COMP120: Creative Computing: Tinkering

5: Tinkering Graphics II

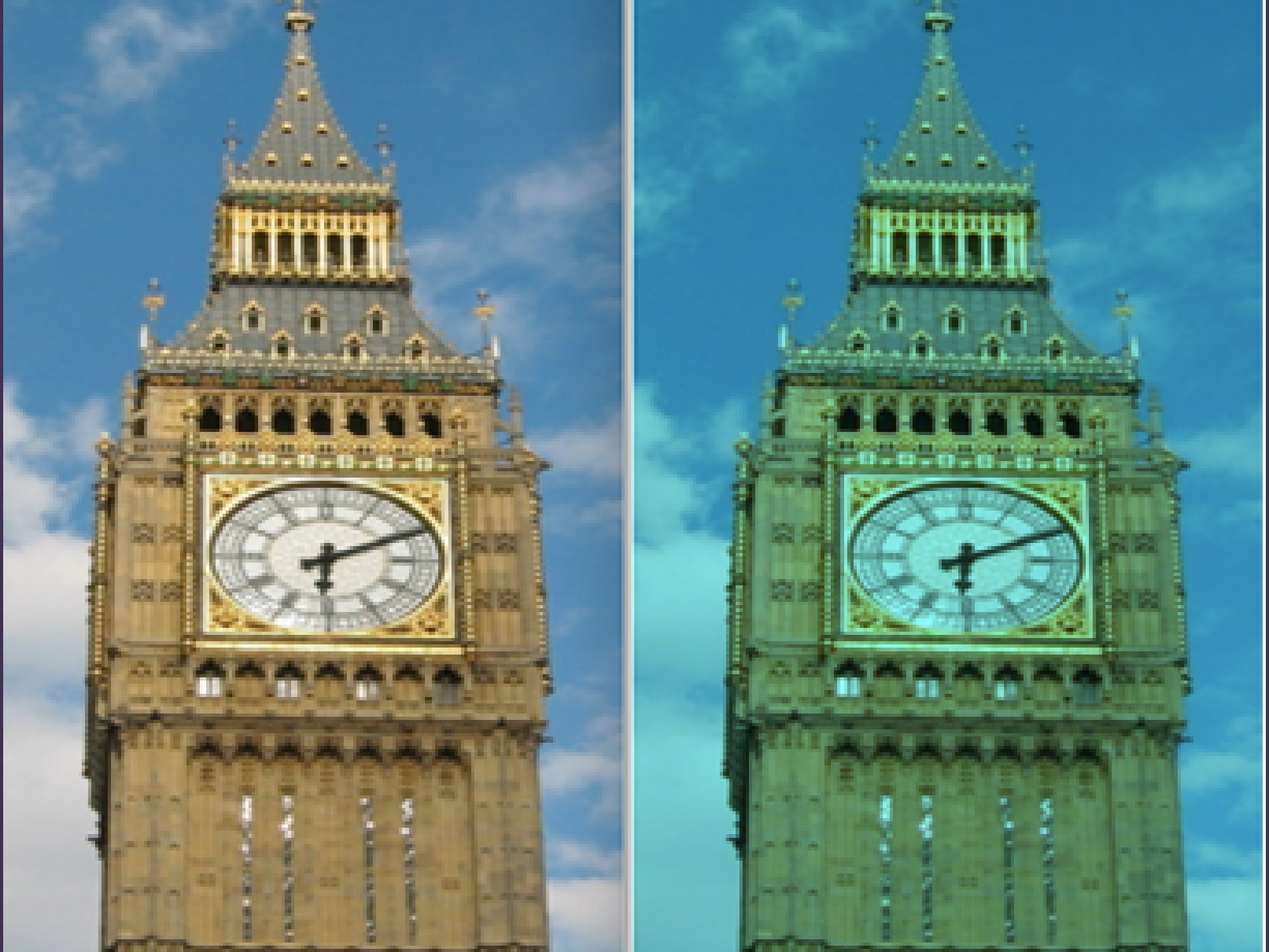
Learning Outcomes

- **Explain how** conditional logic can manipulate the output of a computer program
- **Apply** mathematical knowledge to **write** computer programs that manipulate pixels in a surface
- **Trace** existing computer programs to **predict** its output for a given input

