD. **Figure 13-7** shows a portion of the cheat sheet. Each instruction is listed, along with the options, or parameters, you can use with the instruction. This is called the syntax for writing an instruction. Higher-level languages call it the application programming interface (API). For example, to draw a sphere, use the instruction

```
sphere(radius | d=diameter)
```

Goodheart-Willcox Publisher

Figure 13-7.
A portion of the OpenSCAD cheat sheet showing the 3D primitives syntax with links to the full documentation.

The complete instruction for generating a sphere is given in the API documentation called the OpenSCAD User Manual. In the documentation, all the possible parameters are explained. Click the hyperlink for more information.

There are defaults for each parameter. For the sphere, you can specify if you want a part of a sphere and if you want many polygons drawn to make the sphere smooth. The higher the resolution, the more polygons and the smoother the sphere. The instruction

```
sphere( );
```

creates a complete sphere at the origin with radius 1 in low resolution.

To generate a sphere, type **sphere** into the text editor and use a number within a pair of parentheses for the radius or diameter. The vertical bar in the parentheses tells you to use either the radius or the diameter. The default is the radius, so you must use d= when specifying the diameter. Place a semicolon at the end of this instruction to let OpenSCAD know that you are finished writing the instruction. Use double slashes to write a comment about the code.

```
sphere( 5 ); // draw a sphere of radius 5
sphere( d = 14 ); // draw a sphere with diameter 14
```

Notice that the word sphere in the cheat sheet is linked to the full code documentation. The documentation for OpenSCAD is very good. You do not need to memorize the syntax, because the documentation is always available. Some instructions will be easy to learn because they are used so often. Programmers rely on documentation to write the code. Each scripting language provides pretty much the same capability. It is the syntax that changes from program to program. It is always better to look up the syntax than guess at what it might be. Another feature of the OpenSCAD UI is that as you type, the API is displayed below where you are typing.

Once the script is written, click the Preview button on either Action Bar to view the 3D model. Rolling your mouse over the icons causes a tool tip to pop up and to tell you what the icon represents. The tool tip for this icon shows the message Preview F5. This indicates that if you want to use the keyboard, pressing F5 generates the same result as clicking the icon with the mouse.


Preview

Figure 13-8 shows the preview of a 3D model of a sphere of radius 20. The sphere by default is located with its center at the (x,y,z) origin, (0,0,0). Modifying the sphere uses instructions such as translate, rotate, scale, color, and others.

Goodheart-Willcox Publisher

Figure 13-8.
Code in the text editor specifies the image in the virtual 3D space.

Using Transformations

Transformations change the model. Use **translate( )** to move the model to another location in space. Turn the model using **rotate( )**. Change the size of an object using **scale( )**. For all transformations, there must be at least one object for it to apply to. This object is called the *child*. The following example moves the center of a sphere of radius 10 to the specified point, (10, 5, 5). Notice that the curly braces indicate what object or objects are being moved. In this case **translate( )** is the transformation, and **sphere( )** is the child.

```
translate([10, -5, 5]) { sphere(20); }
```


Hands-On Example 13.2A

Using Instructions in OpenSCAD

This activity explores scripting in OpenSCAD. You will create objects from the OpenSCAD solid primitives and transform them.

1. Launch OpenSCAD and click the **New** button.
2. In the text editor, add a comment on line 1: // HOE13-2A and your name.
3. On line 2, type sphere(10); and press **F5** or click the Preview Button.
4. View the preview of the sphere centered at the origin of the virtual 3D window.
5. Click and hold the mouse button down in the virtual 3D window. Apply what you have learned about mouse use in OpenSCAD to change the view of the sphere. Notice the reference axes in the lower left corner of the window. These tell you where you are in 3D space. They move as you move the mouse.

Goodheart-Willcox Publisher


Reset View

6. Click the **Reset View** icon to reorient the axes.
7. On line 3 in the text editor, type translate([30,15,-5]){sphere(5);}.
8. Compare the two spheres.
9. On line 4 in the text editor, type cube(10); and press **F5**.

Continued

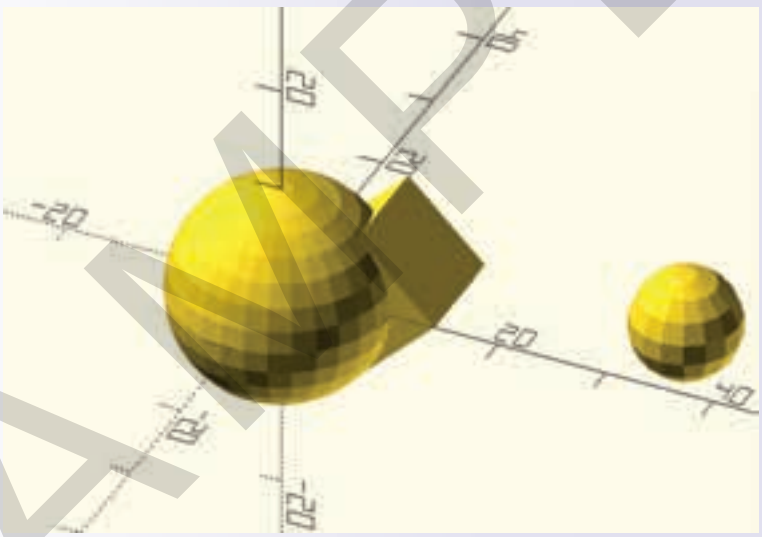
Continued

10. Notice that the cube and the large sphere intersect. They occupy some of the same space. However, they are not joined forever. That would take more instructions. To investigate this claim you will use the **rotate( )** transformation on the cube.
11. Use the cheat sheet to look up the syntax for the transformation **rotate( )**. Three values are specified. Click on the rotate instruction in the cheat sheet to open the documentation on **rotate( )** and learn what these values are. You will find that they are degrees of rotation about each axis. The instruction rotate(0,45,0) rotates the cube 45 degrees about the Y-axis.
12. Comment out line 3 by inserting two forward slashes before the **cube( )** instruction. This tells OpenSCAD to ignore the code on this line.

```
// cube(10);
```

