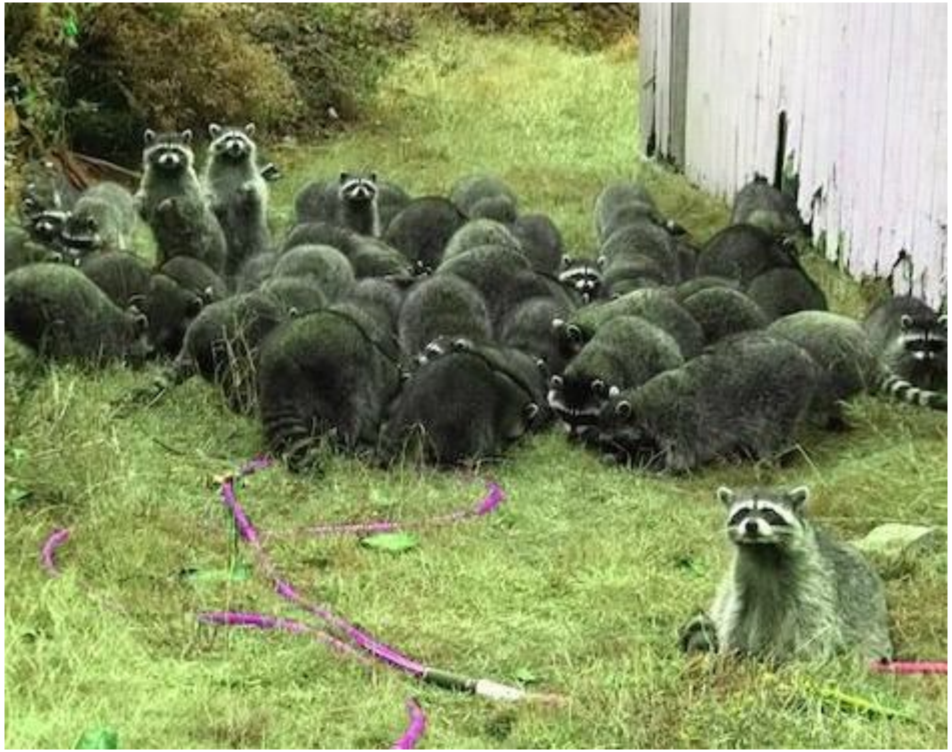
Graphics Programming Assignment - 00

Colors


pic_1_a.png

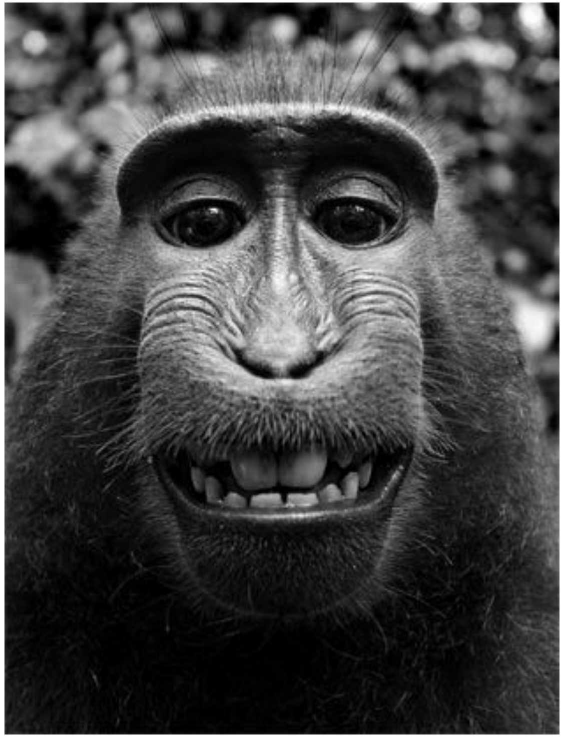
The task was to swap the red and green channels of the original image.

