Lab #1 Report

**Introduction:** By using recursive methods I will be drawing figures that compress on themselves and create interesting patterns.

**Proposed solution design and implementation/Experimental Results:**

* Figure 1- The way I approached the recursive squares was by creating smaller squares and assigning them to each vertex of the square.

\*Create an array with the vertex coordinates to plot your original square to be passed to the recursive method drawSquares().

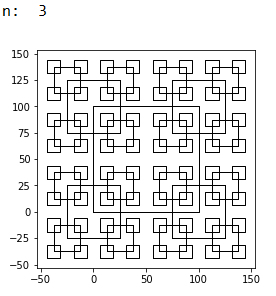
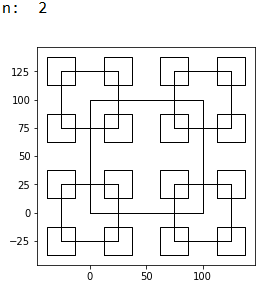
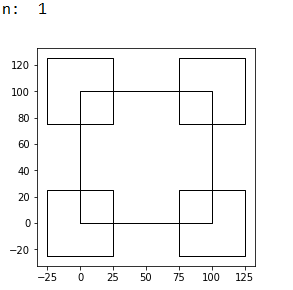
\*Create an array with the index’s designated to each coordinate to use to plot the square and not use a loop.

\*Plot the original Square.

\*Divide the coordinates of the original square by 2 to create smaller squares to assign to the vertex of the original square.

\*Create new arrays (RU,RB,LU,LB) from the Original square array to be modified and go into each vertex of the original square.

\*Lastly, call the recursive calls and sent the new smaller squares instead of the original squares.



* Figure 2- Every time the recursive call is made, I reduced the radius and moved the circle to the left.

\*Make a recursive method with 2 user input variables, the center coordinates, the radius and a temporary number that will be replaced later to be used.

\*Make 3 if statement that will decide which way the recursive call is going to go.

\*The first if-statement is when the user input is that of 9 or less.

1. Set the temporary number previously mentioned as ‘w’=.5 to be used to reduce the circle.

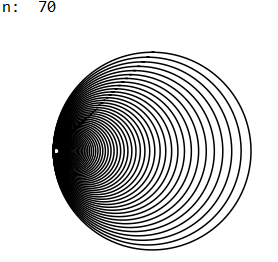
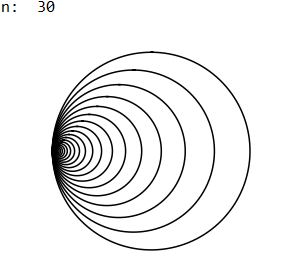
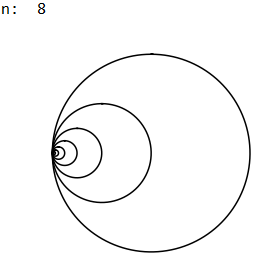
2. Use the circle() method to receive the x and y coordinates that will be used to plot the circle.

3. Plot the circle by changing the x by adding the radius and leaving y as it is.

4. Call the recursive method and change the radius by multiplying it with ‘w’.

\*The second if-statement is when the user input is in between 10 and 39 and ‘w’ will be set to .82 and repeat the previous if-statement steps.

\*The third if-statement is when the user input in bigger or equal to 40 and ‘w’ will be set to .9 and repeat the previous if-statement steps.



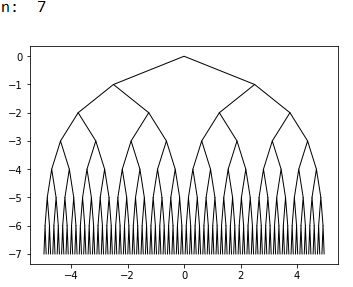
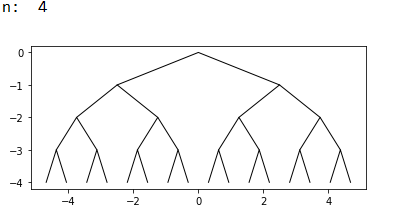
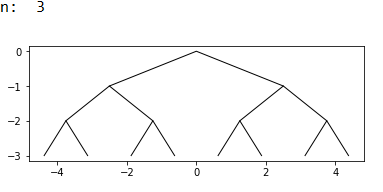
* Figure 3- I set main coordinates to be passed and for every recursive call I would create two new coordinates that would not touch in its ‘x’ coordinate and ‘y’ would lower by one.