13. On line 5, type the instruction to rotate the cube 45 degrees about the Y-axis. Press **Preview**. Compare your results with the image shown.

```
rotate([0,45,0]) {cube(10);}
```



Goodheart-Willcox Publisher

14. Save your work as HOE13-2A.scad. Summarize your observations from this Hands-On Example and prepare to share them with the class.

Try It!

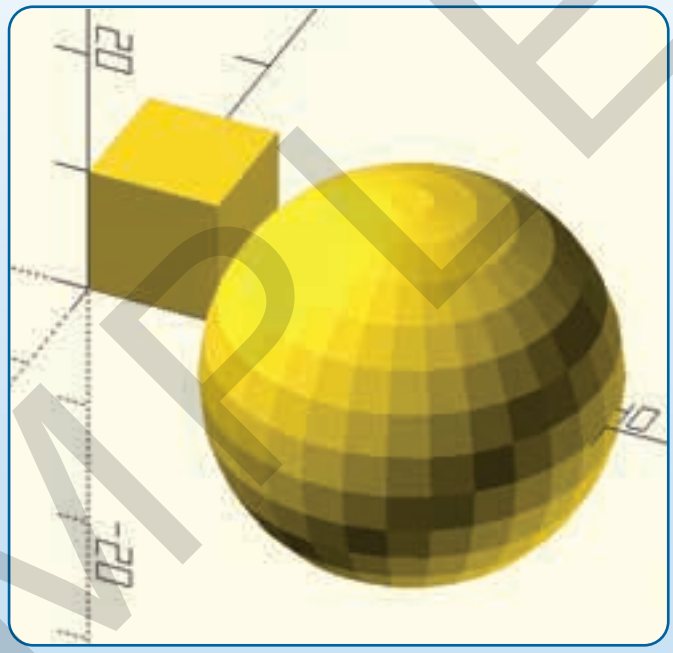
Apply what you have learned to generate a cube of size 10, translate it 20 units along the X-axis, and rotate it 45 degrees about the Z-axis. Save the file in your working folder as HOE13-2ATryIt.scad.

Coding Conundrum

Translation of Child Nodes

In the following example, the intent is to translate both the cube and the sphere 20 units along the X-axis. Only the sphere is moved. The **translate( )** transformation is designed to work on multiple child nodes. Can you find the mistake in the code?

```
1 // move the sphere and the cube 25 units along the x-axis
2 translate([25,0,0])
3     sphere(15);
4     cube(10);
```



Goodheart-Willcox Publisher

In geometry, the shape of a cylinder is defined by its radius and its height. The OpenSCAD cylinder primitive uses parameters to define its shape. The API for the cylinder primitive is shown below, where the parameters **h (height)**, **r|d (radius or diameter)**, and **center** specify what the cylinder looks like. In the second API, there are choices for two radii or two diameters. This allows the designer to make a cone or other derivative shape. The first radius specifies the radius of the base of the cylinder, while the second radius specifies the radius of the top. The values for **center** are true or false. True sets the center of the middle of the cylinder at the origin. False sets the center of the bottom circle at the origin.

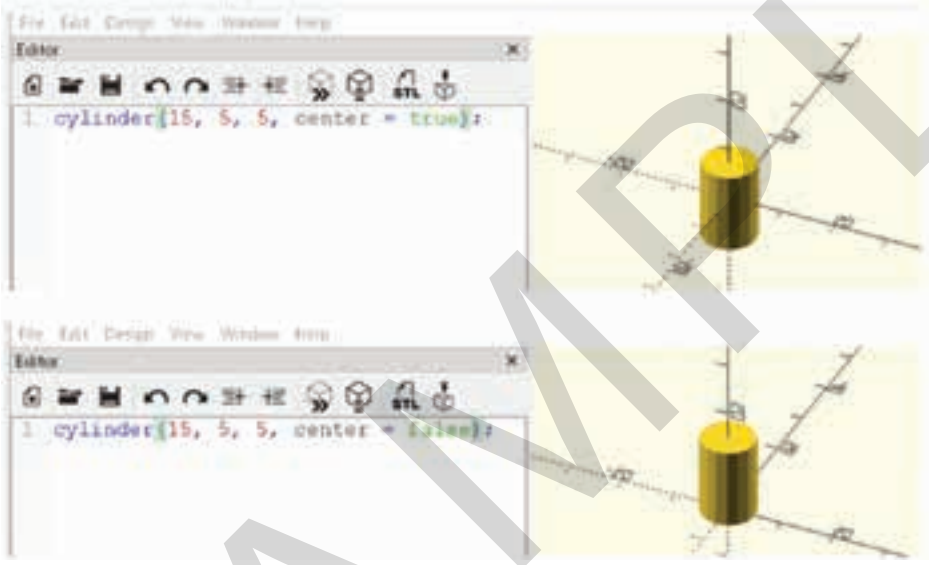
```
cylinder(h,r|d,center)
cylinder(h,r1|d1,r2|d2,center)
```

Figure 13-9 shows an example of two OpenSCAD cylinders and the effect the value of **center** has on their locations in 3D space. The default value of **center** is false.