To do that, I sliced the green and red channels from the original. Then I copied the original image, and swapped the green channel with the red and vice versa.

```
g = img[:, :, 1]
r = img[:, :, 2]

new_img = img*1
new_img[:, :, 2], new_img[:, :, 1] = g, r
```

Colors pt. 2



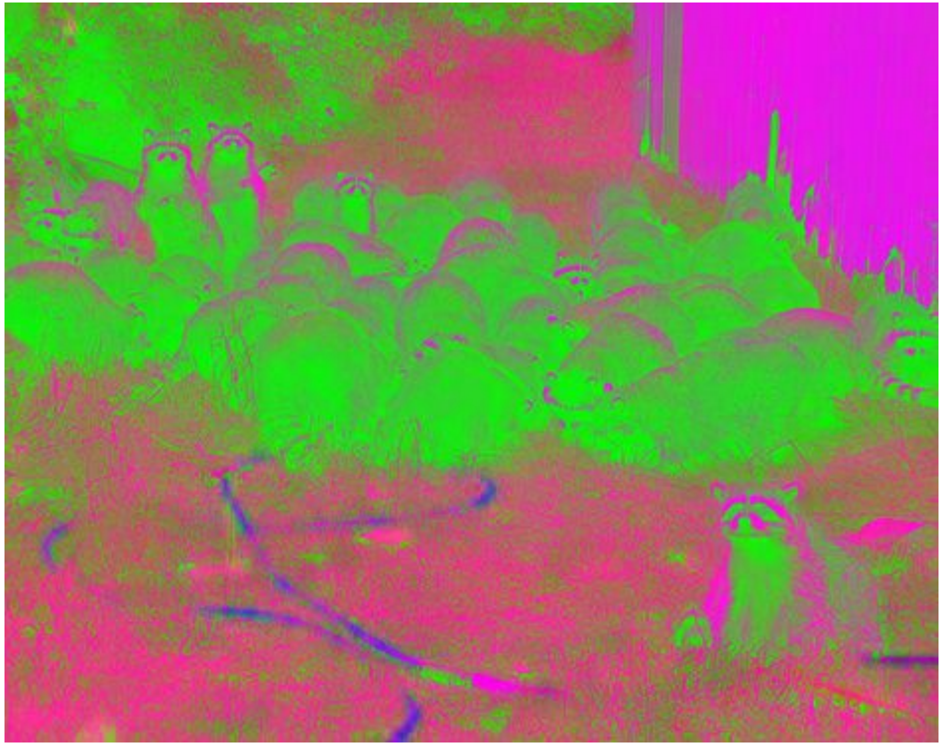
pic_1_b.png

For this one, the task was to splice the blue channel and save that as an image.

The result was an image with only one color channel. This makes the red and green colors darker. If you added the other two channels back, but made them 0s, then everything would be blue.

```
new_img = img[:, :, 0]*1
```

Colors pt. 3

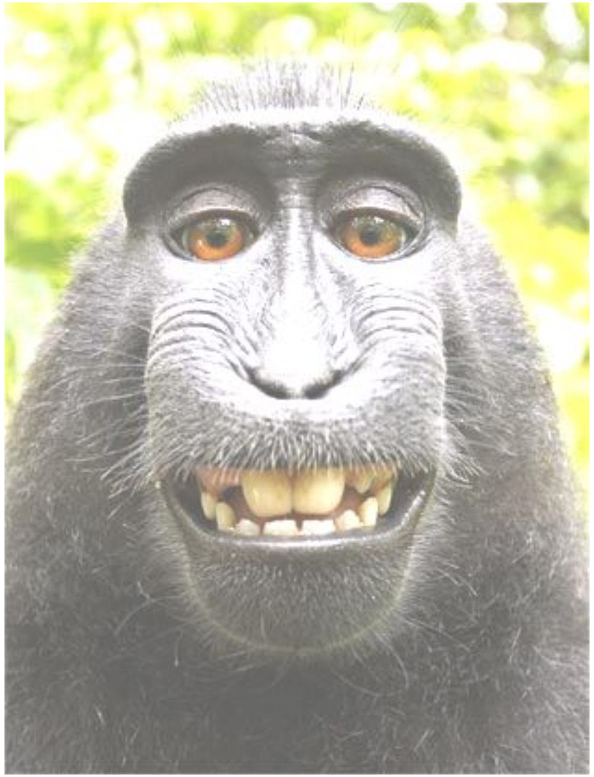


pic_1_c.png

T**h**e t**A**s<u>*k*</u> was to invert the green channel. To do that, you just subtract the current value in the green channel from 255 so that way the values become their opposite.

```
new_img = img*1
new_img[:, :, 1] = 255 - new_img[:, :, 1]
```

