cs151s11proj4

Matthew Willett-Jeffries
Cs151 Project 4
3/1/2011

The overall goal of this project was to convert different images into pixel mappings so that they could be manipulated according to a series of code. Like the image below different aspects of an pixel map were altered by converting the [r,g,b] values of a given set of pixels.

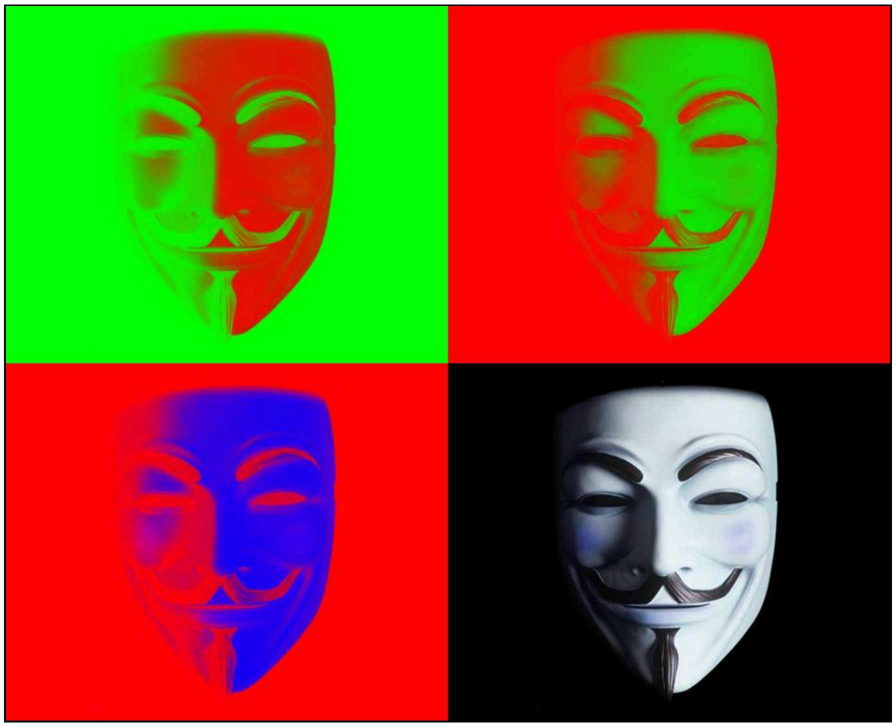

Image1.

This first image, a four part collage of the infamous anonymous mask, was achieved by first identifying the pixel map of the image then applying different filters on each map's general color scheme. These altered images were then transposed onto a pixel map defined by the width and height of the original image. The code can be seen as follows:

```
"pmap = graphics.Pixmap(sys.argv[1])
    map1 =pmap.clone()
    map2 =pmap.clone()
    map3 =pmap.clone()
    map4 =pmap.clone()

    radioactive(map1)
    idk(map2)
    idk2(map3)
    idk3(map4)

    pmap2 = graphics.Pixmap(2*pmap.getWidth(),2*pmap.getHeight())
    putPixmap(pmap2, map1,0,0)
    putPixmap(pmap2, map2, 0, map1.getHeight())
    putPixmap(pmap2, map3, map1.getWidth(), 0)
    putPixmap(pmap2, map4, map2.getWidth(), map3.getHeight())"
```

Here functions, radioactive() and idk1-3(), represent the different filters applied to each mapping of the original image and the putPixmap () function simply places these images.

The filters themselves were achieved by taking advantage of the for x in range(): function where the ranges assigned to the series x were the height and the width of each pixel map. Each pixel at a point in the range of the height and width were then assigned an [r,g,b] value based on their original value. For example, the function, radio active was achieved with the following code:

```
"def radioactive(image):
    for row in range(image.getHeight()):
        for col in range(image.getWidth()):
            (r,g,b) = image.getPixel(col,row)
            image.setPixel(col,row,(b,255-g,0))"
```

Please note that depending on the original values of ?(r,g,b) of a pixel a consistent but altered (r,g,b) value was applied.

This code is carried out by Filter.py under the Main() function. It can be found on fileserver1 under private/project4/filter.py and the image can be found here as well under mask.ppm.

