**Name: Session:**

**Programming I**

**Simple Image Processing in Python**

**Lab Exercise 12.12.2022**

In this activity you will learn about the science of image processing. You will be working with an image of a cat named *Smokey*. Make sure the file *“smokey.gif”* is stored in the same folder as your *imageTest.py* file. You will create 3 very basic image processing procedures (programmers call them algorithms). These algorithms are blackAndWhite, grayscale, and edgeDetection. This activity will also introduce you to the concept of functions.

1. Start IDLE and create a new window (file).
2. Save your file as imageTest.py
3. Add the following line of code.

*from images import Image*

1. This will import the Image class from the image module which I have added to your Python installation.
2. Now write a main function that will call each of the 3 algorithms. This function creates an image object based on the filename. It then draws the unaltered image, the black and white image, the grayscale image and finally the image edges.

def main(filename = "smokey.gif"):

image = Image(filename)

image.draw()

blackAndWhite(image)

image.draw()

grayScale(image)

image.draw()

image = detectEdges(image,10)

image.draw()

1. Next you will create the blackAndWhite function. This function defines a black and white pixel. It then retrieves the color information from each pixel in the image and finds an “average color”. If the average of a pixel is above a certain threshold, it is set to black and if not is set to white.

def blackAndWhite(image):

blackPixel = (0,0,0)

whitePixel = (255,255,255)

for y in range(image.getHeight()):

for x in range(image.getWidth()):

(r,g,b) = image.getPixel(x,y)

average = (r+g+b)/3

if average < 128:

image.setPixel(x,y,blackPixel)

else:

image.setPixel(x,y,whitePixel)

1. Now create a grayscale function. The function retrieves the color information from each pixel in the image and calculates the weighted value of each color and calculates a “grayscale value” for each pixel.

def grayScale(image):

for y in range(image.getHeight()):

for x in range(image.getWidth()):

(r, g, b) = image.getPixel(x, y)

r = int(r \* 0.299)

g = int(g \* 0.587)

b = int(b \* 0.114)

lum = r + g + b

image.setPixel(x, y, (lum, lum, lum))

1. Now create an edgeDetection function. The function defines a black and white pixel and then makes a copy of itself. The function then compares each pixel and adjacent pixel with some threshold amount to determine if there is an edge. If there is, it replaces the new image with a black pixel.

def detectEdges(image, amount):

def average(triple):

(r, g, b) = triple

return (r + g + b) / 3

blackPixel = (0, 0, 0)

whitePixel = (255, 255, 255)

new = image.clone()

for y in range(image.getHeight() - 1):

for x in range(1, image.getWidth()):

oldPixel = image.getPixel(x, y)

leftPixel = image.getPixel(x - 1, y)

bottomPixel = image.getPixel(x, y + 1)

oldLum = average(oldPixel)

leftLum = average(leftPixel)

bottomLum = average(bottomPixel)

if abs(oldLum - leftLum) > amount or \

abs(oldLum - bottomLum) > amount:

new.setPixel(x, y, blackPixel)

else:

new.setPixel(x, y, whitePixel)

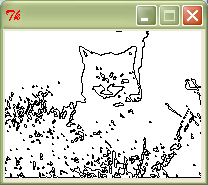
return new

1. Do not forget to call main…..
2. Now try experimenting with the Edge Detection Threshold in the call to the detectEdges function in main.

image = detectEdges(image,10)

Consider commenting out the calls to the other functions and placing this line of code in a loop.

1. Make a screen shot of each of your functions such as this, print them, attach to this sheet, and turn in.



**Optical Illusions**

In this activity you will have the turtle draw an optical illusion on your screen. In order to make this program quick and easy to write you will be using a special graphics module called *turtle.py* which will take the place of the turtle module built into Python. You need to do the following:

1. Create a folder on your desktop called MyPython.
2. In this folder you will store all of your Python source code.
3. Start IDLE (version 3.2)
4. Create a new window and save the file as illusion1.py in the MyPython folder on your desktop.
5. Type in the following program:

from turtle import \*

def main():

diamondshape = Shape("compound")

poly1 = ((-7,-7), (7,-7), (7,7), (7,-7))

diamondshape.addcomponent(poly1, "black")

poly2 = ((-7,-7), (-7,7), (7,7), (-7,7))

diamondshape.addcomponent(poly2, "white")

register\_shape("diamond", diamondshape)

bgcolor("gray55")

shape("diamond")

shearfactor(0.3)

pu()

ht()

tracer(False)

left(90)

for \_ in range(40):

fd(160)

stamp()

bk(160)

rt(9)

tilt(-73.3)

for \_ in range(32):

fd(125)

stamp()

bk(125)

rt(11.25)

dot(12)

goto(0, -270)

write("Stare at the dot, then lean forward and back!",

align="center",

font=("Courier",14,"bold"))

tracer(True)

return "DONE!"

if \_\_name\_\_ == "\_\_main\_\_":

msg = main()

print(msg)

mainloop()

**Nanotechnology Experiment**

You have a 1 inch cube of butter. Your task is to spread it over the entire football field. Assume the football field is 360 feet by 160 feet. You will cover the field by bisecting each face of the cube of butter to create 8 smaller cubes. You will repeat the process iteratively until you cover every square inch of the football field. For example, in the beginning, the cube will cover 1 square inch. After the first iteration, you will have 8 cubes with a side of ½ inch which cover ¼ square inches each thus 8 cubes will cover 2 square inches (8 \* ¼). After the second iteration you will have 64 cubes each with a side of ¼ inch which cover 1/16 inches. 64 cubes will therefore cover 4 square inches. You have the option of continuing this process longhand until the problem is solved or you may write a computer program to iteratively solve this problem. You solution must provide the answer to two questions.

1. How many cubes do you end up with?
2. What is the dimension of the final set of cubes (i.e. How thick is the butter spread on the field in nanometers?)?