Lab #7: Blending Photos

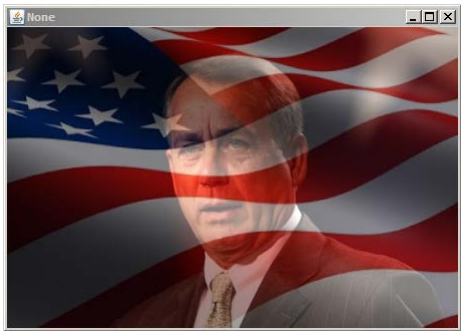
Activity A: Giving the Republicans their turn

In a previous assignment we gave a little publicity to President Obama by using his "Hope" posters from 2008. This week we will give a little publicity back to the folks on the other side of the aisle by creating a poster for Speaker of the House John Boehner (even though this is from the recent past).

We all know that every politician needs a campaign photo of themselves where they are superimposed over/with the waving American flag. In other words, we want to take these two pictures:

+


And add them together to get



For this first activity you may begin by looking at Program 86 (pp.184-185). This program took two images and blended them together, but it did it in such a way that they only partially overlapped. In this activity you have two starting images which are exactly the same size. When that is the case it is really easy to "add them together" to produce a new image which is also the exact same size.

In a file named lab8.py you should:

- Create a function called mixFaceAndFlag().
- This function should take no parameters.
- It should immediately open two pictures. pic1 should be the image of John Boehner (boehner.jpg) and pic2 should be the waving flag (flag.jpg)
- It should create a third, empty image (let's call it output) which is exactly the same size as the first two images (450x300 pixels).
- Since you need to set this new, blank output canvas to have colors in each of the pixels you will need to set up a pair of loops (x/y loops) that consider every pixel in the output canvas.

- For each (x,y) pair you should grab the color from the face and the color from the flag at that exact pixel location. You should calculate a new color where the red is calculated as 50% of the red from the face and 50% of the red from the flag. Similarly, you should calculate the new green and new blue using the same formula.
- Once you calculate the new r, g, b values you should set the pixel to that color.
- When you are done you should show/return this new picture. It should look like the one above.

Activity B: Creating an adjustable version

The problem with previous function is that it produces an image which has too much flag and not enough John Boehner. It would be better if we had a little more control over how much of each picture is included in the final image. The other thing that is a problem with the previous function is that it is "hardcoded" to work only with the pictures of John Boehner and the American flag. Suppose I wanted to create a blended image of President Bill Ruud superimposed with the campanile using 70% president Ruud and 30% campanile. It would take several changes to make that adjustment.

In this part of the lab, we will work on making modifications that convert your specialized program from the previous activity into a much more generalized program that can be reused for similar tasks.

- Begin by copying all of your code from Activity A into a new function.
- Call this function blendTwoImages.
- Change the program definition line so that instead of taking in no parameters it takes in two parameters. These two parameters should be labeled pic1 and pic2.
- Now you should remove the lines of code inside of the program that opened up boehner.jpg and flag.jpg. They are no longer necessary since those images are being passed in as parameters.
- Finally, we need to modify the line of code where you created the blank output canvas. You can no longer assume that you want a canvas that is 450x300. Instead you should look at the width and height of pic1 that was passed into this function and make the output canvas the same size as pic1 (We will assume that pic1 and pic2 are always the same dimensions. It is acceptable to have the function crash if the user violates that assumption).

Once you have this working you can type:

```
>>> p1 = makePicture("boehner.jpg")
>>> p2 = makePicture("flag.jpg")
>>> result = blendTwoImages(p1,p2)
```

to get the exact same picture you made in Activity A. But you can ALSO type in:

```
>>> p1 = makePicture("ruud.jpg")
>>> p2 = makePicture("UNI.jpg")
>>> result = blendTwoImages(p1,p2)
```

And get

Notice that this has taken a very specialized program (one that only blended Speaker Boehner and the flag) and turned it into a generalized program that can blend any two pictures that are the same size.

But let's not stop there. Let's make it this new function even more generalized.

- Change the header definition of blendTwoImages to take four parameters. The first two should remain as pic1 and pic2. The second two parameters should be labeled weight1 and weight2.
- It will be assumed that rather than using 50% of pic1 and 50% of pic2 you will use weight1% of pic1 and weight2 % of pic2. Modify the lines of code that you wrote earlier so that instead of multiplying by 0.5 you multiply by weight1 and weight2.

Those are the only changes you need to make. NOW if I run

```
>>> p1 = makePicture("boehner.jpg")
>>> p2 = makePicture("flag.jpg")
>>> result = blendTwoImages(p1,p2,0.7,0.3)
```

I get a picture that is 70% Speaker Boehner and 30% American flag. Notice that this is much better looking overall.