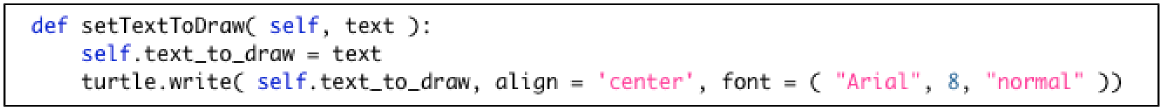
Lingar Project 8

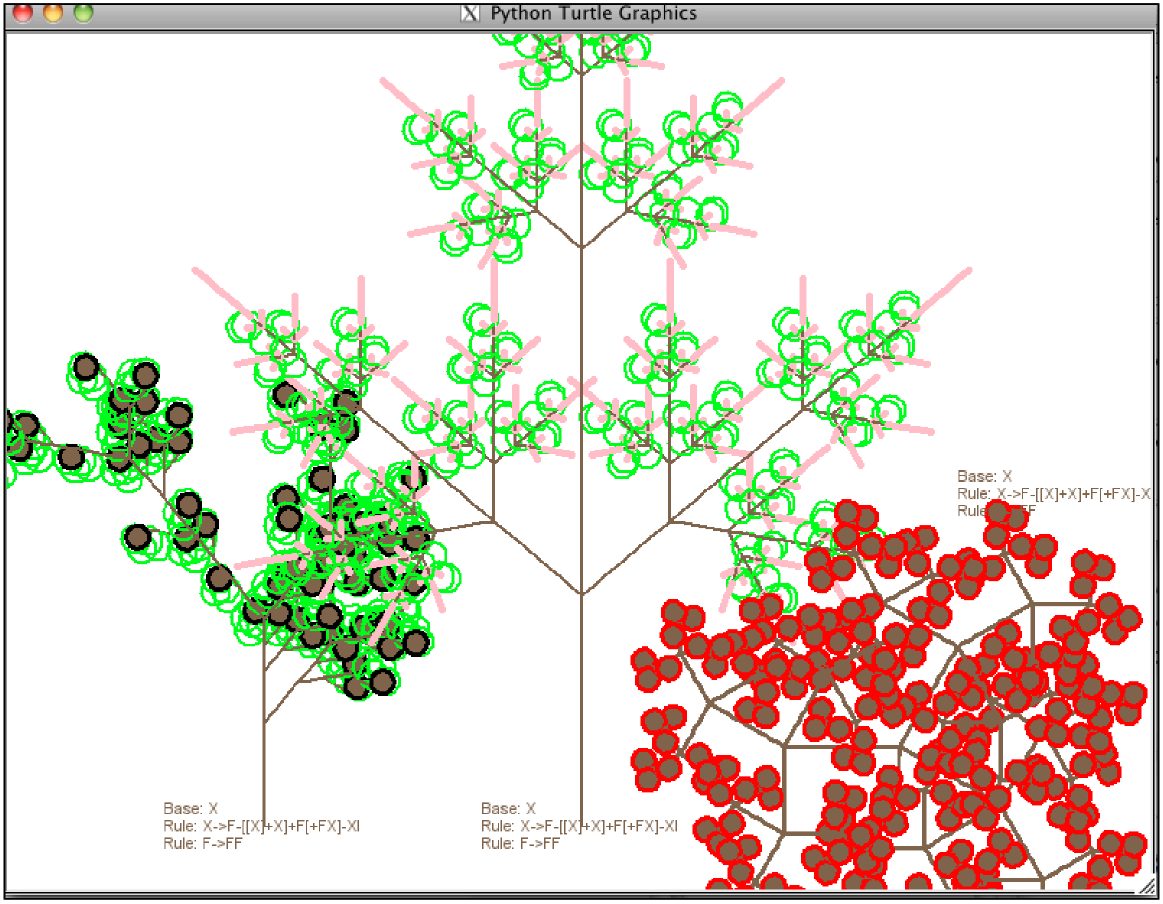
In today's project, I made fractal shapes and trees while not using the turtle module in my scene files. Instead, I used the interpreter and drawString. I learned how to add variables to the drawString function. I also experimented with adding "elif" statements and "if"s within the "elifs" in the interpreter.

In the first task, I created an image using three different lstrings. I drew the base and rule near each of the three shapes by referencing a "setTextToDraw" function which I created in the interpreter:



```
def setTextToDraw( self, text ):
    self.text_to_draw = text
    turtle.write( self.text_to_draw, align = 'center', font = ( "Arial", 8, "normal" ))
```

I added elif statements to the drawString in order to place acorns, flowers, and berries on the trees.

