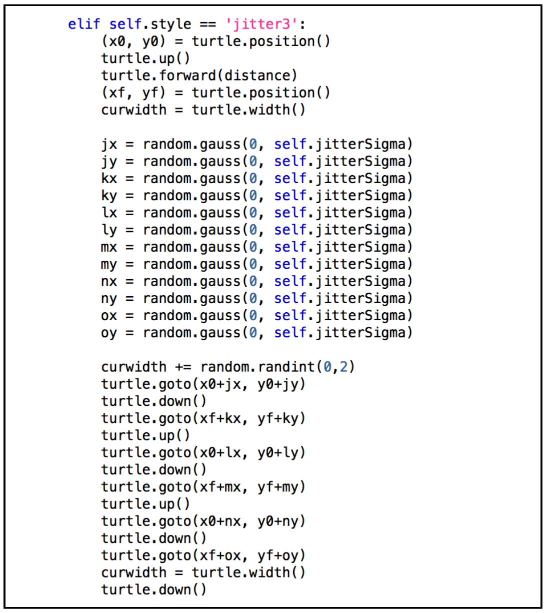
Project10

In this project, we developed our understanding of non-photorealistic rendering. Using my knowledge class, I created a couple new and different drawing styles for my drawing function gave the images a much more "hand-drawn" effect. I was able to use this new knowledge to adapt one of my prior scenes to look less perfect and more realistic. Also, in this project, I gained more practice with writing and understanding stochastic and parameterized L-systems.

Task 1. For task 1, I had to implement a new drawing style into my turtle_interpreter.py forward function. This new case was called "jitter3" and it drew three, criss-crossing jittered lines to represent one line. This style is very similar to the "jitter" style which drew one slightly (and randomly) slanted line instead of a perfecty straight line. To do this, I had to use a lot of similar code, just more of it. I needed 3 pairs of starting and ending x,y values randomly generated using random.gauss Gaussian distribution. Next, I had to add a few more goto() functions. Here is a snippet of my "jitter3" style code:

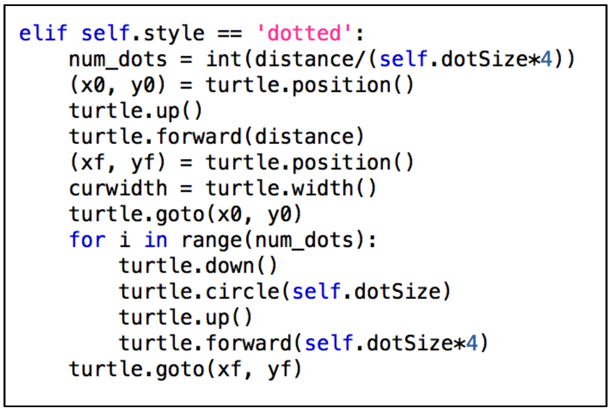


```
        elif self.style == 'jitter3':
            (x0, y0) = turtle.position()
            turtle.up()
            turtle.forward(distance)
            (xf, yf) = turtle.position()
            curwidth = turtle.width()

            jx = random.gauss(0, self.jitterSigma)
            jy = random.gauss(0, self.jitterSigma)
            kx = random.gauss(0, self.jitterSigma)
            ky = random.gauss(0, self.jitterSigma)
            lx = random.gauss(0, self.jitterSigma)
            ly = random.gauss(0, self.jitterSigma)
            mx = random.gauss(0, self.jitterSigma)
            my = random.gauss(0, self.jitterSigma)
            nx = random.gauss(0, self.jitterSigma)
            ny = random.gauss(0, self.jitterSigma)
            ox = random.gauss(0, self.jitterSigma)
            oy = random.gauss(0, self.jitterSigma)

            curwidth += random.randint(0,2)
            turtle.goto(x0+jx, y0+jy)
            turtle.down()
            turtle.goto(xf+kx, yf+ky)
            turtle.up()
            turtle.goto(x0+lx, y0+ly)
            turtle.down()
            turtle.goto(xf+mx, yf+my)
            turtle.up()
            turtle.goto(x0+nx, y0+ny)
            turtle.down()
            turtle.goto(xf+ox, yf+oy)
            curwidth = turtle.width()
            turtle.down()
```

