Graphics Processing

Doia Bogdan-Mihai

30431

Cuprins

2. Subject specification

The subject of the project consists in the photorealistic presentation of 3D objects using OpenGL library. The user directly manipulates by mouse and keyboard inputs the scene of objects.

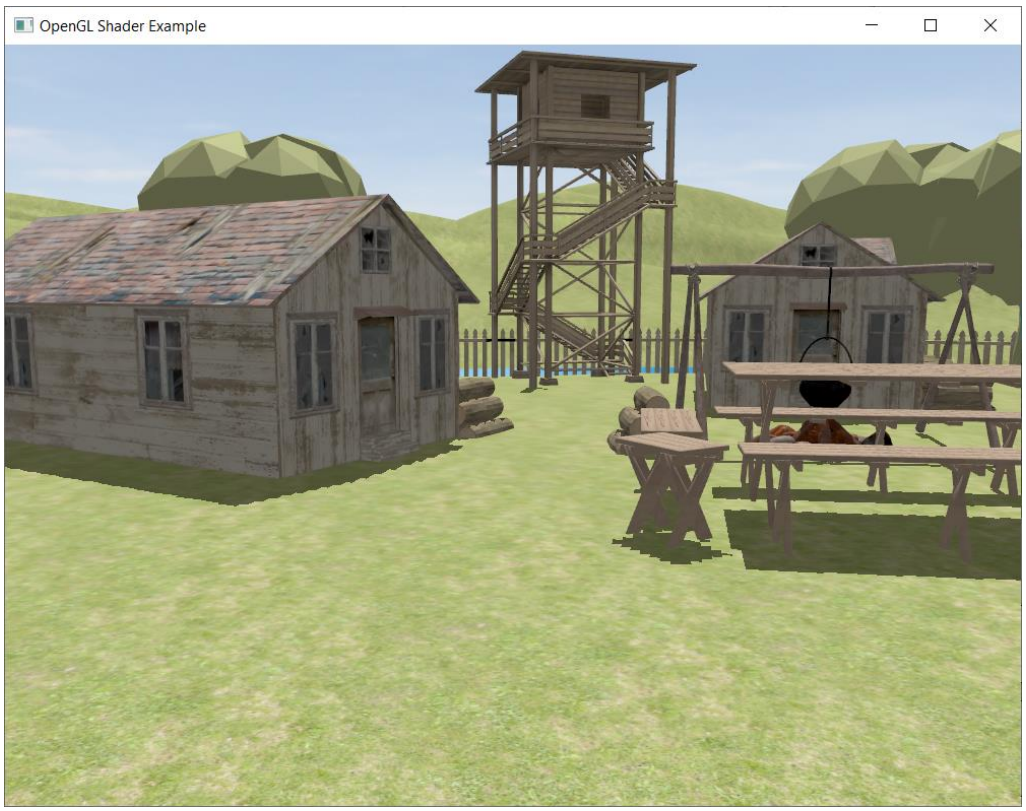
3. Scenario

The project consists of a 3D space with multiple objects in a forest that could serve as the base for a survival game.

3.1. Scene and object description

The scene consists of a small island on a pond in a forest. The main idea behind the buildings on the island is that they are a survival camp in wilderness. The island is surrounded by a wooden fence, there is a wooden bridge connecting the camp with the rest of the forest and 3 boats on the water that can serve as transport

or for fishing. Inside the camp there are 3 houses built around a campfire that has a pot for cooking hanging over it. Beside the campfire there is a wooden table and 2 benches where people can dine or sit to chat. On the sides of the houses there are logs that can be used for fire or for construction. Five watchtowers encircle the camp and can serve as vantage points for surveillance. A few trees are in between the buildings for shade and a windmill that can aid in food production is also placed on the island.



3.2. Functionalities

The scene can be visualized using both mouse and keyboard inputs just like a classic 1st person game. The global light casts shadow that can also be visualized by moving the light source. The campfire can act as a pointlight source and can simulate in contrast to the global light the difference between day and night in the camp. The mill can be made to spin or stop and the whole scene can be visualized in a cinematic style with a slow automatic rotation.

4. Implementation details

4.1. Functions and special algorithms

The features of the project were implemented using various functions and algorithms studied at the laboratory and course throughout the semester. Those include camera movement processing using both keyboard and mouse, computing lighting components using the Blinn-Phong Lighting Model, computing shadows using the Shadow Mapping technique and generating the depth map, generating fog and other various rotation and animation functions.

4.1.1. Possible solutions

In the development process, I considered various solutions to address the implementation of the functions and algorithms above by first trying to understand the code already provided in the laboratory and then modifying and adapting it to my needs. In this mode I was able to put together the code for computing the colour of objects based on the position of light sources, after that the depth map used in shadow generation. I have animated my windmill using the translation, rotation, translation method. Using the OpenGL library functions I implemented viewing solid, wireframe objects, polygonal and smooth surfaces. To make the camera movement intuitive and smooth I completed the Camera.cpp file with the appropriate mathematic operations in OpenGL.

