TECHNOLOGY IN ACTION™

3D Printed Science Projects

Ideas for Your Classroom, Science Fair, or Home

Joan Horvath
Rich Cameron

MatterHackers

EXTRAS ONLINE

Apress®

3D Printed Science Projects

Ideas for Your Classroom, Science Fair, or Home

Joan Horvath
Rich Cameron

Apress®

3D Printed Science Projects: Ideas for Your Classroom, Science Fair, or Home

ISBN-13 (pbk): 978-1-4842-1324-7

ISBN-13 (electronic): 978-1-4842-1323-0

Managing Director: Welmoed Spahr
Lead Editor: Michelle Lowman
Editorial Board: Steve Anglin, Pramila Balan, Louise Corrigan, Jonathan Gennick, Robert Hutchinson, Celestin Suresh John, Michelle Lowman, James Markham, Susan McDermott, Matthew Moodie, Jeffrey Pepper, Douglas Pundick, Ben Renow-Clarke, Gwenan Spearing
Coordinating Editor: Mark Powers
Copy Editor: Corbin Collins
Compositor: SPi Global
Indexer: SPi Global
Artist: SPi Global

Distributed to the book trade worldwide by Springer Science+Business Media New York, 233 Spring Street, 6th Floor, New York, NY 10013. Phone 1-800-SPRINGER, fax (201) 348-4505, e-mail orders-ny@springer-sbm.com, or visit www.springeronline.com. Apress Media, LLC is a California LLC and the sole member (owner) is Springer Science + Business Media Finance Inc (SSBM Finance Inc). SSBM Finance Inc is a **Delaware** corporation.

For information on translations, please e-mail rights@apress.com, or visit www.apress.com.

Apress and friends of ED books may be purchased in bulk for academic, corporate, or promotional use. eBook versions and licenses are also available for most titles. For more information, reference our Special Bulk Sales–eBook Licensing web page at www.apress.com/bulk-sales.

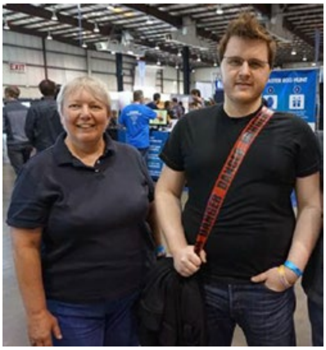
Any source code or other supplementary materials referenced by the author in this text are available to readers at www.apress.com/9781484213247. For detailed information about how to locate your book's source code, go to www.apress.com/source-code/. Readers can also access source code at SpringerLink in the Supplementary Material section for each chapter.

In memory of Zillabell “Jane” Friesen

Contents at a Glance

Contents

About the Authors



Joan Horvath and **Rich Cameron** are the cofounders of Nonscriptum LLC based in Pasadena, California. Nonscriptum consults for educational and scientific users in the areas of 3D printing and maker technologies. Joan and Rich are particularly interested in finding ways to use makertech to make scientific research cheaper and more accessible to the public.

This book is their latest collaboration, following their earlier works *Mastering 3D Printing* (Apress, 2014), *The New Shop Class: Getting Started with 3D Printing, Arduino, and Wearable Tech* (Apress, 2015), and *3D Printing with MatterControl* (Apress, 2015). They also teach online classes in 3D printing and makertech for LERN Network's U Got Class continuing education program. Links for all of the above are on their website, www.nonscriptum.com.

In addition to her work with Rich, Joan also has an appointment as Core Adjunct faculty for National University's College of Letters and Sciences. She has taught at the university level in a variety of institutions, both in Southern California and online. Before she and Rich started Nonscriptum, she held a variety of entrepreneurial positions, including Vice President of Business Development at a Kickstarter-funded 3D-printer company. Joan started her career with 16 years at the NASA/Caltech Jet Propulsion Laboratory, where she worked in programs including the technology transfer office, the Magellan spacecraft to Venus, and the TOPEX/Poseidon oceanography spacecraft. She holds an undergraduate degree from MIT in Aeronautics and Astronautics and a master's degree in Engineering from UCLA.

Rich (known online as "Whosawhatsis") is an experienced open source developer who has been a key member of the RepRap 3D-printer development community for many years. His designs include the original spring/lever extruder mechanism used on many 3D printers, the RepRap Wallace, and the Deezmaker Bukito portable 3D printer. By building and modifying several of the early open source 3D printers to wrestle unprecedented performance out of them, he has become an expert at maximizing the print quality of filament-based printers. When he's not busy making every aspect of his own 3D printers better, from slicing software to firmware and hardware, he likes to share that knowledge and experience online so that he can help make everyone else's printers better too.

