Mathematics Through 3D Printing

a GMU capstone class

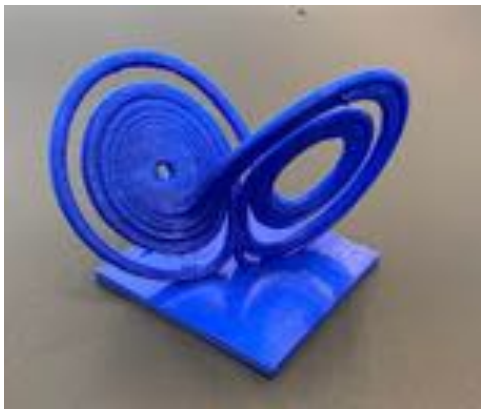
Evelyn Sander

Construct 3D, February 2020, Rice University, Houston

Mathematics through 3D printing Fall 2019

What:

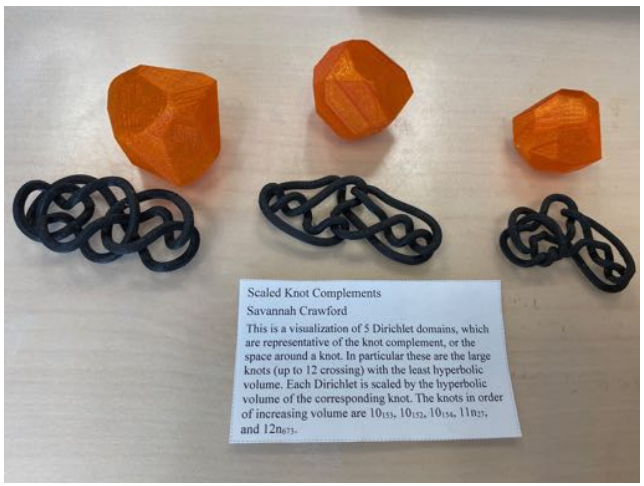
- A capstone class to synthesize mathematical knowledge
- Prerequisites: A proof course and a 300 level math course
- No textbook: readings of papers, websites



Mathematics through 3D printing Fall 2019

How:

- Weekly 3D printed object each on a different topic
- Weekly presentations: oral, written, blog, and Thingiverse
- Public display in department's display case

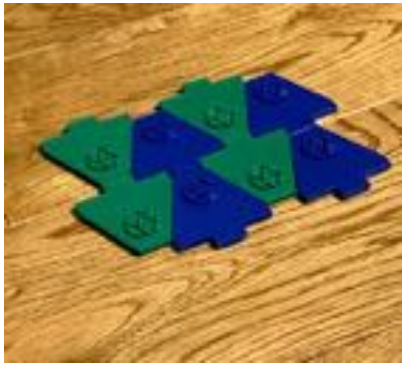


Mathematics through 3D printing Fall 2019

Why:

- Teach topics that slip through the cracks that "every major should have seen." (not unique to math!)
- Breadth over depth
- Creativity: No identical creations
- Patience required

3D printing specifics



- Too many pieces of software is overwhelming: Stuck to OpenSCAD and Mathematica
- Taking full ownership: Students are required to do their own printing. Thus they learned about slicers, manifold and watertight objects, supports, etc.
- Learning assistants helped with setup and printing.
- Lecture each week on the math and the coding.
- Provided with an initial step by step tutorial on each software
- Provided with sample code

Two types of tilings the plane

Irregular convex pentagons

That there are exactly 15 distinct classes is a new result, still under peer review.

Group symmetries

There are 17 distinct wallpaper group symmetries which result in tilings of the plane

Mathematical Optical Illusions: Sugihara Cylinders

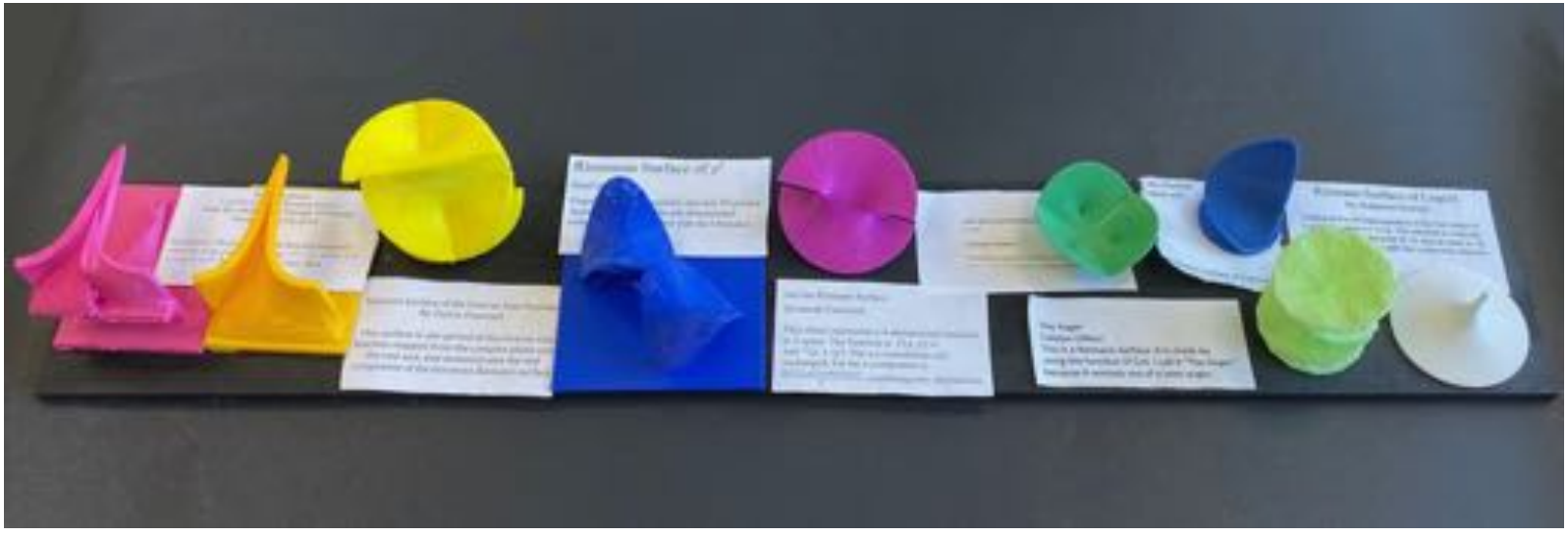
A mathematical optical illusion
in the department display case
Take a look at each object and its reflection.
They're not the same!
Right side up heart

