Lab Assignment 10

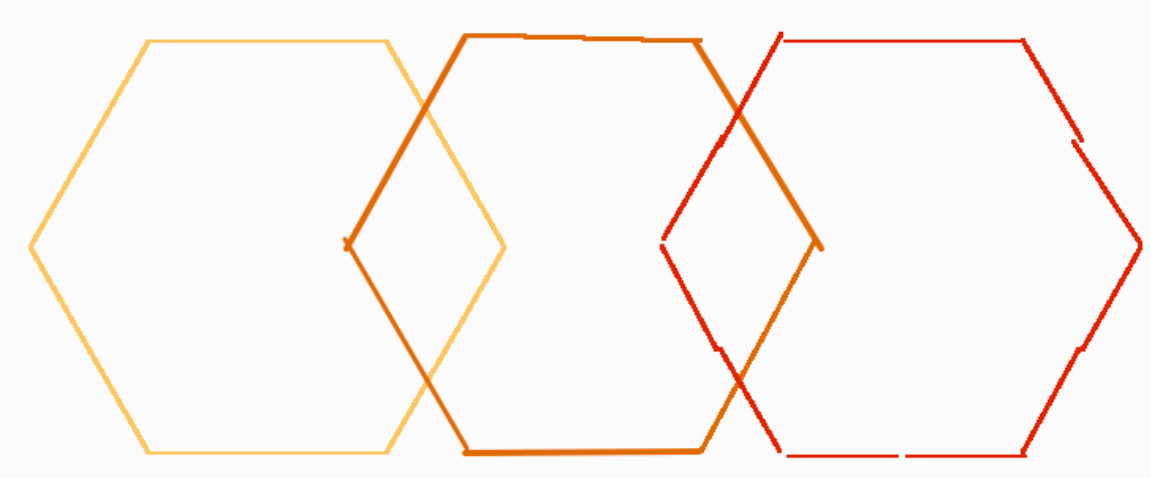
Kevin Bennett

Lab 10

The first task for this lab was to add the line style "broken" to the forward method of the interpreter. I basically copied and pasted the initial code from the jitter method and followed the instructions for the actual moving around and drawing stuff part of it. The one exception was that I changed the part of the code:# Go to (xm + jx, ym + jy)

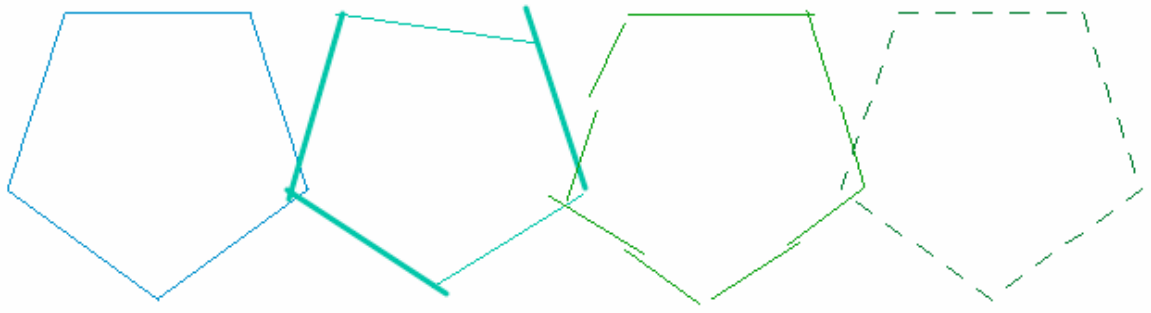
1. Pick the pen up
2. Go to (xm + jx, ym + jy)
   to:
   turtle.goto(xm + jx, ym + jy)
   turtle.up()
   turtle.goto(xm + kx, ym + ky)

If I left it as jx and jy for the second pair, it doesn't "break" because it is drawing from the same place.
That was an easy fix, however, and with little pain, I ended up with this picture:

