Image Transformations

Computational Photography: Project 4

1 - Objective

The goal of this project is to implement some basic image transformations and use them to create a collage in the style of David Hockney from your own images.

2 - Deadline

It should be submitted on T-Square by 11:55PM on Monday, June 16th (after the break).

3 - Process

3.1 Download the base source

Download and unzip the folder with the base code for this project.

3.2 Project description

For this project, you will be implementing 2 of the 3 basic image transformations. You implemented Translation in the Blending project, which leaves Scale and Rotate. We have provided you with a project template (see the "Source code" section for more information) which implements Layers as in the last project but without blending and masks. The template also come with user mouse and key interactions as well as translation. All that remains to be filled are the rotated() and scaled() functions. This is where you come in.

Programming:

As with the last project, this one is designed to allow you to test each function separately and then both together.

The first stage of this project is the scale function. We have an image **I** with width **w** and height **h**. We want to write a scale function scaled(**I**, **sx**, **sy**) such that our function returns a new image **I'** with width **sx***w** and height **sy***h** and there exist no empty pixels. To do so, we will first create a new empty image of the appropriate size. Then, for each pixel in this image, we will find the pixel in the original image to which this new pixel belongs. We will find this pixel by converting a value **x'** in range [0, **sx***w] to **x** in range [0, **w**]. We will do the same for **y'** in range [0, **sy***h] to **y** in range [0, **h**]. Then we set the color **I'(x',y') = I(x,y)**. We now have a bigger or smaller image **I'** containing colors evenly sampled from **I.**

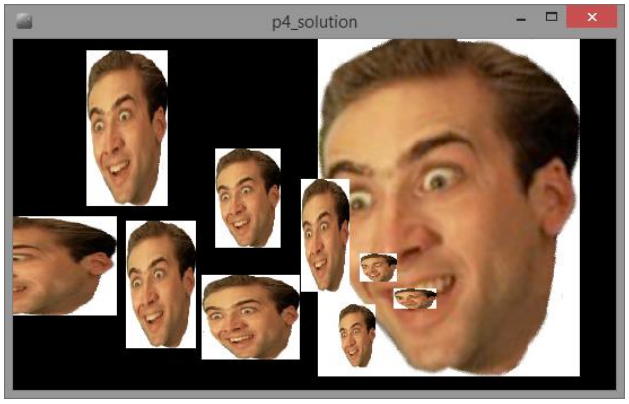

Figure 1: A test of your scale function loaded by pressing 'q'

The second and more involved stage of this project is the rotate function. We have an image **I** with width **w** and height **h**. We want to write a rotate function rotated(**I**, **angle**) such that our function returns a new image **I'** which consists of the old image **I** centered and rotated by **angle** degrees such that any pixel which does not belong to the rotated image **I** has an alpha value of 0 (and is therefore translucent). Remember the discussion of this process in class. We can define the operation of rotating a set of points as applying a matrix $\begin{bmatrix} \cos(angle) & \sin(angle) \\ -\sin(angle) & \cos(angle) \end{bmatrix}$ to the vector $\begin{bmatrix} x \\ y \end{bmatrix}$, the coordinates of the point, resulting in a new, rotated point $\begin{bmatrix} x' \\ y' \end{bmatrix}$. If we view each pixel of an image as a point, we can rotate then rotate the image by applying a single rotation matrix to each pixel of the image. However, as discussed in class, we once again have to do the inverse of this in order to avoid artifacts. So, we want to create our new image **I'** with a width and height which will fit the rotated image **I** and then iterate through each of the pixels **I'(i,j)** to find the pixel in I to which it is associated by applying the inverse rotation matrix to the pixel (remember that the inverse of a rotation matrix is simply the transpose of that matrix). If the pixel found after applying the inverse rotation is outside of the range of **I** (it is negative, or greater than width or height), then we will set the pixel's alpha value to 0 instead, making it invisible.

If we break with process into discrete steps, they are as follows:

1. Create the inverse rotation matrix for a 2D rotation of **angle** degrees
2. Find the correct dimensions of **I'** such that the rotated **I** will fit, centered, inside it.
3. For each pixel of the new matrix **I'(i,j)**
    a. Apply the inverse rotation matrix to the pixel to find the associated pixel **(i',j')** in **I**
    b. If **(i',j')** are both within the bounds of **I**, set **I'(i,j) = I(i',j')**, otherwise, set **I'(i,j)** alpha to 0.

Having done this successfully, you should be able to run the 'w' key example below:

Figure 2: A test of your rotate function activated by pressing 'w'

With both of these tasks complete, you should be able to press ‘e’ and see the example below:

Firgure 3: A test of both rotate and scale together, run by pressing ‘e’.

Photography and Mosaic:

Once you have your code working, your goal is then to collect a series of images from your environment (possible while you are on break) and stitch them together with your program to create a mosaic similar to those created by David Hockney. Search “David Hockney Photography” for examples. You must use at least 8 layers in your result image and your result will be shown to the class. You will create a PDF document with your mosaic image, a couple sentences about the context and inspiration of your image and optionally any comment you may have about this assignment’s clarity, time consumption, level of interest, difficulty, etc.

3.3 Source code

The source code does everything for you except the rotated and scaled functions. All of your code should go in those 2 functions.

The number keys 0-9 switch between layers and keys ‘q’,’w’,and ‘e’ run the test examples.

Holding ‘r’ and using the mouse wheel will rotate the current layer.

Holding ‘x’,’y’, or ‘b’ and using the mouse wheel will scale the current layer.

The ‘s’ key will save the current image.

The ‘p’ key will print all layer properties to the console so you can reload the scene from its current state as is done by the test cases.

You should modify the source code as explained in the description and comment your code (include your name in the header). The source code is written in Processing. Visit “Processing.org/reference/” for more information on built in functions and structure. Please note that **you are not allowed to use built in processing functions** to accomplish the tasks listed in the project description. We want you to access and directly manipulate the pixels in question to perform each task.

When in doubt, ask.

3.4 Authorship Rules

The code that you turn in entirely your own. You are allowed to talk to other members of the class and to the Professor and the TA about general implementation of the assignment. It is, for example, perfectly fine to discuss how one might organize the data for a matrix stack. It is also fine to seek the help of others for general Processing/Java programming questions. You may not, however, use code that anyone other than yourself has written. Code that is explicitly not allowed includes code taken from the Web, from books, from previous

assignments or from any source other than yourself. You should not show your code to other students. Feel free to seek the help of the Professor and the TA's for suggestions about debugging your code.

3.5 Submission

For this assignment, you will submit a Processing sketch and a PDF as stated in the project description.

Your unedited pictures should be located in the data folder where the default photos also reside. Your write-up in PDF format should be in the same folder as your code. In order to run the source code, it must be in a folder named after the main file. When submitting any assignment, leave code in this folder such that all code will run without any changes to folder structure or code. For this assignment, compress the parent folder containing your code folders and PDF and submit that via T-square.