Multivariable calculus objects: Saddles and surfaces

Learning multivariable calculus involves a lot of algebraic manipulations, but it all becomes much easier when getting to see what the concepts look like in 3D.

Student found a starfish to go atop the starfish saddle

Graphs of complex valued functions



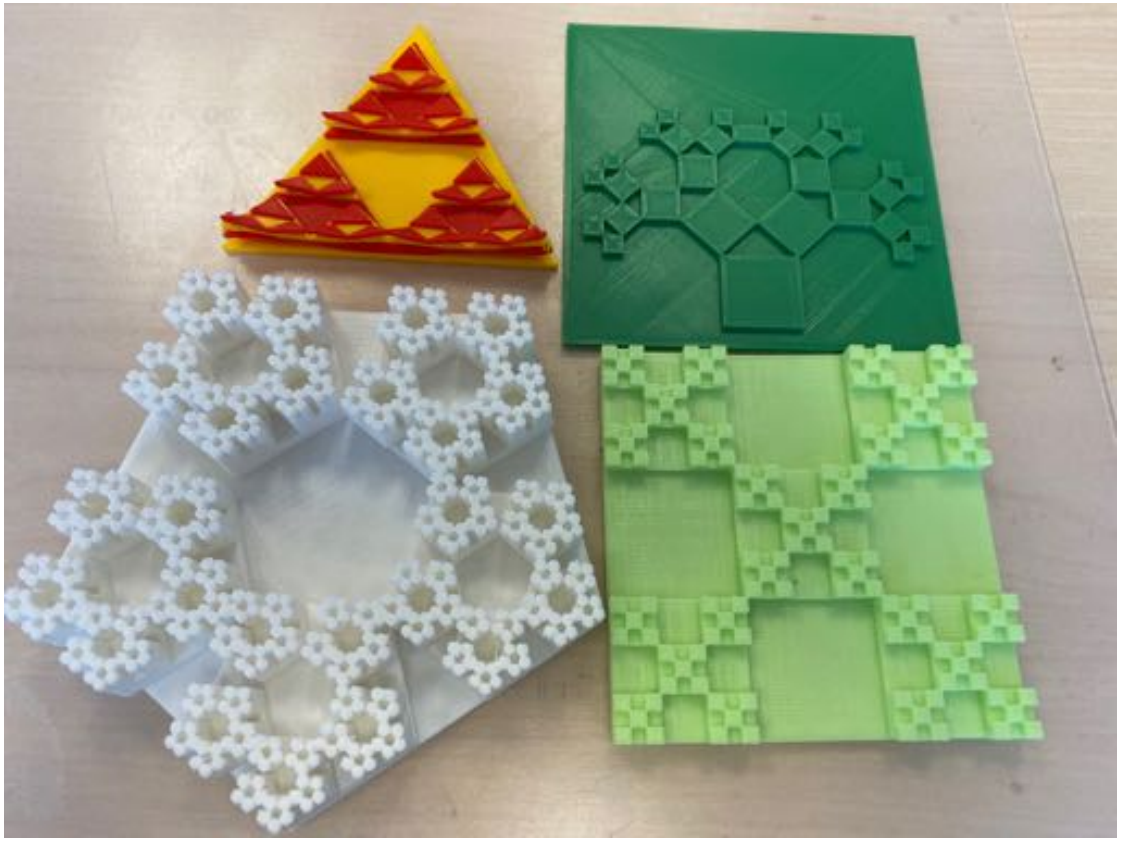
The complex plane is two-dimensional, meaning that the graph of complex functions are four dimensional. We cannot see in four dimensions, so we project to three dimensions. This gives a sense for what the these graphs look like in 4D.

