Rendering and manipulation of 3D models

Aditya Vardhan (IMT2019003)

What was done in the assignment?

1. We created 3D meshes for axes and other 3D objects, by making use of the boolean operations on the simple 3D objects(like cone, cylinder etc.) which were already provided by Blender. Then those meshes were exported as ".obj" files.

2. We imported the ".obj" meshes, using the "objLoader" and rendered them on screen using gl.drawElements(). There were some changes needed to be made in the "shader.js" file for rendering the imported 3D models, as the mesh info was provided in terms of list of unique vertices and their index data, which was different from our initial format of array of repeated vertices.

3. There are 2 modes:

   a. Mode 1: Top view
      i. A model can be selected by clicking on it. Then, it can be:
         1. Rotated(from -π to π) about X, Y and Z axis

         2. Translated(form -1 to 1) along X, Y and Z axis

         3. Scaled up and down

         4. Animated. The animation can be sped up or slowed down

   b. Mode 2: 3D view
      i. Users can choose a rotation axis for the camera(default is set to y-axis).

      ii. Users can then rotate the camera about the chosen axis.

Answers to the questions in the assignment document:

1. *To what extent were you able to reuse code from Assignment 1?*

   Almost all of it was reused in this assignment. Only some changes were made in "vertex.js" and "renderer.js" to incorporate the view and the projection matrices, and some extra functions were added, like the bindIndexBuffer() and drawElements() in "shader.js" to handle the new format of mesh data, provided in terms of vertices and indices.

2. *What were the primary changes in the use of WebGL in moving from 2D to 3D?*

   a. To visualise the 3D models better and change the camera angles, we added view and projection matrices. While changing the camera angle, we needed to update the parameters(such as "eye") of the view matrix.

   b. Since we had a large set of vertices, we had to store the meshes in vertex-index format. Hence, we used gl.drawElements() instead of gl.drawArrays().

   c. We needed to keep into account any updations that took place along or about the z axis, which was not the case with 2D.

3. *How were the translate, scale and rotate matrices arranged? Can your implementation allow rotations and scaling during the animation?*

   Order of transformations:
   a. Scale
   b. Rotate about Z
   c. Rotate about Y
   d. Rotate about X
   e. Translate

   Yes, we can rotate and scale the shape during animation.

4. *How did you choose a value for t1 in computing the coefficients of the quadratic curve? How would you extend this to interpolating through n points (n > 3) and still obtaining a smooth curve?*

   I chose t1=0.5 and solved the 3 quadratic equations for computing coefficients **a, b** and **c.** For n points, we'll be needing n-1 degree polynomials, instead of a quadratic polynomial, for interpolation.

NOTE: Youtube link of the video for this assignment: https://youtu.be/eXVWj8cejo4

References:

1. https://www.youtube.com/watch?v=IsyHfmq18ec

2. https://www.youtube.com/watch?v=MF1qEhBSfq4&list=PLa1F2ddGya_-UvuAqHAksYnB0qL9yWDO6&index=1

3. https://webglfundamentals.org