Changing one of the radii values can produce a cone. **Figure 13-10** shows the object specified by cylinder(15,5,0.1);.

Another transformation applies color to a primitive or a group of primitives. OpenSCAD API for **color** allows use of prenamed colors, or a custom mixing of red, green, and blue hues. In addition, an **alpha** parameter is available to specify how transparent the color is. This is useful for looking inside a 3D model. The API for color is

```
color("colorname", alpha)
```



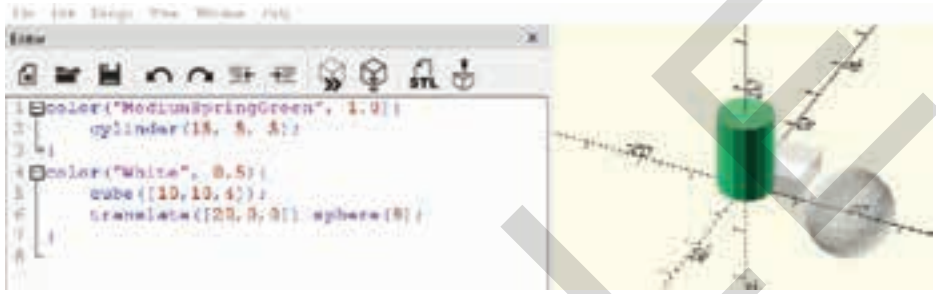
Goodheart-Willcox Publisher

Figure 13-9.
The value of the center parameter specifies where the cylinder is placed.

Goodheart-Willcox Publisher

Figure 13-10.
The value of each radius parameter specifies the radius of one of the bases of the cylinder.

The value of the **colorname** parameter is taken from a list provided by the World Wide Web Consortium shown in the User Manual. Note that the color name is a string and requires quotation marks. The spelling of the **colorname** is in camel case. Some examples of color names are "Aqua", "Yellow", and "MidnightBlue". For a complete list, consult the OpenSCAD User Manual for the **color** transformation. The syntax for using the **color** transformation includes the object or objects to be colored—the child objects. **Figure 13-11** shows **color** transformation in use. Use curly braces to group one or more objects to be colored.



Goodheart-Willcox Publisher

Figure 13-11.
Examples of the **color( )** transformation.

Hands-On Example 13.2B

Coding a 3D Snowman

This activity produces a 3D object of a snowman. Planning for this object first may save time and errors while coding. Thinking about a snowman, you may realize that three different sized white spheres can be used for its body. Two brown cylinders can be combined for a hat. Two small, black spheres can be used for its eyes, and an orange cone for its nose. Applying the **translate( )**, **rotate( )**, and **color( )** transformations, you can write the code to combine these smaller objects into a larger 3D model of a snowman.

1. Launch OpenSCAD and click the **New** button.
2. Create a header of comments to identify this as your work. Use the double slash to start a comment on a line. Add one line of white space.

```
1 // 3D Snowman file
2 // (enter your name and the date)
3
```

3. Write the code to create three spheres of radius 20, 15, and 10 respectively. Add a comment to say what you are going to accomplish.

```
4 // draw three snowballs
5 sphere(20);
6 sphere(15);
7 sphere(10);
```

Continued

Continued

4. Click **Preview**.
5. Notice that it appears that only one sphere was drawn. You can only see the largest one. Use the **translate( )** transformation so that you can see all three spheres. Modify lines 5 through 7 to include the **translate( )** transformation. Move the spheres up the Z-axis to make the snowman. Click **Preview**. Adjust the Z values to your preference.

```
5   translate([0,0,15]) {sphere(20);}
6   translate([0,0,45]) {sphere(15);}
7   translate([0,0,65]) {sphere(10);}
```

Goodheart-Willcox Publisher

6. Apply the **color( )** transformation to make the snowman white. Enclose all the three translate instructions in the **color( )** instruction with curly braces. The transformation will then apply to all three snowballs. Insert the **color( )** transformation before the first **translate( )**. Click **Preview**,

```
5 color("White"){
6   translate([0,0,15]) {sphere(20);}
7   translate([0,0,45]) {sphere(15);}
8   translate([0,0,65]) {sphere(10);}
9   } // end color( ) transformation
```

7. Add a comment to line 10 to indicate you are going to code for the hat as shown.

```
10 // draw hat
```

8. Draw two cylinders: a flat one for the brim and a tall one for the crown. Move them up to the head of the snowman. Click **Preview**.

```
11  translate([0,0,71]){
12      cylinder(h=2,r=13,center=false); // brim
13      cylinder(h=9,r=8,center=false); // crown
14      }
```

Continued

Continued

9. Insert a **color( )** transformation to make the hat brown. Add comments for clarity.

```
color("Brown"){
  translate([0,0,71]){
    cylinder(h=2,r=13,center=false); // brim
    cylinder(h=9,r=8,center=false); // crown
    } // end translate group
} // end color group
```

Goodheart-Willcox Publisher

10. Add a comment for the eyes. Draw two small black spheres and move them to the face. Relocating the eyes to this point is the result of experimentation. Use a Y value and its negative to place the eyes on either side of the face. Adjust these coordinates to your preference. Click **Preview**.

```
// draw eyes
color("Black"){
    translate([8,5,67])sphere(2);
    translate([8,-5,67])sphere(2);
} // end color group
```

