Stephan's Project 10

Stephan Chaikovsky

4.28.15

CS151A

Project 10: Non-Photorealistic Rendering

Summary: For this project, I added 4 drawing styles to my turtle_interpreter's forward() method ('jitter3', 'dotted', 'zigzag', 'zipper'). Then, I created a file that tests the 'normal', 'jitter', 'jitter3', and 'dotted' styles with variations. After this, I modified my indoor_scene.py file from last week and made it look more like a drawing by implementing these styles. Finally, I created my own parameterized stochastic multi-rule L-System and added it to my indoor_scene.py file.

Description of Files:

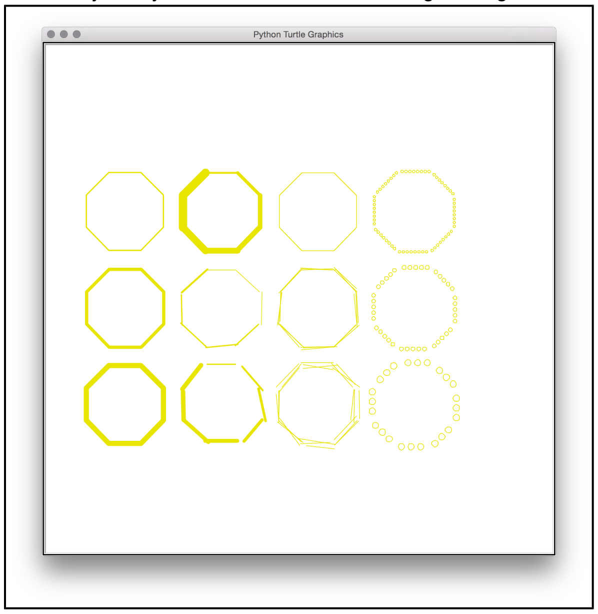
turtle_interpreter.py: For this project, I added four styles to the turtle_interpreter class:

Required:

'jitter3': This style draws 3 lines beginning at short, random distances from the initial position and ending at short, random distances from the desired end point. To achieve this, I made a for loop that generates start/end points using random.gauss and draws the lines between them.

'dotted': This style draws a series of circles separated by spaces. To achieve this, I use a loop to draw circles evenly spaced from one another, and after making it through that loop, I move the turtle to the intended end point as to make sure my distance stays consistent. In creating this style, I also had to create dotSize fields and setter in this file and shapes.py.

Required Image 1: Demonstrates required styles with different parameters as specified by the assignment. The styles (from left to right) are 'normal', 'jitter', 'jitter3', and 'dotted'. This image was generated by my **demo_line_styles.py** file which imports shapes.py.



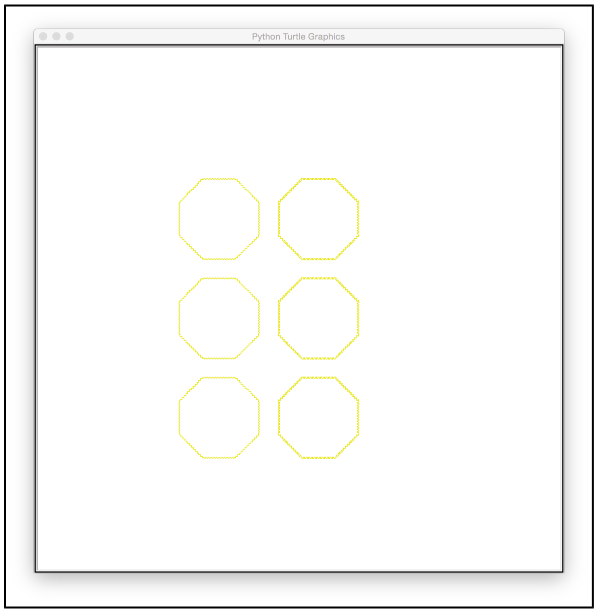
Extensions:

'zigzag': This style draws a zig-zagging line from the start to the end point, and uses the sqrt function from the math import to correctly calculate the distances to travel with each "zig".

'zipper': This style draws 'zipper' style lines (see picture below); these lines are most easily explained by an L system: (90)L(1)F(90)R(1)F(90)R(2)F(90)LF(1)(90)LF(2)(90)R (the part of the system in red loops continuously until the line is completely drawn).

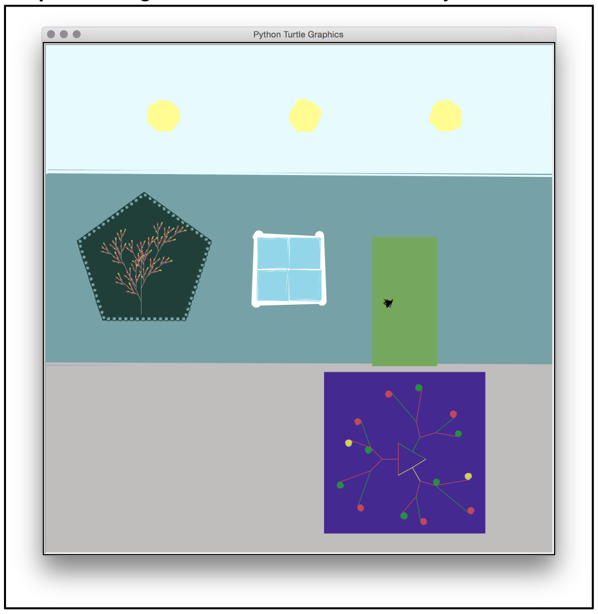
Here is a test of my 'zigzag' style (on left side) and my 'zipper' style (on right side):

(It might be easier to see the styles by zooming in)



indoor_scene.py: For this file I modified my indoorscene.py file in order to make it look more like a drawing by implementing some of the styles I created. I used the "jitter3" style to draw the lights and door handle, "dotted" style to draw the pentagonal picture frame, and "jitter" to draw the window and background. I also added a rug to the scene that contains a rendition of the L-system I created for this project (info about the system below image).

Required Image 2/3: A modified redntion of my indoorscene that incorporates the L-system I created for this project on a rug.



My L-System: The L system I created contains branching, multiple rules, multiple replacement strings, color changes, and ornaments. In designing this L system, I wanted to create a triangle from which 3 smaller L trees grow. Additionally, I wanted for each side of the triangle to be of a different color, and for the first branch coming off of each side to be the same color as the side. The base case I provided draws this triangle, with different color sides, and initiates the branches coming off of it. The branches then branch out to additional branches, which vary in angle, color, and whether or not they have berries. See my file myLSys.txt for reference.

Extensions: For extensions, I created 2 additonal drawing styles, "zigzag" and "zipper". sSe the description of turtle_interpreter.py above for more information and a test image.

What I learned: In this project I learned how to create and implement different drawing styles.

Outside Help:

None