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

FALMOUTH
UNIVERSITY

Painting with Pixels

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

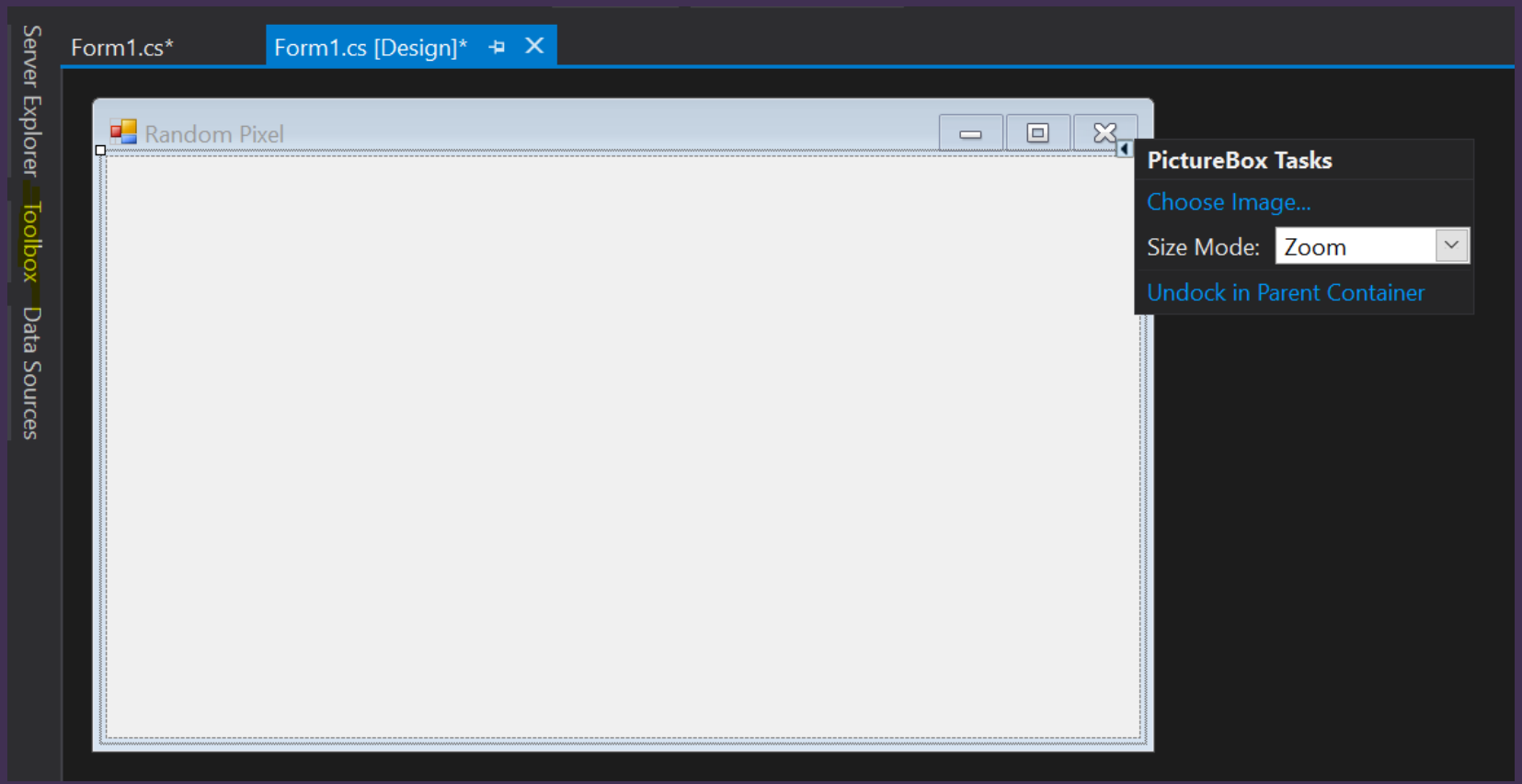
```
int width = 640, height = 320;
Bitmap bmp = new Bitmap(width, height);

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 the **Toolbox** panel. Set the **Picturebox** to **Zoom**.



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

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

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.

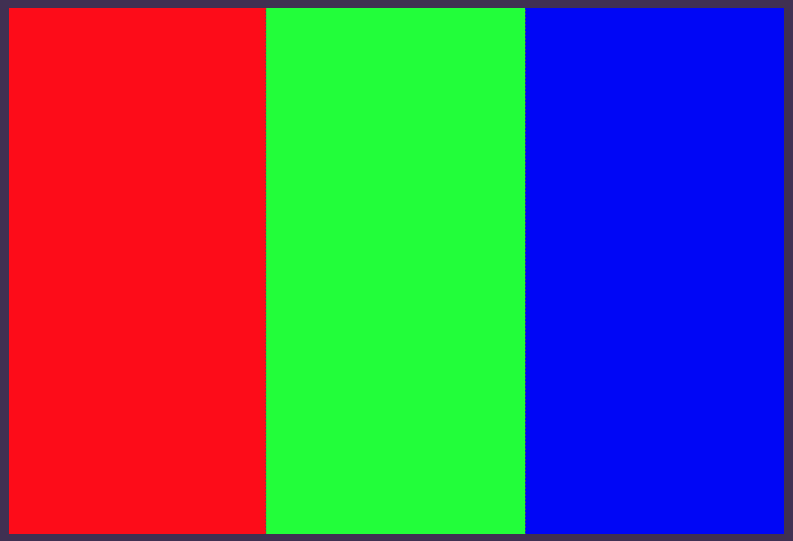
Activity #1b – Setup

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 #3 - 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.

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.

FALMOUTH
UNIVERSITY

Manipulating Bitmaps

Activity #4 – Less Red

In pairs:

- Define a function to load an image file to the Windows `Form` UI
- Then, define a function to reduce it's **redness**
- Refer to the following documentation:
  - `https://docs.microsoft.com/en-us/dotnet/api/system.drawing.bitmap`

Activity #4– Less Red

