CST 205 - Lab 7 – 2/16/19
Partners: Paul Whipp, Jesus Andred Bernal Lopez, Michael Avalos-Garcia

Task 1 Summary: This task was fairly straight forward. We used a findmax routine targeting red and looped through the whole image. We did this with an x and y axis so that we could provide those at the end. Since the maximum value is 255 and we only need to find 1 pixel, we short-circuited the loop when we found a pixel with a red value of 255.

The pixel we found: (255, 254, 248)
Location: x = 1080 y = 455

Below is the code used for task1:

```
from PIL import Image

def find_max_red(im):
    # get data as flat list, assign names
    pixel_list = list(im.getdata())
    max_red = -1
    curr_max_tup = None
    pos = 0
    # find max routine
    for p in pixel_list:
        if(p[0] > max_red):
            curr_max_tup = p
            max_red = p[0]
        if(max_red == 255):
            break;
        pos += 1
    # get image dimensions
    width, height = im.size
    # calculate coordinates
    y_coord = pos // width
    x_coord = pos % width
    # return all values
    return curr_max_tup, x_coord, y_coord

max_tup, x_c, y_c = find_max_red(im)

print(f"tuple: {max_tup} xcoord: {x_c} ycoord: {y_c}")
```

Task 2 Summary: Task 2 was a little bit hard at first, but after looking over the lecture slides, we found a starting point. The toughest part of this task was getting the three images to show.

Below is the code used for task2:

```
from PIL import Image

def blank_canvas():

    canvas = Image.new("RGB", (2000, 600), "white") #creating blank canvas
```

```
    #storing pictures
    first = Image.open("Jesus.png")
    second = Image.open("Paul.png")
    third = Image.open("wes.png")

    #code copied from lecture
    #copied 3 sperate times to show all 3 pictures
    target_x = 0
    for source_x in range(first.width):
        target_y = 0

        for source_y in range(first.height):
            color = first.getpixel((source_x, source_y))  # get pixels
from the source
            canvas.putpixel((target_x, target_y), color)  # put pixels
onto target
            target_y += 1
        target_x += 1

    for source_x in range(second.width):
        target_y = 0

        for source_y in range(second.height):
            color = second.getpixel((source_x, source_y))  # get pixels
from the source
            canvas.putpixel((target_x, target_y), color)  # put pixels
onto target
            target_y += 1
        target_x += 1

    for source_x in range(third.width):
        target_y = 0

        for source_y in range(third.height):
            color = third.getpixel((source_x, source_y))  # get pixels
from the source
            canvas.putpixel((target_x, target_y), color)  # put pixels
onto target
            target_y += 1
        target_x += 1

    canvas.save("three_pictures.jpg")

blank_canvas()
```