\*For the recursive call I passed the number 5 to be used to compute the difference between the x coordinates distances and I passed the main coordinates separately(x and y).

\*In the method first you get your new coordinates. Get the divided coordinate of ‘div’ by dividing ‘I’(the number 5) by 2, then to get ‘x1’ you added ‘div’ to ‘x’ and to get ‘x2’ you subtracted ‘div’ to x, and to get ‘yi’ you just subtract 1 every recursive call.

\*We would put the new coordinates in an array, called root, in reference to the x coordinate, that way we plot from left to right or right to left and we get a nice branch.

\*After plotting the array, we call two recursive call each with its respective ‘x’ coordinate.



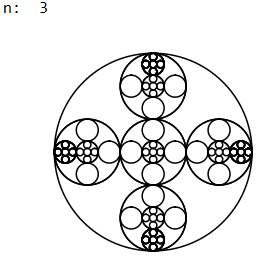
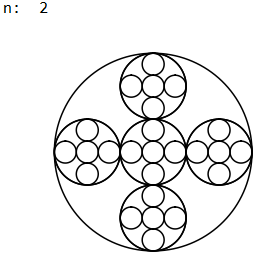
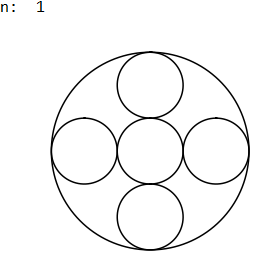
* Figure 4: I attempted to create the figure by dividing the radius of each recursive circle by one third and then I would move its center to alignment them to correspond to the figures. This did not work.

\*I passed the same coordinates as the method in figure 2 with the exemption of only passing one ‘n’ and by passing a temporary value that will be changed

\*First calculation in the method are used to set up the new coordinates that will be passed recursively, these coordinates will move the smaller circles to their corresponding space.

\*Then I would get the coordinates of the circles using the circle() method and I would manipulate the radius parameter accordingly to set the circle where I wanted it.

\*Then I would plot the smaller circles and then I would make the recursive call with a modified center and radius.



**Conclusion:** This is my first-time using code and I learned how to use arrays to plot coordinates to draw figures. I used more variables in my recursive methods than I anticipated, I used them to get around the coordinate manipulation to get my desired figures

**Source Code:**

#Course: CS 2302: Data Structures

#Author: Bruno Roque

#Assignment: Lab #1

#Instructor: Olac Fuentes

#Teacher Asistant: Anindita Nath

#Last Modification: Feb 14, 2019

#Purspose:Drawing Figures Using Recursion

import numpy as np

import matplotlib.pyplot as plt

import math

def drawSquares(ax,n,p):

#This method creates squares with smaller squares in each vertex

if n>0:#ends the recurcion call once it reaches the conditional

#print(n)

#print(p)

a1=[0,1,2,3,0]

ax.plot(p[:,0],p[:,1], linewidth=1, color='k')#Prints the original square

p=p/2#Divide the original square's parameters by half to us on the

#vertex of the original square

RU=p[a1]+75#Moves the reduced square to the upper right vertex

#print(q1)

ax.plot(RU[:,0],RU[:,1], linewidth=1, color='k')

LU=p[a1]+[-25,75]#Moves the reduced square to the upper left vertex

#print(q2)

ax.plot(LU[:,0],LU[:,1], linewidth=1, color='k')

RB=p[a1]+[75,-25]#Moves the reduced square to the bottom right vertex

#print(q3)

ax.plot(RB[:,0],RB[:,1], linewidth=1, color='k')

LB=p[a1]-25#Moves the reduced square to the bottom left vertex

#print(q4)

ax.plot(LB[:,0],LB[:,1], linewidth=1, color='k')

drawSquares(ax,n-1,RU)#Continues the recurcion call by making the upper right square the original square

drawSquares(ax,n-1,LU)#Continues the recurcion call by making the upper left square the original square

drawSquares(ax,n-1,RB)#Continues the recurcion call by making the bottom right square the original square

drawSquares(ax,n-1,LB)#Continues the recurcion call by making the bottom left square the original square

plt.close("all")

origSize=100

p=np.array([[0,0],[0,origSize],[origSize,origSize],[origSize,0],[0,0]])

fig, ax= plt.subplots()

try:

n = int(input('Please enter the number of iterations for figure 1: '))

except ValueError:

print('Error')

n= int(input('Please enter a valid number: '))

print('n: ',n)

drawSquares(ax,n,p)#Original recursive call

ax.set\_aspect(1.0)

plt.show()

fig.savefig('squares.png')