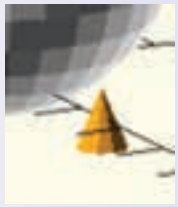
Goodheart-Willcox Publisher

Continued

Continued

11 Apply the cylinder primitive to make a cone for the carrot nose. Place it in front of the snowman.

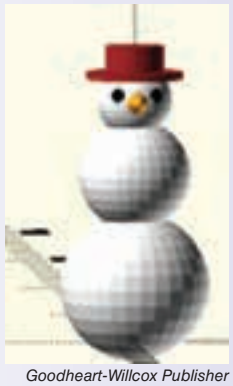
```
22  // draw nose
23  color("Orange"){
24      translate([11,0,65])cylinder(h=7,r1=3,r2=0.1,center=false);
25  }
```



Goodheart-Willcox Publisher

12 Apply the **rotate( )** transformation to the cone along the Y-axis and move it to the face. Note the curly braces are required only if two or more instructions are used with a transformation. Here the cylinder is generated, then **rotate( )** is applied to the cylinder and **translate( )** is applied to the rotation.

```
22  // draw nose
23  color("Orange"){
24      translate([11,0,65])
25          rotate([0,90,0])
26              cylinder(h=7,r1=3,r2=0.1,center=false);
27  } // end color group
```



Goodheart-Willcox Publisher

13 You have completed the snowman 3D model. Save the file as HOE13-2B.scad in your working folder.

Continued

Continued

14. If you have access to a 3D printer, click **Render**, and **Export** the file in the format compatible with your 3D printer. The colors applied to the model in OpenSCAD will not be printed automatically. Advanced knowledge of the 3D printer may permit creating this model in color.
15. Experiment with moving the nose and eyes about the face by changing the values for X, Y, and Z in the **translate( )** transformation on lines 19, 20, and 24. Adjust the parameters for the cylinder to yield a better-looking nose. To get a better view, apply the alpha parameter to the **color( )** transformation in line 5. Choose 0.2 for 20 percent opacity, or 80 percent transparency. Click **Preview** and notice that you can see inside the head. Save your changes to a file named MySnowman.scad.

```
5    color("White",0.2){
```

16. Summarize your findings and prepare to report to the class what you learned.

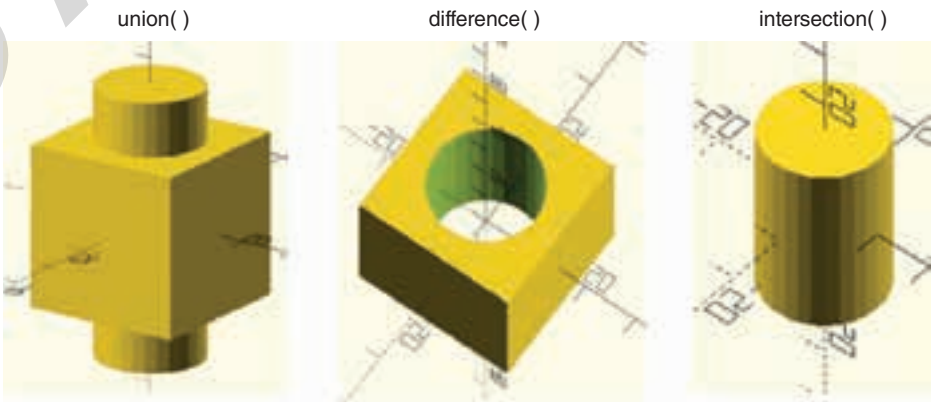
Try It!

There are many additional objects you could add to the snowman model. You might add a smile or buttons made out of coal. Alternatively, you might add arms or rotate the hat to a jaunty angle. Open the file HOE13-2B in OpenSCAD. Apply what you have learned to modify the 3D snowman model. Save your work as TI13-2B.scad.

Combining Primitives

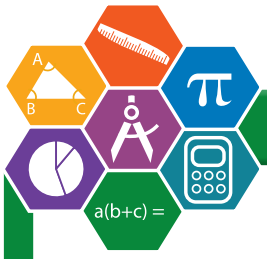
OpenSCAD provides Boolean operations to allow the combination of primitives. The operation **union( )** is used to combine all child primitives into one object. To remove a portion of one primitive, use the **difference( )** operation. The center panel in **Figure 13-12** shows a cylinder subtracted from a cube. The operation intersection yields the common parts of the children. The right panel in **Figure 13-12** shows the intersection of the same cylinder and cube. The API for the Boolean operations is below.

```
union( ) { child nodes }
difference( ) { child nodes }
intersection( ) { child nodes }
```



Goodheart-Willcox Publisher

Figure 13-12.
Results of the Boolean operations.



Math and Coding

Model an Open Container

Open containers are useful for organizing a variety of items. Flowers, pencils, chopsticks, silverware—this list is limited only by your imagination. Use geometry to design a fascinating exterior. A container such as the one shown is generated using hexagons and circles. In OpenSCAD, regular polygons are generated using the circle instruction. Specify the number of sides in the polygon, and the circle instruction draws a regular polygon in 2D. The number of sides in the regular polygon is determined by the fragment number. For example, $fn=5 generates a pentagon. To draw a pentagon in 2D, write the following code.

```
circle(15, $fn=5);
```

To draw a smooth circle, increase the number of fragments. A smooth circle can be generated with a fragment number of 45. In general, the more fragments, the smoother the circle. However, there is a point of diminishing returns where the rendering and printing uses exceptional resources, but the visual result is not better. Save time and energy using a "good enough" number. In this case, 45 does the trick.