Solving differential equations: Strange chaotic attractors

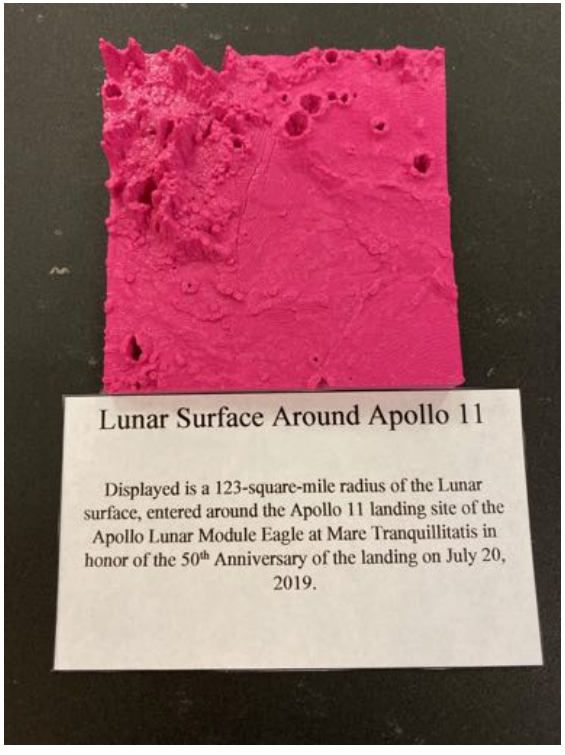
Solutions to 3-dimensional systems of differential equations can be quite simple, such as a point that never moves, or a periodic motion that repeats forever. It can also be quite complicated:
Strange *the object has interesting fractal shape*
Chaotic *nearby initial conditions move away with growing time*
Attractors *any nearby initial condition has a solution limiting to the shape you see.*

Fractals: Iterated function systems

This type of fractal arising from a multivalued transformation.
Easily achieved using any CAD system with recursion.



Visualization of data: Individuality!



Successful methods

- Lecture balanced between mathematical background and coding instructions
- Time to work in class
- Clear detailed instructions and rubrics
- Detailed sample codes

A blog post: wallpaper groups

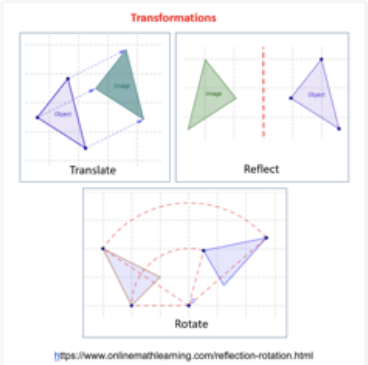
Wednesday, September 25, 2019

Wallpaper Patterns: P1

For the week of september 16th 2019, the class of Math 401: Mathematics Through 3D Printing, was to demonstrate different wallpaper groups. Wallpaper patterns have been dubbed that for the fact that these patterns look like wallpaper and how they tile the plane as well as wallpaper covering the wall. We see these patterns occur frequently everyday whether it be art, floor tiles, and other architecture.

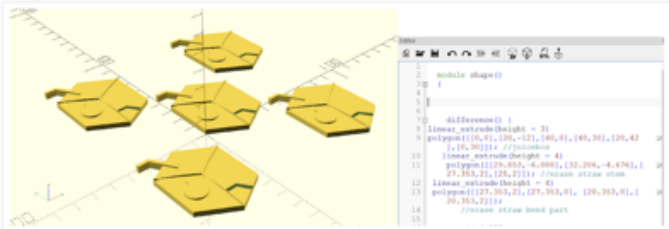
(https://www.youtube.com/watch?v=5UbMFiK3LY0)

In 1891, Russian mathematician, Evgraf Fedorov proved that only 17 distinct wallpaper patterns exist. In these patterns there are 3 different transformations that can occur. A shift or slide may occur when the tile/pattern is shifted producing the same tile as started with, this is a translation. A turn or rotation can occur where a tile is rotated around a certain axis leaving the new tile to be rotated a certain degree from the original. The last transformation that can happen is a flip or reflection where a tile is reflected in a direction and the resulting tile is a mirror of the original. There is a special case where a tile may have a glide reflection where a tile is shifted then reflected across an axis. Examples are shown below.

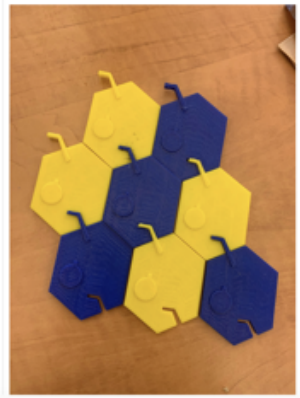


https://www.onlinemathlearning.com/reflection-rotation.html

(https://en.wikipedia.org/wiki/Wallpaper_group#/media/File:Wallpaper_group-p1-3.jpg)



I created a "juicebox" shape and I realized it was a simple hexagon that could be reflected so to combat that I added a "straw" to the design to not get it confused. Adding the straw was the hardest part as I had to find measurements. It took me a bit to create the exact measurements to later find out that when printing there is a bit of an error in printing which I explained in the last paragraph. As shown in my print, the "juice box edges" aren't printed but showed up in the stl file. As someone new to 3D printing I am not sure what happened there.



If I were to do this project again, there were a few mistakes I had learned.First, I needed to add a bit of a buffer room for the "straw" piece, as they were not fitting together at first. As I printed on Friday, I didn't have much time to reprint so I did take an exacto knife to it and shave off a bit of the print until I could connect the pieces together. It ended up working, but I know for next time to leave the space. I would also see what

A blog post: Mandelbrot sets

Wednesday, December 4, 2019

An Inspiration of The Mandelbrot Set

Dr. Sander

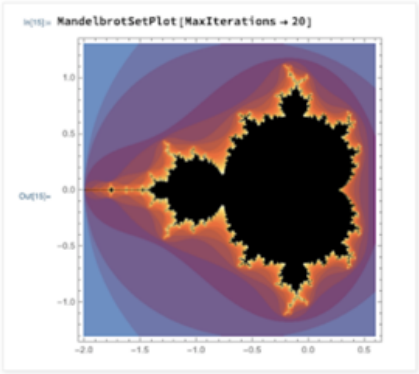
December 6, 2019

An Inspiration of The Mandelbrot Set

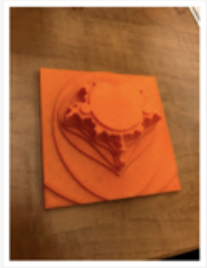
What does the Mandelbrot Set represent?

