Koray Aslan
28180

Introduction
In this project, I was tasked with implementing various features in a WebGL-based solar system simulation. The project was divided into three distinct tasks, each focusing on a specific aspect of WebGL programming, including scene graph implementation, shader programming, and hierarchical transformations.

Task 1: Implementing the Scene Graph Draw Function
Objective: The primary goal in Task 1 was to implement the `draw` function within the `sceneNode.js` file. This function is crucial for ensuring that transformations applied to parent nodes in the scene graph propagate correctly to their child nodes.

Methodology:
1. I started by analyzing the existing `SceneNode` class, which serves as the foundation for the scene graph structure.
2. I then implemented the `draw` function, ensuring that it recursively applied transformations from parent to child nodes. This involved:
   - Computing new transformation matrices by combining the current node's transformations with its parent's.
   - Drawing the current node's mesh using the updated transformation matrices.
   - Recursively calling the `draw` function on each child node.

Result: The implementation successfully propagated transformations down the scene graph. The rendered output correctly displayed the hierarchical relationship between nodes, with parent transformations affecting their children. This was visually verified through the proper arrangement and orientation of celestial objects in the solar system simulation.

Task 2: Shader Programming for Lighting Effects
Objective: In Task 2, my objective was to update the fragment shader in the `meshDrawer.js` file. The initial shader only supported ambient lighting, and my task was to enhance it to include diffuse and specular lighting effects.

Methodology:
1. I first reviewed the existing shader code to understand its structure and how it was integrated with the rest of the WebGL code.
2. I then modified the fragment shader by implementing the following lighting calculations:
   - Diffuse Lighting: Calculated using the dot product between the light direction and the surface normal. This created a more realistic representation of how light interacts with surfaces at different angles.
   - Specular Lighting: Implemented using the Phong reflection model, which considers the viewer's perspective, resulting in bright spots on shiny surfaces.

3. I carefully adjusted the strength and shininess parameters to achieve a visually appealing balance of diffuse and specular effects.

Result: The shader enhancements significantly improved the visual realism of the scene. The celestial bodies in the solar system simulation exhibited more dynamic and realistic lighting, demonstrating the impact of both diffuse and specular components on the surfaces.

Task 3: Adding Mars to the Solar System
Objective: The final task involved adding Mars as a new celestial body in the solar system. This required creating a new scene node for Mars and correctly positioning, scaling, and texturing it.

Methodology:
1. I created a `MeshDrawer` instance for Mars, utilizing the sphere mesh used for other planets.
2. I applied the Mars texture from the provided URL to this new mesh.
3. I instantiated a `TRS` object for Mars, setting its translation, rotation, and scale as per the task requirements.
4. I added Mars to the scene graph as a child of the Sun node.
5. Within the `renderLoop` function, I updated Mars's rotation to be 1.5 times that of the Sun's rotation.

Result: Mars was successfully added to the solar system simulation. It was correctly positioned as a child of the Sun and displayed the appropriate texture and scale. The rotation implementation ensured Mars's movement was in sync with the rest of the celestial bodies, enhancing the dynamism of the scene.

Conclusion
Throughout this project, I employed a methodical approach to solve complex problems in WebGL programming. Each task built upon the previous one, cumulatively resulting in a comprehensive solar system simulation. The project not only demonstrated my technical abilities in WebGL but also highlighted the importance of an iterative and detail-oriented approach in graphical programming. The final simulation effectively showcased a dynamic and visually compelling representation of a solar system, complete with realistic lighting and hierarchical transformations.