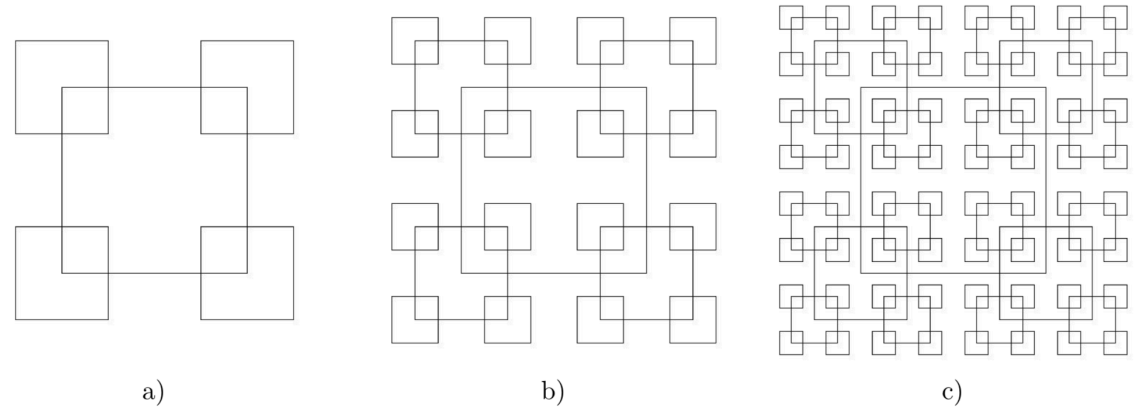
Maria F. Corona Ortega
ID: 80560734
E-Mail: mfcoronaortega@miners.utep.edu
CS 2302 | MW 1:30-2:50 am
Instructor: Dr. Olac Fuentes
TA: Anindita Nath