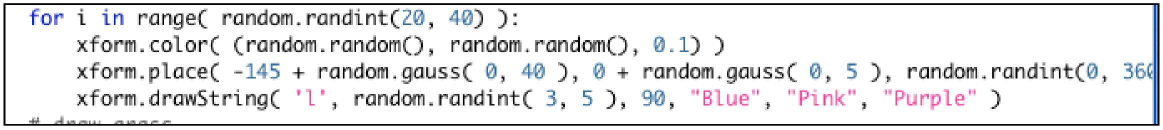


```
Base: X
Rule: X->F-[[X]+X]+F[+FX]-X
Rule: F->FF
```

```
Base: X
Rule: X->F-[[X]+X]+F[+FX]-XI
Rule: F->FF
```

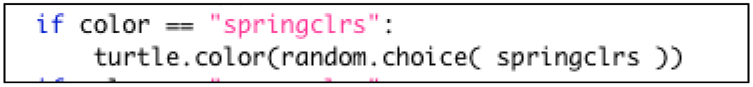
```
Base: X
Rule: X->F-[[X]+X]+F[+FX]-XI
Rule: F->FF
```

In the second task, I made four different trees using the CL system. I added background color, line color, and fill color variables to the drawString in order to control colors from the main code. I added code using the gauss function in order to place grass, flowers, and snow banks around each tree depending on the season:



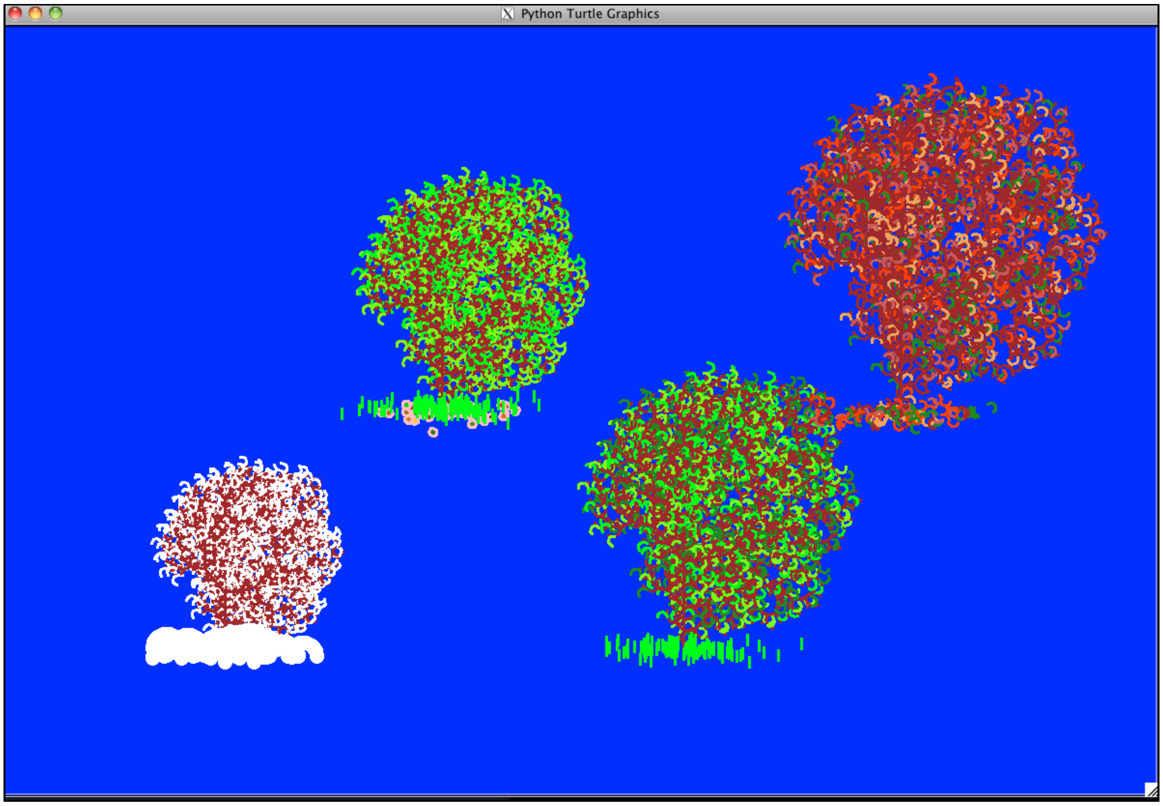
```
for i in range( random.randint(20, 40) ):
    xform.color( (random.random(), random.random(), 0.1) )
    xform.place( -145 + random.gauss( 0, 40 ), 0 + random.gauss( 0, 5 ), random.randint(0, 360
    xform.drawString( 'l', random.randint( 3, 5 ), 90, "Blue", "Pink", "Purple" )
# draw grass
```

I also created lists of colors for each season, and called them using if statements in the drawString:



```
if color == "springclrs":
    turtle.color(random.choice( springclrs ))
```

I increased the number of iterations with each season so it would appear that the tree was growing throughout the year.