Source Code: Less Red

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

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



Socrative: Less Red

Socrative room code: `FALCOMPMIKE`

If 'bb' is the original picture of Big Ben, which of the below function calls created the change:

- changeRed(bb, 1.5)
- changeRed(bb, 2.0)
- changeRed(bb, 0)

FALMOUTH
UNIVERSITY

Tinkering Graphics Activities

Calculating Distance Between Colors

Sometimes we need to measure when something is `close enough´:

- Distance between two points in the Cartesian coordinate system:
- $\sqrt{(x_1 - x_2)^2 + (y_1 1 - y_2)^2}$
- Distance between two colours in the RGB colour representation system:
- $\sqrt{(red_1 - red_2)^2 + (green_1 1 - green_2)^2 + (blue_1 1 - blue_2)^2}$

Activity: Color Distance

In pairs:

- Setup a basic project in PyGame
- Use the distance equation from the previous slide to write a function which accepts a two colours and returns the distance
- Test your solution
- Then, post your solution on Slack
- 10 minutes

Output: Color Distance

```
>>> print distance(WHITE, BLACK)
441.6729559300637
>>> print distance(WHITE, PINK)
113.13708498984761
>>> print distance(BLACK, PINK)
355.3519382246282
>>> print distance(MAGENTA, PINK)
192.41881404893857
```

Source Code: Conditions and Return Values

```
def closeEnoughToBrown(colour):
    if distance(colour, BROWN) < 50.0:
        return True
    else:
        return False
```

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

Source Code: Conditional Tolerance

```
def turnRed():
  brown = makeColor(42,25,15)
  file="/Users/guzdial/Desktop/mediasources/katieFancy ↩
      .jpg"
  picture=loadPicture(file)
  for pixel in getPixels(picture):
    color = getColor(pixel)
    if distance(color, BROWN) < 50.0:
      red=getRed(pixel)*2
      green=getGreen(pixel)
      blue=getBlue(pixel)
      setColor(pixel,makeColor(red,green,blue))
  return picture
```

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

Activity: Colour Tolerance

In pairs:

- Setup a basic project in PyGame
- Implement the function `closeEnough((colour, colour), tolerance)` that returns a boolean value
- Test your solution
- Then, post your solution on Slack
- 10 minutes

Red Eye

- When the flash of the camera catches the eye just right (especially with light colored eyes), we get bounce back from the back of the retina.
- This results in `red eye'
- We can replace the red with a color of our choosing
- First, we figure out where the eyes are (x,y)


C:\Documents and Settings\Mark Guz...
Zoom
X: 109 Y: 91 R: 129 G: 97 B: 76

Activity: Red Eye

In pairs:

- Setup a basic project in PyGame
- Refer to the following documentation
  - `http://www.pygame.org/docs/ref/rect.html`
- Implement the function: `removeRedEye(picture, area, colour)`
- Test your solution
- Then, post your solution on Slack
- 20 minutes


C:\Documents and Settings\Mark Guz...
Zoom
X: 183 Y: 97 R: 0 G: 0 B: 0

Calculating Luminance in RGB

- Luminance is the overall brightness of a pixel
- In RGB, it is the *mean* average value of each component:
  - $lum = (red + green + blue)/3$

Posterization

- Posterization is simply reducing the number of colours in an image
- We look for a range of colours, then map them to a single colour:
  - If red is between 63 and 128, set it to 95
  - If green is less than 64, set it to 31
- The end result is that a bunch of different colours, get set to a few colours
- Beware of naive solutions with a large number of `if' statements

Zoom

X: 0 Y: 0

R: 155 G: 153 B: 94 Color at location:

Zoom

X: 0 Y: 0

R: 159 G: 159 B: 95 Color at location:

Source Code: Black and White

```
def blackAndWhitePosterize(picture):
  for pixel in getPixels(picture):
    red = getRed(pixel)
    green = getGreen(pixel)
    blue = getBlue(pixel)
    luminance = (red + green + blue) / 3
    if luminance < 64:
      setColor(pixel, BLACK)
    else:
      setColor(pixel, WHITE)
```

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

Activity: Posterization

In pairs:

- Setup a basic project in PyGame
- Refer to the following documentation
- Implement the function: `makeGreyscale(picture, colourCount)`
- Test your solution
- Then, post your solution on Slack
- 20 minutes