____

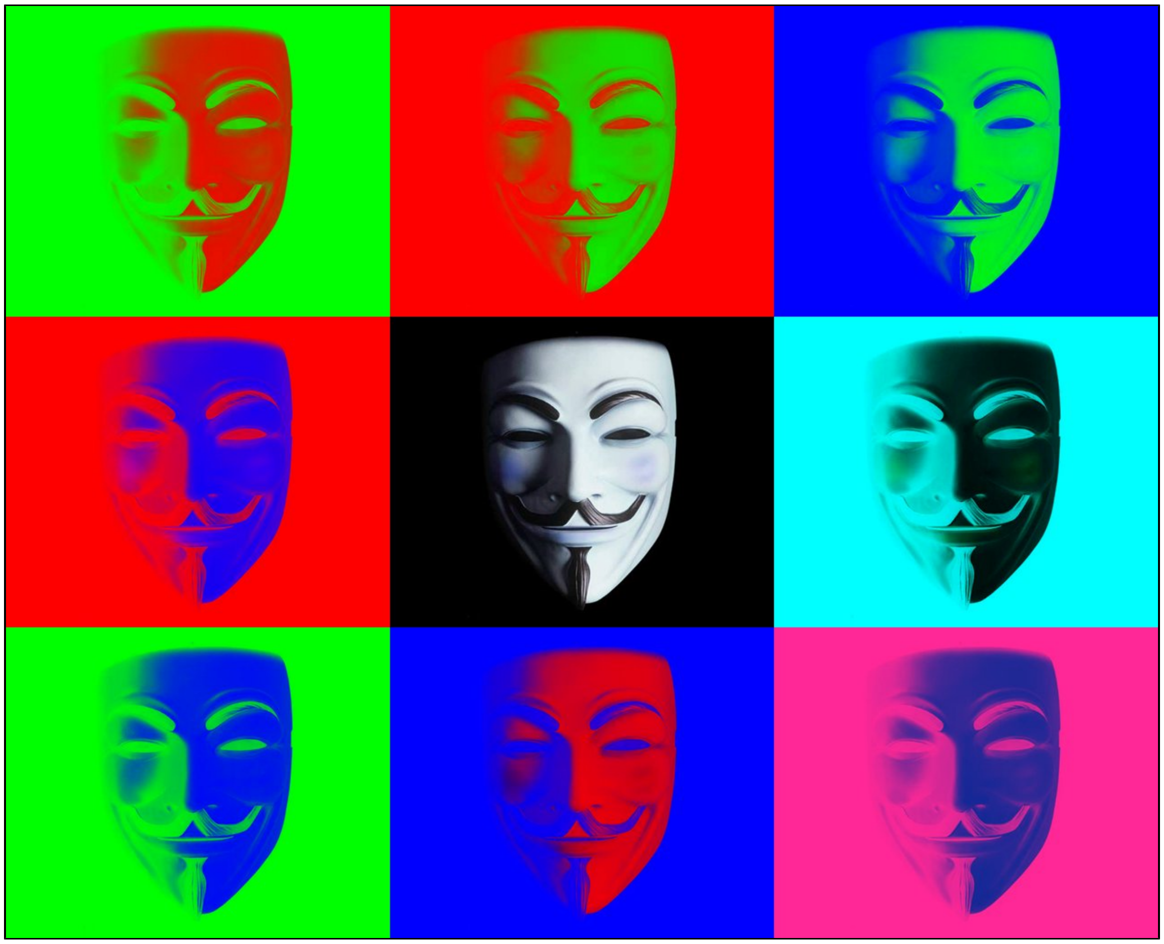

Image 2 (Extension)

Image 2 similar to image 1 was achieved via the same mechanics only differing in how it used the putPixmap() function. First a larger 3*height x 3*width empty pixmap was created then using the putPixmap() the images were place again so that they would fill the new empty pixmap,

```
"filter.putPixmap(pmap2, map1,0,0)
    filter.putPixmap(pmap2, map2, 0, map1.getHeight())
    filter.putPixmap(pmap2, map3, map1.getWidth(), 0)
    filter.putPixmap(pmap2, map4, map2.getWidth(), map3.getHeight())
    filter.putPixmap(pmap2, map5, map2.getWidth()*2, map3.getHeight())
    filter.putPixmap(pmap2, map6, map2.getWidth(), map3.getHeight()*2)
    filter.putPixmap(pmap2, map7, map2.getWidth()*2, map3.getHeight()*2)
    filter.putPixmap(pmap2, map8, map2.getWidth()*2, 0)
    filter.putPixmap(pmap2, map9, 0, map3.getHeight()*2)."
```

The image was then carried out in an separate execution file which imported filter.py. This file can be found on fileserver1 under private/project4/BigCollage.py. The original image, mask.ppm, can be found in this folder as well.