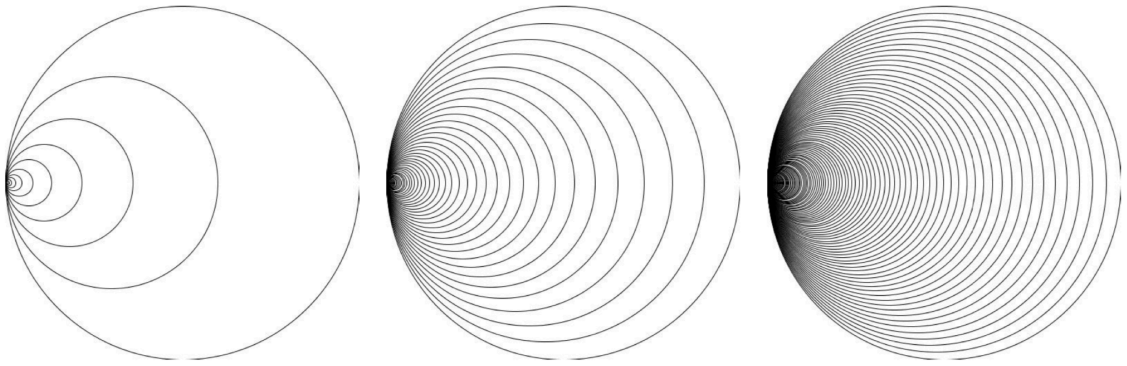
CS 2302 Lab 1 Report

Introduction

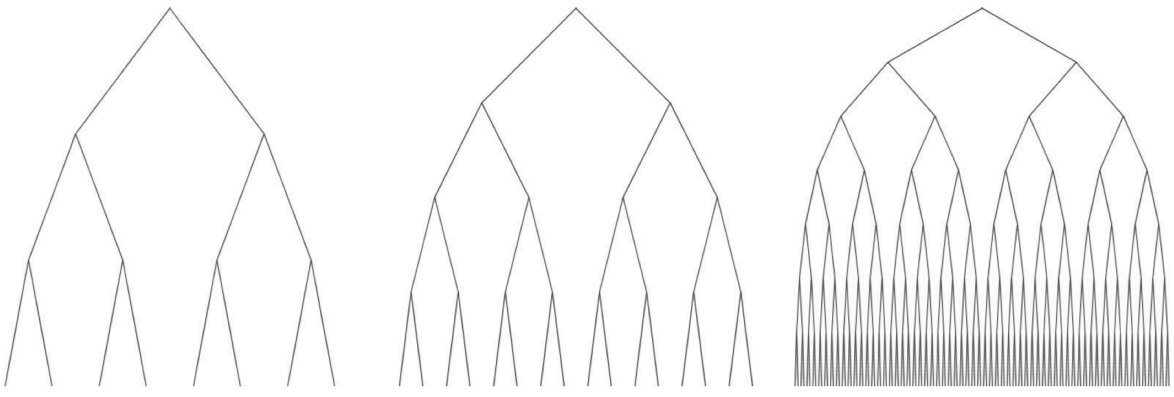
Though the use of recursion we are attempting to reproduce the following images:



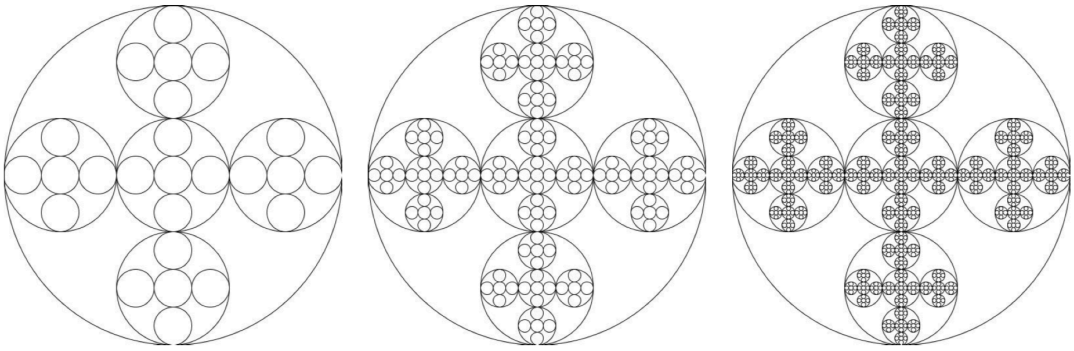
a) b) c)



a) b) c)



a) b) c)



a) b) c)

Proposed Solution

I have grouped the figures with similar shapes that will use similar code together. Therefore figure 2 and 4 share code. Figure 1 shares code with the given example and figure 3 is independent.

- **Task 1 (Figure 1):** First task asks to recursively draw squares in order to duplicate original configuration around the corners of the previous. Four squares need to be drawn around a center square per call. I will begin by having the initial five squares drawn out then with each recursive call draw out the next four squares each time about 1/3 our the size of the previous one placed directly on every corner.
- **Task 2 (Figure 2):** Second task is to recursively draw a circle each consecutively smaller by a constant. the center of each circle changes since the edges with each recursive call remain conjoined at the x-axis.
- **Task 3 (Figure 3):** Third task is to draw a tree, with each side being proportionally smaller depending on the number of levels. For example if there are 3 levels, level one's angle would be narrower that that of a tree with 5 or 6 levels since more splits need to fit within the root's initial angle.
- **Task 4 (Figure 4):** Fourth task is to create a circle with five circles within. These circles are to be directly aligned and the same size along the x and y axis. Middle circle is always concentric and the radius is modified by one sixth. With each recursive call the figure is repeated concentrically within each initial circle and so-on.

Methods

Part 1 (DrawCircles): ***circle, draw_circles, figure2, figure4***

- ***circle* -** The circle method was pre-programmed and given to us in order to facilitate plotting. This method takes a center and radius as a parameter center being the

coordinates of a circle and the radius being the size. Returning variables x and y it helps us plot a circle in a given place with a given size.

