**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
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?

The position is the first number (255, 0, 0)

* 1. What position is the number that controls the amount of green (g) in the pixel?

The position is the second number (0, 255, 0)

* 1. What position is the number that controls the amount of blue (b) in the pixel?

The position is the third number (0, 0, 255)

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

The shade of the colour gets darker.

* 1. What happens when the colour value is close to 0?

It 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.

To create a pixel containing a shade of orange you need redColor = (255,80,0).

* 1. Create a pixel containing a shade of the colour yellow.

To create a pixel containing a shade of yellow you need redColor = (255,230,0).

* 1. Create a pixel containing a shade of your favorite colour.

To create a pixel containing a shade of blue you need blueColor = (0,0,240).

1. Black, white, and shades of grey are created using combinations of equal r,g,b number values.
   1. Create a completely white pixel.

To create a pixel containing white you need blueColor = (255,255,255).

* 1. Create a completely black pixel.

To create a pixel containing black you need blueColor = (0,0,0).

* 1. Create a pixel containing a shade of middle grey.

To create a pixel containing a shade oof grey you need redColor = (127,127,127).

**Level 2: Resolution**

1. Download the image “Resoultion\_284x177.jpg” from Topic B folder in the class repository.
   1. Open the image in a program like Paint or Photoshop.
   2. What is the size of this image? How many pixels does it contain?

It has a width of 284 pixels and a height of 177 pixels and it contains 50268 pixels.

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

In the background you can see the pixels because its blurry.

* 1. Zoom in the view to enlarge the image
  2. Describe how the image looks (e.g. Can you see the pixels?)

You can start to see the pixels that make up the bird’s details like the eyes and the beak.

1. Download the image “Resoultion\_16x16.jpg” from Topic B folder in the class repository.
   1. Open the image in a program like Paint or Photoshop.
   2. What is the size of this image? How many pixels does it contain?

It has a width of 8 pixels and a height of 8 pixels and it contains 64 pixels.

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

Yes I can see the pixels and the image is unrecognizable.

* 1. Zoom in the view to enlarge the image
  2. Describe how the image looks (e.g. Can you see the pixels?)

You can see the pixels clearly.

1. Create a new Repl for Python with Turtle.
2. Copy and paste “Sample Program #1” from the listing at the end of this module.
3. Run the program and examine the Turtle output
4. Compare the program output to the “Resoultion\_16x16.jpg” image in question #2 above.

Both of them have the same size and contain the same number of pixels, you can see the pixels clearly and when you zoom in you can see the pixels clearly for both images.

1. Explain how the program code in lines 52 to 58 works. (i.e. The main program code.)
2. How the program prints out pixels to produce and 8 by 8 resolution image.

This line of code tells Python how many rows and columns there are in the image and when to start a new row.

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

Two variables have given a value, these are pixelMemory and pixelAddress which is put in lines 52 and 58 which tells each pixel what colour it is.

1. Explain the purpose of the code in lines 12 to 21
2. How this code is related to the pixels produced by the main program.

This code is related to the pixels because it tells what colour each pizel is supposed to be

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

The RGB value of the 19th pixel in the image is (28, 12, 12).

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

The RGB value of the 5th column on the 4th row in the image is (154, 140, 22).

1. Modify the main program to print the image upside-down (i.e. pixels in reverse order).
2. Show your modified image to Mr. Nestor.
3. Explain your changes to the program code below.

I used the reverse code and reversed the list so that the pixels would be printed in reverse.

1. Modify the main program to print the image at a resolution of 12 by 4 pixels.
2. Show your modified image to Mr. Nestor.
3. Explain your changes to the program code below.

The only change I made was the colour for each pixel and the rows and columns.

**Level 3: TBD**

1. Use and modify the sample pixel program code to create your own custom image.
   1. Create a larger resolution image than provided in the sample.
   2. Make sure the image is recognizable (or a clear pattern).
   3. Show your image to Mr. Nestor.
2. List and explain your modified image code below.

import turtle

myPen = turtle.Turtle()

myPen.speed(0)

# 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

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

#

pixelAddress=0

pixelMemory= [ (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),(0,0,0),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(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),(0,0,0),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(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),(0,0,0),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(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),(0,0,0),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(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),(0,0,0),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(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),(0,0,0), (0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255),(0,0,255), ]

for i in range(10):

for i in range(10):

drawPixel(pixelMemory[pixelAddress])

pixelAddress+=1

newRow()

I modified pixelMemory which tells Python what colour each pixel is and also changed the rows and columns.

**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),(907,132,79),(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()