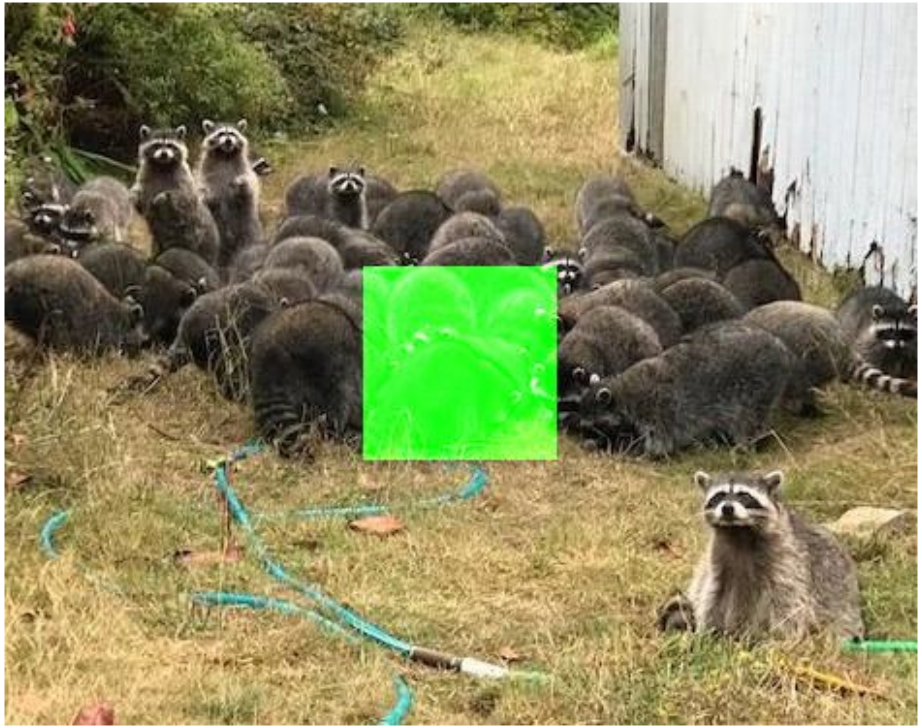
Colors pt. 4


pic_1_d.png

The task was to increase every color value in the image by 100. This just made the whole image look ‘brighter’. To prevent it from looking like a hellish hellscape or something I had to keep it from wrapping over by setting values over 255 to 255.

```
new_img = img*1
new_img[new_img >= 155] = 255
new_img[new_img < 255] += 100
```

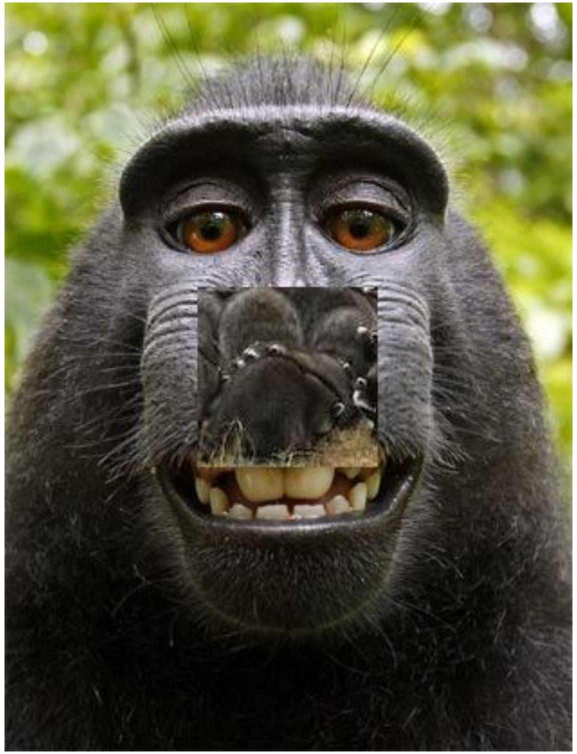
Copy & Paste


pic_2_a.png

The **TASK** was to create a 100x100 center of green.

```
new_img = img*1
h, w = img.shape[:2]
new_img[(h//2)-50:(h//2)+50,
(w//2)-50:(w//2)+50, 1] = 255
```

