CS 405 Project 3

Mine Ergin
26.12.2024

1. Introduction

This project's objective is to use WebGL to replicate a solar system. The Sun, Earth, Moon, and Mars are examples of celestial objects that should be rendered with appropriate scale, rotation, and hierarchical transformations. In order to improve the visual authenticity, the project also uses diffuse and specular lighting.

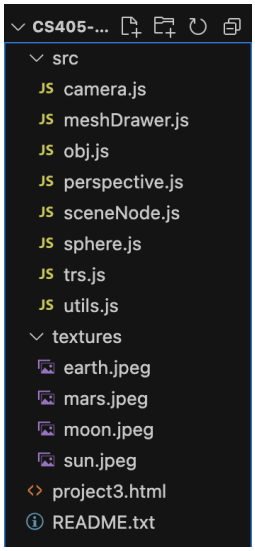
With an emphasis on rendering, transformation, animation, and lighting approaches, this study outlines the methodology utilized to create the simulation. It also discusses the difficulties encountered during development and how they were overcome.

2. Methodology

a. Project Setup

1. **Tools Used**:
    - **WebGL**: For rendering 3D objects in the browser.
    - **Python HTTP Server**: To bypass browser restrictions on texture loading (CORS policy).
    - **Visual Studio Code with Go Live**: For development and local testing.

2. Folder Structure:



b. Rendering Celestial Bodies

The following process was used for illustrating the Sun, Earth, Moon, and Mars.

1. **MeshDrawer Class**: A helper class (`MeshDrawer`) was used to parse sphere objects and create vertex buffers.
2. **Textures**: Textures for each celestial object were loaded using the `setTextureImg` function, and the images were stored in the `textures/` folder.
3. **Parent-Child Relationship**:
    - Earth,Moon were children of the Sun.
    - Mars was also set as a child of the Sun.
    - This relationship was achieved using the `SceneNode` class, which allows transformations to propagate hierarchically.

c. Transformation and Animation

1. **Rotation**:
    - Celestial objects rotate around their respective axes.
    - The rotation speeds were calculated based on time elapsed.

```
var zRotation = timeOffset * 40 * Math.PI / 180;
earthNode.trs.setRotation(0, 0, zRotation * 2);
marsNode.trs.setRotation(0, 0, zRotation * 1.5);
```

2. **Translation and Scaling**:
    - The TRS (Translation, Rotation, Scaling) class was used to apply transformations.

    For example, Mars was translated by `-6` units on the X-axis and scaled to `0.35`:
    ```
    marsTrs.setTranslation(-6, 0, 0);
    marsTrs.setScale(0.35, 0.35, 0.35);
    ```

d. Lighting

Diffuse and specular lighting were implemented in the fragment shader to improve realism:

1. **Diffuse Lighting**: Based on the angle between the light source direction and the surface normal.
    ```
    diff = max(dot(normal, lightdir), 0.0);
    ```

2. **Specular Lighting**: Added a reflective highlight based on the viewer's direction:
   ```
   vec3 reflectDir = reflect(-lightdir, normal);
   spec = pow(max(dot(viewDir, reflectDir), 0.0), phongExp);
   ```
3. **Fragment Shader Example**:

   ```
   vec3 lightPos = vec3(0.0, 0.0, 5.0);
   vec3 lightdir = normalize(lightPos - fragPos);
   float ambient = 0.35;

   gl_FragColor = texture2D(tex, vTexCoord) * (ambient + diff +
   spec);
   ```

e. Local Server Setup

Since modern browsers block texture loading from local file paths due to CORS restrictions, a local server was set up:

Python HTTP Server:
`python3 -m http.server`

- **Testing**: The project was accessed via `http://localhost:8000/project3.html`.

f. Challenges and Solutions

1. **CORS Errors**: External texture URLs (e.g., Imgur) resulted in `403 Forbidden` errors. Solution: Download all textures and stored them locally in the `textures/` folder.
2. **Go Live Issues**: The `GoLive` server failed to load textures properly. Solution: Preferred Python HTTP Server for consistent results.
3. **Lighting Tuning**: Balancing diffuse and specular lighting required multiple iterations.

3. How to Run the Project

This section provides clear steps for running the project (also, I added details in README file):

1. **Setup**:
   - Ensure all files (HTML, JS, and texture files) are in the correct folder structure as shown in the **Project Folder Structure** section.
2. **Run Using Python HTTP Server**:
   - Open a terminal
   - Run the following command to start a local server:
     `python3 -m http.server`

Open your browser and navigate to:
http://localhost:8000/project3.html

Or

http://[::]:8000/project3.html

3. **Inaccurate link: Runing Using VS Code Extension Go Live**:
    - Open the project folder in Visual Studio Code.
    - Start the **Go Live** extension from the bottom toolbar.
      `http://127.0.0.1:5500/project3.html`
    - Note: This extension and link gives inaccurate results

4. Results

5. Conclusion

This project effectively illustrated WebGL illumination, object rendering, and hierarchical transformations. Even though there were issues with lighting balance and CORS failures, they were fixed with careful implementation and debugging.

Future enhancements might include more planets, orbiting routes, or user-interactive controls.