Acknowledgments

The consumer 3D-printing ecosystem would not exist in its current form without the open source 3D-printing hardware and software community, and in particular Marius Kintel, the main developer and maintainer of OpenSCAD software, and his collaborators for their software, which was used to develop the objects in this book. We are also grateful for the support of the MatterHackers team and their MatterControl software, particular Lars Brubaker, Kevin Pope, and Mara Hitner for their support during the writing of this book. The maker community as a whole has also been very supportive. The picture of Joan and Rich in the “About the Authors” section was taken at the 2015 San Mateo Makerfaire by Ethan Etnyre; we appreciate all the inspiration we have gotten by looking at projects made by everyone at maker events large and small.

The Apress production team made this process seamless for the most part, and was there with virtual needle and thread for the occasions where it was not. We dealt most directly with Mark Powers, Michelle Lowman, James Markham, Corbin Collins and Welmoed Spahr, but we also appreciate the many we did not see.

We picked a lot of scientists’ brains as we thought about how to model some of the concepts in this book. We particularly thank high school teacher Michael Cheverie for his insights into teaching chemistry. Joan’s longsuffering astronomer husband, Stephen Unwin, was a huge help as we went back into some basic physics or just tried to get past the 3D modeling equivalent of writer’s block. Frank Carsey, Dan Berry, Tim Thibault, and many others helped us out by reading a chapter draft or helping us think through alternative ways that we might model something.

We thank the staff, teachers, and students of the Windward School in Los Angeles for inspiration and discussions of how students learn, particularly Cynthia Beals, James Bologna, Lyn Hoge, Simon Huss, Dorothy Lee, Geraldine Loveless, Kevin Kloeker, Ernie Levoney, Tri Nguyen, Karalyn Raymon, Colin Rose, and Regina Rubio. We also were inspired to create these models in part by discussions with people in the community of teachers of the visually impaired, notably Mike Cheverie, Lore Schindler, Yue-Ting Siu, and the participants in the Benetech workshop organized by Lisa Wadors.

Finally, we are grateful to our families for putting up with our endless brainstorming, kitchen table commandeering, and test runs of explanations. This book has been a long pull for everyone, but we think it will be well worth the wait.

Introduction

When we started our 3D-printing and makertech consulting company, we joked that we were going to call it "Now What?" because that was what schools seemed to say after they bought a 3D printer. We hope that this book takes a step toward answering that question.

We saw that students, parents, and teachers would be excited about using a 3D printer, might download a 3D model, print it, and then wonder what to do next. Or, they might get into creating models from scratch, and get discouraged by the limitations of easier 3D-modeling programs or the learning curves of the more capable ones.

In this book, we try to create a middle path: models that you could just print, but that would be reasonably easy to alter if you wanted to do more. Further, we designed the models so that they would be useful for learning science or math principles while you were changing their features. In particular we wanted to create some seeds of science fair or extra-credit projects—that is, open-ended, meaty explorations that could be explored at a variety of levels.

We were surprised at how hard this turned out to be. Most textbooks and online sites endlessly recycle versions of the same 2D projection of models of science concepts. In each chapter, we have a "Learning Like a Maker" section where we talk about our adventures in defining and implementing the models—which in some cases involved finding online copies of 1935 manuscripts (signed off by a Wright brother and Charles Lindbergh!) and in others meant figuring out what to do after a scientific experts who implied that everyone teaches the subject at hand the way it is drawn in textbooks, but you unlearn all that in grad school to get to the *real* way.

This book presumes that you know a little bit about 3D printing already. If not, Appendix A and the resources linked there should get you up to speed. The models are all written using the OpenSCAD free and open source 3D-modeling program. If you know how to program in a language like C, Java, or Python, that will help, but it's not strictly necessary to alter the models. Appendix A and the OpenSCAD materials linked there will help you out with that too.

We have found that teachers use 3D printers in one of two fundamental ways. Either they want to create a model to pass around in class to help students visualize a concept, or they want students to use a printer either to learn engineering and design per se or to cement physical concepts like levers and gears. Since most of these models would lend themselves to being used either way, we have not included a grade level or explicit lesson plans.