Sepia Tone

- Pictures that are sepia-toned have a yellowish tint to them that we associate with older pictures.
- It's not directly a matter of simply increasing the yellow in the picture, because it's not a one-to-one correspondence:
  - Instead, colors in different ranges get mapped to other colours.
  - We can create such a mapping using IF statements
- The end result is that a bunch of different colours, get set to a few colours
- Beware of naive solutions with a large number of `if' statements

Source Code: Sepia (1)

```
def sepiaTint(picture):
  #Convert image to greyscale
  makeGreyscale(picture)

  #loop through picture to tint pixels
  for p in getPixels(picture):
    red = getRed(p)
    blue = getBlue(p)

    #tint shadows
    if (red < 63):
      red = red*1.1
      blue = blue*0.9
...
```

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

Source Code: Sepia (2)

```
...
    #tint midtones
     if (red > 62 and red < 192):
       red = red*1.15
       blue = blue*0.85

     #tint highlights
     if (red > 191):
       red = red*1.08
       if (red > 255):
         red = 255

       blue = blue*0.93

     #set the new color values
     setBlue(p, blue)
     setRed(p, red)
```

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

Sepia Tone

- First, we're calling greyScaleNew (the one with weights).
- We then manipulate the red (increasing) and the blue (decreasing) channels to bring out more yellows and oranges.
  - It's perfectly okay to have one function calling another.
  - Why are we doing the comparisons on the red? Why not? After greyscale conversion, all channels are the same!
- The end result is that a bunch of different colours, get set to a few colours
- Why these values? Trial-and-error: Tinker the values!

Activity: Sepia Tone

In pairs:

- Setup a basic project in PyGame
- Refer to the following documentation
- Refactor the function: `sepiaTint(picture)` to use constants rather than literals
- Tinker with the values of the constants to test your solution
- Then, post your solution on Slack
- 20 minutes

