CGD6214 COMPUTER GRAPHICS FUNDAMENTALS

FINAL PROJECT

TECHNICAL DOCUMENTATION

TT1L GROUP 1

Members:

Name	ID
Wisyal Faridz Aimizil bin Mohd Fauzi	1221304904
Imran Faizal	1221301693

1. Architecture Overview

1.1 System Design

Our application is a real-time 3D environment built using OpenGL. It combines advanced modelling, lighting, shading and terrain systems to create an interactive forest-and-village simulation.

The rendering pipeline follows this flow:

```
Models/Assets → Scene Graph → Shader Programs → Render Loop → Screen
```

Key Components:

Lighting System

- Supports directional light (sun), point lights (lamps), and optional spotlight (flashlight).
- Shadow mapping with depth textures for realism.
- Dynamic day/night cycle adjusts light direction and color.

Shader System

- Vertex and fragment shaders handle Phong lighting, fog, and shadow blending.
- Depth shaders used for shadow mapping pass.

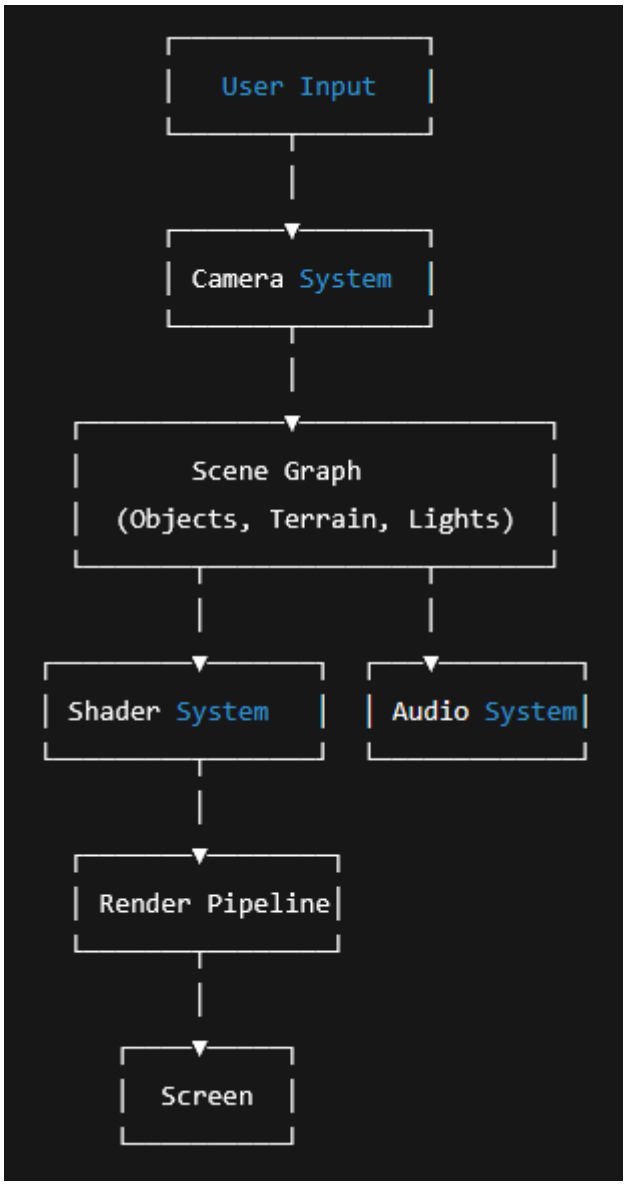
Audio System

- Ambient sounds (birds, wind) looped using OpenAL/irrKlang.

Input and Navigation System

- First-person camera navigation with WASD and mouse.
- Flashlight toggle via F.

1.2 Component Interaction Diagram



2. Technical Implementation

2.1 Heightmap Terrain

- **Loading:** Heightmap PNG is read via stb_image.h. Pixel brightness values are mapped to vertex heights.
- **Mesh Generation:**

- Each pixel generates a vertex (x, height, z).
- Neighbor sampling computes normals for smooth lighting.
- Indices form a triangle grid (two triangles per quad).

- **Usage:** The terrain is positioned at the edge of the world to act as **mountains**.

2.2 Forest Wall & Instancing

- Trees, rocks, and ferns are randomly positioned along the boundary.
- Small random rotations and scaling add natural variation.
- This ensures efficient rendering while avoiding uniform placement.

