Ismael Villlalobos
February 12, 2019
ID:80586103
Professor Fuentes

Lab 1

Introduction: The purpose of this lab is to use recursive methods using Python and its libraries to draw interesting figures.

Proposed Solutions:

Figure1- For this this Image my proposed solution was to draw a square from a given origin point. First, a midpoint was established using a length that was an input parameter. Next, staring from the given origin it would plot the five coordinates needed to draw an initial square. After the first square has been plotted it would make 4 variables for each point in the square and use them as new origin points for the second call of the recursive method. This would continue recursively for n number of times.

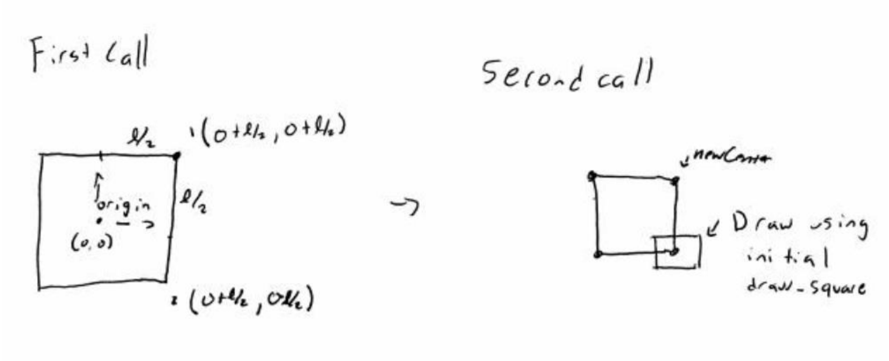


Figure 2- For this image I referenced the given draw_circles code as a skeleton. I began by first drawing a circle with origin point being the same distance as the radius. Next, the radius would be decreased by some given w. The recursive call would scale the new circle by some w using the previous circle as a reference. This would continue recursively n number of times.

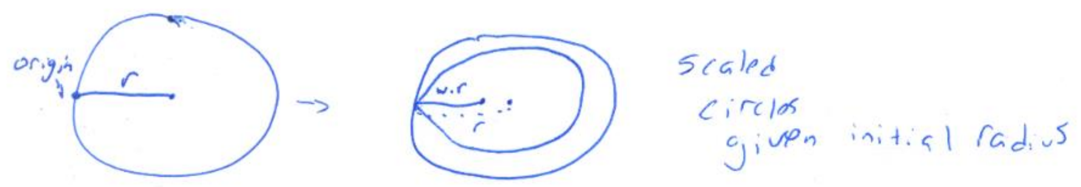


Figure 3 – To recreate this figure I would began at an origin point. I would then call the index of the origin array and create points that would be used to plot the initial branch. I would then plot lines given the new points created. The call would run the following time adjusting the x value by halving it and reducing the y value by a given scale of 0.9. The call would run recursively for n number of levels in the tree.

Ismael Villlalobos
February 12, 2019
ID:80586103
Professor Fuentes

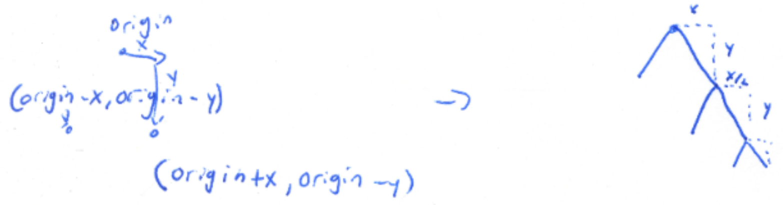
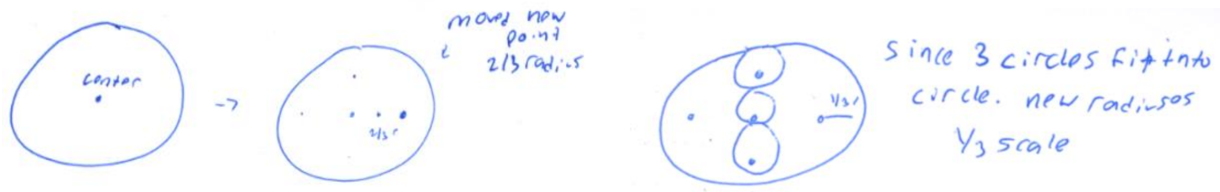
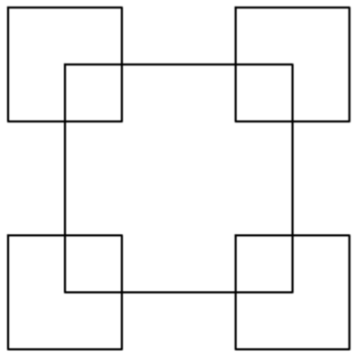


