COMP120: Creative Computing

# 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 Visual Studio Form App
- **Use** functions, arguments, and basic data structures such as arrays

## Activity #1a – Setup

In pairs:

- Open Visual Basic
- Create a 'Windows Forms Application'
- Refer to the following documentation for details:

`https://docs.microsoft.com/en-us/visualstudio/ide/create-csharp-winform-visual-studio`

## Activity #1a – Setup

```
int width = 640, height = 320;
Bitmap bmp = new Bitmap(width, height);

for (int y = 0; y < height; y++)
    {
    for (int x = 0; x < width; x++)
    {
        bmp.SetPixel(x, y, Color.FromArgb(255, 0, 0, 0));
    }
}
pictureBox1.Image = bmp;
bmp.Save("D:\\images\blackImage.png");
```

Note: This is an example that is contained in the ‘Form’ class

# Activity #1a – Setup

Add a **Picturebox** from **Tools**. Set the **Picturebox** to **Zoom**.
NOTE - ADD SCREENSHOT TO ILLUSTRATE

# Key C# Methods Used

- `Bitmap` - consists of the pixel data for a graphics image and its attributes.
  - `New` - Initializes a new instance of the Bitmap class with the specified size or from an existing file.
  - `Save` - Saves the Image to the specified file or stream.
- `SetPixel` - Sets the color of the specified pixel in a Bitmap.
- `GetPixel` - Gets the color of the specified pixel in a Bitmap.
- `Color.FromArgb` - Creates a colour structure from the four 8-bit ARGB components (alpha, red, green, and blue) values.

## Key Concepts

**Nested** `for` **Loops** - to iterate through all the positions in a two dimensional array. For example: all the pixels in an image which are arranged in rows and columns.

```
for (int hours = 0; hours < 24; hours++)
{
    for (int minutes = 0; minutes < 60; minutes++)
    {
        //do something for every minute in the day
    }
}
```

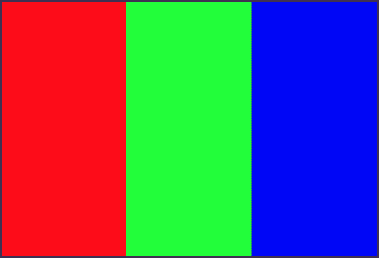
# Activity #1b – Setup

In pairs:

- Render a green `Bitmap` image
- Refer to the following documentation:
  - `https://docs.microsoft.com/en-us/dotnet/api/system.drawing.color.fromargb`
  - `https://docs.microsoft.com/en-us/dotnet/api/system.drawing.bitmap.setpixel`

## Activity #2 - Test Card

- Create a `Bitmap` image that displays 3 equal vertical bars of red, green and blue
- The image must be **640 x 480** in size.
- Consider how you will allocate the painting of pixels to the different areas of the screen.



## Activity #3 - Random Pixels

- Create a `Bitmap` image that displays random pixel for every pixel in the image. Like snow on an old TV.
- Consider how you will generate random values for ARGB
- You will need to explore these methods associated with the `Random` class:

```
 new Random();
```

Initializes a new instance of the `Random` class.

```
Next();
```

Returns a non-negative random integer.

FALMOUTH UNIVERSITY

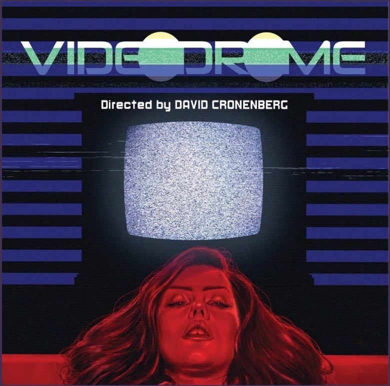
## Activity #3 - Random Pixels

```
Random rand = new Random();
```

Create a variable to contain the `Random` class.

```
int a = rand.Next(256);
```

Assign a variable for each colour channel and use `Next` with the new random variable to randomly choose a value.



# 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:
  - $NewPixelValue = \frac{\Sigma CurrentChannelValue}{NumberOfChannels}$

## 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:
    - $NewChannelValue = 255 - CurrentChannelValue$

# 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, descreasing luminance
- Refer to the following documentation:
    - `//www.pygame.org/docs/ref/time.html`

FALMOUTH
UNIVERSITY

## 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.