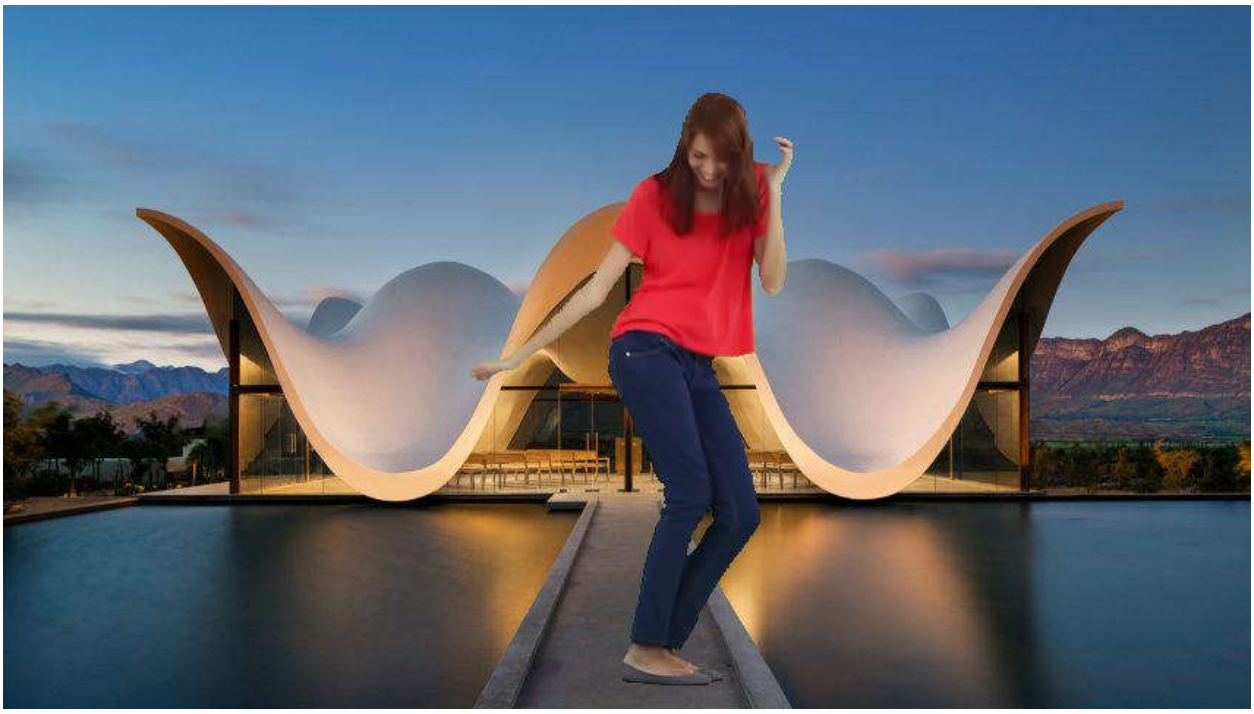
Below is the snapshot of task 2 completed:

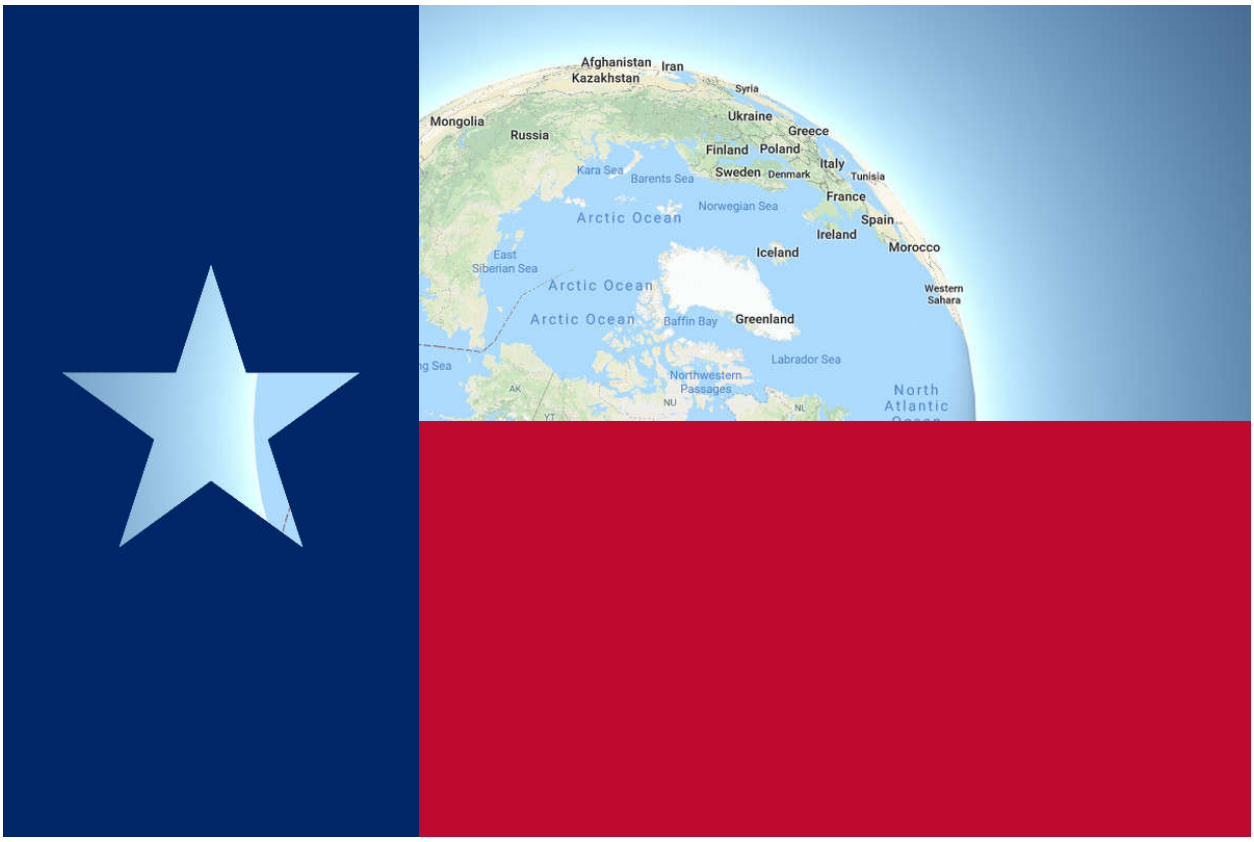
Task 3 Summary: Overall task 3 was fairly simple. The part that was a bit troublesome was figuring out how to make it work with images of different sizes. So we went to programmer's heaven(stackoverflow) and got the answer we were looking for. So overall, thanks for the code! We had fun with this one.