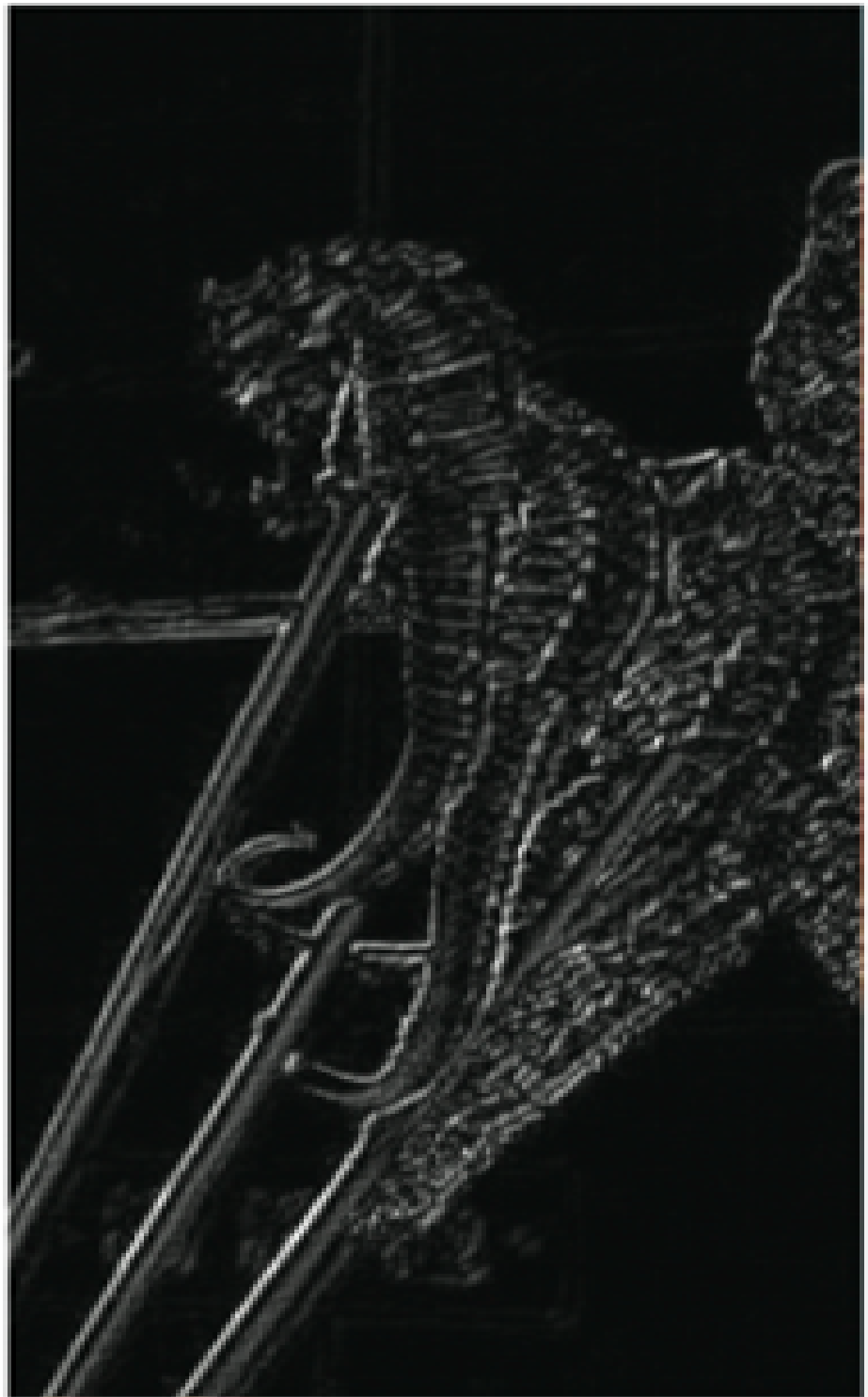
Edge Detection

- Blurring is averaging across pixels
- Edge detection is looking for *differences* between pixels:
  - We draw lines that our eyes see — where the luminance changes
- If the pixel changes left-to-right, or up-and-down, make a pixel black. Else, white.

Source Code: Edge Detection

```
def drawEdges(picture):
  for x in xrange(0, picture.length - 1):
      for y in xrange(0, picture.height - 1):
          pixelSum = getRed(pixel) + getGreen(pixel) + ↩
               getBlue(pixel)
          nextPixel = getPixel(picture, x+1, y+1)
          nextPixelSum = getRed(nextPixel)+ getGreen( ↩
              nextPixel) + getBlue(nextPixel)
          diff = abs(nextPixelSum-pixelSum)
          newColor = makeColor(diff,diff,diff)
          setColor(pixel, newColor)
```

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



Source Code: Edge Detection (1)

```
def drawBetterEdges(picture):
  orig = makePicture(filename)
  makeBw = makePicture(filename)

  for x in xrange(0,getWidth(picture)-1):
    for y in xrange(0,getHeight(picture)-1):
      here=getPixel(makeBw,x,y)
      down=getPixel(orig,x,y+1)
      right=getPixel(orig,x+1,y)

      hereL=(getRed(here)+getGreen(here)+getBlue(here)
          )/3
      downL=(getRed(down)+getGreen(down)+getBlue(down)
          )/3
      rightL=(getRed(right)+getGreen(right)+getBlue(
          right))/3
...
```