Copy & Paste


pic_2_b.png

*l***E** t**A**s<u>k</u>: Take the 100x100 center of the image 1 center and stick it on the 100x100 center of the second image. (this was a very long screenshot so I had to squish it in)

I just made a copy of the second image, got the dimensions, and copied the according pixels.

```
new_img = img2*1
h1, w1 = img1.shape[:2]
h2, w2 = img2.shape[:2]

new_img[(h2//2)-50:(h2//2)+50,(w2//2)-50:(w2//2)+50] =
img1[(h1//2)-50:(h1//2)+50, (w1//2)-50:(w1//2)+50]
```

Stats

I multiplied the width and height to get number of pixels. The minimum and maximum pixel values are 0 and 255.

I expected the mean intensity value to be somewhere around 128, and 120 is about right to me. The standard deviation being 62 means that most of the other pixel values are that far away from the mean, either positive or negative.

```
Number of Pixels: 175683
min: 0, max:255
Standard Deviation: 62.422979001095854
Mean Intensity: 120.62877075945501
```

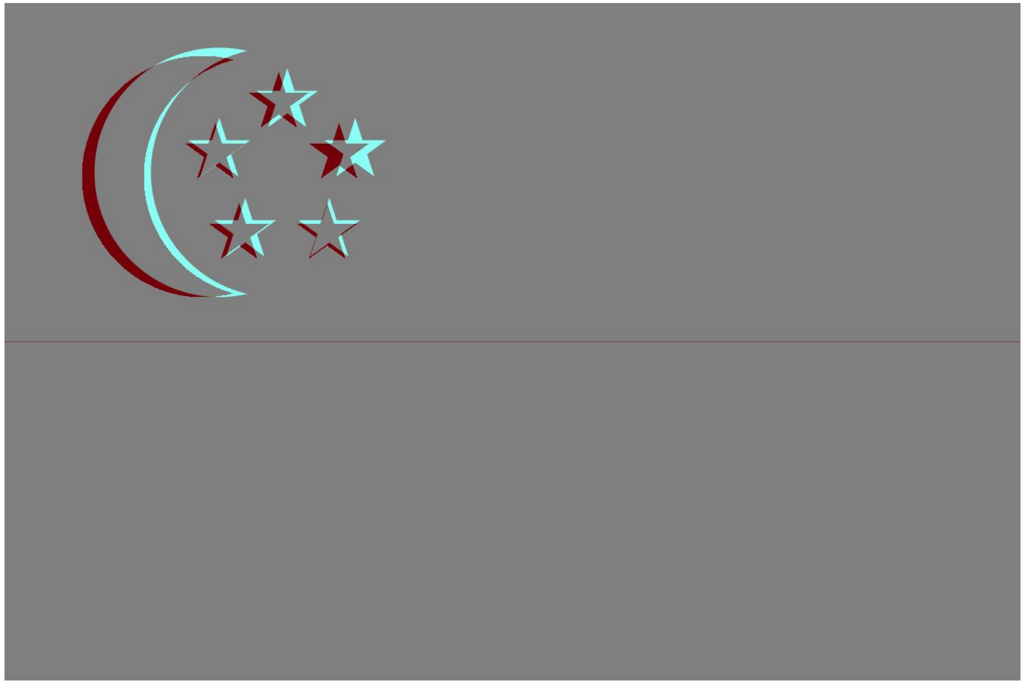
The standard deviation is 62. the mean intensity is 120 ~ 121, meaning that the average color value of all the pixels is 121. So, 68 percent of the pixels will have values 62 +- of 121.

FLAG

their flag

MY FLAG (pic_4_a.png)

FLAG pt.2



(pic_4_b.png)

I think this is right. I took the difference between the initial flag and my flag, then normalized and cast it to a uint8 in numpy. The black is where the pixels were the same in both images, and the difference in colors is where the 'difference' was. The moon was slightly off and the stars were too. But considering I used a ruler and very little math, I think it turned out sexy.

Code on next slide.

FLAG pt.3

```
orig_diff = (img1*1.0 - img2)
new_diff = normalize(orig_diff)
new_diff = new_diff.astype(np.uint8)
```

Difference code

```
img = img - np.min(img
img = img / img.max()
img *= 255.9999

return img
```

Normalize code