Introduction

Project Overview

This document provides a comprehensive overview of the Solar System simulation project, showcasing the orbits and rotations of the planets around the sun. It covers the motivation behind the project, the choices made during development, the algorithms and techniques used, detailed implementation steps, and a performance analysis of the final application.

Objectives

- Create a 3D scene of the solar system with interactive camera controls
- Implement real-time lighting and texturing for realistic rendering
- Optimize performance to maintain a high frame rate

Project Details and Choices

Coordinate System

- Positive X axis points right
- Positive Y axis points up
- Positive Z axis points “outside” the screen

Key Libraries

- GLFW: handles window creation, context management, and input
- GLAD: manages OpenGL function pointers
- GLM: provides mathematical operations for transformations
- STB Image: Loads images for textures
- Shader.h: handles shaders
- Model.h: handles models
- Camera.h: handles camera operations

Algorithms and Techniques

Camera control

- The camera class manages user input for moving around the scene. It supports forward, backward, left, and right movements, and rotates based on mouse movement. Diagonal movements are compensated to maintain consistent speed, ensuring smooth and intuitive navigation through the 3D space.

Object transformation

- Each celestial body in the scene is represented by a model. Transformations for positioning, rotating and scaling are applied using GLM matrices. Each planet orbits around the sun and rotates around its own axis. These transformations are calculated using time-based delta values to ensure consistent and smooth animations.

Lighting

- The scene uses Blinn-Phong lighting for realistic shading. Key parameters include ambient, diffuse, and specular components, as well as the shininess coefficient. These parameters allow for detailed control over how light interacts with surfaces, creating realistic reflections and highlights.

Texture mapping

- Textures are loaded using STB image and applied to models. Cube maps are used for environment mapping to simulate a surrounding skybox. Each planet has a unique texture that enhances its visual realism. The sun, for instance, has a distinct texture to simulate its fiery surface.

Shaders

- **Illumination shader:** manages lighting calculations, including diffuse and specular components.
    - **Vertex shader:** processes each vertex of the 3D models. Performs transformations and prepares data to eb passed to the fragment shader.
    - **Fragment shader:** calculates the color of each fragment based on lighting, material properties and texture.

- **Skybox shader:** renders the background environment map, giving a sense of depth and immersion.

Implementation Details

Initialization

- The application initializes GFLW, creates a window, and sets up the OpenGL context. GLAD is used to load OpenGL functions. Depth testing is enabled to ensure proper rendering of 3D objects, and the **clear color is set to a light blue to represent the sky.**

Shader Setup

- Shaders are compiled and linked using shader class. Subroutine indices are retrieved and stored for shader swapping. This allows dynamic changes in rendering techniques, such as switching between different lighting models.

Model and texture loading

- Models are loaded using model class, and textures are loaded with STB image. Texture IDs are stored for easy binding during rendering. Each model corresponds to a celestial body, and textures are mapped accordingly to enhance visual detail.

Rendering Loop

- The main rendering loop updates the view and projection matrices based on camera position, clears the buffers, and renders each object in the scene. This loop ensures that the scene is continuously updated and rendered at a smooth frame rate.

Lighting and texture application

- Lighting parameters and textures are set in the shader before drawing each model. The Blinn-Phong model is primarily used for diffuse and specular

calculations, which determine how light interacts with surfaces. This adds realism to the visual appearance of the celestial bodies.