Below is the code used for task3:

```
from PIL import Image
import math


def distance(one, two):
    dist = 0
    for i in range(3):
        dist += math.pow(one[i] - two[i], 2)
    return math.sqrt(dist)


def chrome_key(img, bg, color_to_replace, save_loc):
    width = min(img.width, bg.width)
    height = min(img.height, bg.height)
    for x in range(width):
        for y in range(height):
            pix = img.getpixel((x, y))
            if distance(pix, color_to_replace) < 180:
                img.putpixel((x, y), bg.getpixel((x, y)))
    img.save(save_loc)


if __name__ == "__main__":
    img = Image.open("img/one.jpg")
    bg = Image.open("img/bg.jpg")
    save_loc = "img/result.png"
    green = (0, 255, 0)
    chrome_key(img, bg, green, save_loc)
```

Below is the snapshot of task 3 completed:





Task 4 Summary: Task 4 followed the same structure as task 3, but finding the correct conversion to 'LabColor' was tricky. Many docs later, Torin posted the correct way to handle this type conversion and we were off to the races. We tried 5 different distance values: 7, 12, 18, 24, and 30. The best of these was perhaps 18. At 24, there was still some green and yet there were holes in the image, but this seems to be because of the starting images.

Below is the code used for task4:

```
from PIL import Image
import math
from colormath.color_objects import sRGBColor, LabColor
from colormath.color_conversions import convert_color
import colormath.color_diff as cm_cd # delta_e_cie2000

im = Image.open('img/preyer.jpg')

def de(color_1, color_2):
    return cm_cd.delta_e_cie2000(color_1, color_2)

# turn scaled RGB pixel
def l_c(pixel):
```

```
    rgb = sRGBColor(pixel[0], pixel[1], pixel[2], True)
    return convert_color(rgb, LabColor)

def chromakey_de(source, bg, offset_x, offset_y):
    #source size plus offset must be less than bg size
    green = (0, 147, 34)
    perc_done = .1
    for x in range(source.width):
        for y in range(source.height):
            coord = (x, y)
            curr_pix = source.getpixel(coord)
            if de(l_c(curr_pix), l_c(green)) < 24.0:
                source.putpixel(coord, bg.getpixel((offset_x + x, offset_y
+ y)))
        if x == int(source.width * perc_done):
            print(f"Chromakey is {int(perc_done * 100)}% done.")
            perc_done += .1
    source.save('img/chromakeyed.png')



im_1 = Image.open('img/harry_mf_potter.jpg')
im_2 = Image.open('img/street.jpg')

chromakey_de(im_1, im_2, 200, 200)
```

Below is the snapshot of task 4 completed:


Truly
Unlimited
$79.99/mo
PARK
W 52 St


Truly
Unlimited
$79.99/mo