**CSC 545 Computer Speech, Music and Images**

**Assignment: Color Indexing and Dithering**

**Due April 5, 2018**

Write a Processing program to simulate a indexed imaging and implement Floyd-Steinberg dithering. Set the number of colors in a variable or constant defined near the top of the program; your program should be able to handle 8 to 256 colors. The setup() function loads an image (with the filename defined by a String variable near the top of the program), reads a color palette (determined by the image filename and the number of colors), and calls your functions to create a new image that uses only colors from the palette and then a second image that uses Floyd-Steinberg dithering to reduce artefacts introduced by indexing. Test your program with the following images: ColoredSquares.jpg, colorful-1560.jpg, and Lizard.jpg.

Hot keys are set up as follows: ‘1’ displays the original image (img[0]), ‘2’ displays the indexed image (img[1]), ‘3’ displays the dithered image (img[2]), ‘c’ displays the color palette, ‘h’ displays the histogram of the indexed image (phist), and ‘d’ displays the histogram of the dithered image (dhist). If you prefer, the two histograms may be displayed as text in the display window (not the console). NOTE: ‘h’ and ‘d’ do not display the histogram of the original image—rather, for each entry in the color palette, they display the number of times that color is used; for example, if palette[0] is used 10 times, then counts[0] = 10. Display the color palette as a grid filled with the corresponding color in the table; you cannot simply display the palette image stored in *palettes*.

When dithering, you will have to handle the boundary problem—the first pixel in a row has no lower left neighbor; the last pixel in a row has no right or lower right neighbor; the bottom row has no lower neighbors. A common solution is to leave a one pixel border on the left, right, and bottom that is not dithered.

There are at least two implementations of Floyd-Steinberg dithering, in Processing, on the Internet. One of them is very similar to my implementation. In fact, this is a reasonably generic algorithm and I would be hard-pressed to determine whether you copied Internet code or wrote your own. So I won’t try – you are free to use the Internet implementation of the dithering – and only the dithering – algorithm. But you must make it work within the framework of the code I have given you – and be sure it’s the Floyd-Steinberg algorithm and not a different one. In the end, you might find it easier to write your own.

Name your program Asn3 and put it in your upload folder on the class server by 5 minutes after the start of class on the due date.

**Deliverables**

Processing program, described above

Be sure your program is well documented internally.

**Indexing pixels**

For each pixel

Find the closest palette color //Use Euclidean distance

Replace the pixel with the palette color

If dithering

Add quantization error to neighboring pixels, as specified by the Floyd-Steinberg algorithm