Programming for Design (11055)

Project 1 - *Self Portrait*

Keir Herbert (u3211239)

Explanation/Rationale

The original idea was to create a recognisable portrait by utilising just simple shape geometry. The intention was to avoid any kind of stick figure or cartoon style result.

Using a photograph as a starting point, I executed a vector trace in order to reduce the image into fewer, more simple shapes. The image was then converted back into a bitmap, the colour depth was reduced to black and white and the resolution was reduced to a 200 x 200 pixel matrix. This matrix was analysed and the x, y coordinates of the black pixels were retrieved and stored in a text file.

I'd hoped to get JavaScript to read the text file, but I discovered that for security reasons, JavaScript is unable to access the client's file system without the user actually selecting the file by means of an interactive process/dialogue box. That wasn't the user experience I wanted to achieve so instead I hard-coded all of the pixel coordinates into a multidimensional array.

Due to the large number of lines required to dimension an array with all of the coordinates, I decided to store that part of the code as a separate .js file. Otherwise, it would have made the main code too cumbersome.

Once all of the coordinate data is in memory and available for processing (more than 10,000 points in total), the program can plot the image using different shape methods to create different artistic results. The user is able to modify a range of parameters and see the effect that they have on the image in real time.

There are a few .js files that make up the program. It seemed logical for certain parts of the code to become independent functions and this separation allows the main code to remain quite concise and easy to manage.

All of the programs are wrapped up in HTML that is enhanced by the W3.css framework. This allows the HTML to do quite a lot of the page rendering and really just leaves the dynamic aspects to JavaScript. I intentionally positioned the JavaScript drawing canvas so that it overlaps some of the HTML elements, which I think makes the two methods appear more seamless and integrated.

The program output is best viewed on a high-resolution screen. It will adjust automatically for different resolutions, but the resulting image is more impressive at 1920 x 1080 or greater. At lower resolutions the individual shapes that make up the portrait are more difficult to discern. The canvas won't attempt to resize if just the browser window is resized; such an event will only prompt the program to once again check the screen resolution. I implemented this feature because I was developing the program on a dual monitor setup and I wanted it to be able to gracefully manage being moved between screens.

I have successfully tested the program on Opera, Edge, Firefox, and Safari browsers.

Accepting the default values for the modifiable parameters results in an image that's surprisingly recognisable given the dramatically reduced amount of information, compared with the original source photograph. From there, the output can become very abstract if parameters such as the shape size are increased beyond natural thresholds.