To show our readers who are teachers (in the United States) what we had in mind, though, at the end of most chapters we suggest Next Generation Science Standards that we thought might benefit from these models. These science standards, from the group NGSS Lead States, are documented in *Next Generation Science Standards: For States, By States* (The National Academies Press, 2013). Links are given at the end of relevant chapters. If you are a teacher, you may want to check with your state or school standards as well to see the best fit.

The models span a variety of topics, and we tried to cover as many disciplines as possible. Briefly, here is what you can look forward to:

> Chapter 1 gives you a few options to print many different types of mathematical surface. This ability underlies some of the other models.
>
> Chapter 2 creates models of waves to allow you to explore what happens when various waves overlap and interact. You can print yourself a model of Young's famous double-slit experiment to see how light from two slits can interfere.
>
> Chapter 3 takes us back to Newton and Kepler to learn about planets and stars and how they speed up and slow down in their orbits.
>
> Chapter 4 allows you to create wings with classic airfoil shapes from the early days of flight. You will be able to make yourself a very simplistic test stand that you can use to measure the lift from the wing with just a fan and a postal scale.
>
> Chapter 5 lets you create basic models of all the "simple machines"—wedge, inclined plane, lever, pulleys, and screws.
>
> Chapter 6 allows you to model plants and their ecosystems, and to design plants for different environments. Maybe you can create a garden for another planet (or for Earth after another few hundred years of climate change).
>
> Chapter 7 lets you begin to explore carbon atoms, and how water molecules come together to make two different types of ice crystals.
>
> Chapter 8 explores 2D and 3D trusses and how you can use them in various explorations.
>
> As we noted earlier, Appendix A reviews how to 3D print, and Appendix B aggregates all the links in the book.

Finally, we are making the 3D-printable models used in this book (although not the book itself!) open source, licensed under a Creative Commons Attribution-NonCommerical-ShareAlike 4.0 International License (`https://creativecommons.org/licenses/by-nc-sa/4.0/`). That means you can use them for noncommercial purposes (in a classroom, for instance) and alter and remix them as long as you credit us. Appendix A has some notes about where to find the repositories if you would like to add to these models. We hope these models are the beginning of a set of models that students everywhere can play with and learn from.

CHAPTER 1

3D Math Functions

Since so much scientific visualization starts with looking at underlying mathematics, we are beginning this book on 3D printing for science projects with a chapter on 3D printing mathematical functions. The basic models in this chapter are intended to be a starter set that you alter to 3D print whatever function you like, within the boundaries we will get to in a later section.

Math Modeling for 3D Printing

It seems like it should be easy to just put an equation into a program and have the printer "draw" it somehow, like some sort of 3D pen plotter. However, if a 3D printer head just tried to follow an equation, it would have no way of knowing how to avoid material that had already laid down, so we have to go about it in a bit more roundabout way.

3D Printing

3D printers require a several-step process from the first idea to a finished print. First, you need to develop a 3D model, as we will in this chapter. Models in this chapter (and most of the remainder of the book) are based on the free, open source 3D solid modeling program OpenSCAD (`www.openscad.org`). OpenSCAD allows you to encode geometrical models in a language that is sort of a subset of the C programming language. Good documentation is available by clicking the Documentation button on the OpenSCAD site's home page.

Then, other software takes this model and slices it into layers, which the printer will then create one at a time, typically from the bottom up. We will use the open source MatterControl host program throughout this book, available free from `www.mattercontrol.com`. Appendix A talks more about things you should know about OpenSCAD, MatterControl, and 3D printing in general.

Electronic supplementary material The online version of this chapter (doi:10.1007/978-1-4842-1323-0_1) contains supplementary material, which is available to authorized users.

■ **Tip** This book presumes you are generally familiar with 3D printing practices. If not, you can learn how to use a printer from Joan's previous book, *Mastering 3D Printing* (Apress, 2014) or our book *3D Printing With MatterControl* (Apress, 2015). Unless we specifically note otherwise, prints in this book were created on a Deezmaker Bukito in polylactic acid (PLA) plastic, using the MatterSlice engine in MatterControl (although we could have used any software compatible with an open source printer).

Math Background

This chapter presumes you know what a *function* is—a relationship among a number of variables. In this case, we are dealing with functions using three variables, which we will call x, y, and z. Function notation looks like this: $z = f(x,y)$. All that means is that our variable, z, can be computed for any given pair of values for the x and y variables. Having three variables means we can define shapes in three dimensions, with one variable corresponding to each dimension. Normally these three-dimensional shapes would be shown on a page with two-dimensional projections. Often, this is fine and you can see what is going on. Sometimes, however, it really helps to hold a 3D model in your hand and turn it this way and that. This chapter will give you the ability to do that for many types of functions.

