Universitatea Technica

din Cluj-Napoca

Graphic Processing Project

Wildwood Refuge scene

STUDENT: Moga Diana Marcela

GROUP: 30435

1. Contents:

2. Subject Specification

This documentation describes a 3D computer graphics project that captures a serene natural landscape, titled Wildwood Refuge. The scene is set in a tranquil environment where nature and human habitation coexist in harmony. It features a variety of animals, such as graceful swans gliding across the water, deer peacefully grazing in the orizont, and ducks moving about the area. Scattered across the landscape are charming cottage houses, nestled in the natural surroundings. The cottages are placed strategically near the water, nestled in the forest, offering a blend of rural and natural beauty. The terrain is complex with various heights, a lake slopes, and dense woods that create a serene, idyllic atmosphere. The houses are diverse in style, reflecting handcrafted details like unique roof designs and cozy entrances. The overall scene portrays a peaceful refuge, where wildlife and humans live side by side in a harmonious, undisturbed setting. The surrounding nature, with its forests, waters, and wildlife, enhances the sense of serenity and tranquility.

3. Scenario

3.1. Scene and Objects Description

The scene takes place in nature featuring:

- Some rustic houses surrounded by trees.
- A windmill with animated proppeler.
- Animals like deers, swans, ducks, and a dog.
- A lake.
- A campfire.
- A wheelbarrow.

Each object in the scene is rendered with high-quality textures and casts realistic shadows, influenced by the directional light of the main sun. To further enhance the atmosphere, the scene has an option for fog. Users can also adjust the direction of the sunlight. The best way to explore the key elements of the scene is through the camera animation, which is available on demand.

Figure 1 1 Scene Overview

3.2. Functionalities

1. Camera Control and Animated Tour

The user has the ability to freely navigate the scene, moving forward, backward, sideways, up and down as well as rotate the camera using mouse look.
Additionally, an animated tour of the scene guides the user's viewpoint along key locations, highlighting important landmarks.

2. Directional and Point Lights

The sunlight (directional light) serves as the main source of illumination, with its angle controlled via the keyboard. It is the only light source that casts shadows in the scene. Additionally, point lights representing the campfire can be toggled on and off using the keyboard.

3. Shadows

Shadow mapping is used to cast realistic shadows by directional sunlight. Point light doesnt cast shadows.

Figure 1 2 Firecamp point light and shadows

4. Fog

A fog system is implemented in the scene, which can be toggled on or off, adding to the atmosphere of the natural landscape. The fog is intentionally thick and spans the entire map, creating a mystical and serene effect that enhances the beauty and tranquility of the environment.

