CS 606: Computer Graphics
Report for Assignment 2

Swasti Shreya Mishra (IMT2017043)
International Institute of
Information Technology Bangalore
Email: SwastiShreya.Mishra@iiitb.org

1. Introduction

The aim of this assignment is fourfold—1. To model 3D objects using Blender, 2. To develop an application that achieves 3D rendering with 3D translation, rotation and scaling of the rendered objects which are read from a .obj file, 3. To enable picking of objects and/or their faces, and 4. Camera rotation. The geometric shapes implemented in this project are cubes and a cylinder topped with a cone to specify axes. There is interactive GUI to switch between different modes of the application. Objects can be picked based on the mouse click position. Finally, the camera can be rotated about the x-axis using mouse drag.

2. Methods

The entire application is implemented using WebGL and dat.GUI is used for the GUI controls. Figure 1 depicts the controls of the application. The following are the possible operations on the objects (excluding axes) depending on the control button:

- **Object orientations**: There are three different object orientations—1. Origin: Here the 3 objects are present at the origin of the canvas, 2. Corners: Here the 3 objects are present at the 3 corners of an invisible equilateral triangle present at the origin, and 3. Sides: Here the 3 objects are present at the center of the edges of the invisible triangle mentioned in the previous setting.
- **Mode**: There are two different modes—1. Camera: Here the user can use mouse drag to rotate the camera about the x-axis, and 2. Selection: Here the user can click on an object to select an object and do operations like either changing the face selected face color or the entire object color.
- **Selection Mode**: This value is activated when the user is in the 'Selection' setting of the above specified mode. There are two selection modes—1. Face: Here when the user clicks on the object, the face that lies beneath the mouse click is selected and its color can be changed, and 2. Object: Here when the user clicks on the object, the entire object is selected and the color of all its faces can be changed.

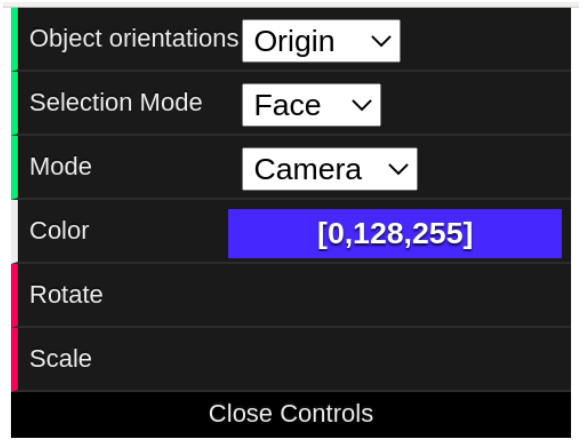

Figure 1. Controls of the application

- **Color**: This is a color picker in an RGB format. The user can choose a color using this color picker to change the color of faces/objects in the 'Selection' mode.
- **Rotate**: This performs the operation of rotating the 3 objects by 90 degrees along the x, y and z rotation axis respectively.
- **Scale**: This performs the operation of scaling the 3 objects by 0.5x, 2x, 3x respectively.

2.1. Parsing

The parser takes the text of the specified .obj file and parses it line by line to check for vertices, texture coordinates, normals, faces and other attributes. If the line starts with 'v', then the following coordinates are vertex coordinates and similarly, 'vt' for textures and 'vn' for vertex normals. The vertices are extracted and are subsequently stored in vertices which are passed into the vertexBuffer. The lines specifying the faces start with 'f' and are followed by v/vt/vn strings separated by spaces. These face coordinates are extracted and formatted (the coordinates might be duplicated) to represent the face values in the form of triangles. The coordinates for these triangles are stored in the corresponding order in the indices to generate data for

the indexBUffer. Using the vertices and indices, the objects can be rendered according to their geometry on the canvas.

Additionally, *webgl-obj-loader* [1] has been used for compound objects.

2.2. Picking

The picking of an object is done in two ways—picking of a face and picking of an entire object. If the mouse click coordinates lie on an object/face, that object/face is selected; if the mouse click coordinates do not lie on any object, there is no object selected and therefore the variable 'selected' in the application code is set to NULL.

The picking of objects and faces in the 3D space is implemented by using the 'A' value of the RGBA of the objects' pixels as a unique ID for objects and their faces. During selection using mouse click, the *gl.readPixels* function is used to read the pixels underneath the mouse click and the 'A' value is used to extract the object ID as well as its face ID.

The IDs are computed by multiplying object ID (specified during initialisation and goes as 1, 2, 3, and so on...) with a value α and then the face number (specified as per the order they appear in the faces array of the object) is added to it. Therefore, final object ID = $ID * \alpha + face\_num$.

This approach has its own drawbacks as total only 256 different IDs can be specified due to the constraints of the RGBA value of pixels. This is not scalable when we have a lot of objects in the scene. Also, this results in different 'A' values for each face, but, as in this assignment we do not specify lighting, I personally felt that this helps to distinguish different faces of an object. Also, we note that the 'A' values do not differ much as the ID remains same for one object. I have used α as 80 for this assignment.

The color change for a specific face/object is done by modifying the color data which is passed into the coloBuffer. Additionally, we also deleted the previous color buffer and reinitialize it in order to render the correct color for a face/object.

3. Answers To Assignment Questions

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

We could reuse the entire base code for the application from the previous assignment. Minor modifications to mesh object were done to adapt them to 3D rendering. The depth buffer had to be enabled explicitly. In place of **drawArrays**, **drawElements** was used in Assignment 2. New tranforms were added such as **perspective** (to specify the frustum) and **lookAt** for camera position. The remaining transform matrices (translation, rotation, scaling) remained the same. The shaders largely remained constant. A parser and GUI controls were added to the existing code base to achieve the effects desired for Assignment 2.

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

In order to achieve 3D rendering, an additional ELEMENT_ARRAY_BUFFER was introduced along with the existing ARRAY_BUFFER for vertices and colors for the respective vertices, which carries the indices of the faces that are supposed to be drawn. Along with this, 3D models can be loaded from specified files into WebGL for rendering. This makes it easier for rendering 3D models as it is quite difficult to input the vertex data manually. We also familiarized ourselves with perspective and view matrices. Due to these various transforms and the 3D nature of objects, the picking becomes inconvenient using the centroid approach as we did in the previous assignment. Therefore, different approaches have been discussed for 3D picking of objects/faces.

3.3. How were the translate, scale and rotate matrices arranged such that rotations and scaling are independent of the position or orientation of the object?

We always translate the object first to the origin, scale and rotate them and translate them back to their original location. This is done as rotation and scaling are by default done with respect to the origin, i.e. (0,0,0).

4. Discussion

There are other approaches are possible for picking. For example, we could differ faces colors by a small amount and assign it as an object's ID. But, this has other drawbacks such as, when we intend to change the color of a face of the selected object, the ID will have to be updated accordingly. I have read and understood different approaches, so as to implement it according to an application's requirements.

5. Conclusion

This assignment helped us learn the basics of 3D rendering using WebGL such as perspective and views. It improved our understanding of transformation operations using matrix algebra for camera rotations. We also learnt to use dat.GUI and implemented functions that execute when an event occurs. We can now build more complex 3D objects, render them using WebGL and add animations.

References

[1] Aaron Boman. frenchtoast747 / webgl-obj-loader, Apr 2020.