Figure 4- For this image I began by first constructing an initial circle given an initial center point of (0,0) I would then reference the earlier given draw_circles method as a skeleton. Once in the recursive method I would create 4 points for the outer circles. Given the output file I noticed the radiuses were scaled by 1/3 given the original circle. My new center points for the 4 outer circles would be 2/3 the size of the original so my centers would shift up, down, left and right respectively given that center point. I would then draw 5 circles the first being using the same given center point and my other four with my new centers. This would be called n number of times.

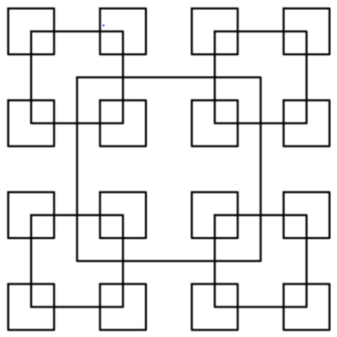

center
moved new point 2/3 radius
2/3 r
1/3 r
Since 3 circles fit into circle. new radiusos 1/3 scale

Experimental Results

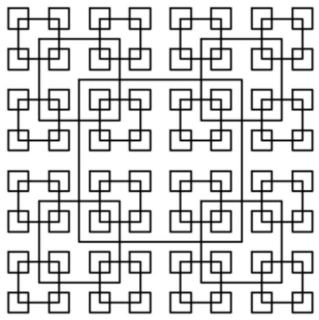
Figure 1



draw_square_flower(ax, [0, 0], 2, size)



draw_square_flower(ax, [0, 0], 3, size)

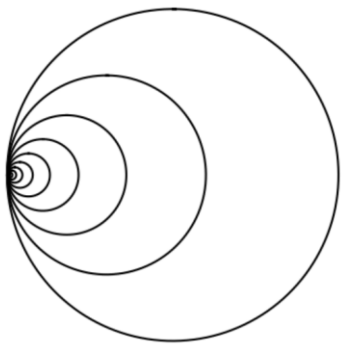


draw_square_flower(ax, [0, 0], 4, size)

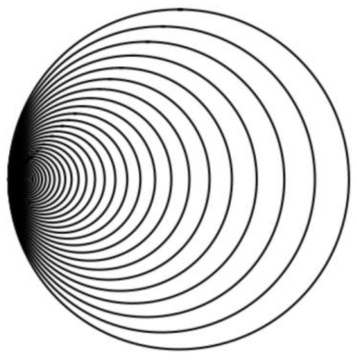
Results For Figure1 using n as 2,3 and 4 respectively.

Ismael Villlalobos
February 12, 2019
ID:80586103
Professor Fuentes

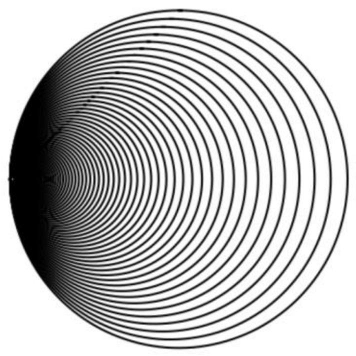
Figure 2



draw_circles(ax, 25, [100, 0], 100, .6)



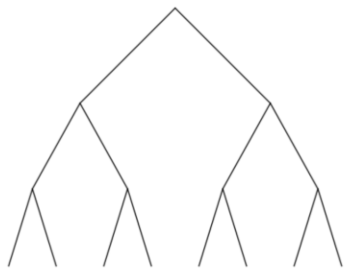
draw_circles(ax, 50, [100, 0], 100, .9)



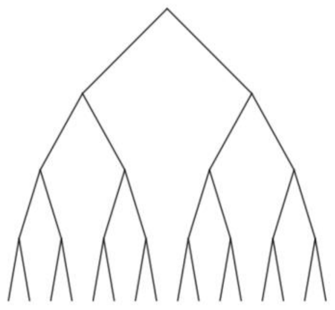
draw_circles(ax, 75, [100, 0], 100, .95)

Results For Figure2 using n as 25,50 and 75 respectively.