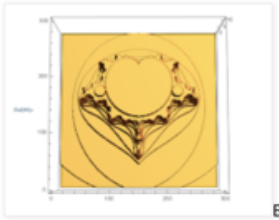
In the purest sense, the Mandelbrot set is simply a set of numbers that display certain unusual property, but since the set is always associated with an image that represents this property, we often use the term “Mandlebrot Set” to describe the image itself. The Mandlebrot set is a visual representation of an iterated function on the complex plane.

The Mandelbrot set is a set of all numbers that DO NOT grow exponentially in the iterated function and are typically represented by the color black. If we have a black circle, an image of the complex plane that shows all the possible numbers in the Mandelbrot set displayed in black, and iterate the function , each time in a different color, infinitely many times, we would end up with this shape:

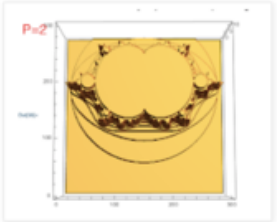


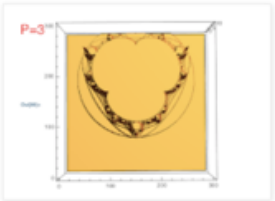
For my specific print, as shown in the code, I manipulated my function for Manderlot set a bit to show how it would still be similar to the initial function. I raised my function to the power of 0.9.



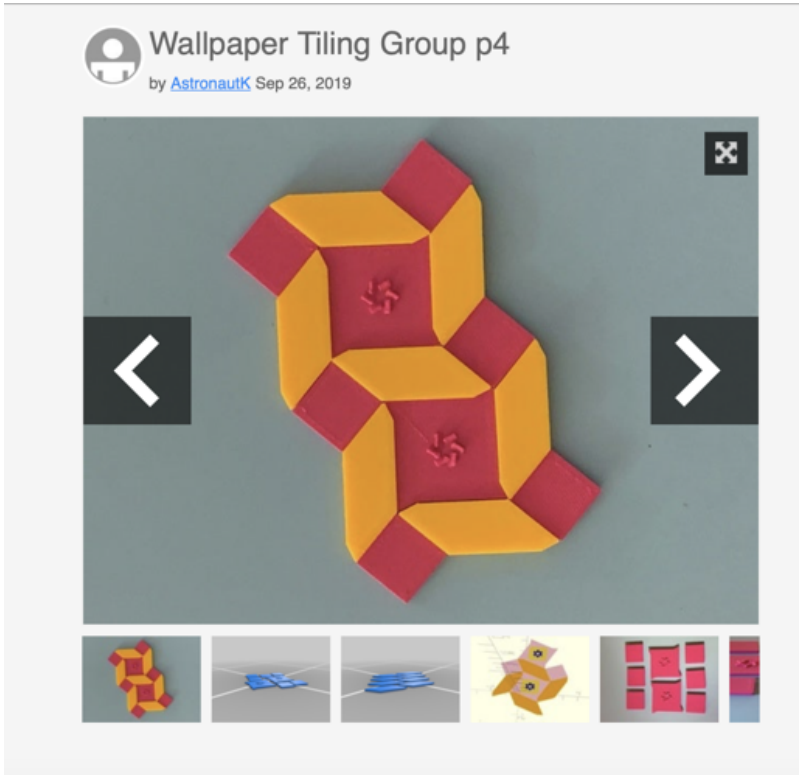


But I also raised it to 2,3,4, and 5 and got very cool illustrative results:





A Thingiverse entry: wallpaper groups



Summary

My name is . I'm a student at George Mason University taking Math 401: Mathematics Through 3D Printing. This object is my print from a project in the class where we were given a wallpaper group and had to construct a tile (or tiles) to tile a plane. Two of my tiles, there are three in total, are combinations of multiple geometric shapes. Attached is a computer rendered image of the original pattern I created which is what my wallpaper tiles are based off of. Uploaded are images of the my fully-constructed wallpaper tiles, as well as the two sets that the tiles were printed in.

How I Designed This

The rendering I made to idealize and create my wallpaper pattern is an online site linked below called Wallpaper Symmetry (I would highly recommend, it's really good and helps one to visualize the differences in wallpaper patterns). The tiles were designed based off of the wallpaper pattern I designed using Wallpaper Symmetry. In order to save some time, I measured the lengths of the sides of all the triangle, squares, and the trapezoid, then coded those lengths into the respective functions in OpenSCAD.

OpenSCAD. Is the program I used to construct, check that the plane was completely tiled with no gaps, and render the tiles to be printed. Pictures of the tiles in the program are uploaded as well as the OpenSCAD code that I wrote. In the code I made three main modules, one for each tile. The center tile is, as mentioned earlier, the center square and the four triangles encasing it plus the decoration on top. That was designed in yet another module that was later added to the module that contained the code for the shapes. The design is squares rotated 60 degrees from 0 to 360 degrees and tilted at a 30-degree angle in the x-axis to achieve that orientation. The trapezoid mentioned earlier was combined with two different shaped triangles than in the center tile to create the tile in both the original rendering and in the final printed object.