To make the opening in the container, draw a circle of a smaller radius and find the difference( ) between the pentagon and the circle. The result is still a 2D image.

Use the **linear_extrude** instruction to create the 3D image. Add the twist parameter to add interest to the exterior of the container.

```
linear_extrude(height=40,center=true,
convexity=10, twist=90)
```

To create the complex shape in the illustration, simply repeat the steps with a twist of –90.

To complete the container, you need to add a bottom. Code a circle of radius 12, extrude it to a height of 2, and translate it to the bottom of the container. Experiment with different regular polygons, different heights, and different angles for the twist. The printed model will make a unique container of your own design.

Preview and render the model. If you have access to a 3D printer, generate the STL file. Save your work as MathAndCode13.scad.

Goodheart-Willcox Publisher

The **linear_extrude** instruction can be used in OpenSCAD to create a container.

Extruding 2D to 3D

The process of generating a 3D object from a 2D shape is called extrusion. The root of the word extrusion is ***extrude***, which means to force out. There are two types of extrusion actions in OpenSCAD: **linear_extrude( )** and **rotate_etrude( )**.

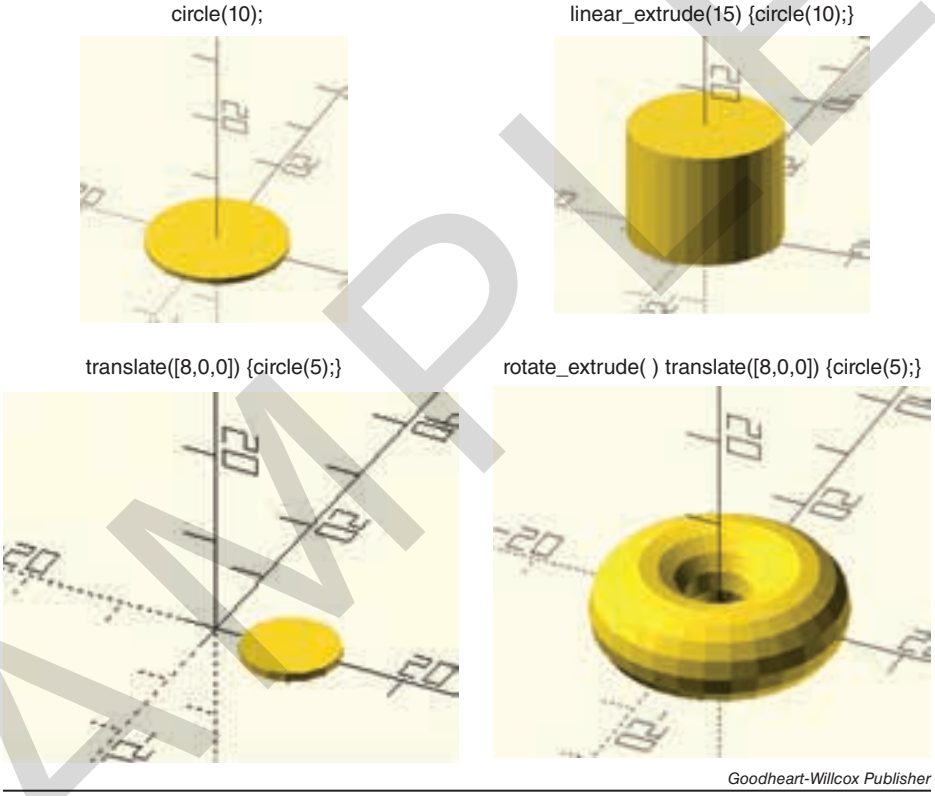
```
linear_extrude( )
rotate_extrude( )
```

The **linear_extrude( )** operation creates a 3D solid from a 2D object in the XY plane. Each linear extrude grows the object along the Z-axis. Think of a cylinder as an extrusion of its circular base.

The **rotate_extrude( )** operation creates a 3D solid from a 2D object by rotating it around the Z-axis. Think of a circle in the XY plane that grows into a donut. **Figure 13.13** shows simple examples of the two types of extrusions.

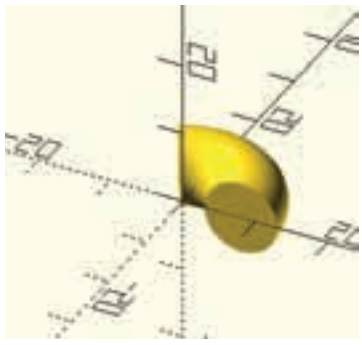
There are several parameters for each type of extrusion. To smooth the 3D result in a **rotate_extrude( )** operation, increase the number of fragments generated. The parameter $fn sets the number of fragments generated in a 360-degree rotation. The angle parameter sets the number of degrees to rotate extrude. **Figure 13-14** shows the result of the instruction.

```
rotate_extrude (angle= 120, $fn=50) translate ([8,0,0])
{circle (5);}
```



Goodheart-Willcox Publisher

Figure 13-13.
Two types of extrusions in OpenSCAD.



Goodheart-Willcox Publisher

Figure 13-14.
Result of the instruction to extrude a circle 8 units from the origin with a rotation of 120 degrees and the number of fragments set to 50.

