Lab 9: The Colors of Silver

Goals

Develop more experience working with images in Processing.

Warm Up

Recall the following points from Assignment 12:

- Photos need to be in the same folder as the .pde file; jpg, .png and .gif are OK.
- One or more `PImage` variables, like `myImage`, must be declared.
- For us, it is much easier if the canvas size exactly matches the size of the photo.
- Photos need to be *loaded* into the program and assigned a name as in
  `myImage = loadImage("greatPic.jpg");`
- The photo must be placed on the canvas at a specific position, as in
  `image(myImage,0,0);`
- To work with the actual pixels on the canvas, they need to be placed into the `pixels[ ]` array with the `loadPixels( )` command.
- To update the screen with the revised pixels, use the `updatePixels( )` command.

You will use all of that information in this lab.

Display The Image

Find a color photo that you'd like to use for the assignment, and write a Processing program to display it. Divide the items listed in the Warm Up above so that all but the last two items are performed in `setup( )`; the remaining two should be performed in `draw( )`. I will use a photo of Nate Silver, a statistician who predicted the presidential election and explained by the Seahawks last play was smart (even if it didn't work).

Extract A Color

Next we want to display only the red pixels when the user clicks the `'r'` key. As usual, this uses an if-statement in a `keyPressed( )` function. If the user clicks `'r'`, then refill the `pixels[ ]` array with only the red component of the pixels. To do that, we make an assignment of the form

```
pixels[i] = color(red(pixels[i]), 0, 0);
```

which must be inside of a for-loop in which `i`, the index, runs from 0 up to the `width*height`. Notice that this assignment effectively "zeroes out" the green and blue components of the pixel, keeping only the red. Once the `pixels[ ]` array is refilled, the screen update in the `draw( )` function will display the "reddened"

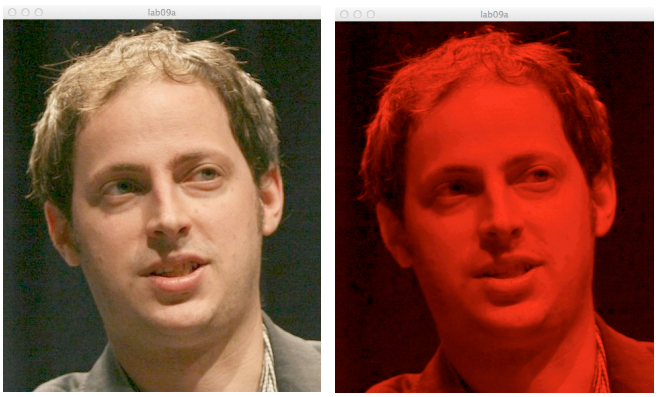

image. (Be sure `draw( )` **does** the `updatePixels( )`.) Try it out!

Restoring the Image

Once the pixels have been modified, as described in the last step, then we'd like to restored them so we can try other changes. For that, we add a `mousePressed( )` function that simply repositions the original image (`myImage`) back at position 0,0. Now, running the program can change to only red pixels by pressing `'r'`, and then restore the original image by clicking the mouse. Try it!

Finishing Up

Return to your `keyPressed( )` function. Using copy/paste/edit, replicate the if-statement recognizing each of the letters and displaying the image as follows:

`g` displays only the green component of the pixel
`b` displays only the blue component of the pixel
`c` displays the green and blue components of the pixel
`m` displays the red and blue components of the pixel
`y` displays the red and green components of the pixel

where `c` stands for cyan, the complement of red, `m` stands for magenta, the complement of green, and `y` stands for yellow the complement of blue.
Once that's done, it is possible to look at the photo in all of its parts.

Wrap Up

You have practiced working with images in Processing, and displayed an image with less than the full RGB components of each pixel.

Turn In

Submit your **commented** program and photo to the class dropbox.