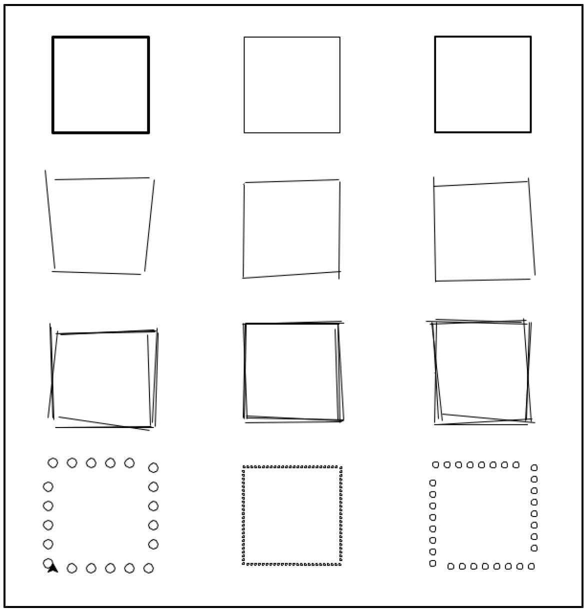
Task 2. This task also asked me to create a new style, but instead of a jitter style it was to be called "dotted." The purpose of this style was to draw a series of circles in the shape of the line. To do this, I had to create field in the TurtleInterpreter class called dotSize, and I had to create setDotSize methods in both my TurtleInterpreter class and Shapes class. Here is the snippet of code for my "dotted" style:



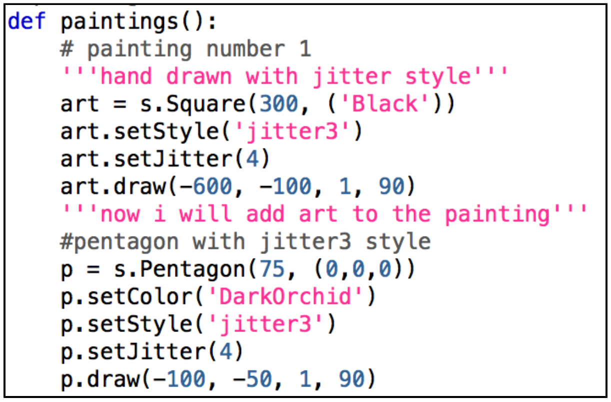
```
elif self.style == 'dotted':
    num_dots = int(distance/(self.dotSize*4))
    (x0, y0) = turtle.position()
    turtle.up()
    turtle.forward(distance)
    (xf, yf) = turtle.position()
    curwidth = turtle.width()
    turtle.goto(x0, y0)
    for i in range(num_dots):
        turtle.down()
        turtle.circle(self.dotSize)
        turtle.up()
        turtle.forward(self.dotSize*4)
    turtle.goto(xf, yf)
```

Task 3. This task just asked me to create a new file titled demo_line_styles.py that drew a bunch of a shape that depicts each of the drawing styles with a couple different varieties. Here is what the resulting image from this file looked like:

Required Image 1:



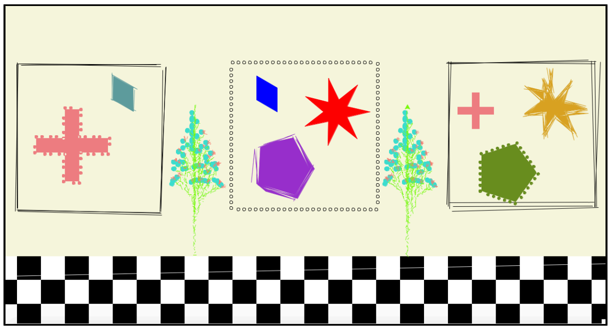
Task 4. This task asked me to use these new drawing styles in my scene from the past project and change aspects of the scene to make it look more hand drawn. I decided to use my scene from the last project and change the frames of the museum paintings and the actual shapes in the paintings with new drawing styles. I had to create a new file titled indoor_scene.py. Here is a snippet of code:



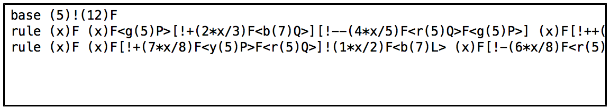
```
def paintings():
    # painting number 1
    '''hand drawn with jitter style'''
    art = s.Square(300, ('Black'))
    art.setStyle('jitter3')
    art.setJitter(4)
    art.draw(-600, -100, 1, 90)
    '''now i will add art to the painting'''
    #pentagon with jitter3 style
    p = s.Pentagon(75, (0,0,0))
    p.setColor('DarkOrchid')
    p.setStyle('jitter3')
    p.setJitter(4)
    p.draw(-100, -50, 1, 90)
```

Here is the resulting image from a number of these changes:

Required Image 2:



Task 5. This task asked me to create a new stochastic parameterized multi-rule L-system. To do this, I looked at some of the files we have been provided in the past and created my own new L-system tree. Here is a snippet of my new lsystem in the file "project10lsystem.txt":

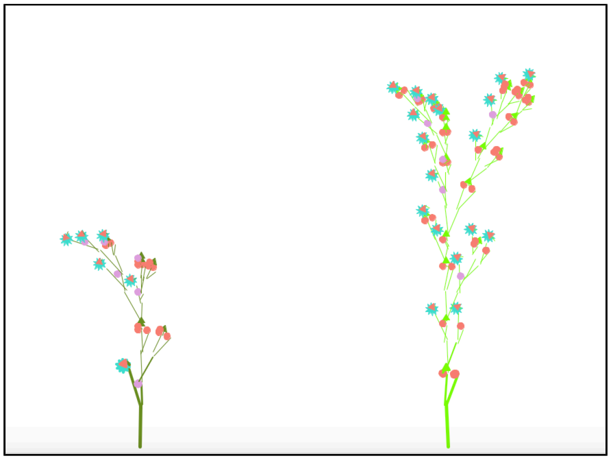


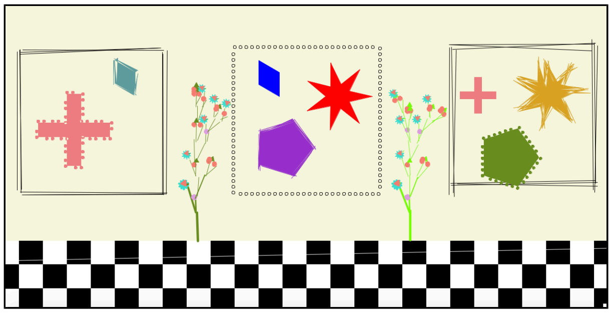
```
base (5)!(12)F
rule (x)F (x)F<g(5)P>[!+(2*x/3)F<b(7)Q>][!--(4*x/5)F<r(5)Q>F<g(5)P>] (x)F[!++(
rule (x)F (x)F[!+(7*x/8)F<y(5)P>F<r(5)Q>]!(1*x/2)F<b(7)L> (x)F[!-(6*x/8)F<r(5)
```

Then, I put my new trees into my scene!