Figure 1 3 Fog effect

5. Animated proppeler

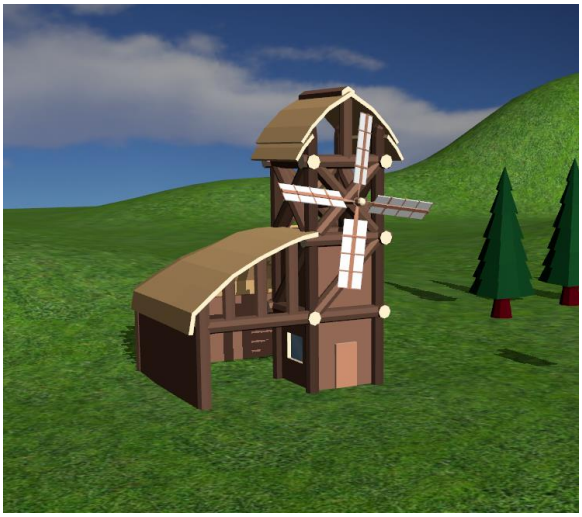

Figure 1 4 Animated Propeller

6. Multiple options for visualizing the surface (solid, wireframe, or polygonal)

Figure 1 5 Solid mode

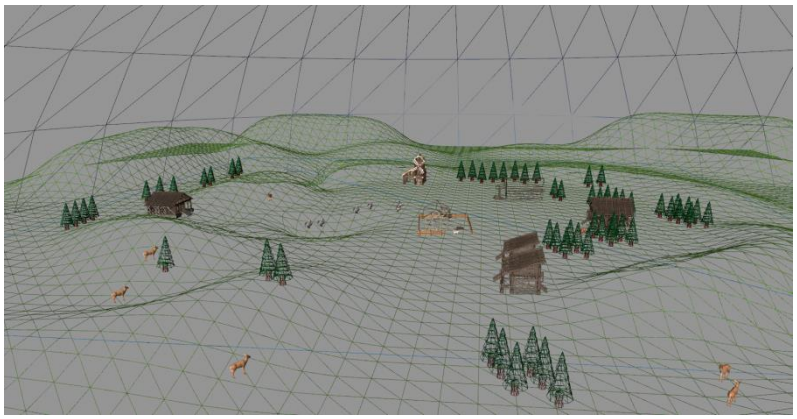

Figure 1 6 Wireframe mode

Figure 1 7 Polygonal mode

4. Implementation Details

4.1 Functions and special algorithms

Camera Animation

The camera animation system is designed to provide a smooth, predefined path for the user's viewpoint, guiding them through important landmarks in the scene. This system includes the ability to start, stop, and control the movement of the camera along a series of 3D points in the scene.

The camera follows a series of predefined 3D coordinates that represent specific key positions within the scene. These coordinates define the trajectory that the camera will take during the animation.

The user can start or stop the camera tour. When the tour is started, the camera begins at the first path point and follows the path to the next, continuing until all points are visited. If the tour is stopped, the camera halts at the current position.

The movement speed is controlled by a time parameter, and the camera automatically progresses to the next point after reaching each one. The system is updated in real-time, allowing for continuous movement until the final point, at which the tour stops. This setup provides a dynamic and controlled exploration of the scene.

Multiple Light Sources

The implementation incorporates both directional and point light to create a dynamic and realistic scene.

The **directional light** (simulating sunlight) is computed using the computeDirLight function. It includes ambient, diffuse, and specular components, and the light direction (lightDir) and color (lightColor) are used to calculate the lighting. This matches the description of a distant light source with parallel rays, which affects the scene's lighting, including shadows.

The **point light** is represented by the computePointLight function. The point light's intensity is attenuated based on distance using the constant, linear, and quadratic attenuation factors. This ensures realistic light falloff, as described.

The **Phong lighting model** is applied to both directional and point lights, which combines:

- **Ambient light** for uniform base illumination.
- **Diffuse light** for intensity changes based on the angle of the light.
- **Specular light** for reflective highlights.

The model is applied using ambient, diffuse, and specular components for each light source.

Shadows

The shadows were added using the Shadow Mapping technique, discussed in **Lab 9[1]**. Shadows contribute to photorealism, being an inevitable phenomenon encountered in real life when a light source hits an object. Shadow Mapping is a multi-pass technique that uses depth textures to decide whether a point is in shadow or not, as described in the lab guide. The algorithm consists of two stages: first, rasterizing the scene from the light's point of view (creating the depth map), and second, rasterizing the scene from the observer's point of view (the camera's position). The advantage of using OpenGL is that it handles all rasterization (color, depth, and stencil information) in a framebuffer. Completing the requirements from **Lab 9** facilitated my work on the project since I was able to use the same shader programs without modifying them for my objects.

Proppeler Animation

To implement a rotating propeller, the rotor blades were modeled separately in Blender, with careful attention given to positioning the pivot point at the exact center of rotation of the windmill. The rotor's coordinates were then converted to OpenGL's coordinate system, and its rotation is driven by time-based updates to the angle, producing a realistic spinning effect.

Fog

- Adds an atmospheric effect where distant objects gradually fade into fog, simulating mist or haze.

The **fog effect** is computed based on the distance of each fragment from the camera.

A **fog density** value determines how thick or thin the fog is. The further away an object is from the camera, the more fog is applied.

The fog effect is blended with the scene's color, and it can be toggled on or off using the isFogRunning variable. When fog is enabled, the final color is blended with a gray fog color, creating a misty appearance.

4.2 Graphics model

The project utilizes OpenGL 4.1 Core for high-performance rendering, supported by GLFW for cross-platform window management and real-time user input handling. GLEW is used to load OpenGL extensions as needed, while GLM provides the mathematical foundation for vector and matrix transformations.

