Project 3: Scene Graph and Illumination - Technical Report

Let me rewrite the report in three detailed paragraphs focusing on each task's methodology.

The first task involved implementing the scene graph functionality through the SceneNode class's draw method, which was crucial for establishing proper hierarchical transformations in our solar system simulation. The implementation required careful consideration of matrix multiplication order and transformation propagation from parent to child nodes. The key aspect was ensuring that each node's local transformation matrix was properly combined with its parent's transformations, which was achieved through sequential matrix multiplications. Specifically, we took the node's transformation matrix obtained from the TRS (Translation-Rotation-Scale) object and multiplied it with the incoming model, view, and projection matrices. This multiplication order was critical - we multiply the incoming matrices with the node's transformation matrix to ensure proper transformation hierarchy. The draw method also handled the recursive traversal of child nodes, passing the transformed matrices down the hierarchy, which allowed transformations to properly cascade through the scene graph. This meant that when the Earth moved, for example, the Moon would automatically follow while maintaining its relative position and orientation.

The second task focused on implementing proper lighting calculations in the fragment shader using the Phong illumination model, which consists of ambient, diffuse, and specular components. The key challenge was implementing the diffuse and specular calculations within the designated section of the shader code. For diffuse lighting, we calculated the dot product between the normalized surface normal and the light direction vector (diff = max(dot(normal, lightdir), 0.0)), which gives us the cosine of the angle between these vectors. The max function ensures we don't get negative lighting values, as negative light doesn't make physical sense. For specular highlights, we implemented the calculation by first computing the view direction as the normalized vector from the fragment to the camera (normalize(-vPosition)), then calculating the reflection direction using GLSL's reflect function with the light direction and surface normal. The specular component was then computed using the dot product of the view direction and reflection direction,

raised to a power (phongExp) to control the size and sharpness of the highlights. This implementation created realistic lighting on our celestial bodies, with the sun being handled specially as a light source.

The third task involved extending our solar system by adding Mars as a new celestial body, which demonstrated the extensibility of our scene graph implementation. The implementation followed specific requirements for Mars's properties, including its position, scale, and rotation relative to the sun. We created a new MeshDrawer instance for Mars, using the same sphere mesh as other celestial bodies but with Mars-specific texture mapping. The transformation setup was crucial - we created a TRS object that positioned Mars 6 units away from the sun on the negative X-axis, scaled it uniformly by 0.35, and implemented its rotation to be 1.5 times the sun's rotation rate. Mars was added as a direct child of the sun node in the scene graph, ensuring it would orbit around the sun while maintaining its own rotation. The implementation demonstrated how our scene graph structure could easily accommodate new celestial bodies while maintaining proper hierarchical relationships and transformations. The result was a visually coherent addition to our solar system, with Mars exhibiting proper orbital motion and lighting effects consistent with the existing celestial bodies.