4.1.2. The motivation of the chosen approach

My motivation for choosing this approach in finalising the project's functionalities is a combination of computation efficiency and visual fidelity, while keeping my code easy to understand and the coding style that was presented to us in the various graphics classes.

4.2. Graphics model

The graphics model encapsulates the foundational elements for rendering a 3D scene in OpenGL. It contains the initialization and management of essential components such as shaders, 3D models, a camera, and lighting effects. Shaders define how light interacts with surfaces, implementing features like the Blinn-Phong Lighting Model. The camera is positioned within the scene, and user input is processed to control its movement and orientation. Lighting features include point lights and directional lights, contributing to the realism of the rendered scene. The implementation of shadow mapping adds depth to the visuals by

simulating shadows. The scene rendering process involves multiple passes, including the creation of a depth map and rendering the final scene with shadows.

4.3. Data structures

The code uses several data structures from the OpenGL Mathematics (GLM) library, as well as some standard C++ data structures. Those include GLM data structures: matrices and vectors (mat4, mat3, vec3); OpenGL data structures: GLFWindow, GLunit, GLenum;

4.4. Class hierarchy

The class hierarchy is organized to encapsulate various components related to 3D graphics rendering and interaction. At the core of the hierarchy is the Camera class, representing the virtual camera's position and orientation within the 3D scene. Derived from the Camera class, the Model3D class encapsulates the functionality for loading, managing, and rendering 3D models. The Shader class is responsible for handling OpenGL shader programs, while the SkyBox class manages the skybox rendering. Additionally, the SkyBox class is utilized within the main loop to render the skybox. The project also includes a variety of utility classes, such as the glm classes for mathematical operations, providing support for vectors, matrices, and transformations. Overall, the class hierarchy is designed to promote modularity, encapsulation, and ease of maintenance, enabling efficient management and rendering of 3D graphics within the OpenGL environment.

5. Graphical user interface presentation/user manual

Once the program runs, the user interface should open shortly after.

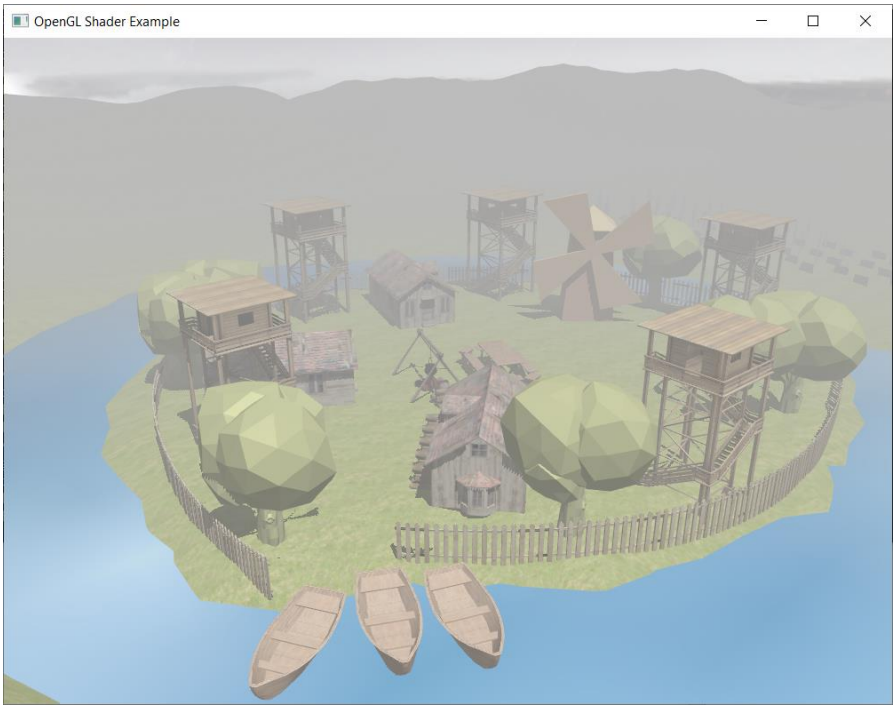
When the scene finishes loading, the mouse will be locked inside the window to facilitate the camera movement. After that the scene can be visualized and inspected however the user chooses to. The inputs the user can use to aid in this are listed below:

- Mouse input – moving the mouse acts like a 1st person camera
- W – Zoom in
- S – Zoom out
- A – Move camera left
- D – Move camera right