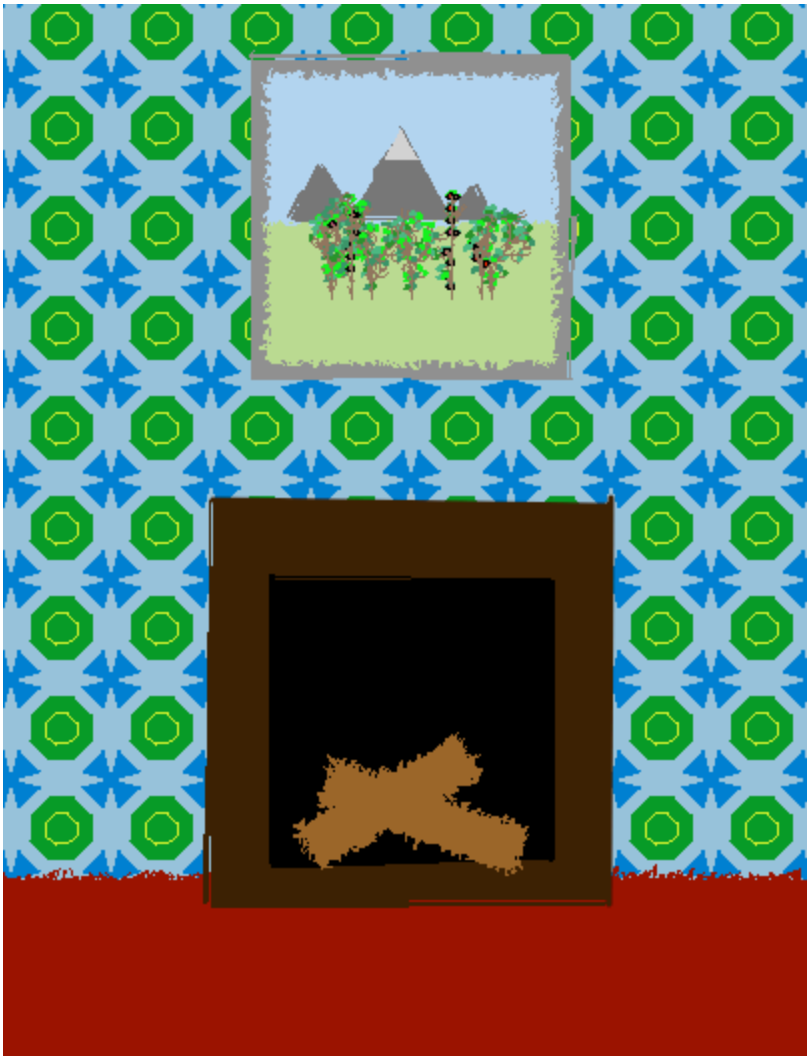


The second task was to create a dashed line style.
Like in the other styles, I assigned the beginning and end points to (x0, y0) and (xf, yf).
Then, inside a for loop in range int(distance/(self.dashLength*2)), I just had the turtle go forward, then pick it-
self up and go forward again. Once it was done, I had it pick up, go to the end point, and put down.
My test code was accidentally written over, but I recreated it with incorrect colors and positioning, but the code is basically the same.

Here was the picture of all four:



The third task was to alter our scene code from the previoius lab. All I did was add one line for each shape I wanted to change: [shape].setStyle([either 'jitter' or 'broken']).
Here is the picture:



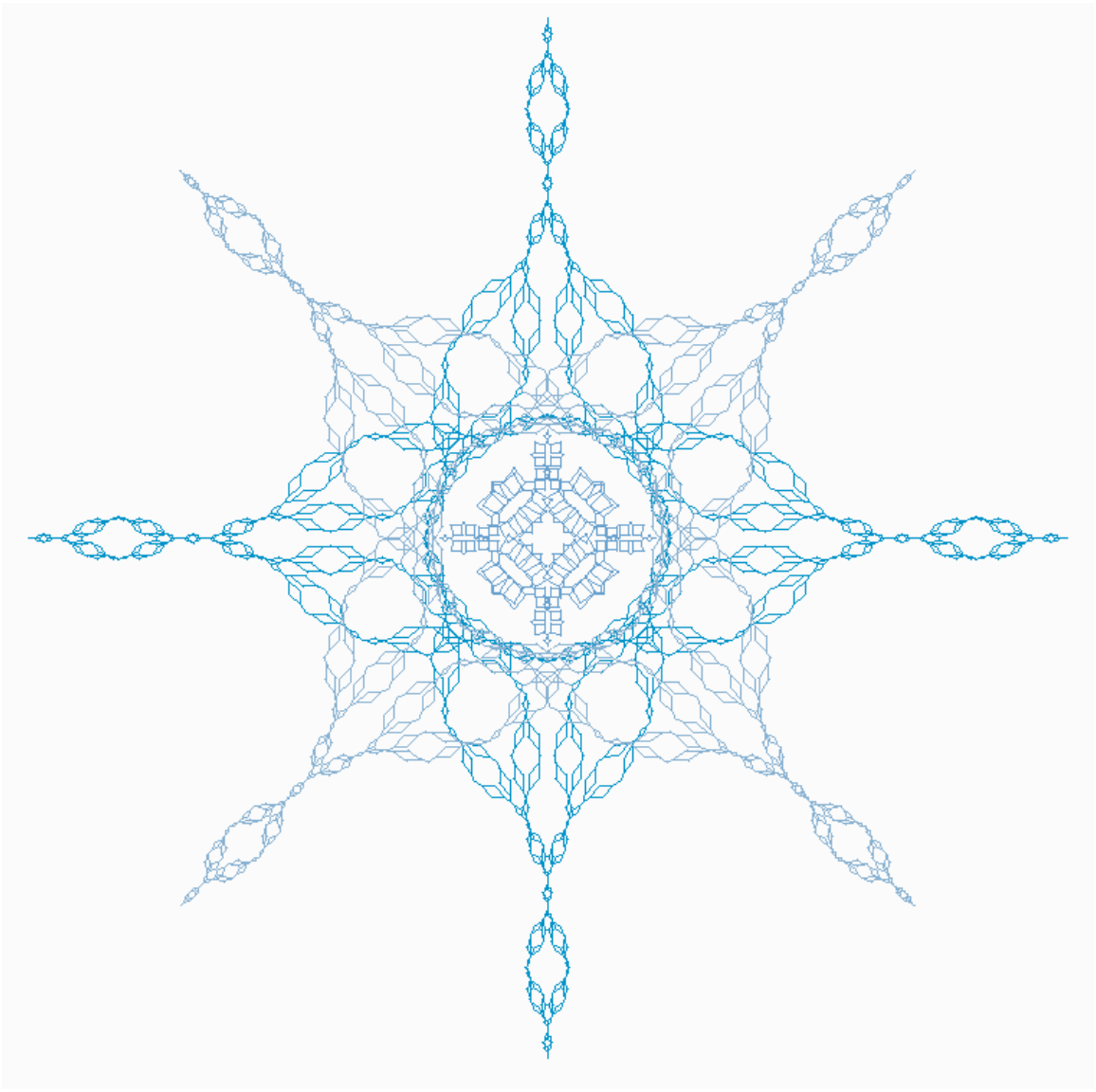
The rectangles were interesting because I did not parameterize their strings. The result: a random jitter each time the turtle moves forward. So the carpet, logs, grass, and sky look crazy.