For content creation, Blender was used to design and assemble the scene. The objects were downloaded from the internet, some of which came with textures, while for others, I created custom textures. The objects were scaled, translated, and rotated directly in Blender, which proved to be a significant advantage.

4.3 Data Structure

In my project, the following key data structures were used:

1. **std::vector**: A dynamic array to store collections of objects like camera path points or light sources.
2. **Arrays**: Used for fixed-size collections such as point light constants or predefined parameters.
3. **GLM Matrices (mat4, mat3)**: Essential for 3D transformations (scaling, rotation, and projection) in the scene.
4. **Uniform Variables**: Used to pass data like light positions, camera transformations, and textures from the CPU to the GPU.
5. **Shader Programs**: Contain GPU instructions for rendering and lighting calculations.

These structures ensure efficient organization and management of the 3D scene, supporting transformations, lighting, and rendering.

4.4 Class Hierarchy

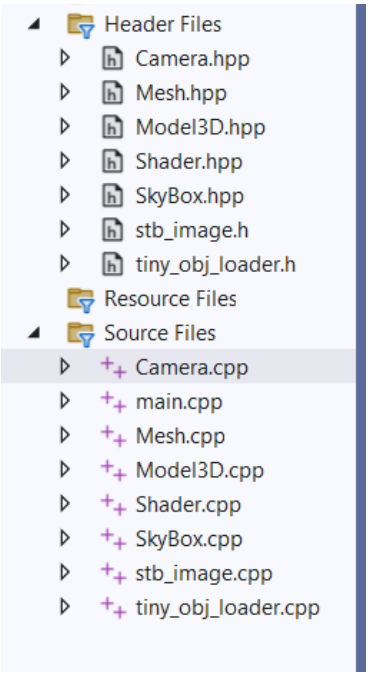

Figure 1 8 Class Hierarchy

Among the given classes, we can mention the functionality of some of them as follows:

- Camera.cpp – implements the camera movements;
- Mesh.cpp – represents a 3D object;
- Shader.cpp – contains methods for creating and activating shader programs;
- main.cpp – includes the implementation of the project and the functionalities mentioned above.

5. Graphical user interface presentation / user manual

Using the mouse and keyboard, we can navigate and view the created scene, performing translation, scaling, rotation, and adjusting the movement and positioning of the camera.

Keyboard Controls:

- **W / A / S / D**: Move the camera forward, left, backward, and right.
- **R / F**: Move the camera up and down.
- **1, 2, 3**: Switch between wireframe, points, and filled polygon rendering modes.

- **Z**: Toggle fog on.
- **X**: Toggle fog off
- **T**: Start the scene tour.
- **G**: Stop the scene tour.
- **P**: Toggle the point light on.
- O: Toggle the pointlight off.
- K: Move right the angle of the directional light.
- L: Move left the angle of the directional light.
- **ESC**: Exit the application.

Mouse Controls:

- Move the mouse horizontally/vertically to adjust the camera's yaw and pitch.

6. Conclusions and further developments

This project demonstrates the integration of advanced graphics techniques using OpenGL to create an immersive and dynamic 3D environment. By combining elements such as lighting, shadows, animations, and environmental effects, it successfully delivers a visually engaging scene. The use of Blender for modeling, along with efficient data structures and robust input handling, enabled seamless creation and interaction within the environment.

The project offers significant potential for expansion. Future enhancements could include extending the scene by adding more elements, incorporating additional point lights, more animations, or implementing advanced algorithms like collision detection. Additional weather effects(like rain and thunder) are also potential areas for development.

7. References

[1] https://moodle.cs.utcluj.ro/pluginfile.php/202223/mod_resource/content/6/Laboratory%20work%209.pdf

[2] https://moodle.cs.utcluj.ro/pluginfile.php/202209/mod_resource/content/3/Laboratory_work_8.pdf

[3] https://www.youtube.com/playlist?list=PLrgcDEgRZ_kndoWmRkAK4Y7ToJdOf-OSM

[4] https://free3d.com/3d-models/farm

[5] https://www.blender.org/