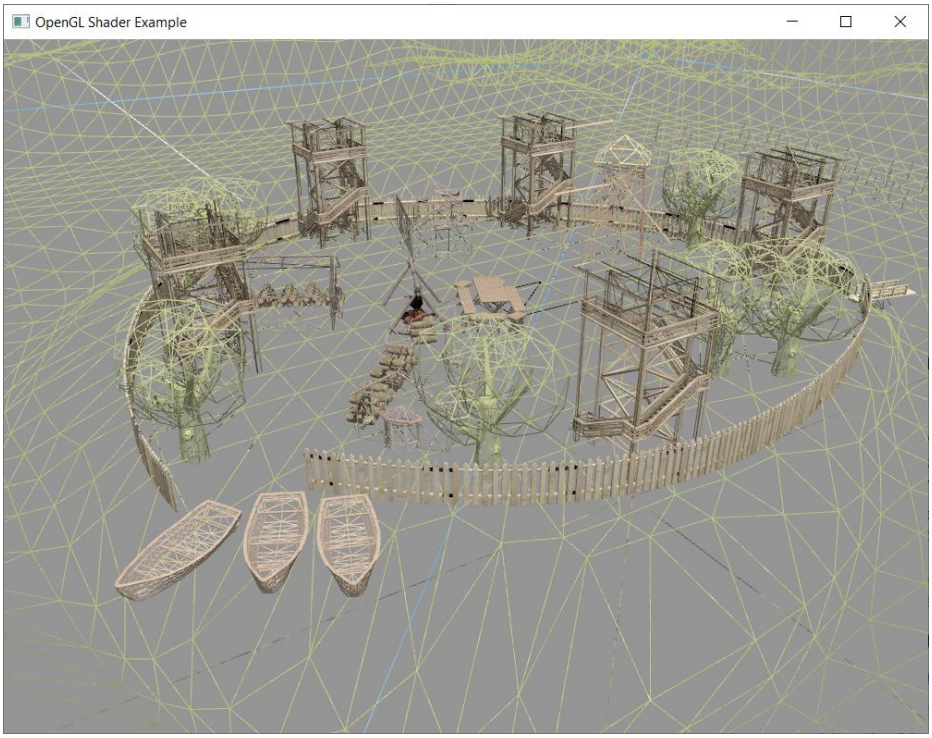
- Q – Rotate camera counterclockwise
- E – Rotate camera clockwise
- P – Turn on automatic presentation
- J – Rotate global light clockwise
- L – Rotate global light counterclockwise
- F – Turn on fog
- C – Turn on point light
- M – View shadow map
- 1 – View wireframe
- 2 – View polygonal
- 3 – View smooth
- 4 – View solid
- 7 – Start windmill rotation
- 8 – Stop windmill rotation
- Esc – Close window

Point light on:

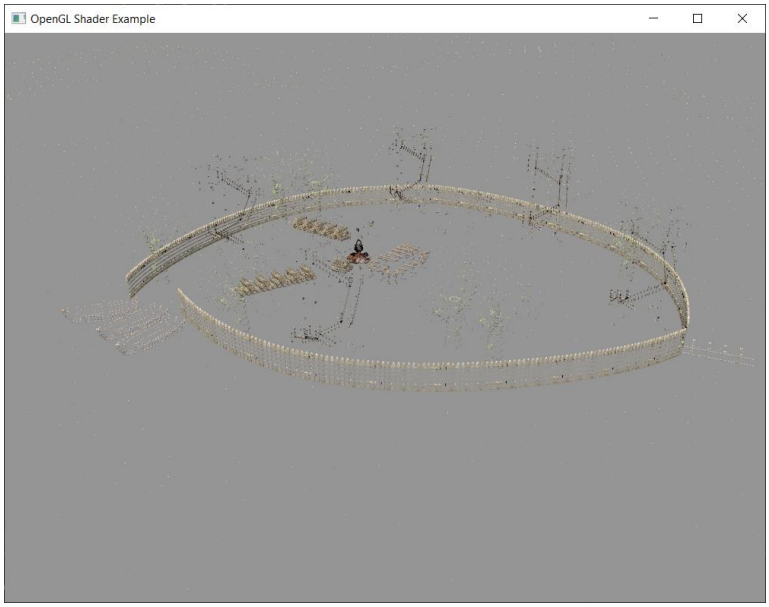
Fog on:



Wireframe:



Polygonal:



6. Conclusions and further developments

In conclusion, this project has proven to be very entertaining to work on, in spite of the arguably big workload necessary, it has proved to be one of if not the most fun project to work on up to this point. Learning how to use Blender on top of the OpenGL functionalities is a valuable skill in my opinion and modelling my scene from scratch gave me great satisfaction in seeing it come to life in the final project.

As further developments, certain hazards or dangers can be added to make the camp actually have a purpose, for example a zombie apocalypse style game can be made using the scene. Additional features that could be added include being able to rise/lower the bridge, drive the boats, add spotlights to the towers, add wildlife and characters that live inside the camp.

7. References

https://docs.blender.org/manual/en/latest/index.html

https://learnopengl.com/Getting-started/Camera

https://free3d.com/

https://docs.google.com/document/d/1njtWPMmOQNIaD_z9ve8iPRUqQTWdIV_PO-NvPD0nOuM/edit#heading=h.7h28ckb0ku81

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