1 Running the Application

Build instructions can be found in the README.md.

2 Creating the Application

The first thing that I did was to embed my ThreeJS application inside of a website, this allowed me to develop the application in the same way that I am used to working, as well as allowing me to abstract functionality for the application into several javascript files, use ES6 transcompiling to ES5, and it also gave me access to the npm repository to easily download and manage libraries.

I used the excellent 'canvas-testbed' package to create the HTML page + canvas for my application automatically; I had done it manually, however this was a cleaner solution and provided better CSS. I then used the 'three' and 'three-orbit-controls' packages to import the ThreeJS libraries I needed.

My plan was to recreate my bedroom in three dimensions, to start out I placed a series of Points on the canvas that highlighted the X, Y and Z axis. This helped me to visualise the different dimensions and orientate myself to the scene before I had actually built anything that I could use as a reference point.

The next thing I added was orbit controls, this allows the user to interact with the scene using a mouse and keyboard. This is essential to do whilst developing 3D models, as otherwise you have no easy way of viewing the models you have created quickly.

From here I attempted to manually create the walls and floors with Vertices and faces, however this proved to be a very slow method of building objects and I was keen to get something visual to work on top of.

This lead me to create my walls and floors with primitive shapes (this can be seen in the objects.js file), this was a lot quicker to develop and allowed me to finally have a room on display.

Once I had a room setup it was time to add some furniture, I decided that it wouldn't be time efficient to manually model these large items myself, so I used the online three.js editor https://threejs.org/editor/ to create a bed frame and a door.

These objects were then exported to JSON, which I was then able to add to my project and load in via the Three.js ObjectLoader function.

After seeing how easy it was to model objects in the three.js editor I decided to model my whole room this way. Creating the floor, walls, lighting and furniture was a lot easier this way so I quickly replaced the manually developed version (The code is still there though) with one ObjectLoader call that loads in the whole scene from a JSON file.

This approach seems a lot simpler and cleaner from an implementation standpoint, although I know that for this assignment you want to see us using primitives and vectors, so I have left the original code in the project which does produce effectively the same output as the code that loads in the scene from the editor.

I then decided to create my own complex shape, I decided upon the coffee mug on my desk. I first created a hollow cylinder to form the walls of the mug, then created a flat cylinder to form the base. To get the different cylinders to match up I shifted the vertices all down by the Y axis to meet with the end of the mug walls. I was then able to merge the geometries to create one whole mug geometry.

I added a green material texture and set the mug to receive and cast shadows, however when I had both of these options there appeared to be some rendering errors, so I had to opt to turn off the receiving shadows.

I now wanted to create my own handle for the mug using vertices I plotted myself. This proved to be rather challenging. My plan was to use a sine curve to generate a wave of points, then select one curve of the wave to use as my handle. I was able to this and generate four waves to make each side of the handle.

With the vertices created I now needed to add faces to them to create a wireframe that a material could be applied to. Unfortunately this is where I was unable to find a suitable solution. I could have manually added each face to the geometry but this seemed like an incorrect way of doing things as it would have been repeating code and been a painstaking manual process.

Instead I opted to use the triangulateShape function provided from three.js, this did allow me to create faces on the points, however I couldn't manage to get them to be created in the positions that I had imagined.