Project Documentation: 3D WebGL Renderer

**1.Introduction**

The name of the project is Cube.In this project we used WebGL 2 library of JavaScript programming language.There is a normal scene and also an interactive scene where cube is rotating.We used lightning,textures,camera ,user interface.We will describe some parts of our project In upcoming slides.

**2.Usage**

The 3D WebGL renderer allows users to interact with the rendered scene using mouse or touch controls. Here are some basic interaction instructions:

Rotate Scene:

Click and drag the mouse to rotate the entire scene.

For touch devices, swipe your finger to rotate.

Zoom In/Out:

Scroll up/down with the mouse wheel to zoom in and out.

For touch devices, use a pinch gesture to zoom.

Users can explore more advanced modifications based on their preferences and understanding of WebGL.

***Using ctrl key + mouse to rotate camera***

***Arrows to move camera***

**3.Interacting with the Scene**

Interactivity is a crucial aspect of any 3D graphics application, providing users with a dynamic and engaging experience. In the WebGL 3D Renderer project, interacting with the scene involves camera movement and user-controlled transformations. Here's a detailed overview:

**3.1 Camera Control**

The primary means of interaction is through the manipulation of the virtual camera. The camera allows users to view the 3D scene from different perspectives, enhancing the overall experience. In this project, the camera supports rotation along the X, Y, and Z axes.

Rotation Controls:

The RotateZ, RotateY, and RotateX functions in the code enable users to rotate the camera around the respective axes.

These rotations are based on the elapsed time, creating smooth and animated transitions.

User Input:

User input, such as keyboard or mouse events, can be integrated to provide real-time control over the camera's orientation.

Developers can extend the project to incorporate custom input handling for a more interactive experience.

**3.2 Object Transformations**

Apart from camera movement, the project supports transformations on multiple objects within the scene. These objects include a rotating cube and additional surfaces.

Surface Translations:

The surface\_matrix allows for translation of the surface along the X-axis.

Users can modify this matrix to move the surface to different positions within the scene.

Randomly Positioned Cubes:

The project generates an array of matrices (movementMatric) to randomly position multiple cubes within the scene.

Developers can further customize the positions, allowing for a diverse and visually appealing layout.

**4. Animating the Scene**

The animate function controls the animation loop, updating the scene at each frame. This function facilitates smooth transitions between different states, creating a visually appealing and dynamic 3D environment.

Delta Time (dt):

The concept of delta time (dt) ensures that animations remain consistent across different devices and frame rates.

Multiplying transformations by dt ensures that movements are proportional to the elapsed time.

**5. Extending Interactivity**

Developers are encouraged to extend interactivity by introducing additional features, such as:

Zooming:

Implement a zoom functionality to allow users to move closer to or farther away from the scene.

User Controls:

Integrate user controls using libraries like Three.js or custom input handling for more user-friendly interaction.

Object Selection:

Implement picking to enable users to select and interact with specific objects within the scene.

**6.Buffers**

A vertex buffer is a GPU memory buffer that stores the geometric data of vertices defining the 3D objects in the scene. In this project:

**6.1 Index Buffer**:

The index\_buffer is created similarly to the vertex buffer using gl.createBuffer() and loaded with indices using gl.bufferData().

Binding:

Like the vertex buffer, the index buffer is bound with gl.bindBuffer() before rendering.

**6.2 Texture Buffer**:

Although not explicitly discussed in this section, a texture buffer (boxTexture) is created to store the image texture used in the project. It follows similar principles of buffer creation, binding, and data loading.

**7.Animation**

**Cube Animation:**

4 cube objects (movement\_matrix, movement\_matrix2, movement\_matrix3, movement\_matrix4) undergo rotation along different axes, producing a visually interesting and dynamic effect.

The rotation angles are adjusted based on the elapsed time (dt), resulting in smooth and continuous motion.

The cameraObj instance manages the camera's movement and orientation. The Update and various Rotate methods applied to the camera contribute to the overall animation:

**Update Method:**

The cameraObj.Update() method is called in each animation frame, adjusting the camera's internal parameters.

**Rotation Methods:**

cameraObj.RotateX, cameraObj.RotateY, and cameraObj.RotateZ introduce rotations to the camera matrix, influencing the view of the scene.

**8.References**

justkurtle.github.io/farm3d/deps/gl-matrix.js  
This js file is for to able to reach running code(gl-matrix.js).

<https://opengameart.org/>

We use wood texture from this website.