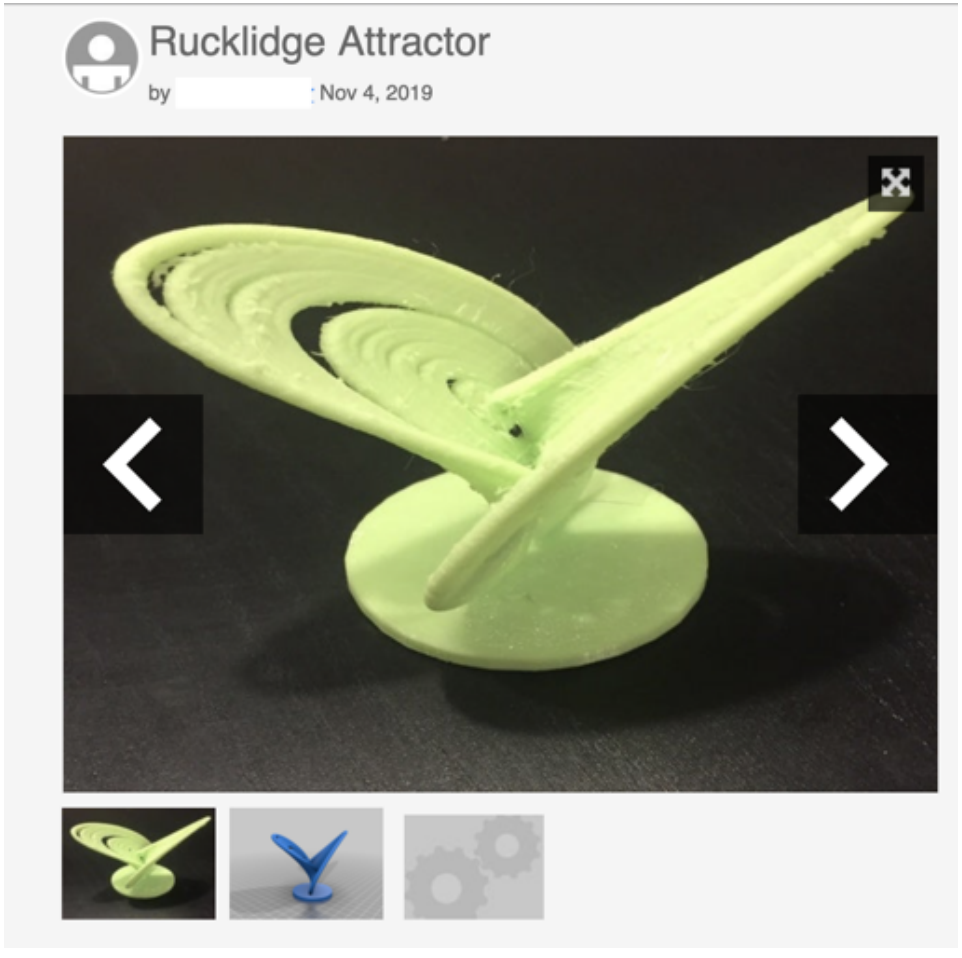
I successfully printed all the tiles in just under four hours on a Ultimaker 3D printer. A raft was used to ensure the safety of the print and no warping as it cooled as it was to be 3mm thick, but no other supports were necessary. As I measured all of the tiles from the original rendering, in the code they are very tiny in size. I had some issues with the scale function in OpenSCAD so when I printed the two files (one for each of the two colors) I had to scale the object to 1300% in the x- and y- directions and I set the z-scale to be 3mm. I printed both sets of tiles with these parameters. Also, the infill density was set to 7, and the printing speed for the objects and the raft was 70 mm/s.

Overview and Background

A pattern of multiple shapes that completely tiles a plane with no gaps, known as a wallpaper pattern. The printed tiles are of a p4 pattern which has no reflections, including glide reflections, and has a square lattice. There is a schematic that I found helpful in illustrating this concept which I have attached. The group p4 has two rotation centers of order four (90°), and one rotation center of order two (180°), and because of this its point group is C4. A p4 pattern can be looked upon as a repetition in rows and columns of equal square tiles with 4-fold rotational symmetry. Also, it can be looked upon as a checkerboard pattern of two such tiles, a factor $\sqrt{2}$ smaller and rotated 45°.

http://math.hws.edu/eck/js/symmetry/wallpaper.html
https://en.wikipedia.org/wiki/Wallpaper_group

A Thingiverse entry: Chaotic attractor



Summary

MATH 401 - Math with 3D Printing
George Mason University

A strange attractor is an attractor of a dynamical system (ie. the set of values that a dynamical system tends towards) that takes on a fractal structure, which is usually indicative of the system being chaotic. A chaotic system is a dynamical system that exhibits sensitive dependance on initial conditions, and it's attractors can be subclassed as chaotic attractors: sets of values that towards which the system exhibits global stability, despite it's arbitrary pieces exhibiting instability. One such strange attractor is the Rucklidge model, which is the dynamical system described by the set of equations:

dx/dt = -kx+ly-yz
dy/dt = x
dz/dt = -z+y^2

This system exhibits chaotic behavior where k=2 and l=6.7, with initial conditions (x0,y0,z0) = (1, 0, 4.5).

To construct a 3D object of this system, I input the system into Mathematica (see attached file), evaluated the system with the initial conditions up to an a certain time timelenghta, re-evaluated with the initial conditions being the value of the system at t = timelengtha, and then plotted the interpolated functions in 3-space. After scaling, thickening, and creating the base, the file was then exported as the STL attached.