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,i,center,radius,w):

#This method creates smaller circles and shifts the center of each smaller circle

if n>0:

if i<10:#Depending on the user input the recursive method will change

w=.5

x,y = circle(center,radius)#Method call used to get the x and y cordinates

ax.plot(x+radius,y,color='k')#Plots the circle

draw\_circles(ax,n-1,i,center,radius\*w,.5)#Recursive method that changes te radius

if i>=10 and i<40:

w=.82

x,y = circle(center,radius)

ax.plot(x+radius,y,color='k')

draw\_circles(ax,n-1,i,center,radius\*w,.82)

if i>=40:

w=.95

x,y = circle(center,radius)

ax.plot(x+radius,y,color='k')

draw\_circles(ax,n-1,i,center,radius\*w,.95)

plt.close("all")

fig, ax = plt.subplots()

try:

n = int(input('Please enter the number of iterations for figure 2: '))

except ValueError:

print('Error')

n= int(input('Please enter a valid number: '))

print('n: ',n)

draw\_circles(ax,n,n, [100,0], 100,0)#Original recursive method

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('circles.png')

def recursiveRoots(ax,i,x,y,n):

#This method creates a root like diagram

if n>0:

#These are the equations to create new cordinates to be

#plotted and passed to the recursive methods

div=i/2

x1=x+div

yi=y-1

x2=x-div

root=np.array([[x1,yi],[x,y],[x2,yi]])#Creates an array to be used for plotting

ax.plot(root[:,0],root[:,1], linewidth=1, color='k')#Plots the array

recursiveRoots(ax,div,x1,yi,n-1)#Recursive method that passes the new coordinates

recursiveRoots(ax,div,x2,yi,n-1)

plt.close("all")

fig, ax= plt.subplots()

try:

n = int(input('Please enter the number of iterations for figure 3: '))

except ValueError:

print('Error')

n= int(input('Please enter a valid number: '))

print('n: ',n)

recursiveRoots(ax,5,0,0,n)#Original recursive method

ax.set\_aspect(1.0)

plt.show()

fig.savefig('squares.png')

def draw\_inside\_circles(ax,n,center,radius,w,prevM):

#This method makes smaller circles within an original circle in a cross formation

if n>0:

#These functions create the new coordinates to be used in the recursive methods

m=radius\*2/3

LC=[radius\*1/3,0]

RC=[100+(m+prevM),0]

UC=[100,m+prevM]

DC=[100,-m-prevM]

x,y = circle(center,radius)

ax.plot(x,y,color='k')

x,y = circle(center,radius/w)

ax.plot(x,y,color='k')

draw\_inside\_circles(ax,n-1,center,radius/w,w,m+prevM)

x,y = circle(center,radius/w)

ax.plot(x+m,y,color='k')

draw\_inside\_circles(ax,n-1,LC,radius/w,w,m+prevM)

x,y = circle(center,radius/w)

ax.plot(x-m,y,color='k')

draw\_inside\_circles(ax,n-1,RC,radius/w,w,m+prevM)

x,y = circle(center,radius/w)

ax.plot(x,y+m,color='k')

draw\_inside\_circles(ax,n-1,UC,radius/w,w,m+prevM)

x,y = circle(center,radius/w)

ax.plot(x,y-m,color='k')

draw\_inside\_circles(ax,n-1,DC,radius/w,w,m+prevM)

plt.close("all")

fig, ax = plt.subplots()

try:

n=int(input('Please enter the number of iterations for figure 4: '))

except ValueError:

print('Error')

n= int(input('Please enter a valid number: '))

print('n: ',n)

draw\_inside\_circles(ax,n, [100,0], 100,3,0)#Original method, [100,0] is the center coordinate

#100 is the radius, 3 is 'w' which will be the number that will be used to decrease the circles

#0 is precM wich is used with m for the coordinate equations

ax.set\_aspect(1.0)

ax.axis('off')

plt.show()

fig.savefig('circles.png')