CS 405 Project 3: Scene Graph + Illumination

December 2024

Filiz İpek Oktay - 30780

The assignment requires modifying a scene graph implementation to simulate **a solar system**. Tasks include implementing hierarchical transformations, adding lighting with shaders, and integrating a new planet Mars with specified transformations and textures.

Task 1

In task 1, the **draw function** in the SceneNode class has been implemented.

The function handles hierarchical transformations and rendering within the scene graph. It begins by initializing transformation matrices (mvp, modelView, normalMatrix, and modelMatrix) passed to the function. These matrices are updated by multiplying them with the current node's transformation matrix (this.trs.getTransformationMatrix()). This ensures that transformations such as translation, rotation, and scaling are applied **hierarchically**, propagating from parent nodes to child nodes. If the current node contains a meshDrawer, it uses these updated matrices to render the node, applying the appropriate transformations and lighting.

The function then recursively processes all child nodes by calling their draw methods with the updated matrices. This approach ensures that each child node inherits the transformations of its parent while applying its own transformations. The recursive structure enables a tree-like rendering of the scene graph, where objects like planets and moons are transformed and rendered relative to their parent nodes, simulating a solar system with proper hierarchical relationships.

Scene Graph Hierarchy: In the HTML code, the hierarchical structure ensures that the earthNode is transformed relative to the sunNode, and the moonNode is transformed relative to the earthNode. The recursive rendering system uses this hierarchy which is visualized below:

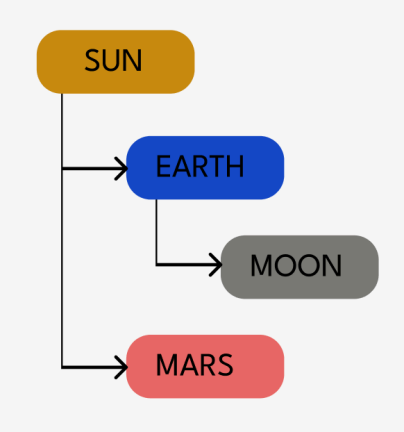

Figure 1: *Scene Graph Hierarchy*

Task 2

This task, updates the fragment shader to implement diffuse and specular lighting, enhancing realistic lighting effects in the scene.

In the mesh drawer, diffuse and specular lighting were applied following the **Phong reflection model**. For diffuse lighting, the dot product of the normalized surface normal (normal) and the light direction (lightdir) determines the intensity of light hitting the surface. This ensures that the lighting is angle-dependent, simulating how light illuminates a surface based on its orientation. The “max” function ensures that negative values (caused when light is behind the surface) are clamped to zero, preventing unrealistic effects.

For the specular lighting, the light direction is reflected around the normal using the “reflect” function, giving the direction of the reflected light (reflectDir). The viewDir, the normalized vector from the fragment to the camera, is then compared to the reflectDir using their dot product. The result is raised to the power of phongExp, controlling the sharpness of the highlight. This logic produces a realistic specular highlight, dependent on the viewing angle, material shininess, and light direction.

Task 3

In this task, Mars has been added as a SceneNode child of sunNode, consistent with the scene graph hierarchy described earlier. Transformations including translation (-6 units along the X-axis), scaling (uniformly to 0.35), rotation (1.5 times the Sun’s rotation), and texture mapping are applied to achieve realistic rendering. A new MeshDrawer is created to handle Mars’s geometry, utilizing the pre-defined sphere mesh in the project. Mars's transformations are managed through a TRS object, applying the specified translation and scaling. The texture for Mars is set using the setTextureImg function with the provided URL. Finally, Mars is added as a child of sunNode, establishing its place within the solar system hierarchy.

In the renderLoop function, Mars’s rotation about the Z-axis is dynamically updated to be 1.5 times the Sun’s rotation, ensuring its motion is synchronized with its parent node. This hierarchical structure allows Mars to inherit transformations from the Sun while maintaining its distinct scale, position, and rotation properties. The recursive draw function ensures that Mars is rendered appropriately alongside other nodes, seamlessly integrating it into the solar system simulation as a dynamically rendered object.

Unexpected Behavior of Mars:

After implementing Task 3, it was observed that the texture of Mars shows a noticeable irregularity at a specific point during its rotation around the Sun. This behavior is not a mistake or error but a natural outcome of the given rotation logic, which is based on the **time offset** and **rotational speed**. The rotation angle, derived from the timeOffset, experiences periodic resets due to the modulus operation, leading to abrupt transitions in the transformation matrix and temporary texture irregularities. Extensive testing was conducted by changing the texture, altering the rotation parameters, and comparing Mars's behavior with other planets. These tests confirmed that the observed issue is specific to Mars's rotation settings and not indicative of a problem with the implementation or texture mapping. The behavior aligns with the expected results under the given parameters and conditions.