**Outline**

Develop an understanding of how images and graphics are drawn and stored in a computer. Learn about the RGB colour space. Apply Python concepts related to lists and loops.

**Objectives**

* tbd

**Materials**

* tbd

**Level 1: RGB Color Space**

1. Create a new Repl for Python with Turtle.
   1. Copy and paste “Sample Program #1” from the listing at the end of this module.
   2. Run the program and examine the Turtle output  
      Red, dark red, green, dark green, blue, dark blue
2. Colours can be specified by using a combination of three numbers. These three numbers together define a “Pixel” point in a graphic image.
   1. What position is the number that controls the amount of red (r) in the pixel?
   2. What position is the number that controls the amount of green (g) in the pixel?
   3. What position is the number that controls the amount of blue (b) in the pixel?  
      (r,g,b)

red, green, blue

1. Colour number values can range from 0 to 255.
   1. What happens when the colour value is less than 255?
   2. What happens when the colour value is close to 0?

If it is less than 255 the colour of the circle becomes more dark.

If the value is close to 0 the colour becomes black.

1. Other shades of colours can be created using a combination of r,g,b number values.
   1. Create a pixel containing a shade of the colour orange.
   2. Create a pixel containing a shade of the colour yellow.
   3. Create a pixel containing a shade of your favorite colour.

Finished.

Orange= redColor = (255,101,10)

drawPixel(redColor)

Yellow= greenColor = (255,255,0)

drawPixel(greenColor)

**B**lue= blueColor = (80,40,266)

drawPixel(blueColor)

1. Black, white, and shades of grey are created using combinations of equal r,g,b number values.
   1. Create a completely white pixel.
   2. Create a completely black pixel.
   3. Create a pixel containing a shade of middle grey.

White= redColor = (255,255,255)

drawPixel(redColor)

grey= greenColor = (128,128,128)

drawPixel(greenColor)

black= blueColor = (0,0,0)

drawPixel(blueColor)

**Level 2: Resolution**

Level 2: Images Using Pixels

1. Download the image “Resoultion\_284x177.jpg” from Topic B folder in the class repository.

a. Open the image in a program like Paint or Photoshop. 

b. What is the size of this image? How many pixels does it contain?

c. Describe how the image looks (e.g. Can you see the pixels?)

d. Zoom in the view to enlarge the image

e. Describe how the image looks (e.g. Can you see the pixels?)

The size is 284 by 177, the amount of pixels in the picture is 50,976.

The pixels in the picture look like very small squares all put together.

The image looks like a bird with a hat on his head.

2. Download the image “Resoultion\_16x16.jpg” from Topic B folder in the class repository.

a. Open the image in a program like Paint or Photoshop.

b. What is the size of this image? How many pixels does it contain?

c. Describe how the image looks (e.g. Can you see the pixels?)

d. Zoom in the view to enlarge the image

e. Describe how the image looks (e.g. Can you see the pixels?)

The size of the picture is 16 by 16. The amount of pixels in the picture are 256.

The pixels in the picture are big and large squares all put together.

The picture looks like a robot bird.

3. Create a new Repl for Python with Turtle.

a. Copy and paste “Sample Program #2” from the listing at the end of this module.

b. Run the program and examine the Turtle output

c. Compare the program output to the “Resoultion\_16x16.jpg” image in question #2 above.

This picture has small circles instead of big larger squares. There white space in between the circles in the image. The picture does not look like bird with a flower on the top of his head.

4. Explain how the program code in lines 52 to 58 works. (i.e. The main program code.)

a. How the program prints out pixels to produce and 8 by 8 resolution image.

b. How the program decides which colour information to use for each pixel.

The way line 52 to 58 work is it the amount of pixels you want to display in the python turtle.

The program decides on the colour by the coordinates of the numbers.

5. Explain the purpose of the code in lines 12 to 21

a. How this code is related to the pixels produced by the main program.

b. The RGB value of the 19th pixel in the image

c. The RGB value of the pixel in the 5th column on the 4th row.

This is the main program because because this is where it decides where all the colours of the pictures are. The value of the 19th pixel is (28,28,12). The pixel of the 5th column 4th row is (154,140,22).