- ***figure2:*** Using parameters *ax* which is the figure in which to plot. *n* which determines the number of iterations. *center* which includes the initial coordinates for the circle along with *radius* which determines the size. *w* is a constant variable used in order to reduce the size of each iteration equally. The difference between this figure and the previous is actually when it comes to plotting. when it comes to variable x which is passed in through the *circle* method, does plot the circle "+*radius."* This does shift the circles to the left. Recursively if *n* is not zero each call plots a circle in which we change radius by constant *w*, circle int this case does not remain concentric every call.

- ***figure4:*** Using parameters *ax* which is the figure in which to plot. *n* which determines the number of iterations. *center* which includes the initial coordinates for the circle along with *radius* which determines the size. *w* is a constant variable used in order to reduce the size of each iteration equally. This figure plots a total of 6 circles a large on and 5 smaller ones that fit in the shape of a cross. When it comes to variable x which is passed in through the *circle* method, does plot the circle in a variation go modifications to the left, right or top of the initial concentric smaller circle. Recursively if *n* is not zero each call (in this case 6 recursive calls) plots five smaller circles along with the original in which we change radius by constant *w* or in this case the radius as well so that they remain proportionate. Only middle circles remain concentric.

Main method uses standard plotting code which we first see in the given source code and a single call to each figure.

Part 2 (DrawTrees) : *figure3*

- ***figure3:*** Using parameters *ax* which is the figure in which to plot. *n* which determines the number of iterations. *p* is a two-dimensional array which contains the given coordinates for all x and y values . *w* is a constant variable used in order to reduce the size of each iteration equally. This figure plots a total of two lines which meet on top. We use an array of indices in order to modify previous array all together. Recursively if *n* is not zero each call plots two lines per existing line which meet at the middle these are modified in size by constant *w* so that they remain proportionate.

Part 3 (DrawSquares) : *figure1*

- ***figure3:*** Using parameters *ax* which is the figure in which to plot. *n* which determines the number of iterations. *p* is a two-dimensional array which contains the given coordinates for all x and y values which in this case plot a square. *w* is a constant

variable used in order to reduce the size of each iteration equally. This figure plots a total of five squares. We use an array of indices in order to modify previous array all together. I created four arrays which all get modified (*q, q1-q4*) Recursively if *n* is not zero each call plots four squares per existing square (total of 4 recursive calls) which are located at the corners these are modified in size by constant *w* so that they remain proportionate.

Experimental Results:

