**Name: Session:**

**Programming I**

**Lab Exercise 11.15.2019**

**Creating a Paint Program**

In this exercise, we will be creating a basic painting program. To be consistent with other programming languages, we will start by creating a main function. In this activity in addition to drawing, we will be looking for two specific Pygame events: MOUSEMOTION and KEYDOWN. The MOUSEMOTION event will allow us to draw lines (very short ones) and the KEYDOWN event will allow us to change drawing parameters.

**When your program is complete and working, print a screen shot of your name painted on the canvas and attach it to this sheet.**

Start by importing the modules we will require:

import pygame, sys

Now create a main function:

#Initialize Pygame

pygame.init()

#Setup the screen

screen = pygame.display.set\_mode((1024, 768))

pygame.display.set\_caption("Paint: (r)ed, (g)reen, (b)lue, (w)hite, " +

"blac(k), (1-9) width, (c)lear, (s)ave, "

+ "(l)oad, (q)uit")

#Create a background to draw on

background = pygame.Surface(screen.get\_size())

#Make background white

background.fill((255, 255, 255))

#Create a clock

clock = pygame.time.Clock()

#Initialize variables

keepGoing = True

lineStart = (0, 0)

drawColor = (0, 0, 0)

lineWidth = 3

#Start game loop

while keepGoing:

#Set clock rate

clock.tick(30)

#Check events

for event in pygame.event.get():

#QUIT event

if event.type == pygame.QUIT:

keepGoing = False

#MOUSEMOTION event

elif event.type == pygame.MOUSEMOTION:

#Store mouse position in lineEnd

lineEnd = pygame.mouse.get\_pos()

#Draw a line if left mouse button is pressed

if pygame.mouse.get\_pressed() == (1, 0, 0):

pygame.draw.line(background, drawColor, lineStart,

lineEnd, lineWidth)

#Store lineEnd in lineStart

lineStart = lineEnd

#KEYDOWN event

elif event.type == pygame.KEYDOWN:

#place myData into a tuple to be passed to checkKeys

myData = (event, background, drawColor, lineWidth, keepGoing)

#Call checkKeys and store returned tuple in myData

myData = checkKeys(myData)

#unpack myData tuple

(event, background, drawColor, lineWidth, keepGoing) = myData

#BLIT background

screen.blit(background, (0, 0))

#Call showStats and store returned result in myLable

myLabel = showStats(drawColor, lineWidth)

#BLIT myLabel

screen.blit(myLabel, (850, 730))

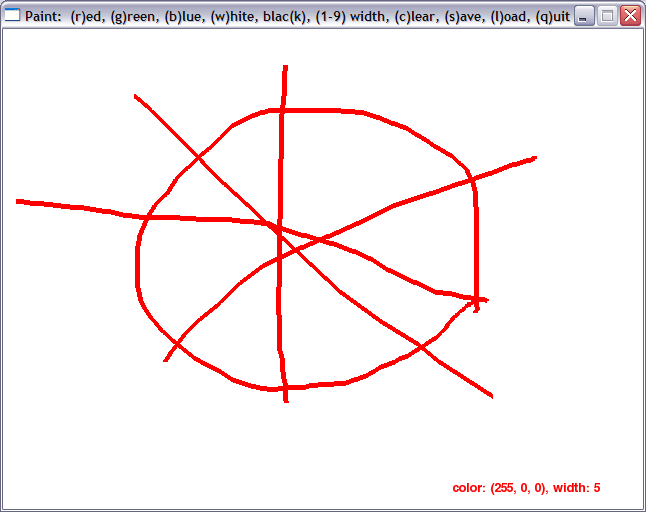
#flip the display

pygame.display.flip()

#End program

pygame.display.quit()

sys.exit()



The above screen is what we will be creating. In addition to the drawing surface, there is an information label to tell us the status of our drawing color as well as the width of our paintbrush. The caption of the window gives us information on how to use the program.

Next we must add a function to process the keys. We will call this function checkKeys.

def checkKeys(myData):

#Define filename to store bitmapped image

filename = 'myPainting.bmp'

The checkKeys function is passed a variety of data which must be extracted into a tuple as such:

#extract the data

(event, background, drawColor, lineWidth, keepGoing) = myData

This takes the data from myData and places it into the appropriate fields of the tuple.

Next we need to add the code to handle the KEYDOWN event for the following keys:

‘q’, ‘c’, ‘s’, ‘l’, ‘r’, ‘g’, ‘w’, ‘b’, ‘k’, ‘1’, ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’, ‘9’

if event.key == pygame.K\_q:

#quit

keepGoing = False

elif event.key == pygame.K\_c:

#clear screen

background.fill((255, 255, 255))

elif event.key == pygame.K\_s:

#save picture

pygame.image.save(background, filename)

elif event.key == pygame.K\_l:

#load picture

background = pygame.image.load(filename)

#colors

elif event.key == pygame.K\_r:

#red

drawColor = (255, 0, 0)

elif event.key == pygame.K\_g:

#green

drawColor = (0, 255, 0)

elif event.key == pygame.K\_w:

#white

drawColor = (255, 255, 255)

elif event.key == pygame.K\_b:

#blue

drawColor = (0, 0, 255)

elif event.key == pygame.K\_k:

#black

drawColor = (0, 0, 0)

#line widths

elif event.key == pygame.K\_1:

lineWidth = 1

#you need to add the line width code for keys 2 – 9

Now you need to return the data from the checkKeys function using the following code:

#return all values

myData = (event, background, drawColor, lineWidth, keepGoing)

return myData

Finally, we need to write a showStats function that will return a summary of color and line width information that can be displayed to the user.

def showStats(drawColor, lineWidth):

#shows the current statistics (lineWidth and drawColor)

myFont = pygame.font.SysFont("None", 20)

stats = "color: %s, width: %d" % (drawColor, lineWidth)

statSurf = myFont.render(stats, 1, (drawColor))

return statSurf

**Lastly, be sure to call main()**