■ **Note** 3D printing convention holds that x and y are in the plane of the platform that your model is being built up on, and z is vertical height above that. In other words, the bottom of the surfaces generated in this chapter is always the $z = 0$ plane. In this convention, you always have to *transform* what you are printing to have z greater than or equal to zero, since you cannot build under the platform. In other words, if you know that z would be negative for some values of x and y that you want to use, you may have to add an offset to your equation so that z is always greater than zero and remember that the offset is there when you think about what your model represents.

We will get you started with a model entirely in OpenSCAD that creates surfaces of functions $z = f(x,y)$, where x and y are the plane of the 3D printer's build platform, and z is the height of the surface above that plane. First we will show you how the basic 3D math model Rich has written and included here works, and what kind of functions you can print. Then we will show you a simple model that creates surfaces that might be a starting point for your own projects in OpenSCAD.

Alternatively you may have code you developed that produces a surface you would like to 3D print. It may not be practical to port that code to an OpenSCAD model. We will also show you an example in which we wrote a separate Python script that produces a file, which is then read into OpenSCAD and made into a surface. Finally we will give you some ideas about how you might use these tools as a teacher or as part of a student project.

Others (see links at "Where To Learn More" later in this chapter) have used more sophisticated mathematics modeling programs, but our desire here is to make this completely accessible and free so that you can get started without investing in software, at least at the beginning of your explorations.

Creating Surfaces Entirely in OpenSCAD

In this section we show you how to create a polynomial with a flat base. In the next section we show you what a print of a double-sided surface looks like using the same OpenSCAD code with different parameters. This model was written to be simple and easy to alter, which means that it does not check for complicated problems, like functions that go to infinity or other mathematically bad behavior. It does, however, let you input a function $f(x,y)$ (as you can see in Listing 1-1). It uses OpenSCAD's polyhedron module to accomplish this.

Making a Smooth Surface with a Flat Bottom

Listing 1-1 is the OpenSCAD model that creates a flat-bottomed "slice" of a surface, like a chunk cut out of a mountain range. The function in this example is $z = f(x, y) = 0.01 (x - 50) (y - 50) + 30$, and the 3D print will go from $x = 0$ to 99 and $y = 0$ to 99. This creates a "saddle point" structure, as shown in Figure 1-1. The model will create a surface *xmax* mm in *x* and *ymax* mm in *y*, with *z* height computed in mm. If the resulting structure is too big (by default, 99 mm by 99 mm on the bottom, or a bit less than 4 inches square), then you can scale the whole piece in your 3D-printing software. Figure 1-1 was scaled down to one-quarter scale.

Figure 1-1. *Saddle function with a flat bottom. Printed at quarter scale, so each side is about 25 mm, or just under an inch long. Layer height was 0.2 mm*

The values of *x* and *y* go from 0 to *xmax* and *ymax* respectively. A maximum of 100 points in each dimension (`xmax = 99` and `ymax = 99`) is allowed. The model will step in units of 1, which cannot be changed. If you want your model to step in smaller or larger increments, you will need to scale the variable in the equation.

For example, if in your original equation you had a function you wanted to increment by 0.02 in each step, replace the *x* in your original equation by (0.02 * *x*) to accomplish the same thing when you increment by 1. Or if you want to step from –500 to –400 in the original function, replace the *x* in your original function with (*x* – 500) everywhere to accomplish the same thing. Be careful if variables are raised to a power, or are inside a function like cosine, that you do this scaling correctly and consistently.

■ **Caution** Because we are creating a flat bottom, the equation being represented here is actually 0 <= *z* <= *f*(*x, y*). As a result, *z* = *f*(*x, y*) must be greater than zero everywhere for the flat-bottomed (`thick = 0`) version. A model will still be produced if there are *z* values that are less than zero, but it will be an invalid model, and even if you manage to repair it, it won't print easily.

■ **Note** The OpenSCAD models in this book are written by Rich Cameron and licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International License—see http://creativecommons.org/licenses/by-nc-sa/4.0/ for details.