```
string img = "C:\\Images\\myPic.jpg";
Bitmap bmp = new Bitmap(img);
int width = bmp.Width;
int height = bmp.Height;
Bitmap rbmp = new Bitmap(bmp);

for (int y = 0; y < height; y++)
{
    for (int x = 0; x < width; x++)
    {
        Color p = bmp.GetPixel(x, y);
        int a = p.A;
        int r = p.R;
        int g = p.G;
        int b = p.B;
        int rI = Convert.ToInt32(r*.5);
        rbmp.SetPixel(x, y, Color.FromArgb(a, rI, g, b));
    }
}
```

Activity #5 – 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 #5 – Swap Channel

```
//get pixel value
Color p = bmp.GetPixel(x, y);

//extract ARGB value from p
int a = p.A;
int r = p.R;
int g = p.G;
int b = p.B;

//swap red for blue
RtoBbmp.SetPixel(x, y, Color.FromArgb(a, r, g, r));

//swap blue for red
BtoRbmp.SetPixel(x, y, Color.FromArgb(a, b, g, b));
```

Note: This code occurs inside the nested for loop and presumes you have setup 2 bitmap variables - RtoBbmp and BtoRbmp.

Activity #6 – 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 #6 – Greyscale

```
//get pixel value
p = bmp.GetPixel(x, y);

//extract pixel component ARGB
int a = p.A;
int r = p.R;
int g = p.G;
int b = p.B;

//find average
int avg = (r + g + b) / 3;

//set new pixel value
bmp.SetPixel(x, y, Color.FromArgb(a, avg, avg, avg));
```

Note: This code occurs inside the nested for loop and presumes you have setup a bitmap variable - bmp.

Activity #7 – Negative

In pairs:

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

Activity #7 – Negative

```
 //get pixel value
Color p = bmp.GetPixel(x, y);

//extract ARGB value from p
int a = p.A;
int r = p.R;
int g = p.G;
int b = p.B;

//find negative value
r = 255 - r;
g = 255 - g;
b = 255 - b;

//set new ARGB value in pixel
bmp.SetPixel(x, y, Color.FromArgb(a, r, g, b));
```

Note: This code occurs inside the nested for loop and presumes you have setup a bitmap variable - bmp.