This object was printed on the Makerbot Replicator 5th Generation. The first attempt at printing resulted in a size smaller than what was required, so the model was redesigned so that minimal rescaling would be needed in the slicer. The print required supports and rafts, and printed with no issues in 3hr and 35min.

https://search-proquest-com.mutex.gmu.edu/docview/2259455417/B833C5AAF7724110PQ/
https://en.wikipedia.org/wiki/Attractor#Strange_attractor

Print Settings

Printer Brand:

MakerBot

Printer:

MakerBot Replicator (5th Generation)

Rafts:

Yes

Supports:

Yes

Infill:

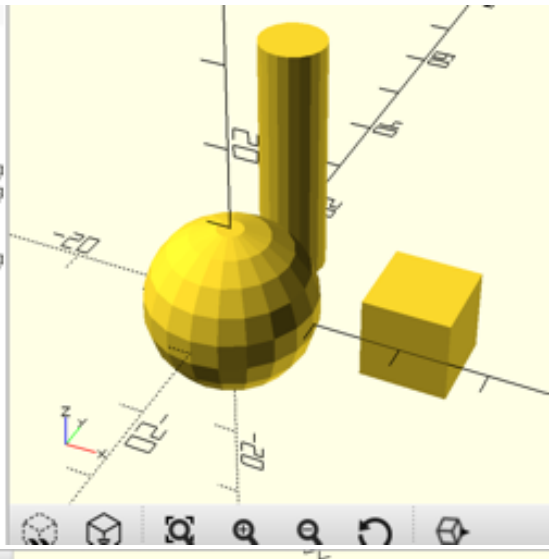
10

Successful methods

- Lecture balanced between mathematical background and coding instructions
- Time to work in class
- Clear detailed instructions and rubrics
- Detailed sample codes with many comments!!

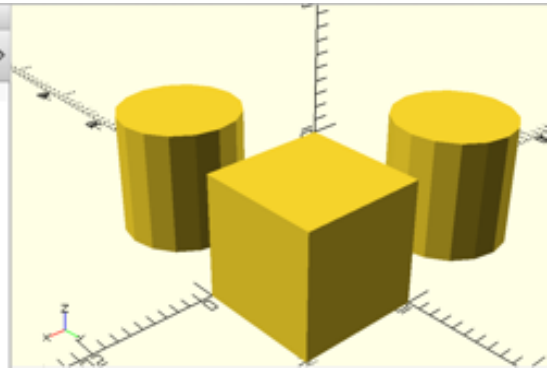
OpenSCAD sample code: Short simple OpenSCAD tutorials

```
tructions:
//1. a. Rotate (one finger)
//   b. Zoom (two fingers)
//   c. Preview (>>)
//   d. Render (cube button)
//   e. Save (File menu)
//   f. Export as STL (File menu)
//2. Change $fn to different numbers and preview.
    This controls the resolution of your shapes -
    very important for a high quality print.
//3. Modify this code to make a sphere of radius 5.
//4. Move the cube close to the sphere so that they
    overlap.
//5. Make the cube have a height of 2.
//6. Change one radius of the cylinder to be 0.
//7. Move the cylinder to the middle of the sphere.

$fn = 20;


sphere(r=10);

translate([15.5,0,-5])
cube([10,10,10]);
```



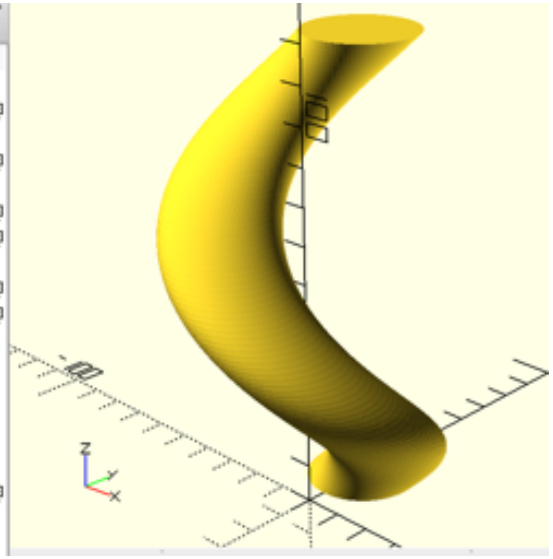
Editor

```
1  // Heart Puzzle Instructions
2  // Use these pieces to make piece in the shape of a heart.
3
4  cube([10,10,10]);
5
6  translate([5,-10,0])
7  cylinder(h=10,r1=5,r2=5);
8
9  translate([-10,5,0])
10 cylinder(h=10,r1=5,r2=5);
```