Figure 1

n	w
1	0.5
2	0.5
3	0.5
4	0.5

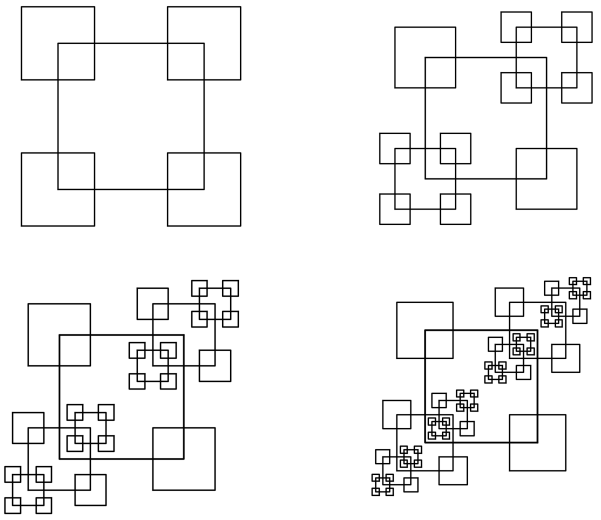


Figure 2

n	w
1	0.5
5	0.5
50	0.9
100	0.95

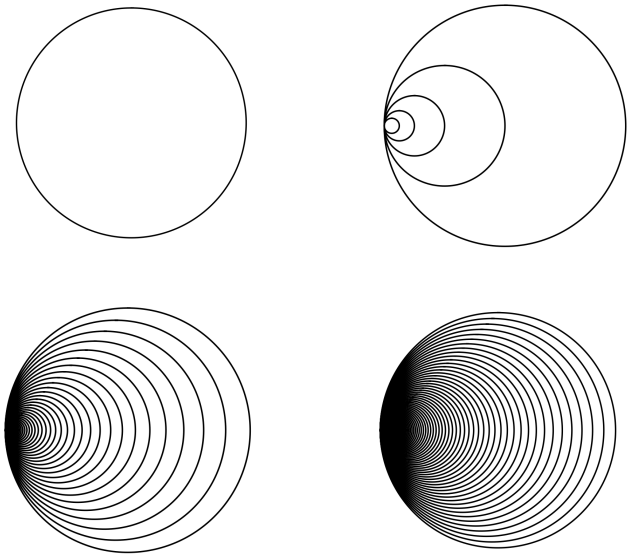


Figure 3

n	w
1	0.3
2	0.3
3	0.3
4	0.3

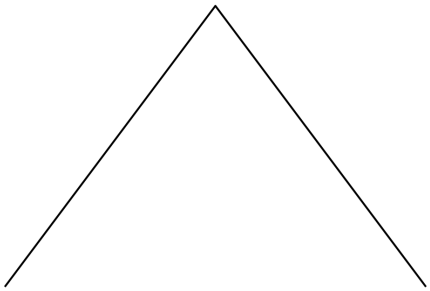
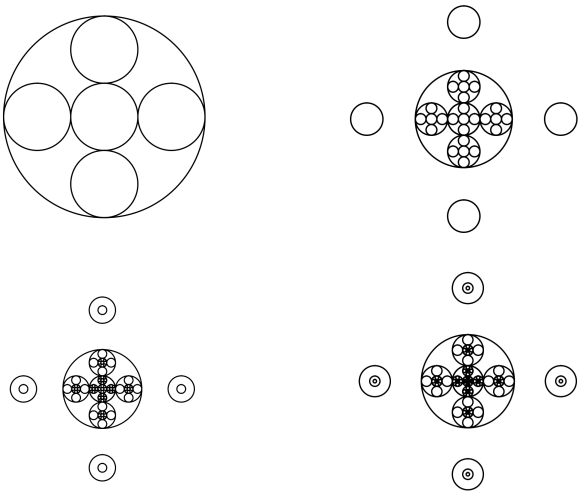


Figure 4

n	w
1	0.3
2	0.3
3	0.3
4	0.3



Conclusion

In conclusion, from this lab I have learned and experienced how recursion works. Even though my approaches might not of been implemented correctly, I did for the most part was able to make progress with each recursive call. I explored the rules of recursion on my best attempt to implement the algorithm correctly. In the future I look forward to perfecting recursion and being able to use it to approach most problems in which it can apply as a viable solution.

Appendix

```
#Course: CS 2302 Data Structures | Spring 2019
#Author: Maria Fernanda Corona Ortega
#Assignment: Lab 1 | Part 1 Figures 2 and 4
#Instructor: Olac Fuentes
#Purpose of Code: The purpose of this code is to duplicate images
presented
#through plotting and recursion
#Last Modification: 03/09/2019 8:27pm

mport 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 figure2(ax,n,center,radius,w):
    if n>0:
        x,y = circle(center,radius)
        ax.plot(x+radius,y,color='k')
        figure2(ax,n-1,center,radius*w,w)

plt.close("all")
fig, ax = plt.subplots()
figure2(ax, 100, [100,0], 100,.9)
ax.set_aspect(1.0)
ax.axis('off')
plt.show()
fig.savefig('figure2.png')


def figure4(ax,n,center,radius,w):

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

```
        ax.plot(x*w,y*w+radius-radius*w,color='k')
        figure4(ax,n-1,center,radius*w,w)
        figure4(ax,n-1,[radius+radius,0],radius*w,w)
        figure4(ax,n-1,[radius-radius/w,0],radius*w,w)
        figure4(ax,n-1,[0,radius+radius],radius*w,w)
        figure4(ax,n-1,[0,radius-radius/w,],radius*w,w)
        ax.plot(x,y,color='k')


plt.close("all")
fig, ax = plt.subplots()
figure4(ax, 3, [0,0], 100,1/3)
ax.set_aspect(1.0)
#ax.axis('off')
plt.show()
fig.savefig('figure4.png')

