Nicholas Campagna

Homework Project Writeup

I chose to start this project off by building off of my code for assignment 3. This was of course due to the fact that the project requires a lot of what was accomplished in assignment 3, including the need to create some of the same shapes and applying the same transformations. Since I made a compromise in assignment 5 where I essentially had two canvases, one for canvas API and one for WebGL, I decided to fix that issue by simplifying everything to use canvas API instead, including rewriting the render function to support this, as well as the default vertices used when clicking “Draw Shape”. To enable the user to draw on the canvas, I also implemented an “Enable Shape Drawing” button, which after being clicked, allows the user to mark points on the canvas for their shape. Most shapes just require two clicks to set up, triangles need a third click, but if the user is drawing a polygon or poly-line, they can click an infinite number of times for an infinite number of coordinates, and can use a right-click to finish their drawing. I also had to change the program to draw multiple shapes rather than just one, and to do this I created a “Shape” class, whose variables are “id” (for selecting which shape to transform when transforming), “shapeType” (for telling render() how to draw the shape’s vertices), “verts” (the shape’s vertices), and “r”, “g”, “b” for the shape’s color. As mentioned earlier, transforming can only apply to one of these shapes at a time, so to do that and fulfil the “selection” requirement, the user can input the ID of which shape to transform in the “Shape ID to transform” box. The following fields, “dX” and “dY” for Degrees of Transformation tell the program just how much a shape should be

transformed by (rotate only uses dX). Next are three fields for RGB values of the shape to be drawn. While this was extra credit, I didn't see too much trouble in implementing it, especially since it would make things look much better and distinguish shapes more. The "Transform Shape" button works as it did in assignment 3, but applies to the shape with the "selected" ID. "Clear" likewise works the same, but it clears all shapes. "Enable Shape Drawing" works as described, but prevents the use of the former three buttons to avoid potential crashes and problems. A user must either finish drawing a shape or select "Cancel Shape Drawing" to resume using the other three buttons. Also, to prevent frustration involving telling exactly where the canvas ended, I changed the background of the main page to a soft blue shade to make the canvas easily visible. "Download Canvas as JPEG" downloads the canvas and everything drawn on it as a JPEG - there was an issue where the background would be black, so instead of using clearRect for clearing/refreshing the canvas, I just used a big white rectangle covering everything. Saving and loading JSON files proved very tricky to me, though with enough time and research I properly implemented saving the array of Shape classes as a JSON file. Loading from JSON, however, was more difficult than expected. I've tried many methods I've looked at, but was unable to get anything functioning by the deadline. From what I've found, JSON files seem notoriously difficult to load from one's local storage for security purposes. Since this was only one requirement that was left unfinished, and since I had implemented the extra-credit RGB color feature, I decided to leave it as is and submit.