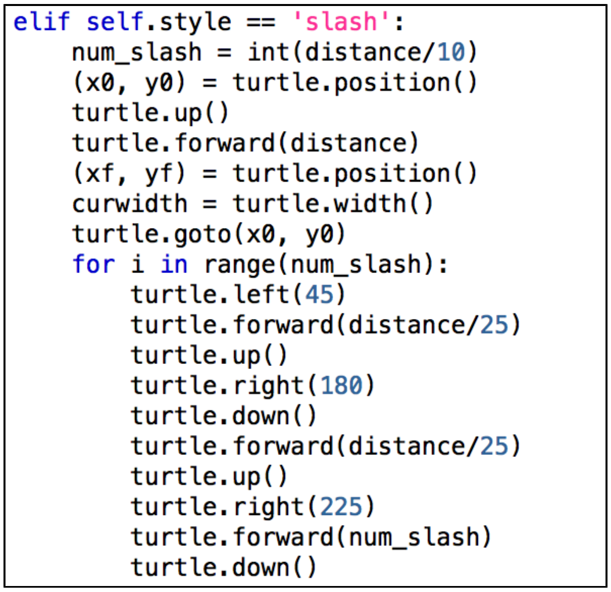
Here is the tree my new L-system created:

Required Image 3:

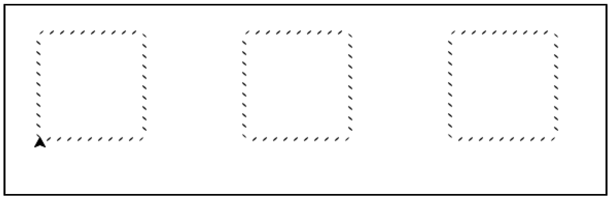




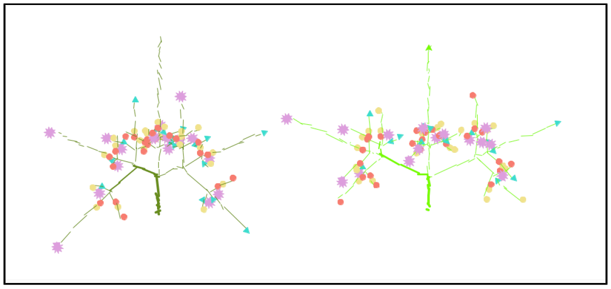
After I completed the required tasks, I went ahead and completed 3 extensions. First off, for **extension 1**, I completed one of the recommended extensions and created a new drawing style of my own. This style was called "slash" and drew a series of slashes to represent the line. Here is the snippet of code for the slash style:



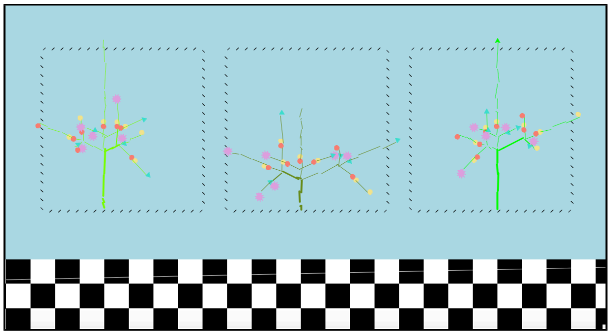
```
elif self.style == 'slash':
    num_slash = int(distance/10)
    (x0, y0) = turtle.position()
    turtle.up()
    turtle.forward(distance)
    (xf, yf) = turtle.position()
    curwidth = turtle.width()
    turtle.goto(x0, y0)
    for i in range(num_slash):
        turtle.left(45)
        turtle.forward(distance/25)
        turtle.up()
        turtle.right(180)
        turtle.down()
        turtle.forward(distance/25)
        turtle.up()
        turtle.right(225)
        turtle.forward(num_slash)
        turtle.down()
```



Extension 2. I completed another one of the recommended extensions and designed another new L-system. My goal in this L-system was to create something that didn't look much like a tree, but more like a train underground map. This file is titled "project10extension2.txt": Here is what the images it produces look like:



Extension 3. Finally, this extension basically took the first two extensions and put them together in a scene in my project10extension3.py file. Here is what the resulting image looked like:



What I learned. I learned how to give images, shapes, and scenes a more realistic touch as if though someone had drawn the picture by hand. I became more familiar with the use of the class systems, field, and methods. Also, I now have a better understanding of stochastic parametrized multi-ruled L-systems and how to create my own. Overall, this project made me much more comfortable with my ability to create a realistic scene and more complex L-systems.

Who helped me. I received help from TA Mike, Julia Saul, and Professor Taylor.