CPSC 478/578 Computer Graphics
Fall 2017
Assignment #1
Assigned: Wednesday, August 30, 2017
Due: Wednesday, September, 13, 2017, 11:59pm
(This is after shopping period ends, no extensions for being a late shopper.)

**This assignment has five questions. Note that the requirements for each question may vary depending on whether you are registered for 478 or for 578. The areas addressed in this assignment are the structure of images, and defining and displaying 2D triangles using WebGL.**

## General Note

To get your programs to work, you will need to run them with a localhost HTTP server, e.g. with the `python -m SimpleHTTPServer` command (python 2.x) or `python -m http.server` (python 3.x) in the directory where the program files are stored, and then use
http://localhost:8000/

## Turn-in Procedure

You should submit your work as a zip file using the Canvas server. Please name your file as
LastNameFirstName-Assignment1.zip
When your file is unzipped there should be subdirectories for each question named q1, q2, q3, q4 and q5. Name your files as directed in each question. In each directory you should have:
1. The HTML and Javascript programs you have written. You should use files in the form of the samples given, rather than producing files from scratch. This will help us follow your code.
2. Sample images created by your program. You can save these by clicking and saving results in your browser, or by taking a screenshot.
3. A readme.{txt, doc} that answers any questions posed, and that lists the input used to create the images you include. You should also list the operating system (e.g. Linux, Windows 7, 8.1, 10, Mac OS 10.4.4) and browser (e.g. Firefox 40.0.2, Safari, IExplorer, Edge) that you used. If you programs fail on the machines used for grading, you may be asked to bring in your system to demonstrate that the files you submitted functioned in the environment you worked in.

**Question I.** Creating and displaying an image using html and Javascript.

**478 and 578** The file q1-example.html uses the script js/onmycanvas.js to create an image and assign colors to the image pixels. This shows you how you can address each pixel, and what the effect of combining values of r,g, and b are. Use the file q1-

checker.html and modify js/checker.js to create a checkerboard of two specified colors. Use the value entered in html to determine how many squares should appear in each row of the checkerboard.

**578** Use some other analytical functions based on image location (e.g. polynomials, exponentials) to vary color across the image to make new abstract designs. Select some different parameters (rather than number of squares) to control the result. Name your files q1-other.html and js/other.js

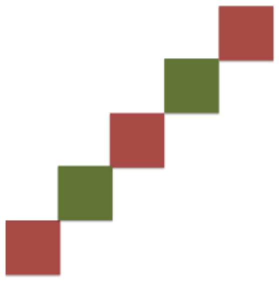
**Question 2.** Reading and filtering an image using html and Javascript.

**478 and 578** The file q2-example.html and js/filtering.js show how to browse for an image and then display the original image and the image after filtering. In this case the filter is very simple – the green and blue channels are changed. Create new files q2-bilateral.html and js/bilateral.js that take two integers in addition to browsing for a file. The first integer n should specify the size of a median filter. That is, if n is 2 each pixel (i,j) in the filtered image should be the median of the 25 pixels from (i-2,j-2) to (i+2,j+2). The second integer m should specify an intensity matching value from 0 to 255. For each pixel calculate an intensity (.3r+.5g+.2b). The median filter should be calculated by only including pixels in the sum that have an intensity within m of the pixel value of (i,j). If m is 255, all of the pixels should be used. If m is 0, a pixel will be the average of other pixels with the same intensity. In the readme file for this question, explain the visual effect of changing n and m.

**578** Write a new filter that displays an image that shows the absolute value of the difference between the original image and an image filtered with a median filter of size m. Explain what features of the original image this new image shows.

**Question 3.** Drawing a line, pixel by pixel.

**478 and 578** Create a new html file q3-line.html that takes as input the value of two points p1= (x1,y1) and p2= (x2,y2), and an image size nhorizontal and nvertical. The values of the x's and y's should all be between 0 and 100. Write a script js/bresenham.js that draws a line from p1 to p2, considering the lower left hand corner of the image as (0,0), and the upper right hand corner as (100,100). Draw the line using Bresenham's algorithm. Alternate the colors of the pixels along the line between blue and red. DO NOT use WebGL and DO NOT use any built in "draw" functions.


Example of a line draw pixel by pixel with alternating colors.

**578** Create a new html file q3-triangle.html that takes as input three points p1,p2, p3, in the same form as in the first part of question 3. Create a file js/triangle.js that draws the three lines that form a triangle. In the readme file, describe how drawing these lines could be used to efficiently draw a filled triangle (i.e. a triangle where all of the pixels inside the triangle are some color other than the background).

**Question 4.** Uniformly colored triangles in WebGL.

**478 and 578** The file hello_draw_another.html is a variation of the file hello_draw.html that is part of the code provided with the Ganovelli et al. textbook. Write a new file hello_draw_ten.html that draws 10 non-overlapping triangles. There should be at least one triangle in each of the four quadrants of the image (not just the upper right, or not just the top of the image). You can choose your own two colors for background and triangle color.

**Question 5.** Triangles approximating another shape in WebGL

**478 and 578** The files rendering-variation.html and js/rendering-variation.js are variations of files provided with the Ganovelli et al. textbook. Write your own files q-filled-circle.html and js/rendering-filled-circle.js that produces a filled circle approximated by an even number 2n triangles that either has one half the circle colored with one color and one half the other color, or which has one color in the middle that smoothly varies to second color at the edges. Your html file should let the user specify how many triangles are used in the approximation, two colors, and whether the uniform color or smoothed colors are used.