#Course: CS 2302 Data Structures | Spring 2019
#Author: Maria Fernanda Corona Ortega
#Assignment: Lab 1 | Part 2 Figure 3
#Instructor: Olac Fuentes
#Purpose of Code: The purpose of this code is to duplicate images
#presented
#through plotting and recursion
#Last Modification: 03/09/2019 8:27pm

import numpy as np
import matplotlib.pyplot as plt

def figure3(ax,n,p,w):
    if n>0:
        index = [0,1,2]
        q = p[index]/2+p[index]
        ax.plot(p[:,0],p[:,1],color='k')
        figure3(ax,n-1,q,w)


plt.close("all")
size = 3
levels = 5
p = np.array([[-size,-size-(size/size)],[0,0],[size,-size-(size/
size)]])

fig, ax = plt.subplots()

figure3(ax,levels,p,1/3)
```

```
ax.set_aspect(1.0)
ax.axis('off')
plt.show()
fig.savefig(‘figure3.png')

#Course: CS 2302 Data Structures | Spring 2019
#Author: Maria Fernanda Corona Ortega
#Assignment: Lab 1 | Part 3 Figures 1
#Instructor: Olac Fuentes
#Purpose of Code: The purpose of this code is to duplicate images
presented
#through plotting and recursion
#Last Modification: 03/09/2019 8:27pm
import numpy as np
import matplotlib.pyplot as plt

def figure1(ax,n,p,p1,p2,p3,p4,w): #FIXME
    if n>0:
        index = [1,2,3,0,1]
        q = p[index]
        q1 = p1[index]*w + 2*w
        q2 = w*p2[index]+ 2*w
        q3 = w*p3[index]+ 2*w
        q4 = w*p4[index]+ 2*w
        ax.plot(p[:,0],p[:,1],color='k')
        ax.plot(p1[:,0],p1[:,1],color='k')
        ax.plot(p2[:,0],p2[:,1],color='k')
        ax.plot(p3[:,0],p3[:,1],color='k')
        ax.plot(p4[:,0],p4[:,1],color='k')
        figure1(ax,n-1,q,q1+4*w,q2+4*w,q3+4*w,q4+4*w,w)
        figure1(ax,n-1,q,q1-4*w,q2-4*w,q3-4*w,q4-4*w,w)
#        figure1(ax,n-1,q,q1,q2,q3,q4,w)
#        figure1(ax,n-1,q,q1,q2*w,q3*w,q4*w,w)

plt.close("all")
size = 4

p = np.array([[0,0],[0,size],[size,size],[size,0],[0,0]])
p1 = np.array([[-size/4,-size/4],[-size/4,size/4],[size/4,size/4],
[size/4,-size/4],[-size/4,-size/4]])
p2 = np.array([[size-size/4,-size/4],[size-size/4,size/4],[size+size/
4,size/4],[size+size/4,-size/4],[size-size/4,-size/4]])
p3 = np.array([[-size/4,size-size/4],[-size/4,size+size/4],[size/
4,size+size/4],[size/4,size-size/4],[-size/4,size-size/4]])
```

```
p4 = np.array([[size-size/4,size-size/4],[size-size/4,size+size/4],
[size+size/4,size+size/4],[size+size/4,size-size/4],[size-size/4,size-
size/4]])


fig, ax = plt.subplots()
figure1(ax,2,p,p1,p2,p3,p4,.5)
ax.set_aspect(1.0)
ax.axis('off')
plt.show()
fig.savefig('fig1.png')
```

Academic Honesty Certification

I certify that this project is entirely my own work. I wrote, debugged, and tested the code being presented, performed the experiments, and wrote the report. I also certify that I did not share my code or report or provided inappropriate assistance to any student in the class.

Maria F. Corona Ortega

09/09/2017