2.3 Lighting & Shading

- **Phong Model:**
    - Ambient, diffuse, and specular components.
    - Directional light (sun), two point lights, and one spotlight (flashlight).
- **Fog:**
    - Exponential attenuation:

```
fogFactor = exp(-pow(distance * density, 2.0));
finalColor = mix(fogColor, sceneColor, fogFactor);
```

- **Shadow Mapping:**
    - Depth framebuffer (1024x1024) stores scene from light's POV.
    - PCF (Percentage Closer Filtering) averages neighboring depth samples.
    - Bias avoids shadow acne.

2.4 Day/Night Cycle

- Directional light color interpolates from yellow (day) to blue (night).

- Light intensity decreases at dusk, increases at dawn.

- Skybox tint adjusts accordingly.

3. Team Contribution Matrix

Member	Contributions
Wisyal	-Environment modeling (trees, rocks, bushes, flowers, farmhouse) - Scene management and instancing system (forest wall, random placement) - Heightmap terrain (mountains at world edge) - Lighting and shading (Phong lighting, multiple light sources, fog system) - Shadow mapping implementation - Texture loading and ground texturing - Ambient audio integration (birds, wind) - Performance tuning (instancing, culling, LOD) - Technical documentation drafting
Imran	- Flashlight spotlight implementation - Day/night cycle system (dynamic directional light, smooth transitions) - Minor shader adjustments for dynamic lighting - Testing and debugging integration with your environment systems

4. Advanced Features Documentation

4.1 Shadow Mapping

- **Pass 1:** Render scene from light's POV into a depth map.
- **Pass 2:** Sample shadowMap in fragment shader.
- Uses PCF with 3×3 sampling to soften shadow edges.

4.2 Fog System

- Implemented as exponential fog in fragment shader.
- Blends scene with sky/fog color at distance.

4.3 Heightmap Terrain

- Provides mountains at the boundary of environment.
- Vertices displaced according to grayscale pixel intensity.

4.4 Ambient Audio

- Birds chirping and wind effects play continuously.

5. Performance Analysis

5.1 Frame Rate Testing

- Hardware: RTX 2060, Ryzen 5 3600, 16GB RAM
- Scene: 500+ objects, 1 farmhouse, terrain, shadows, fog
- Results (1080p):
    - Shadows + Fog: 80–110 FPS
    - No shadows: 120+ FPS
    - With heightmap terrain: ~75–90 FPS

5.2 Optimization Strategies

- **Frustum Culling:** Skip objects outside camera frustum.
- **VAO/VBO Management:** All geometry stored in GPU buffers.
- **Instancing:** Used for forest wall to reduce draw calls.
- **Level of Detail (LOD):** Low-poly models used for distant trees.

6. User Manual

Controls

- WASD → Move camera

- Mouse → Look around

- Scroll → Zoom

- F → Toggle flashlight

- ESC → Quit application

Features

- Day/night cycle plays automatically.

- Fog increases at distance.

- Shadows visible for house, trees, and terrain.

- Ambient sounds loop in background.

7. Developer Guide

7.1 Compilation

Prerequisites:

Windows: Visual Studio 2019/2022 with C++ and OpenGL development libraries installed

---

Compilation Instructions:

Windows (Visual Studio)

Open Project1.sln in Visual Studio.

Ensure that assimp, GLEW, GLM and GLFW include/library directories are correctly set in project properties:

C/C++ → General → Additional Include Directories

Linker → General → Additional Library Directories

Build the project using Ctrl + Shift + B or Build → Build Solution.

The executable will be located in the build folder.

Running Instructions:

Ensure your assets folder (models, textures, shaders) is in the same directory as the executable.

Launch the program:

Windows: Double-click Project1.exe or run via Visual Studio.

Notes:

Ensure your GPU drivers support OpenGL 3.3 or higher.

If the program shows a black screen, check that the shader files are correctly located and paths are valid.

All transformations, animations, and lighting calculations are handled in the respective shaders and main loop.

7.2 Adding Assets

- Place models in /assets/models/
- Place textures in /assets/textures/
- Update renderScene() in main.cpp to include new objects.

7.3 Shader Editing

- /shaders/ folder contains GLSL files:
    - lighting.fs → main fragment shader
    - depth_shader.fs → shadow pass
    - skybox.fs → skybox rendering
- Modify parameters in shader uniforms for quick adjustments.