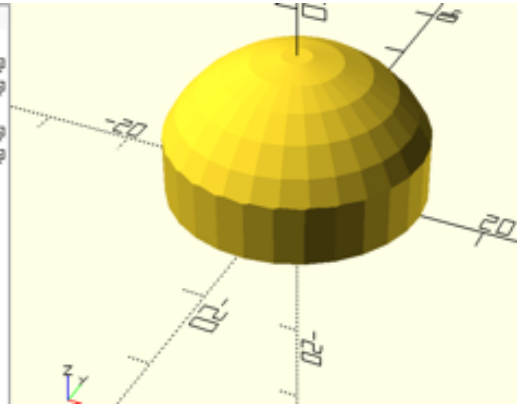


OpenSCAD sample code: Short simple OpenSCAD tutorials

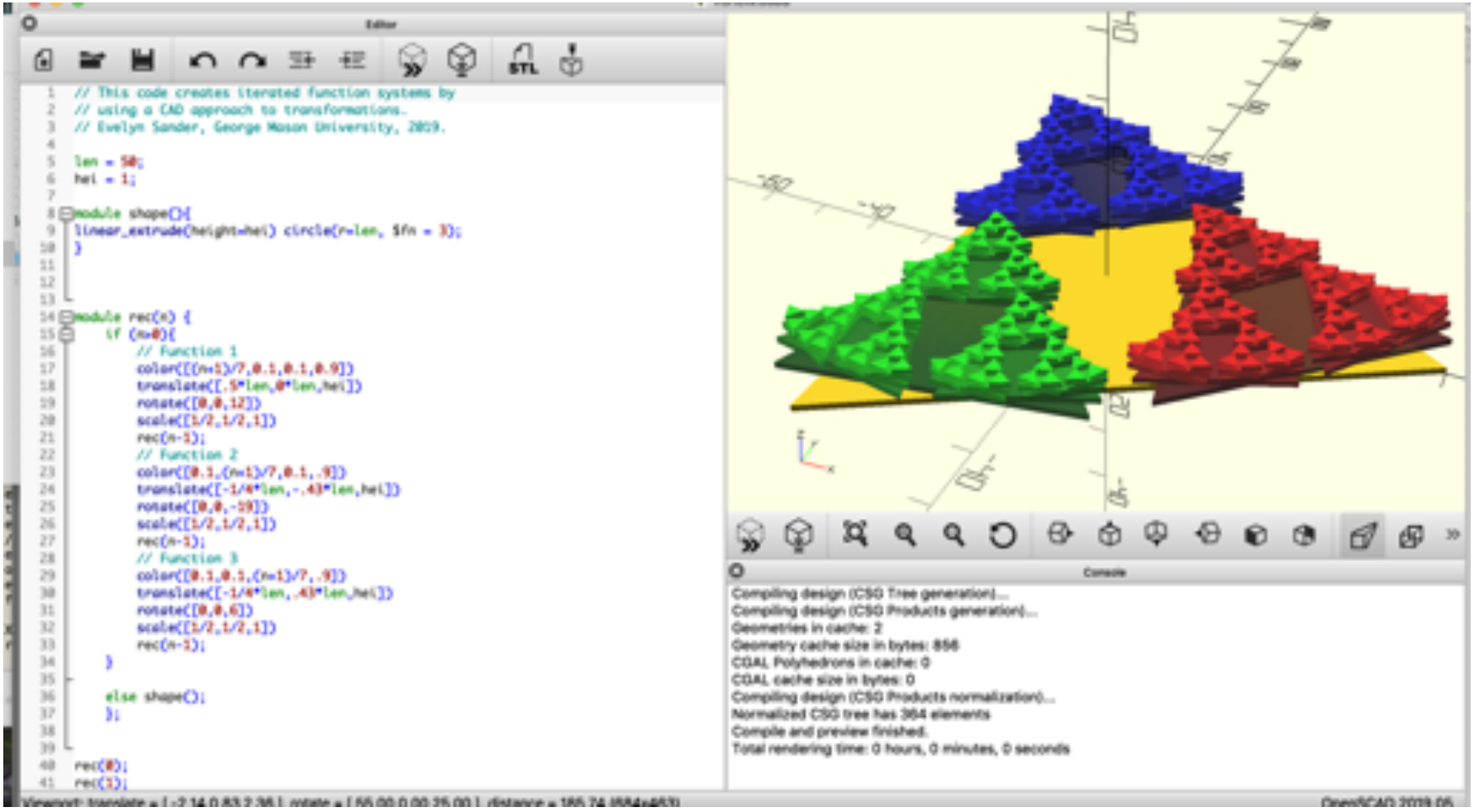
```
>
// Extrude Teardrop Instructions
//You can make 3D objects from 2D ones,
    sort of like with a Playdoe extruder.
// 1. Replace the circle command with the
    square command.
// 2. Show both the circle and the square
    at the same time. What shape is being
    extruded?
// 3. Change the twist from 360 to 180.
    Now figure out how to make it twist
    twice around.



$fn = 100;

linear_extrude(height = 120, twist=360)
    translate([0,20,0]) circle(15);
```



```
//Intersection is a useful way to combine two shapes.
//Scale makes and object bigger or smaller.
// 1. Change the scale amount to be larger or
    smaller. Which number corresponds to the
    original size?
// 2. Combine translations with different scale
    amounts to put a second dome on top of this
    (like on the U.S. Capitol Building).



scale(1.5)
intersection(){
    sphere(r=10);
    cylinder(r=9,h=10);
}
```



OpenSCAD sample code: Iterated function systems



Mathematica sample code excerpt: Chaotic attractor

Creating a strange chaotic attractor

Written by Evelyn Sander 2019

This particular code creates the Chen double-scroll attractor due to Chen and Ueta, 1999.

