**Project Report: Scene Graph and Illumination Implementation**

**by Kaan Karahan**

**Introduction**

This report outlines the methodology and implementation details for CS 405 Project 3: Scene Graph and Illumination. The project’s objective was to create a solar system simulation using scene graph hierarchies, implement advanced lighting techniques, and expand the system by integrating a new celestial body. The project was developed using JavaScript and WebGL.

**The three primary tasks included:**

1. Implementing a scene graph's hierarchical draw method.
2. Enhancing the fragment shader to support diffuse and specular lighting.
3. Adding Mars as a node in the scene graph, ensuring proper texture, transformations, and behavior.

**Task 1: Scene Graph Implementation**

**Objective**

To implement the draw method in the sceneNode.js file, enabling hierarchical transformations to propagate from parent nodes to their children.

**Approach**

1. **Understanding the Scene Graph Structure:**
   * Each node represents an object in the scene, holding transformation data and references to its children.
   * A hierarchical system where transformations (translation, rotation, scaling) applied to a parent node affect all its children.
2. **Implementation Details:**
   * The draw method was designed to:
     + Multiply the parent’s transformation matrix with the child’s local transformation matrix.
     + Pass the resulting matrix to the WebGL rendering pipeline.
   * Recursive traversal was used to ensure transformations propagated through the graph hierarchy.

**Challenges and Solutions**

* **Challenge:** Ensuring correct matrix multiplication order.
  + **Solution:** Verified against examples and tested using debug outputs.
* **Challenge:** Handling multiple children’s nodes.
  + **Solution:** Used recursive functions for clean traversal.

**Outcome**

The solar system’s hierarchical structure was successfully implemented, visualizing the Sun, Earth, and Moon with their relative transformations.

**Task 2: Lighting Implementation**

**Objective**

To enhance the fragment shader in meshDrawer.js to calculate diffuse and specular lighting in addition to ambient lighting.

**Approach**

1. **Understanding Lighting Models:**
   * **Ambient Lighting:** Constant light affecting all surfaces equally.
   * **Diffuse Lighting:** Simulates light scattered uniformly from a rough surface.
   * **Specular Lighting:** Simulates reflective highlights on shiny surfaces.
2. **Shader Modifications:**
   * **The fragment shader was updated to calculate lighting using the following steps:**
     + Compute the normalized light direction vector.
     + Compute the diffuse intensity using the dot product of the light vector and the surface normal.
     + Compute the specular intensity using the reflection vector and the view direction.

**Challenges and Solutions**

* **Challenge:** Debugging the fragment shader without detailed error outputs.
  + **Solution:** Used incremental testing for each lighting component.
* **Challenge:** Adjusting specular shininess for realistic results.
  + **Solution:** Experimented with various shininess values and observed results.

**Outcome**

The lighting system now realistically simulates ambient, diffuse, and specular effects, significantly improving the visual realism of the solar system.

**Task 3: Adding Mars to the Scene**

**Objective**

To add Mars to the scene graph as a child of the Sun, with appropriate transformations, texture mapping, and rotational behavior.

**Approach**

* **Node Configuration:**
  + Mars was created as a sphere mesh object.
  + The following transformations were applied:
    - **Translation:** −6 units on the X-axis relative to the Sun.
    - **Scaling:** 0.35 units for uniform scaling.
    - **Rotation:** 1.5 times the Sun’s rotation speed.

**Challenges and Solutions**

* **Challenge:** Ensuring Mars’s rotation remained synchronized with the Sun.
  + **Solution:** Linked Mars’s rotation to the Sun’s rotation multiplier in the render loop.
* **Challenge:** Applying the texture correctly.
  + **Solution:** Verified texture coordinates and debugged WebGL calls.

**Outcome**

Mars was successfully added to the solar system simulation, enhancing the visual complexity and showcasing hierarchical transformations.

**Overall Challenges and Learnings**

1. **Challenges:**
   * Understanding and implementing matrix transformations.
   * Debugging WebGL shaders without detailed error messages.
   * Managing texture mapping and scene graph hierarchy.
2. **Learnings:**
   * Scene graph hierarchies enable efficient transformations and rendering.
   * Lighting models significantly enhance visual realism.
   * Incremental development is critical for complex graphical systems.

**Conclusion**

**The project was successfully completed, fulfilling all objectives:**

1. The scene graph’s draw method was implemented to support hierarchical transformations.
2. The fragment shader was updated to support diffuse and specular lighting, improving realism.
3. Mars was integrated into the solar system, demonstrating scene graph expansion and texture mapping.

This project provided valuable insights into scene graph implementation, shader programming, and hierarchical transformations in computer graphics.

**References**

* **Mars texture:** [**https://i.imgur.com/Mwsa16j.jpeg**](https://i.imgur.com/Mwsa16j.jpeg)
* **My Github Link For The Course: https://github.com/KaanKarahann/cs405**