Exporting 3D Models

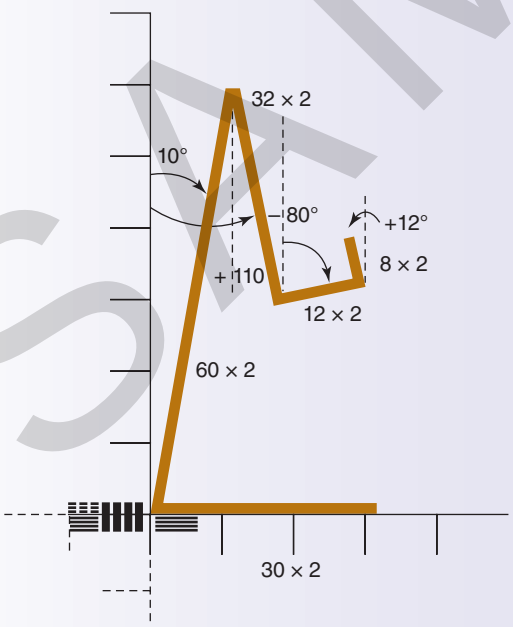
Once a model is ready for use, the file format for exporting the model depends on the next step in the process. First render the model, then select the export format in the **File** menu. OpenSCAD supports nine export formats. A commonly used format is the stereolithography (STL) file type. Because all 3D models are in essence a collection of triangles fused together, you may encounter this acronym also used for Standard Triangle Language. Most 3D printers accept this file format. To export a 2D image of the virtual 3D space, choose the PNG file format. Other formats are used for models that will be used in another modeling program for final use in an animation or a video game, for example.

Hands-On Example 13.2C

Coding a Mobile Phone Caddy

This activity produces a 3D model that can be printed to use as a support for charging or hands-free use of your mobile phone. You will need a base to balance the model, a ledge to hold the phone, and openings to accommodate the charging cord. A sketch will be most useful before we begin. Draw two views of the model, one from the front and one from the side.

The plan is to code a 2D figure and extrude it into a 3D model. Use the right view to guide the creation of the 2D figure of the side of the caddy. Then extrude the 2D model into a 3D shape. Use the **difference( )** operation to make the openings in the shape. Preview, render, and export the 3D model in a format compatible with your printer. Below are drawings used to plan the model for this example.



Goodheart-Willcox Publisher

Side View: Because extrude takes a 2D object in the XY plane and extrudes along the Z-axis, draw the side in the XY plane.

Goodheart-Willcox Publisher

Angle View: Draw cutouts for the charging cords.

Continued

Continued

1. Launch OpenSCAD and click the **New** button.
2. Set a variable for the thickness of the stand in case you want to change it later.

```
1 // Mobile Phone Stand
2 // Your name Today's date
3
4 thick=2; // variable for thickness of 2D primitives
```

3. Apply what you have learned about drawing 2D figures. Use the parameters supplied to draw the side view for this project. Start at line 12, because you have instructions to add above this drawing.

```
12  // draw 2D version
13      // draw slantedback
14      rotate(-10) translate([-0.2,0.5,0]) square([thick,60]);
15
16      // draw base
17      translate([1,0,0]) square([30,thick]);
18
19      // round the corner
20      translate([1,1,0]) scale([0.011,0.011,0.011]) circle(100);
21
22      // draw phone rest back
23      rotate(11) translate([22,24.5,00]) square([thick,32]);
24
25      // draw phone rest shelf
26      translate([16.5,30,0]) rotate(-80) square([thick,12]);
27
28      // draw phone brace
29      translate([28,30,0]) rotate(12) square([thick,8]);
```

Continued

Continued

When you click **Preview**, the 3D space will look like the side view drawing above. Note that a circle was added to smooth the corner at the origin.

4. Add the instruction on line 10 to linear extrude the 2D into 3D. Make the height 40 units and center it. Place a curly brace in front of the 2D drawing (line 10) and after it (line 30). Use comments to remind yourself why you did what you did. The 2D drawing is the child of the extrude instruction.

```
10  linear_extrude(40,center=true){

. . .

30  } // end extrusion from 2D to 3D
```

5. Draw a cube to subtract from the phone rest to allow room for the charging cord.

```
32  // cutout on rest

33  translate([28,32,0])cube([15,15,12],center=true);
```

6. Apply the **difference** operation to subtract the cube from the phone stand.

```
6   // subtract cord openings

7   difference(){
```

7. Remember to place the curly brace at the end of the code to include the new cube in the operation.

```
36

37  } // end difference
```

8. Click **Preview** and **Render** to view the finished phone stand. Save your project as HOE13-2C.scad.

Goodheart-Willcox Publisher

Continued

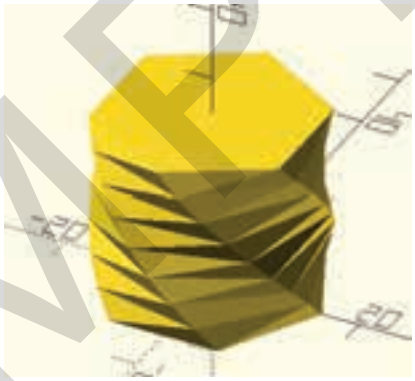
Continued

Try It!

In this example, there is no opening in the back for the cord to pass through. Design an opening of your choice and add it to the difference operation, before the final curly brace. **Difference( )** operates by removing all the child objects from the first object. Save your 3D model. If you have access to a 3D printer, make a physical model of your virtual model. Save your work as TI13-2C.scad.