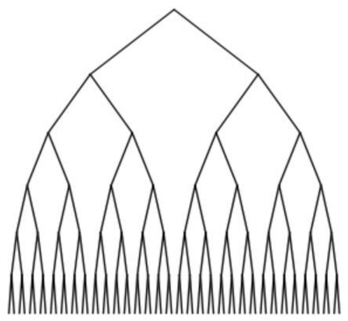
Figure 3



draw_tree(ax, line, x, y, 3)



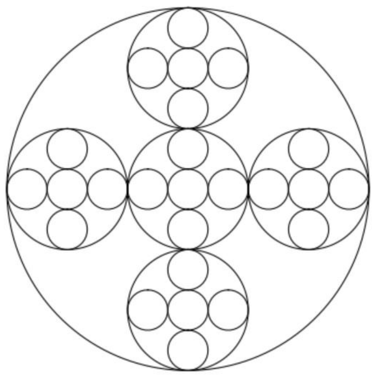
draw_tree(ax, line, x, y, 4)

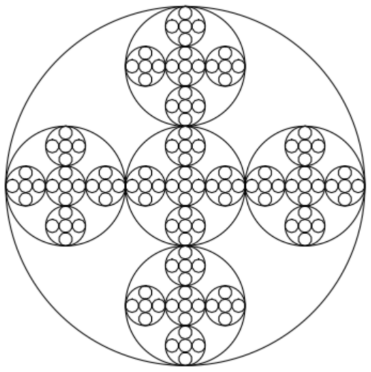


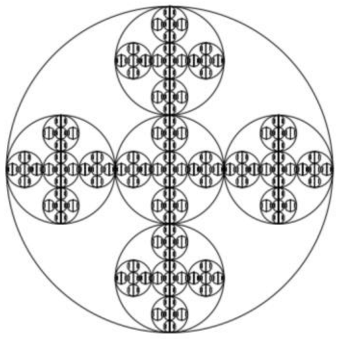
draw_tree(ax, line, x, y, 6)

Results For Figure3 using n as 3,4 and 6 respectively.

Ismael Villlalobos
February 12, 2019
ID:80586103
Professor Fuentes
Figure 4


draw_circles(ax, 3, [0, 0], 50)


draw_circles(ax, 4, [0, 0], 50)


draw_circles(ax, 5, [0, 0], 50)

Results For Figure3 using n as 3,4 and 5 respectively.

Conclusion

From this first challenging lab I gained much needed knowledge using python and it very extensive libraries. This allowed me to use my basic knowledge of recursion and challenged me to apply its principles into making fun images.

Appendix

Source Code provided by Professor Fuentes

draw_squares.py

```
import numpy as np
import matplotlib.pyplot as plt

def draw_squares(ax,n,p,w):
    if n>0:
        i1 = [1,2,3,0,1]
        q = p*w + p[i1]*(1-w)
        ax.plot(p[:,0],p[:,1],color='k')
        draw_squares(ax,n-1,q,w)

plt.close("all")
orig_size = 800
p = np.array([[0,0],[0,orig_size],[orig_size,orig_size],[orig_size,0],[0,0]])
fig, ax = plt.subplots()
draw_squares(ax,15,p,.8)
ax.set_aspect(1.0)
ax.axis('off')
plt.show()
fig.savefig('squares.png')
```

Ismael Villlalobos
February 12, 2019
ID:80586103
Professor Fuentes

draw_circles.py

```
import matplotlib.pyplot as plt
import numpy as np
import math

def circle(center,rad):
    n = int(4*rad*math.pi)
    t = np.linspace(0,6.3,n)
    x = center[0]+rad*np.sin(t)
    y = center[1]+rad*np.cos(t)
    return x,y

def draw_circles(ax,n,center,radius,w):
    if n>0:
        x,y = circle(center,radius)
        ax.plot(x,y,color='k')
        draw_circles(ax,n-1,center,radius*w,w)

plt.close("all")
fig, ax = plt.subplots()
draw_circles(ax, 50, [100,0], 100,.9)
ax.set_aspect(1.0)
ax.axis('off')
plt.show()
fig.savefig('circles.png')
```