Set the parameters

```
a = 40;
b = 3;
c = 28;
```

Set the righthand side for the differential equation

```
F1[x_, y_, z_] := a (y - x)
F2[x_, y_, z_] := (c - a) x - x z + c y
F3[x_, y_, z_] := x y - b z
```

Set the initial conditions

```
x0 = -0.1;
y0 = 0.5;
z0 = -0.6;
timelengtha = 2;
```

Starting at the initial condition, solve the differential equation, but don't keep it because we are waiting for it to converge to the attractor. This is called "removing transients."

```
ap = NDSolve[{
    xa'[t] == F1[xa[t], ya[t], za[t]],
    ya'[t] == F2[xa[t], ya[t], za[t]],
    za'[t] == F3[xa[t], ya[t], za[t]],
    xa[0] == x0, ya[0] == y0, za[0] == z0},
  {xa, ya, za}, {t, 0, timelengtha}, MaxSteps -> Infinity]
```

```
{{xa → InterpolatingFunction[ Domain: {{0., 2.}} Output: scalar ],
  ya → InterpolatingFunction[ Domain: {{0., 2.}} Output: scalar ],
  za → InterpolatingFunction[ Domain: {{0., 2.}} Output: scalar ]}}
```

Even though we are only going to use the end of this as a starting point of the next computation, we look at the plot to see what it does.

```
ParametricPlot3D[Evaluate[{xa[t], ya[t], za[t]} /. ap], {t, 0, timelengtha},
 PlotRange → All]
```

Now we do the same thing, except this time we start at the place we ended the last time.

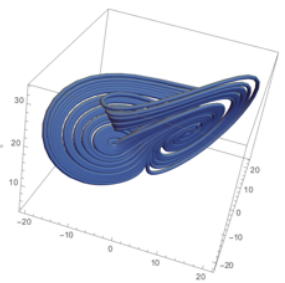
```
bp = NDSolve[{
    xb'[t] == F1[xb[t], yb[t], zb[t]],
    yb'[t] == F2[xb[t], yb[t], zb[t]],
    zb'[t] == F3[xb[t], yb[t], zb[t]],
    xb[0] == x1, yb[0] == y1, zb[0] == z1},
  {xb, yb, zb}, {t, 0, timelengthb}, MaxSteps -> Infinity]
```

```
{{xb → InterpolatingFunction[ Domain: {{0., 50.}} Output dimensions: {1} Data not in notebook; Store now » ],
  yb → InterpolatingFunction[ Domain: {{0., 50.}} Output dimensions: {1} Data not in notebook; Store now » ],
  zb → InterpolatingFunction[ Domain: {{0., 50.}} Output dimensions: {1} Data not in notebook; Store now » ]}}
```

Plot the attractor.

```
ParametricPlot3D[{xb[t], yb[t], zb[t]} /. bp, {t, 0, timelengthb}, PlotRange → All]
```

```
cp = ParametricPlot3D[{xb[t], yb[t], zb[t]} /. bp, {t, 0, timelengthb},
  PlotStyle → Tube[.5], PlotPoints → 100, PlotRange → All]
```



Recall that you need to change the name of the file!

```
Export["yourname_chen.stl", cp]
```

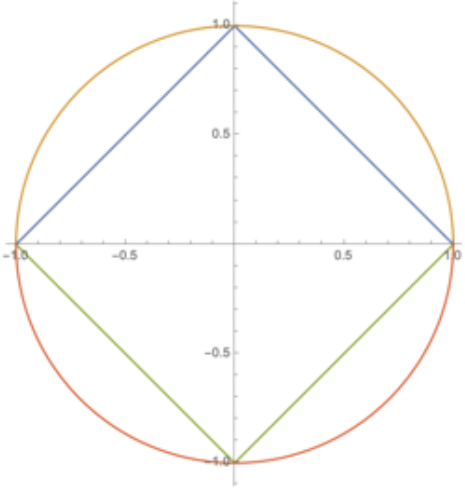
```
yourname_chen.stl
```

Mathematica sample code excerpt: Sugihara cylinders

Creating an ambiguous cylinder

Written by Evelyn Sander 2019

```
Plot[{1 - Abs[t], Sqrt[1 - t^2], -(1 - Abs[t]), -Sqrt[1 - t^2]}, {t, -1, 1},
 AspectRatio → Equal]
```

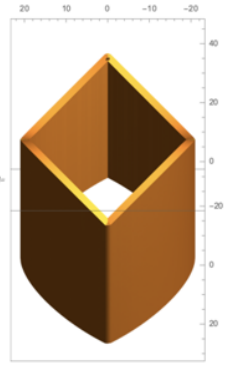



```
f1 = 1 - Abs[t]
f2 = -(1 - Abs[t])
g1 = Sqrt[1 - t^2]
g2 = -Sqrt[1 - t^2]
```

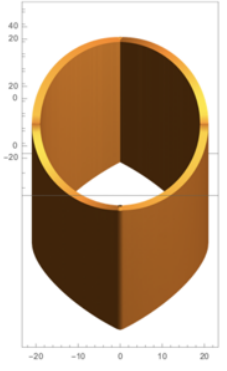
```
1 - Abs[t]
```

```
-1 + Abs[t]
```

```
a3 = ParametricPlot3D[20 {{t, sum1, u (h + diff1)}, {t, sum2, u (h + diff2)}},
  {u, 0, 1}, {t, -1, 1}, PlotStyle → Thickness[2], Mesh → False, PlotPoints → 100,
  AspectRatio → Equal, ViewPoint → {0, Infinity, Infinity}]
```




```
Show[a3, ViewPoint → {0, -Infinity, Infinity}]
```




```
Export["ambiguity.stl", a3]
```

```
ambiguity.stl
```

The payoff

I learned a lot from my students!

- Many coding commands and data structures
- Ideas for data visualization
- Even math: Student talk on complex graphs outlined relationship between sin and sinh as graph projections
- Github function collection contains equations for all superheros!

Thanks!

For further information see:

http://gmumathmaker.blogspot.com

Thanks for your attention!