In the third task, I first modified the CL system to create two new lsystems. The original system was:

base F
rule F FF-[F+F+[F]<x>]+[+F-F[F][F]<x>]

I created new1.txt:

base F

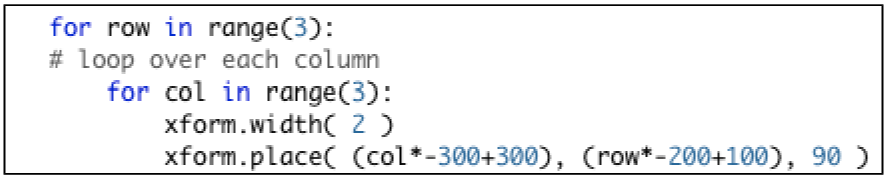
rule F FF-[F-F-[F]<rA>][+F-F-[F]<rA>]

and new2.txt:

base F

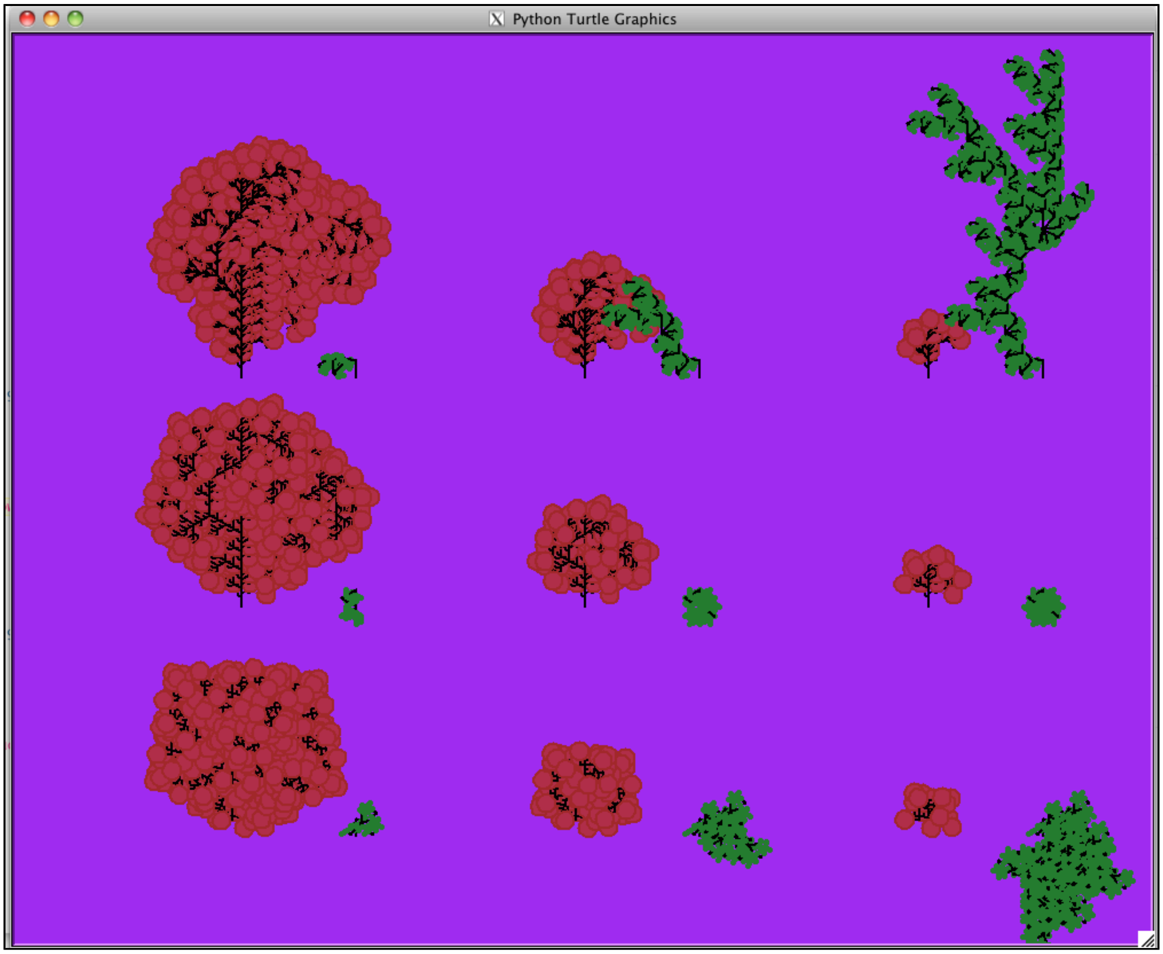
rule F FFF+[F[F]<gL>]+[F[F]<gL>]

I created code which is similar to the grid design in the last project. I created a double for loop for each of the lsystems:



```
for row in range(3):
# loop over each column
    for col in range(3):
        xform.width( 2 )
        xform.place( (col*-300+300), (row*-200+100), 90 )
```

The number of iterations [2, 3, 4], increased from right to left for the first lsystem and from left to right for the second lsystem. The angle [40, 60, 80], increased from right to left for both lsystems.



Extensions:

1) In the second task, I modified the trees by season.

2) I added acorns, flowers, and berries to the first task.

3) I used double for loops in the third task, as well as math to place the trees in relation to each other in the grid (see above structure of the for loop and xform.place.) Also, I used if statements in the interpreter to create seasonal variance. Get fancy with Get fancy with required images 2 or 3. Note that fancy means using programming structures (functions, loops, conditionals, lists, or math) to make a more interesting scene.required images 2 or 3. Note that fancy means using programming structures (functions, loops, conditionals, lists, or math) to make a more interesting scene.