Listing 1-1. The Basic OpenSCAD Model to Create a 3D Print of a Surface

```
// OpenSCAD model to print out an arbitrary surface defined as z = f(x,y)
// Either prints the surface as two sided and variable thick = thickness
// Or if thick = 0, prints a top surface with a flat bottom
// File surfaceprint.scad

function f(x, y) = ((x - 50) * (y - 50)) / 100 + 30; // Saddle point

//z height, in mm

thick = 0; //set to 0 for flat bottom. else mm thickness of surface
xmax = 99; //Number of points in x direction - 99 is the max
ymax = 99; // Number of points in y direction - 99 is the max

// If you want a rough surface (to make it more tactile) set blocky=true.
// Otherwise surface will be smoothed
blocky = false; //if true, xmax and ymax must be less than 100.

//number of points that will be plotted
toppoints = (xmax + 1) * (ymax + 1);
```

```
//next section generates the points in the array
points = concat(
    [for(y = [0:ymax], x = [0:xmax]) [x, y, f(x, y)]], // top face
    (thick ? //bottom face
        [for(y = [0:ymax], x = [0:xmax]) [x, y, f(x, y) - thick]] :
        [for(y = [0:ymax], x = [0:xmax]) [x, y, 0]]
    )
);

zbounds = [min([for(i = points) i[2]]), max([for(i = points) i[2]])];

//create triangles from quad
function quad(a, b, c, d, r = false) = r ? [[a, b, c], [c, d, a]] : [[c, b, a],
[a, d, c]];

faces = concat(
    //build top and bottom
        [for(bottom = [0, toppoints], i = [for(x = [0:xmax - 1], y = [0:ymax - 1])
            quad(
                x + (xmax + 1) * (y + 1) + bottom,
                x + (xmax + 1) * y + bottom,
                x + 1 + (xmax + 1) * y + bottom,
                x + 1 + (xmax + 1) * (y + 1) + bottom,
                bottom
            )], v = i) v],
        [for(i = [for(x = [0, xmax], y = [0:ymax - 1]) //build left and right
            quad(
                x + (xmax + 1) * y + toppoints,
                x + (xmax + 1) * y,
                x + (xmax + 1) * (y + 1),
                x + (xmax + 1) * (y + 1) + toppoints,
                x
            )], v = i) v],
        [for(i = [for(x = [0:xmax - 1], y = [0, ymax]) //build front and back
            quad(
                x + (xmax + 1) * y + toppoints,
                x + 1 + (xmax + 1) * y+ toppoints,
                x + 1 + (xmax + 1) * y,
                x + (xmax + 1) * y,
                y
            )], v = i) v]
);

//Now either generate the surface as discrete cuboids
// or smoothly with the polyhedron function
```

```
if(blocky) for(i = [0:toppoints - 1]) translate(points[toppoints + i])
cube([1.001, 1.001, points[i][2] - points[toppoints + i][2]]);
else polyhedron(points, faces);

echo(zbounds);
```

Printing Considerations

Since this model was designed to have a flat bottom, it prints easily. There will not be any overhangs. If your function would get very tall, you may want to scale it down by multiplying the equation by a constant. To check that the function is not getting too big or going negative in the *z* direction, you can either just run it in OpenSCAD and look at the result, or graph it conventionally yourself to see what it looks like. (The model does not check it for you.)

Limitations and Alternatives

To keep the model simple, transparent, and easy to understand, we have not inserted any error checking or special cases. Obviously, if you have a function that goes to infinity or has some sort of discontinuity, you will need to come up with some fix. You could, for example, create a branch in the definition of $f(x, y)$ to handle cases that are poorly behaved in the mathematical sense.

We could only test a few cases, and there are an infinite number of options, so as with any print you should check the software model of your print before committing it to be sure that it worked for your particular example. As discussed earlier, *z* has to be greater than zero.

Another parameter in the code is `blocky`, which in this example is set to `false`. Setting `blocky = true` will create a slightly rough-textured surface instead of the smooth one here, which generally will take a lot longer to render in OpenSCAD—we have pulled out this code in a later section as a freestanding tiny model, so you might want to try using it that way.

■ **Caution** OpenSCAD's math functions will probably look familiar if you are a programmer, but some of them will not be what you are expecting if you do not have that experience already. Exponents, for example, take the form of `pow(base, exponent)` rather than using a superscript, and a square root uses the `sqrt()` function instead of a radical sign. You can find a (nearly) complete listing of the mathematical functions available in OpenSCAD at `https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/Mathematical_Functions`.

Making a Two-Sided Smoothed Surface

Although it is convenient to be able to visualize the top of a surface in 3D, sometimes it is even better to see it from both sides. However, you might think that then you will have to put a lot of support under the surface and wind up more or less with the same thing. The way out of this is to print the surface *sideways*. As long as the surface does not contain