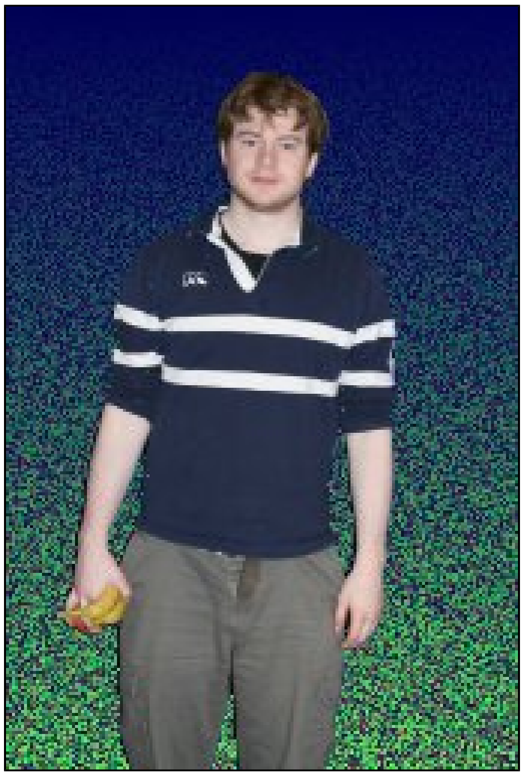
Image 3. (Extension)

The final goal of this project was to alter our own pictures by changing a solid background of a given pixmap. This meant that the pixel map would have to be changed at some values of (x,y) but not all. In order to achieve this an if statement was used, which when the color of a given ?pixel was blue enough would assign a new (r,g,b) value to that pixel. In the case of this picture the non-solid background was achieved by inserting random (r,g,b) values for each pixel that, though random, were semi dependent on a set of parameters which reverence the row number. The filter code is listed below:

```
"def Bluebegone(image):
    for row in range(image.getHeight()):
        for col in range(image.getWidth()):
            (r,g,b) = image.getPixel(col,row)
            if b > (2*r):
                image.setPixel(col,row,(random.randint(1,10)*0.05*row,random.randint(1,20)*0.05*row,85))"
```

The image was then carried out in an separate execution file which imported filter.py. This file can be found on fileserver1 under private/project4/NoBlue.py. The images, with_no_blue.ppm, the completed image, and matt.ppm, the original, can all be found in this folder as well.

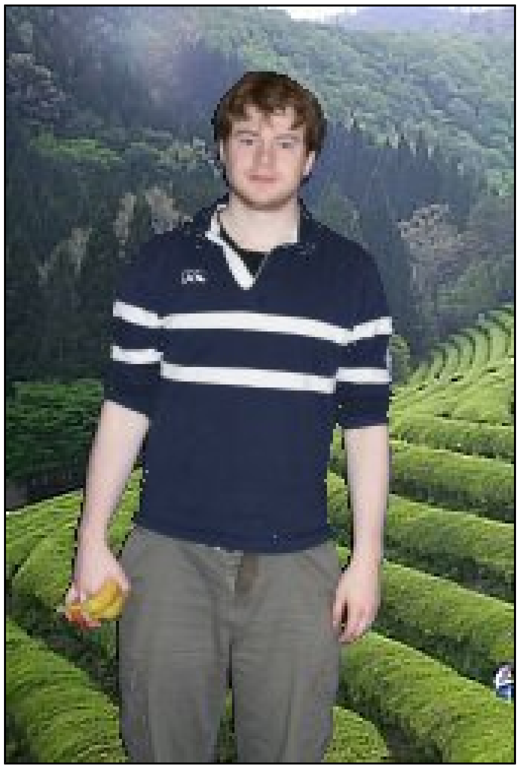
Image 4. (extension)

This image extension was achieved similarly to image 3, by making use of an if statement to determine when to apply a change in effect. The effect however had to reference another image entirely and therefor had to obtain information from that image and transfer it to the starting image. In this way, for each (x,y) value on the original a separate (r,g,b) value was identified based on the (x,y) location in the second image. When the if statement was satisfied the pixel was set to this separate (r,g,b) value. The code for the filter is listed below:

```
"def Scene(image,image2):
    for row in range(image.getHeight()):
        for col in range(image.getWidth()):
            (r,g,b) = image.getPixel(col,row)
            if b > (2*r):
                (t,s,z) = image2.getPixel(col,row)
                image.setPixel(col,row,(t,s,z))"
```

The image was then carried out in an separate execution file which imported filter.py. This file can be found on fileserver1 under private/project4/extension.py. The images, green.ppm, the background, matt.ppm, the foreground, and Field.ppm, the completed picture, can all be found in this folder as well.

Like the previous projects this project demanded a firm understanding of previously learn material and applied this facility to greater tasks. Though while the last several projects worked mainly with the turtle module, this project introduced a whole new module, graphics, which demanded a general understanding of a set of new terms. Specifically the introduction of pixel mappings and the methods for manipulating and identifying consistent patterns within them probably proved to be the most unfamiliar and enlightening element of these exercises.