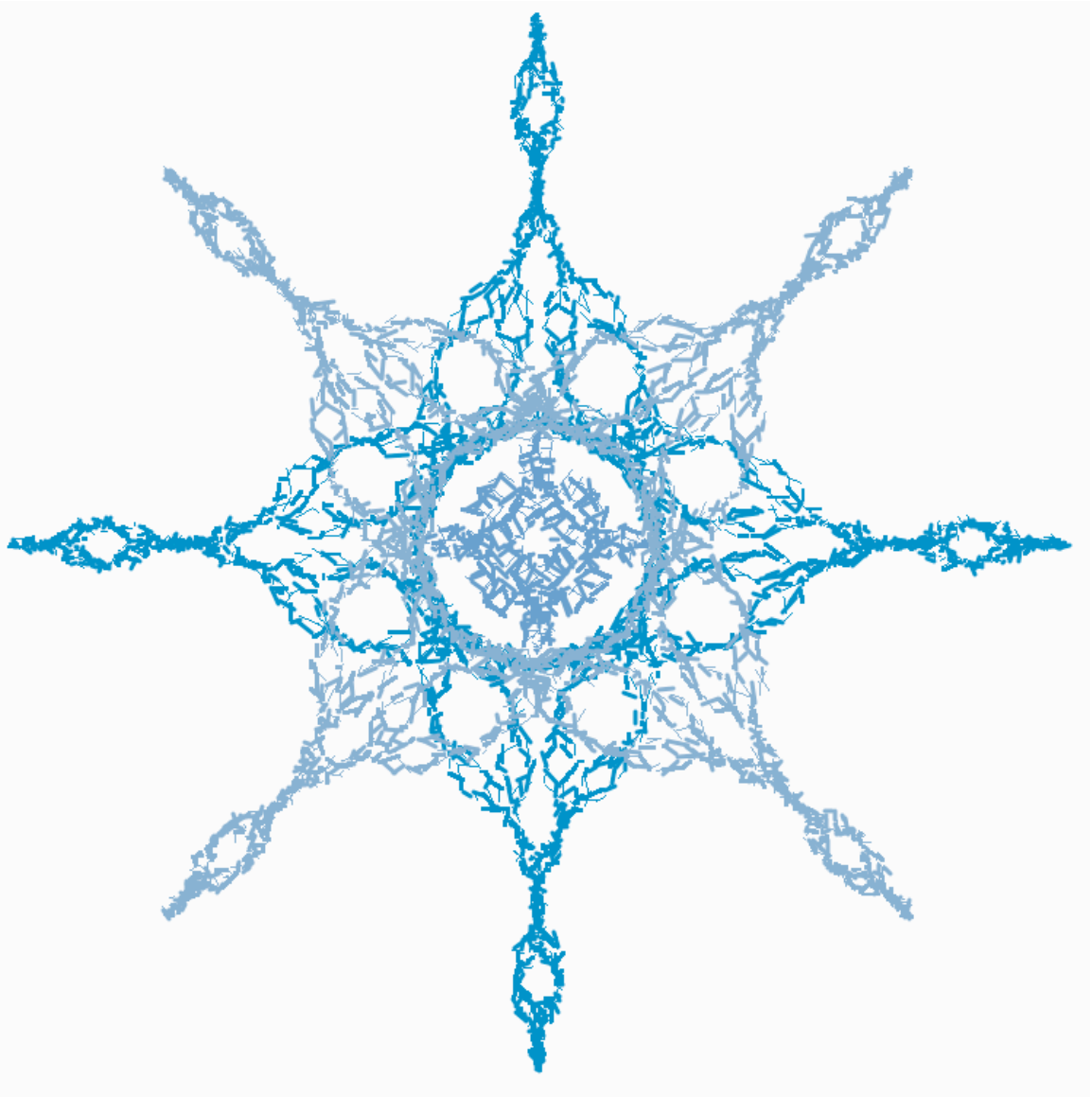
The last task was to create my own l-system. I made one with the following code:

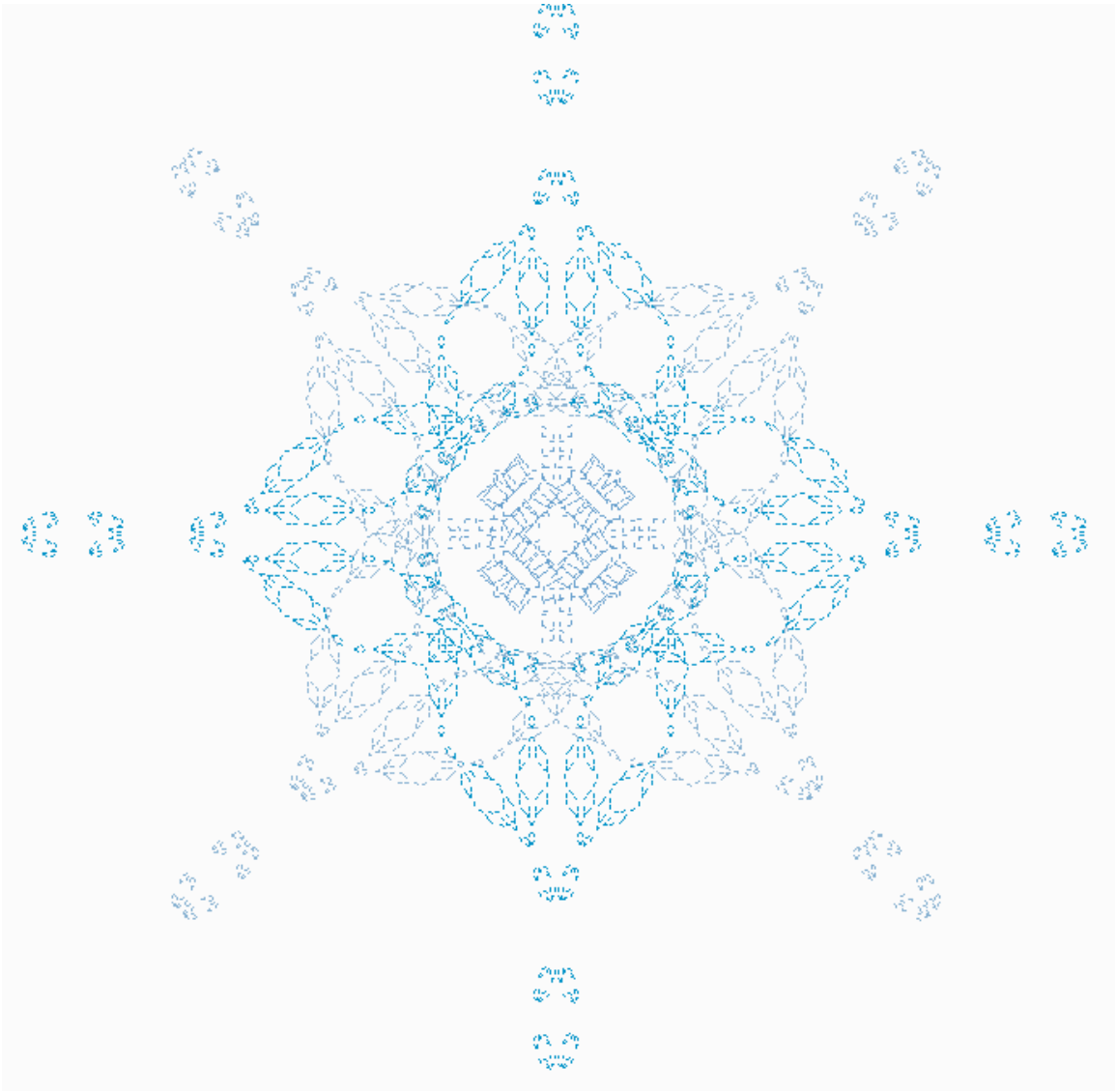
base (5)F
rule (x)F (x/2)F[~~(x)F+(x)F~~](x)F--(x)F+(x/2)F

