Gianna Julio

CMPT435 - Algorithm Analysis and Design
Assignment 0

lineArt.py

```python
#################################
# Gianna Julio
# Period 7-8 HCP
# Line Art Program
# Purpose: draws line art based on user's input
#################################

import pygame, sys, math, random

#initialize game engine
pygame.init()

#open and set window size
w = 800
h = 800
size = (w,h)
surface = pygame.display.set_mode(size)

#set title bar
pygame.display.set_caption("Line Art!")

#color constants
BLACK = (  0,   0,   0)
WHITE = (255, 255, 255)
RED =   (255,   0,   0)
GREEN = (  0, 255,   0)
BLUE =  (  0,   0, 255)
YELLOW= (255, 255, 0  )

#----------------------Functions:

def randomColor():
    r = random.randint(0, 256)
    g = random.randint(0, 256)
    b = random.randint(0, 256)
    COLOR = (r, g, b)
    return COLOR

def drawLines(xAnchor, yAnchor, COLOR, numSegments, quadrant):
    gap = w/numSegments

    for i in range(numSegments):
        pygame.draw.line(surface, COLOR, xAnchor, yAnchor, 1)

        if quadrant == 1:
            xAnchor[0] += gap
            yAnchor[1] += gap
        elif quadrant == 2:
            xAnchor[0] -= gap
            yAnchor[1] += gap
        elif quadrant == 3:
            xAnchor[0] -= gap
            yAnchor[1] -= gap
        elif quadrant == 4:
            xAnchor[0] += gap
            yAnchor[1] -= gap

def getIntBetween(message, low, high):
    num = low - 1

    while(num > high or num < low):     # continue while number is out of range
        try:
            num = int(input(message + " between " + str(low) + " and "+ str(high) + ": ")
                )
            if num > high or num < low:
                print("Error - number out of range. Try again.")

        except ValueError:
            print("Error - invalid number. Try again.")

    return num

#---------------------Main Program Loop:

def main():

    numSegments = getIntBetween("Enter a number", 10, 100)
    COLOR1 = randomColor()
    COLOR2 = randomColor()
    COLOR3 = randomColor()
    COLOR4 = randomColor()

    while(True):
        for event in pygame.event.get():
            if (event.type == pygame.QUIT or (event.type == pygame.KEYDOWN and event.key
                == pygame.K_ESCAPE)):
                pygame.quit()
                sys.exit()

                #game logic:

        #set background color
        surface.fill(BLACK)

        #drawing code:
        drawLines([0, h], [0, 0], COLOR1, numSegments, 1)
        drawLines([w, h], [w, 0], COLOR2, numSegments, 2)
        drawLines([w, 0], [w, h], COLOR3, numSegments, 3)
        drawLines([0, 0], [0, h], COLOR4, numSegments, 4)
```

```
98
99          #last line - update screen
100         pygame.display.update()
101
102 main()
```

Description

I made this simple line art program in my high school intro course using python in 2017. It takes user input and draws a series of lines in randomly chosen colors to create simple geometric line art using pygame graphics.

Inputs

The program asks the user to enter a number between 10 and 100 to determine how many lines will be drawn on each edge. If the user enters a number outside of that range or a non-numerical character, the program gives the user an error message and prompts them to try again using a while loop until a valid answer is entered. The code to get this input from the user as well as the error catching is contained in the getIntBetween function on line 58.

Outputs

The output of this program is a pygame window that displays the lines of different colors to make the line art on a black background. The window can be closed with the escape key or the X button in the top right corner. The window settings are imported initialized in lines 8-20 and updated in the main function starting on line 74.

Procedure

The random colors are generated by three randomly generated colors assigned to the rgb values in the function randomColor on line 32. The math for the line drawing is relatively simple. A gap value is computed based on how many lines the user input, and a for loop draws a line from a starting and ending point that are incremented and decremented as needed each iteration. The function drawLines on line 39 contains all of the drawing and math.