Next Steps

This chapter is the briefest introduction to the world of 3D modeling. To learn more, explore the other features of OpenSCAD in the User Manual. For example, the **linear_extrude( )** operation takes a twist parameter. **Figure 13-15** shows a circle with six fragments extruded with a twist of 90 degrees. Imagine the fun you could have with this option and others.



Goodheart-Willcox Publisher

Figure 13-15.
Result of the instruction linear_extrude(height=25, twist=90) circle(15,$fn=6);.

LO 13.3

Compose a career plan for employment in the 3D field.

Section 13.3 Exploring Careers in the 3D Field

There are many career opportunities for those who work with 3D models. Some of these careers are involved in the creation of 3D models, and some careers are involved in what happens after the 3D model is created. For all careers, it is important to have an educational plan to achieve your goals.

Graphic artists who work in 3D create animations and visual effects for many forms of media, including:

- television;
- motion pictures;
- web design; and
- video games.

They may also work in advertising in any field where a three-dimensional presentation is vital. For example, architects and interior designers often use 3D models. Renderings are used to help clients see design and material choices as they will appear on the finished product.

Animators take 3D models and add motion. Sometimes the models are manipulated to change poses, as shown in **Figure 13-16**. For example, the arms and legs on a human model may be manipulated to create an animation of running. Other times, the 3D models are not manipulated. Rather, the view is animated. An example of this is an architectural walkthrough. The person viewing the animation sees the room or building as if actually walking through it.

Consumers continue to want more realistic video games, movie and television special effects, and movies in 3D. As a result, the demand for multimedia artists and animators will rise. Further, an increased demand for computer graphics for mobile devices will lead to more job opportunities. Artists working in 3D will be needed to create animation for games and applications for other mobile devices.

The number of jobs for multimedia artists and animators is projected to grow 16 percent from 2020 to 2030. This growth will be due to increased demand for animation and visual effects in video games, movies, and television. Multimedia artists and animators held about 62,400 jobs in 2020. A little more than half of these were self-employed.

There are many opportunities for coders in the industries that also employ multimedia artists and animators. For example, studios that use proprietary 3D software need coders to create, update, and maintain that software. In video game studios, coders are needed to program the functionality of the games. In many cases, coders must write programs to control how 3D models are rendered in the game engine.

Oleg Zhevelev/Shutterstock.com

Figure 13-16.
Animators add motion to 3D models. They also may animate changes in poses.

Other industries involving 3D work include virtual reality and CAD/CAM. Virtual reality (VR) is an artificial environment created by a computer in which the user can become immersed and interact. Coders are needed to create the software that drives VR systems. CAD/CAM stands for computer-aided design/computer-aided manufacturing. These systems are used by engineers, designers, and production workers to create various products. Coders develop these software systems. In some cases, coders program the manufacturing machines to perform the tasks of creating the end product.

Hands-On Example 13.3

Investigating Higher Education

An important part of a successful career is learning the skills needed in the career. A career in multimedia art and animation requires education beyond high school. However, you can start developing a portfolio of your work while you are in middle school and high school.

1. Go to the Bureau of Labor Statistics website (www.bls.gov) or a job-search site. Investigate various careers in multimedia art and animation to find one that sounds like something you would like.
2. Identify the educational requirements for the career you selected.
3. Search for a college, university, or trade school that offers a degree related to the career you selected.
4. List the technology classes required to earn the degree.
5. Do the classes at the school you selected appear to provide the skills needed by the career you chose? Be prepared to discuss your findings with the class.

Try It!

Using the information you collected in the Hands-On Example, develop a career plan. Write a brief description of the career you selected. List several goals to achieve on your career path. What education will you undertake? After your education, what job do you think you will have when you first begin working in your career? Outline the other jobs you think you will have in this career. Save the career plan document as TI13-3 in your working folder.

Cooperative Coding

2-IC-22

Openwork Toy

Kittens love to play with toys. Almost anything can become an item of interest for scooting and chasing. Interest can be increased if the toy makes noise. Even more intriguing to a cat is a small object inside another toy.

With a design and code team, examine the toy in the figure. The toy is an open-lattice ball constructed of a number of co-joined rings. Also, there is a small loose object inside. This will rattle when the toy is moved. Together with your team, design an open-lattice ball with a small irregular object inside. Write the code in OpenSCAD to create the 3D model.

Questions to ask include:

- How is the latticework ball achieved?
- Should you start with 2D and extrude to 3D?
- How does the small object get inside?
- If you want the small object inside to be printed a different color, how will that affect your design?
- Can you design the 3D model so that the small object is generated inside the ball in one print session?
- How do you ensure the toy is safe for the kitten?
- Will the size of the final toy affect your design?

Together, write and test the code. If you have access to a 3D printer, print your model and test it with a pet. Save your design as CoopCode13.scad.

Benjamin Simeneta/Shutterstock.com

OpenSCAD can be used to create a fun pet toy.

Chapter Summary

Section 13.1 Learning the Basics of Three-Dimensional Modeling

LO 13.1 Describe the basics of 3D modeling.

- A 3D model is a virtual object that has volume.
- Rendering is the process of generating a 3D model from instructions in 3D software.
- Rapid prototyping, such as 3D printing, is the process of quickly creating a physical model from a 3D model.

Section 13.2 Introducing OpenSCAD

LO 13.2 Use functions to create shapes in OpenSCAD.

