Lab Goals

- Create an “Image Reassembly” program
- Combine together the brightness/contrast and histogram programs into a single application

Assignment Details

This lab consists of two separate parts to build on some of the image processing work that we have done during recent lectures. In the first part, you will create an “Image Reassembly” program that does the inverse of the crumbling program we wrote together. In the second part, you will combine together the brightness/contrast program with the histogram program we looked at on Wednesday.

Image Reassembly (30 points)

In class on Friday, we wrote an “Image Crumbling” program which took a provided image, broke it up into a number of Tile objects, and then allowed those Tiles to fall one-by-one to the bottom of the screen. For the first part of this lab, you will do close to the opposite – begin with Tiles scattered randomly throughout the screen, and then reassemble them into a complete image.

I recommend that you approach this problem via the following steps (though you certainly don’t need to and can approach it however you see fit):

1. Start by breaking the image up into Tiles as we did in class. This should be basically copy/paste code. The difference is that the (x,y) position that you pass into the Tile constructor will be the final position of the Tile rather than the initial position.

2. Once all of the Tiles have been created, give them random positions and orientations in the sketch scene. Then, draw them initially at those locations and angles. To make things easier, these Tiles can initially be positioned off of the sketch window if the random generation puts them there.

3. Next, you will need to compute xStep and yStep variables for each Tile. The purpose of these variables is to give a vector for the direction that each Tile needs to travel to get from its randomized starting position to its final position in the completed image. You can also create an angleStep variable to rotate each Tile back to the correct orientation for the
completed image, or you can simply set a defined `rotateAmt` for all tiles to spin them at the same rate until they reach their final position, then stop the rotation when it gets to the final location. This should complete your `setup()` function.

4. Now, in your `draw()` function, you will need to incrementally change the \((x, y)\) position and `angle` of each `Tile` using the `xStep`, `yStep`, and `angleStep` (or `rotateAmt`) variables that you defined for each `Tile`. Then tiles should then spin and slowly move back into their final positions, stopping when they reach their destination.

The image that you choose to break apart and reassemble is your decision. If you want to experiment around a bit more, here are some suggestions to enhance this program:

- Try making the `Tiles` variable in size.
- Instead of rotating each `Tile` about its center, try rotating it about a random pixel in the `Tile`.
- Make each `Tile` greyscale until it is set in its final position, then fade it into full color.

I uploaded a demo video to BitBucket (labs/Lab6Videos/reassemble.mp4) to better show what the goal of this lab is. Mine begins on a mouse click so that I could better capture the video; yours does not need to do so.

**Brightness, Contrast, and Histograms (20 points)**

In class on Wednesday, we looked at a prewritten sketch that computed the color histogram of an image, as well as a second sketch that allowed the user to modify the brightness and contrast of an image. Though I described the transformations made to the histogram when brightness and contrast were changed, we didn’t actually see these changes in action. In this part of the lab, we will!

The goal of this section is to combine together the `histogram.pde` and `BrightnessContrast.pde` sketches into a single application. The exact layout of the interface is up to you, but I found it easier to draw the histograms on the left side, and the original and enhanced images on the right side with an x-position of 516.

The goal of the sketch is to draw the histogram of `imgEnhanced`, so you’ll need to alter the `histogram.pde` code to load the pixels of that `PImage` rather than `img`. You’ll also need to move all of the code from `histogram.pde` from `setup()` and put it in `draw()` of the combined sketch, so that you see continuous updates as the mouse is moved vertically and horizontally. Because the original histogram code only draws once, don’t forget to zero out all of the values in the array between each iteration of the draw loop; otherwise you will get misleading results.

Again, I uploaded a demo video to BitBucket (labs/Lab6Videos/brightnesscontrast.mp4) to give you a better idea of what I am looking for than what the above text description provides. You can attempt to duplicate this exactly, or come up with your own interface as you see fit.
Here are high-level comments for how my solution behaves. You can either copy/paste code to try to duplicate this, or come up with your own solution:

// Before functions, define the arrays to store the histogram values, as well as
// PImages img and imgEnhanced

void setup() {
  // load the image
  // setup the width/height of the sketch window to fit everything
  // create imgEnhanced with the same dimensions as img
}

void draw() {
  // calculate brightness and contrast vars based on mouse position
  // call ContrastAndBrightness function
  // draw img and imgEnhanced
  // zero out histogram values
  // load pixels in imgEnhanced
  // iterate through imgEnhanced, updating histogram values
  // find max value in all histogram columns
  // scale and draw histograms
}

void ContrastAndBrightness() {
  // no changes made here
}

Submission Details

For this assignment, please submit the following items which you have followed while completing this lab to your cs382f2015-<your user name> repository. Your submission should include the following:

1. Print: Interesting source code that you wrote, or representative sections of your solutions
2. Upload: All of your commented source code for your Image Reassembly program
3. Upload: All of your commented source code for your combined Histogram Brightness Contrast program
4. Upload: The base image that you are reassembling
5. Upload: The base image that you are enhancing and drawing histograms for
6. Upload: A screenshot of the brightness/contrast histograms in action
7. Print and Upload: An Assignment Information Sheet
Additional Group Requirements

You are welcome to submit this assignment on your own, or in a group with one other person. In cases of groups, I would prefer one experienced programmer and one inexperienced programmer in the group, but this is not an absolute requirement. Additionally, each group need only submit one paper copy of their work, but each member of the group should push all items to their own repositories.

Finally, each group must create a short document (0.5-1 pages) detailing the work breakdown of the group members: who worked on which components, which lines of code belong to each group member, etc. Comments in the code will help here too.