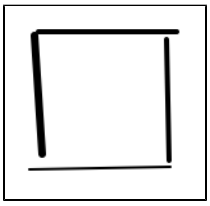
Jonathan's CS151 Project 10

Overview:

The purpose of this project was to learn and implement new methods into our preexisting turtle interpreter, shapes and lsystem files that would allow us to draw the shapes we created last project in new ways. We needed to add new cases to our TurtleInterpreter class in our turtle interpreter file in order to make our new designs. The two new designs that we implemented in this project were the jitter and dot methods.

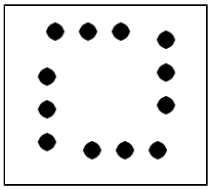
Task 1:

Our main goal for the jitter method was to chose starting and ending points for lines that were extremely close, but not exactly, the original points. This would allow us to draw more realistic drawings, as not everyone can draw perfectly straight lines. An example of the jitter method is below:



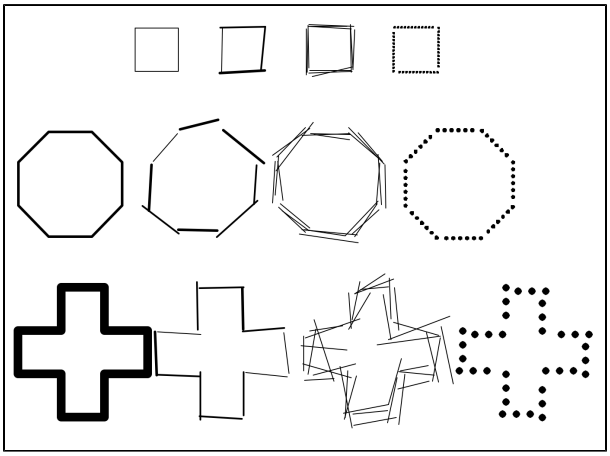
This is supposed to be a perfect square, however with the jitter method, the lines are just slightly off. These designs can make extremely cool images.

Task 2:

In regards to the dot style, we wanted to create a method that would draw circles as the perimeter of the shape, not a solid line. This would allow for a very cool design. An example of the dot method is below: 
Again, this is programmed to be a perfect square drawn with a solid line as a perimeter. However, with our new dot method, we can create a square, or any shape, with a dotted perimeter.

Task 3:

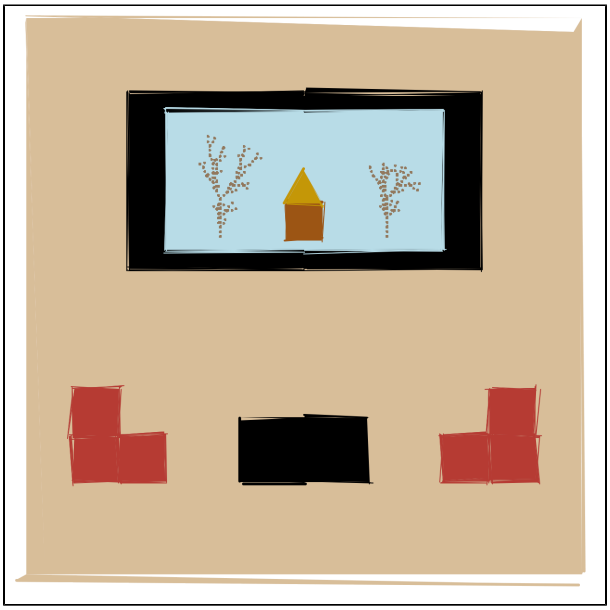
Our third task was to show that we had the ability to completely control the jitter and dot methods. For this task, we needed to show that we can draw shapes that use the normal, jitter, jitter3 (essentially the jitter method, just with 3 lines instead of 1) and the dot method. With all of these methods, there are variables that we can change to slightly change the output. For example, we can change the width of the line that the turtle draws. For jitter, we can change the sigma value. Essentially the sigma value is the distance that the line can be from the original start and stop points. It will be easy to see how the sigma value effects the shapes when an image is shown. Furthermore, we can change the dot size, which changes how many dots perimeters the shape. Below is an image showing that I can properly use all of the new methods:



The first shape in each row uses the normal method. Clearly the cross has a larger line width that the square or the octagon. The next shape in each row uses the jitter method. Next, the third shapes in each row use the jitter3 method. The square has the smallest jitter sigma, and the cross has the largest jitter sigma. Clearly, a smaller sigma will give you a shape that represents the shape the most. A larger sigma will make it look least like the actual shape. Lastly we have the dot method. The square uses the smallest dot size, and the cross uses the biggest dot size.

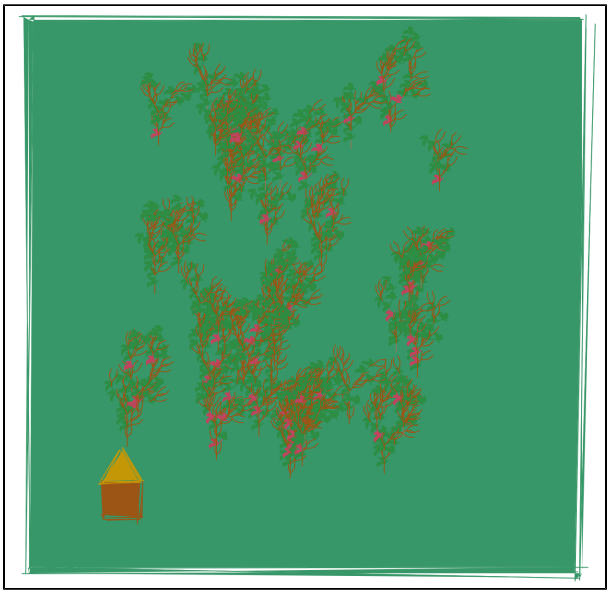
Task 4:

Our fourth task was to edit our indoor scene from project 9 to look more like a real painting. In order to do this, I changed the styles of all the shapes draw to jitter or dotted. The jitter style makes everything look more like a real painting. I used the dotted style for my L-system trees. This gave the trees a cool new look. Below is my new indoor scene:



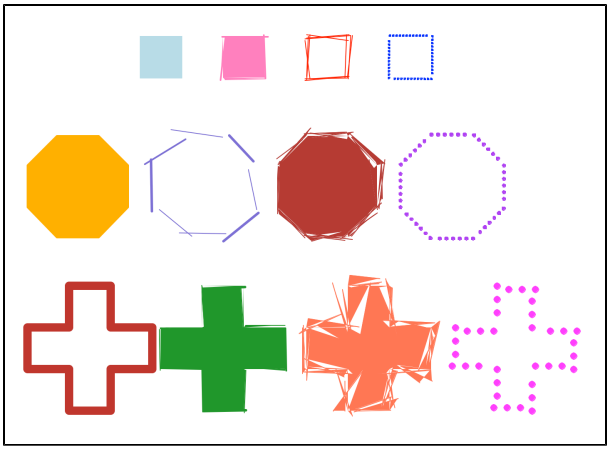
Task 5:

Task 5 asked us to create a new L-system that includes branching, multiple rules, and at least one rule with more than one replacement string. With this new L-system, we needed to create a simple scene showcasing our creation. Below is my scene with the L-system. The L-system is what makes all the trees. Notice how every tree is different:



Extensions:

1.) For my first extension I expanded my main code for task 3. In task three, we had to show off all of our new ways of drawing shapes. In this extension, I decided to expand that main code and add color to task 3. For some shapes I simply changed the outline color. Other shapes I completely filled in. Below is my first extension:



What I learned:

During this lab I learned how to draw shapes differently. It was very important for me to learn how to draw shapes using different styles, such as jitter and dotted. Also, creating my own L-system was very helpful in further understanding how L-systems work.

Acknowledgments: Holli Olsen, Prof. Taylor