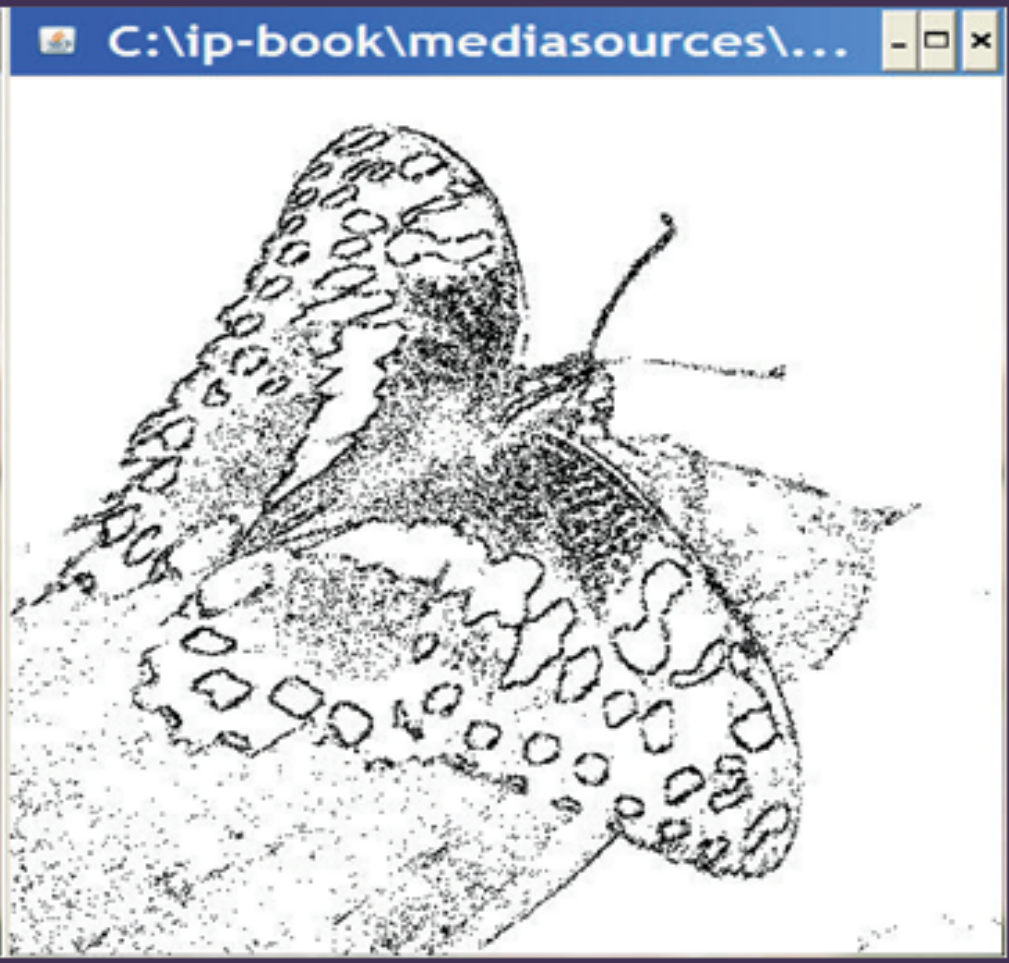
Note: This source code excerpt will not work in PyGame.

Source Code: Edge Detection (2)

```
...
      if abs(hereL-downL)>10 and abs(hereL-rightL)>10:
        setColor(here,black)
      if abs(hereL-downL)<=10 and abs(hereL-rightL) ↩
          <=10:
        setColor(here,white)
  return makeBw
```

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


C:\ip-book\mediasources\...


C:\ip-book\mediasources\...

Activity: Edge Detection

In pairs:

- Setup a basic project in PyGame
- Refer to the following documentation
- Refactor the function: `drawBetterEdges(picture)` to use better variable names
- Test your solution
- Then, post your solution on Slack
- 15 minutes

Background Subtraction

- Let's say that you have a picture of someone, and a picture of the same place (same background) without the someone there, could you subtract out the background and leave the picture of the person?
- Maybe even change the background?
- What we most need to do is to figure out whether the pixel in the Person shot is the same as the in the Background shot.
- Will they be the EXACT same colour? Probably not.

Source Code: Background Subtraction

```
def swapBackground(picture, back, newBack, tolerance):
  for pixel in getPixels(picture):
    x = getX(pixel)
    y = getY(pixel)
    backPixel = getPixel(back, x, y)
    pixelColour = getColour(pixel)
    backColour = getColour(backPixel)
    if (distance(pixelColour,backColour) < tolerance):
      newColour = getColour(getPixel(newBack, x, y))
      setColor(pixel, newColour)
```

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

Problems

- We've got places where we got pixels swapped that we didn't want to swap
- We've got places where we want pixels swapped, but didn't get them swapped
  - Shirt stripes
  - Shadows
  - etc.

Activity: Background Subtraction

In pairs:

- Setup a basic project in PyGame
- Refer to the following documentation:
  - `http://www.cs.utah.edu/~michael/chroma/`
- Implement chroma key as a form of background subtraction
- Test your solution
- Then, post your solution on Slack
- 30 minutes

FALMOUTH
UNIVERSITY

Code Tracing

Live Demonstration

- Start using debug tools when you run into problems:
- `https://www.youtube.com/watch?v=QJtWxm12Eo0`