6. Modify the main program to print the image upside-down (i.e. pixels in reverse order).

a. Show your modified image to Mr. Nestor.

b. Explain your changes to the program code below.

pixelAddress += 1 to pixelAddress += -1

7. Modify the main program to print the image at a resolution of 12 by 4 pixels.

c. Show your modified image to Mr. Nestor.

d. Explain your changes to the program code below.

for row in range (12) :

for column in range(4) :

Level 3: Your Custom Image

1. Use and modify the sample pixel program code to create your own custom image.

a. Create a larger resolution image than provided in the sample.

b. Make sure the image is recognizable (or a clear pattern).

c. Show your image to Mr. Nestor.

]

(0,0,0),(0,0,0),(0,0,0),(0,0,0),(0,0,0),(0,0,0),(0,0,0),(0,0,0),

(0,0,0),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(0,0,0),

(0,0,0),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(0,0,0),

(0,0,0),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(0,0,0),

(0,0,0),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(0,0,0),

(0,0,0),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(0,0,0),

(0,0,0),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(255,255,255),(0,0,0),

(0,0,0),(0,0,0),(0,0,0),(0,0,0),(0,0,0),(0,0,0),(0,0,0),(0,0,0)

]

2. List and explain your modified image code below.

The image is a square inside a square inside a square.

SAMPLE PROGRAM #1

import turtle

myPen = turtle.Turtle()

# These variables track the position of the turtle pen

posX = 0

posY = 0

# This user defined function draws a single image pixel

def drawPixel(rgb) :

global posX

myPen.down()

myPen.color(rgb)

myPen.begin\_fill()

myPen.circle(8)

myPen.end\_fill()

myPen.up()

myPen.forward(18)

posX = posX + 18

# THE MAIN PROGRAM CODE STARTS HERE

#

redColor = (255,0,0)

drawPixel(redColor)

drawPixel((128,0,0))

greenColor = (0,255,0)

drawPixel(greenColor)

drawPixel((0,128,0))

blueColor = (0,0,266)

drawPixel(blueColor)

drawPixel((0,0,128))

SAMPLE PROGRAM #2

import turtle

myPen = turtle.Turtle()

# These variables track the position of the turtle pen

posX = 0

posY = 0

# These variables define the image information.

# Each pixel in the image has a (r,g,b) value

# The complete image is simply a list of pixels

pixelAddress = 0

pixelMemory = [

(15,15,5),(13,13,6),(8,10,3),(23,21,10),(32,33,16),(33,52,22),(32,54,21),(25,42,17),

(21,19,17),(20,18,9),(7,7,6),(58,65,11),(42,47,7),(11,8,6),(24,25,8),(21,28,10),

(25,19,5),(16,13,8),(28,28,12),(191,192,18),(205,202,21),(42,42,14),(11,11,4),(16,11,3),

(34,59,10),(35,47,15),(24,35,12),(156,139,26),(154,140,22),(28,43,10),(9,12,1),(19,22,5),

(42,88,15),(48,94,18),(98,120,49),(213,195,123),(109,134,66),(44,91,15),(52,86,22),(43,85,18),

(50,95,13),(63,104,39),(224,213,156),(255,225,140),(120,153,92),(41,99,17),(58,103,28),(42,98,17),

(35,86,13),(71,105,42),(223,208,144),(216,204,146),(100,134,82),(28,87,3),(39,83,12),(32,80,12),

(49,102,29),(57,109,33),(92,125,53),(66,103,36),(29,66,13),(32,76,17),(48,91,26),(47,93,23)

]

# This user defined function draws a single image pixel

def drawPixel(rgb) :

global posX

myPen.down()

myPen.color(rgb)

myPen.begin\_fill()

myPen.circle(8)

myPen.end\_fill()

myPen.up()

myPen.forward(18)

posX = posX + 18

# This user defined function starts a new row of pixels

def newRow() :

global posX

global posY

myPen.up()

myPen.left(180)

myPen.forward(posX)

myPen.left(90)

myPen.forward(18)

myPen.left(90)

myPen.down()

posX = 0

posY = posY + 18

# THE MAIN PROGRAM CODE STARTS HERE

#

# Draw eight rows of the image.

# Each row contains eight pixels

for row in range (8) :

for column in range(8) :

drawPixel(pixelMemory[pixelAddress])

pixelAddress += 1

newRow()