



FALMOUTH
UNIVERSITY



COMP120: Creative Computing: Tinkering

1: Tinkering Graphics I

Learning Outcomes

By the end of this workshop, you should be able to:

- ▶ **Apply** knowledge of colour models to **write** code that manipulates pixels in a surface
- ▶ **Use** functions, arguments, and basic data structures such as arrays

Activity #1a – Setup

In pairs:

- ▶ Launch a basic Python project in PyCharm
- ▶ Import PyGame, setup a main window, and define a game-loop which renders a white background
- ▶ Refer to the following documentation:
 - ▶ `www.pygame.org/docs/tut/tom_games2.html`

Activity #1a – Setup

```
import pygame

pygame.init()

main_window = pygame.display.set_mode((800,600))

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            crashed = False

    main_window.fill((255,255,255))
    pygame.display.update()

pygame.quit()
```

Note: This is a PyGame example.

Activity #1b – Setup

In pairs:

- ▶ Render a green `Surface` in the top corner of the window
- ▶ Define a function to manipulate a single pixel in the `Surface` using a `PixelArray`
- ▶ Refer to the following documentation:
 - ▶ www.pygame.org/docs/ref/surface.html
 - ▶ www.pygame.org/docs/ref/pixelarray.html

Activity #1b – Setup

Firstly, create a new `Surface` and fill it with green:

```
my_surface = pygame.Surface((200,200))  
my_surface.fill((0,255,0))
```

Then, blit the green `Surface` onto the main window at the origin:

```
main_window.fill((255,255,255))  
main_window.blit(my_surface, (0,0))  
pygame.display.update()
```

Note: This is a PyGame example.

Activity #1b – Setup

Finally, define a new function using the `def` keyword to manipulate a single pixel in the `Surface`:

```
def set_pixel(surf, x_pos, y_pos, colour):  
    px_array = pygame.PixelArray(surf)  
    px_array[x_pos,y_pos] = colour  
    del px_array  
  
set_pixel(my_surface, 100, 100, (0,0,0))
```

Note: This is a PyGame example.

Activity #2 – Less Red

In pairs:

- ▶ Define a function to load an image file to a `Surface`
- ▶ Then, define a function to reduce it's redness
- ▶ Refer to the following documentation:
 - ▶ <https://www.pygame.org/docs/ref/image.html>

Activity #2 – Less Red

```
my_surface = pygame.image.load('test.jpg')
```

```
def decreaseRed(pict):  
    pixelMatrix = getPixels(pict)  
    for pixel in pixelMatrix:  
        value = getRed(pixel)  
        setRedPixel(pixel, value * 0.5)
```

Note: Not all of this source code excerpt will work in PyGame.

Activity #3 – Swap Channel

In pairs:

- ▶ Define a function that turns all of the red values of pixels into blue values...
- ▶ ...and all of the blue values into red values

Activity #3 – Swap Channel

```
def swapRedBlueChannels(pict):  
    pixelMatrix = getPixels(pict)  
    for pixel in pixelMatrix:  
        red_value = getRed(pixel)  
        blue_value = getBlue(pixel)  
        setRedPixel(pixel, blue_value)  
        setBluePixel(pixel, red_value)
```

Note: This source code excerpt will not work in PyGame.

Activity #4 – Greyscale

In pairs:

- ▶ Define a function that loads an image and turns it to greyscale
- ▶ Consider the following calculation:
 - ▶
$$NewPixelValue = \frac{\sum CurrentChannelValue}{NumberOfChannels}$$

Activity #4 – Greyscale

```
def loadGrayscale(file):  
    pixelMatrix = getPixels(makePicture(file))  
    for pixel in pixelMatrix:  
        red = getRed(p)  
        green = getGreen(p)  
        blue = getBlue(p)  
  
        pixelValue = (red+green+blue)/3  
  
        setRedPixel(pixel, pixelValue)  
        setGreenPixel(pixel, pixelValue)  
        setBluePixel(pixel, pixelValue)
```

Note: This source code excerpt will not work in PyGame.

Activity #5 – Negative

In pairs:

- ▶ Define a function that loads an image and turns it to its negative
- ▶ Consider the following calculation:
 - ▶ $\text{NewChannelValue} = 255 - \text{CurrentChannelValue}$

Activity #5 – Negative

```
def neg(picture):  
    pixelMatrix = getPixels(makePicture('file'))  
    for pixel in pixelMatrix:  
        red = getRed(p)  
        green = getGreen(p)  
        blue = getBlue(p)  
  
        setRedPixel(pixel, 255-red)  
        setGreenPixel(pixel, 255-green)  
        setBluePixel(pixel, 255-blue)
```

Note: This source code excerpt will not work in PyGame.

Activity #6 – Sunset

In pairs:

- ▶ Define a function that loads an image and produces several images as output, decreasing luminance
- ▶ Refer to the following documentation:
 - ▶ `//www.pygame.org/docs/ref/time.html`

Activity #6 – Sunset

```
def decreaseRed(picture, amount):  
    for p in getPixels(picture):  
        value=getRed(p)  
        setRed(p,value*amount)  
  
amount = 0.1 #tinker with this value  
wait_time = 50 #tinker with this value  
  
for i in range(10):  
    decreaseRed(picture, amount)  
    decreaseGreen(picture, amount)  
    decreaseBlue(picture, amount)  
    wait(50)
```

Note: This source code excerpt will not work in PyGame.

Activity #7 – Top-Copy

In pairs:

- ▶ Define a function that copies the top half of a picture to its bottom half
- ▶ Refer to the following documentation:
 - ▶ `https://docs.python.org/3.7/tutorial/introduction.html#lists`

Activity #7 – Top-Copy

```
def copyHalf(picture):  
    pixels = getPixels(picture)  
    for index in range(0, len(pixels)/2):  
        sourcePixel = pixels[index]  
        sourceRGBValue = getColor(sourcePixel)  
        destinationPixel = pixels[index + len(pixels)/2]  
        setColor(destinationPixel, sourceRGBValue)  
    repaint(picture)
```

Note: This source code excerpt will not work in PyGame.