- OpenSCAD is free IDE software used to create models in a virtual 3D environment.
- You can write code to create spheres, cylinders, and cubes and transform them.
- OpenSCAD includes the ability to preview, render, and save 3D models.

Section 13.3 Exploring Careers in the 3D Field

LO 13.3 Compose a career plan for employment in the 3D field.

- Projected growth for careers in the 3D field is due to consumers wanting more realistic video games, movie and television special effects, and movies in 3D, as well as increased demand for graphics on mobile devices.
- A career plan outlines the steps needed to reach career goals.
- Artists and animators working in the 3D field usually need a bachelor degree in computer graphics, art, or a related field.

Review and Assessment

Chapter 13 Test

Multiple Choice

Choose the best response for each question.

1. Which axis is used in 3D modeling, but not used in a 2D layout?
   A. W
   B. X
   C. Y
   D. Z
2. When rendering, what are the colors based on?
   A. Object color
   B. Applied materials and textures
   C. Ambient conditions
   D. Code written by programmer
3. Which of the following is true of a 3D model that will be 3D printed?
   A. It must be animated.
   B. It does not need materials or lighting.
   C. It should be located along the negative Y-axis.
   D. The Z-axis values are discarded.
4. What does an animator working in 3D do?
   A. Adds motion to 3D models
   B. Creates a 3D-printed model
   C. Creates and maintains 3D software
   D. Designs code for a 3D project
5. **Preview** in OpenSCAD provides an approximation of the final object. What provides the replica of the 3D model?
   A. Translation
   B. Extrude
   C. Render
   D. Code

Completion

Complete the following sentences with the correct word(s).

6. The advantage of creating 3D models in OpenSCAD is that the objects are the result of writing _____.
7. If a physical model exists, a 3D model can be created from it using a(n) _____.
8. _____ work in the 3D field by adding motion to 3D models.
9. The only use for a _____ in OpenSCAD is to manage the view of the 3D preview area.
10. OpenSCAD code help is available in the online _____.

Matching

Match each term with its correct definition.

A. extrude
B. model
C. rendering
D. wireframe
E. rapid prototyping

11. Producing a 3D model from the code.
12. Formed by a collection of triangles in a 3D model.
13. Creating a 3D model from 2D primitives.
14. Instructions to produce a 3D object.
15. Making an object in a 3D modeling software rather than out of clay.

Application and Extension of Knowledge

1. Apply your knowledge of OpenSCAD to explore the twist parameter of the **linear_extrude( )** transformation. Design a vase from a circular base with seven fragments. Use the **difference( )** operation to make the vase hollow. Remember to leave a bottom to the vase. Save the project as AEK13-01 in your working folder.
2. Extend your knowledge of OpenSCAD by making a cube with rounded corners. Study the **minkowski( )** transformation in the online User Manual. Save the project as AEK13-02 in your working folder.
3. Extend your knowledge of OpenSCAD by investigating its color property. Open the AEK13-01 project and save it as AEK13-03 in your working folder. Modify the code using the **color( )** transformation. Explore the various options for mixing your own colors, setting opacity, and using the hexadecimal values for color contribution.
4. OpenSCAD supports control statements similar to the repeat blocks in Scratch. Apply what you have learned about control structure to OpenSCAD. Research the syntax for the flow control statements in OpenSCAD to generate six non-touching spheres along the X-axis. Save the project as AEK13-04 in your working folder.
5. Think about a fidget spinner. Apply what you have learned about OpenSCAD to plan a project that would make a 3D model of a fidget spinner. Detail the design of the spinner, list the challenges and what you would still need to learn, then make drawings of what it might look like. Save the project as AEK13-05.doc in your working folder.

College and Career Readiness

Communication Skills

Reading. Go to the glossary of this text. Identify three new words with which you are unfamiliar. Write a sentence using each term.

Writing. Select a computer programming language you have heard of. Conduct a short research project to answer questions about the history of the language. Determine when it was created and why. Use multiple authoritative print and digital sources. Write several paragraphs about your findings to demonstrate your understanding of the process that led to the development of the language.

Speaking. When applying for a job, and often when applying for college, you will go through an interview process. Research the topic of preparing for an interview. Identify several points that describe good practices for speaking in a job interview.

Listening. Perform an Internet search to find information on the average age of students in college. Discuss your findings and reasoning with the class. As you listen to your classmates' opinions on this topic, take notes, and ask questions about positions or terms you do not understand. If you hear any unfamiliar vocabulary or expressions used by your classmates, draw on prior knowledge to analyze the meaning.

College and Career Readiness

Portfolio Development

Hard and Soft Skills. Employers review candidates for various positions, and colleges are always looking for qualified applicants. When listing your qualifications, illustrate both hard and soft skills. For example, you might discuss software programs you know or machines you can operate. These abilities are often called *hard skills*. The abilities to effectively communicate, get along with customers or coworkers, and solve problems are examples of *soft skills*. These are also important skills for many jobs. Make an effort to learn about and develop the hard and soft skills needed for your chosen career field.

1. Conduct research about hard and soft skills and their value in helping people succeed.
2. Create a word-processor document and list the hard skills you possess that are important for a job or career that interests you. Use the heading "Hard Skills" and your name. Next to each skill, write a paragraph that describes the skill and give examples to illustrate it. Save the document.
3. Create a word-processor document and list the soft skills you possess that are important for a job or career that interests you. Include the heading "Soft Skills" and your name. Next to each skill, write a paragraph that describes the skill and give examples to illustrate it.
4. Update the master spreadsheet to reflect the inclusion of these documents.