Emre Zeytinoğlu 28190

CS405 PROJECT 3

Task 1

This section explains the application requested in Task 1, that is, the process of implementing the draw function for the SceneNode class.

**Combining Transformation Matrices:**

The Draw function combines the local transformation (TRS) of each SceneNode with the transformation of the parent node. This reflects each node's own transformation as well as its hierarchical position within the scene.

**Updating Matrices:**

Using the combined model matrix, mvp (Model-View-Projection), modelView and normalMatrix are updated. These matrices are used to accurately render and display the mesh in 3D space.

**Mesh Drawing:**

If a SceneNode has a meshDrawer, it will be drawn with the updated matrices. This ensures that the visual representation of each node is rendered accurately.

**Recursion for Child Nodes:**

The Draw function is called recursively for each child node. This ensures that all nodes at each level of the scene graph are rendered and represented correctly.

This approach ensures that each element of the scene graph is correctly transformed and rendered, creating a consistent and accurate visual output in 3D graphics.

Task 2

This report details the update of the fragment shader used in our solar system visualization project. This update was made to account for diffuse and specular lighting effects, thus increasing visual realism.

**Ambient Lighting:**

I added a basic light component that illuminates all objects with the same intensity. I control the effect of this light with the ambientStrength factor.

**Diffuse Lighting:**

Calculated based on the angle between the light direction and the normal to the surface. The light was adjusted to be more intense when it hit the surface directly, and less intense when it hit it at an angle.

**Specular Lighting:**

Highlights highlights where light reflects directly into the camera. specIntensity varies depending on the angle between the reflection direction and the viewing direction, and the exponent controls the extent of specular brightness.

**Light Source Control:**

If isLightSource is true, it just uses the texture color. This is ideal for rendering light sources where you don't want to apply the same lighting effects.

With this update, the objects in our visualization project are displayed with more realistic lighting, thus enriching the user experience. In future developments, it is aimed to increase the visual quality of the project by further improving the lighting model.

Task 3

This part includes improvements to the solar system visualization performed by me. My main goal was to add Mars to the existing system and give this planet rotational motion.

**Addition of Mars:**

To represent Mars, I created a new SceneNode called marsNode. This element contains the visual representation (marsMeshDrawer) and position (marsTrs) of Mars. I used a MeshDrawer for the visual properties of Mars, and this element uses the existing sphere data for the spherical configuration. To add a Mars-specific texture, I used the setTextureImg function and provided an image link here. Mars was positioned relative to the sun. With the expression marsTrs.setTranslation(-6, 0, 0) this positioning was set to a -6 unit shift on the X axis relative to the sun. I set the scaling of Mars with the expression marsTrs.setScale(0.35, 0.35, 0.35).

**Rotational Movement:**

I updated the renderLoop function and added a rotation to Mars motion. This movement allows Mars to rotate around its axis. I set the rotation speed with the formula 1.5 \* zRotation.

With these improvements, my solar system visualization now includes Mars and successfully simulates the rotation of this planet around its axis.