Report: Project A

A picture containing sign

Description automatically generated The primary objectives for Project A are to create several rigid 3-dimensional objects using WebGL and to make those objects interactive in interesting, non-trivial ways. Because any object in computer graphics is drawn with triangles, I tried to design my shapes in a such a way that they could be easily drawn with triangles. This made it easier for me to understand which vertices to input into my code as well as how to transform those shapes in the ways that I desired. The first shape that I designed was the 3D hexagram, which is commonly referred to as the “Star of David” for the followers of Judaism. I realized this shape was constructed from two different triangles, one upright and the other rotated and overlayed about the origin. Once I figured this out, I plotted the points on some graph paper and subsequently input those vertices into my code. To make the object appear 3-dimensional, I drew another copy of the same 2-dimensioal shape further back on the z-axis and filled the space in-between. I achieved the following result:

For a user interaction, I decided to use some of the mouse interactions from the ControlMulti.js code and apply it to the 3D hexagram so that it could be rotated with a click and drag. I decided to extend this a bit and included a double-click feature that allows the user to move the shape around the canvas and place it anywhere they would like.

A picture containing object, monitor, lit, dark

Description automatically generatedFor my second shape, I was inspired by the tetrahedron that was in the original ControlMulti.js code and decided to create something similar, except with a square base instead of a triangular base. This turned out to be a pyramid and it ended up looking like so:

A picture containing object

Description automatically generatedAs you can see, there is a star on the top, which I included as my third designed shape. This shape is also rather triangular in construction, and I refer to it in my code as a concave hexagon. I then decided to use this object to create an interesting multi-jointed assembly which I refer to as a “snake” in my code, but in practice, it looks more like a flower. I combined three of these concave hexagons each with the topmost point rotated all the way down in the negative direction. This topmost vertex then creates a joint with the next hexagon to form the “stem” of the flower. I then finish off the flower with my 3D hexagram. I was fairly pleased with the final result which you can see in the image above. Each joint of the stem moves independently of the other at differing rates which creates a neat effect. The “flower” at the top also bobs from side to side very slightly.

For the rest of my user interactions, I decided to use the keyboard to move my pyramid around the screen using the W, A, S, and D keys. These move the pyramid up, left, down, and right respectively. There is also an input box for the user to submit an angle which will instantaneously change the angle of the flower and the pyramid. There are also buttons at the bottom of the page that allow the user to increase the rate at which the pyramid spins as well as the rate at which the angle of the flower’s links change. The button in the center allows the user to play or pause the entire animation.

Overall, I think I learned quite a bit about how WebGL works as well as the matrix and vector mathematics behind computer graphics. I definitely appreciate my studies in linear algebra much more after seeing how it can be applied to create interesting shapes and animations. I hope I can build on these skills in the projects to come and discover even more about the world of computer graphics. Here is the final result:

Shape

Description automatically generated