Using several different instances of this l-system together, I created a snowflake image.

Then I kind of went overboard in drawing lots of versions of it because I found the resulting picture really surprisingly cool. Here it is, in normal, jitter, and dashed form:





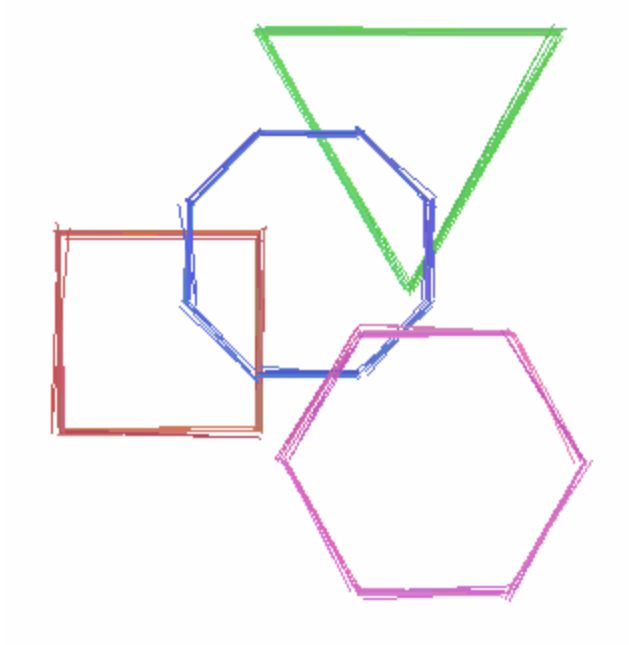


The different pattern on the inside is using an angle of 50 rather than 20.

Then, as an extension, I created a new line style, "brush," which draws 5 jittered lines on top of each other. The hard part of this extension was figuring out how to change the color a little random bit each time. Because after a long time of thinking about this, I did not receive any sudden bursts of genius or creativity, I just had to create a different "if" statement for each case, to make sure that the color never went above 1.0. The code was:

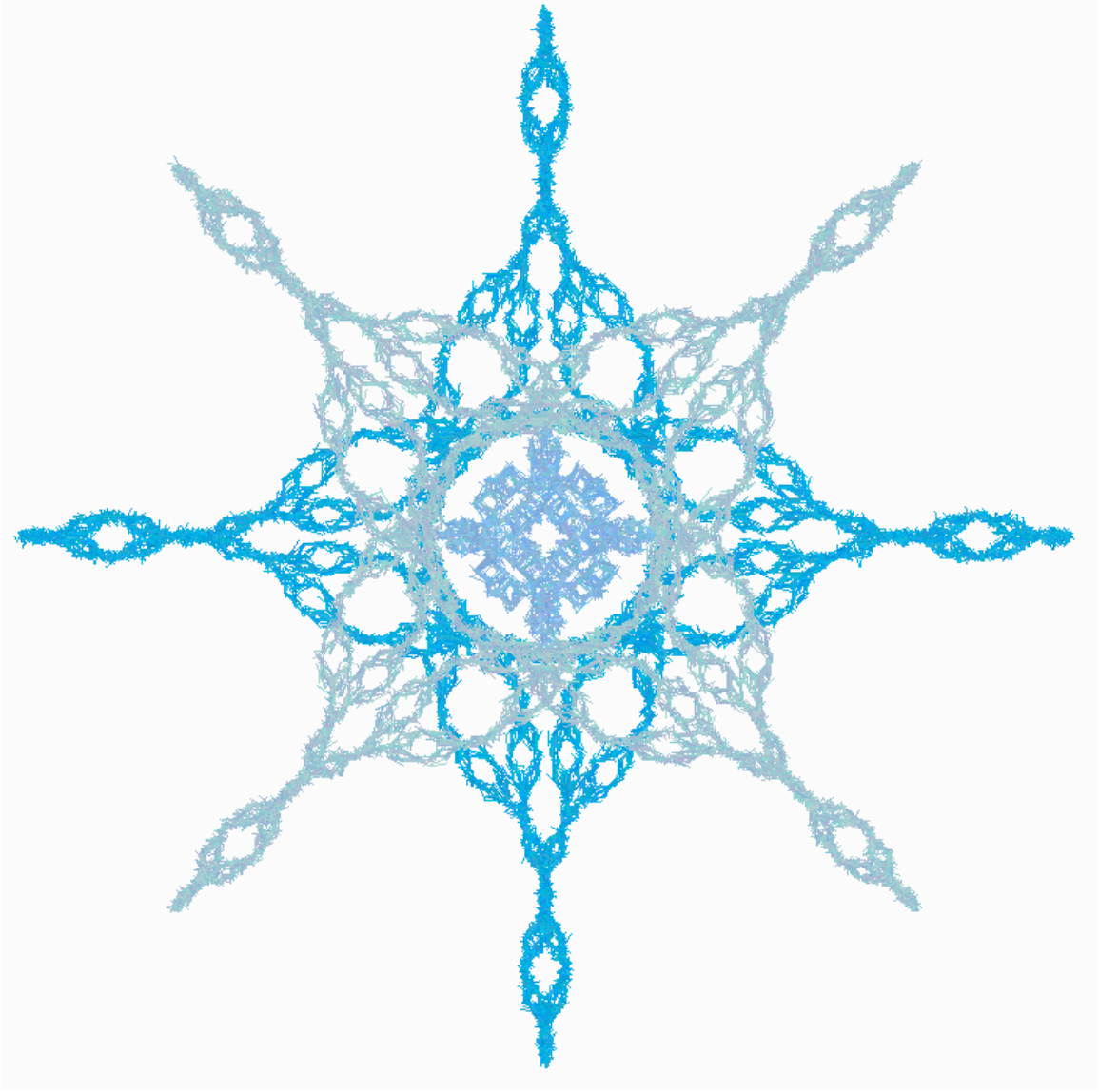
```
if r1 >= 0.8 and g1 < 0.8 and b1 < 0.8:
    turtle.pencolor( ( r1, g1 + random.uniform( 0.0, 0.2 ), b1 + random.uniform( 0.0, 0.2 ) ) )
```

...and so on for every combination of greater than or less than possibility. However, I wrote the code, and tested the style on a few of my shapes. I must say, I am a little proud of how it turned out.



It's a little hard to tell, but there are slight color differences.

And finally, because I just couldn't stop without looking at my snowflake one more time, I applied the brush effect to it, and ended up with:



Though some common little problems came up (including me having to add width to Shape and all of my Shape-derived classes and in a bunch of other places) and it took me a while to realize what I had to do for the dash effect, I didn't really encounter any major problems with this lab and felt pretty comfortable with the l-system, shape, and interpreter code. I definitely felt like I knew what I was doing. Though now (thanks to my width problems) I hope I have a better understanding of-or at least am more comfortable with-the relationship between the three main classes. I actually really enjoyed this lab.