**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 number that controls the amount of red in the pixel is (128,0,0)

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

The position number that controls the amount of green in the pixel is (0,128,0)

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

The position number that controls the amount of blue in the pixel is (0,0,128)

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

When a colour value is less than 255 it becomes a lighter shade

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

When a colour value is less than 0 it becomes a darker shade

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. import turtle
3. myPen = turtle.Turtle()
4. # These variables track the position of the turtle pen
5. posX = 0
6. posY = 0
7. # This user defined function draws a single image pixel
8. def drawPixel(rgb) :
9. global posX
10. myPen.down()

myPen.color(rgb)

1. myPen.begin\_fill()
2. myPen.circle(8)
3. myPen.end\_fill()
4. myPen.up()
5. myPen.forward(18)
6. posX = posX + 18
7. # THE MAIN PROGRAM CODE STARTS HERE
8. #
9. redColor = (240,165,0)
10. drawPixel(redColor)
11. drawPixel((128,0,0))
    1. Create a pixel containing a shade of the colour yellow.

import turtle

1. myPen = turtle.Turtle()
2. myPen.begin\_fill()
3. myPen.circle(10)
4. myPen.color("yellow")
5. myPen.circle(10)
6. myPen.end\_fill()
   1. Create a pixel containing a shade of your favorite colour.
7. import turtle
8. myPen = turtle.Turtle()
9. myPen.begin\_fill()
10. myPen.circle(10)
11. myPen.color("purple")
12. myPen.circle(10)
13. myPen.end\_fill()
14. Black, white, and shades of grey are created using combinations of equal r,g,b number values.
    1. Create a completely white pixel.
15. import turtle
16. myPen = turtle.Turtle()
17. # These variables track the position of the turtle pen
18. posX = 0
19. posY = 0
20. # This user defined function draws a single image pixel
21. def drawPixel(rgb) :
22. global posX
23. myPen.down()
24. myPen.color(rgb)
25. myPen.begin\_fill()
26. myPen.circle(8)
27. myPen.end\_fill()
28. myPen.up()
29. myPen.forward(18)
30. posX = posX + 18
31. # THE MAIN PROGRAM CODE STARTS HERE
32. #
33. redColor = (255,255,255)
34. drawPixel(redColor)
35. drawPixel((255,255,255))
    1. Create a completely black pixel.
36. import turtle
37. myPen = turtle.Turtle()
38. # These variables track the position of the turtle pen
39. posX = 0
40. posY = 0
41. # This user defined function draws a single image pixel
42. def drawPixel(rgb) :
43. global posX
44. myPen.down()
45. myPen.color(rgb)
46. myPen.begin\_fill()
47. myPen.circle(8)
48. myPen.end\_fill()
49. myPen.up()
50. myPen.forward(18)
51. posX = posX + 18
52. # THE MAIN PROGRAM CODE STARTS HERE
53. #
54. redColor = (0,0,0)
55. drawPixel(redColor)
56. drawPixel((0,0,0))
    1. Create a pixel containing a shade of middle grey.

**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?

The size of this image is 8x8 and it contains 5263 pixels

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

The picture looks like Minnie pixels that were put together

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

The pixels look all together and bigger when you zoom in, the actual picture looks burley

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?

The size of this image is 8x8 and has 72 pixels contained in it

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

The image can barely be seen the pixels cannot be seen from that distance

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

When I enlarge the image the pixels can barely be seen and the image is very blurry and hard to spot

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

**Level 3: TBD**

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),(223,208,144),(32,33,16),(33,52,22),(32,54,21),(25,42,17),

(21,19,17),(20,18,9),(223,208,144),(223,208,144),(223,208,144),(11,8,6),(24,25,8),(21,28,10),

(25,19,5),(16,13,8),(223,208,144),(223,208,144),(223,208,144),(42,42,14),(11,11,4),(16,11,3),

(34,59,10),(35,47,15),(223,208,144),(223,208,144),(223,208,144),(28,43,10),(9,12,1),(19,22,5),

(42,88,15),(48,94,18),(223,208,144),(223,208,144),(223,208,144),(44,91,15),(52,86,22),(43,85,18),

(50,95,13),(223,208,144),(223,208,144),(223,208,144),(223,208,144),(223,208,144),(58,103,28),(42,98,17),

(223,208,144),(223,208,144),(223,208,144),(223,208,144),(223,208,144),(223,208,144),(223,208,144),(32,80,12),

(49,102,29),(57,109,33),(223,208,144),(223,208,144),(223,208,144),(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()

**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()