Graphic Processing
Project Documentation

Iulia-Olivia Cipleu
Group 30433

Contents

Table of Figures

Subject Specification

The objective of this project is to craft a lifelike scene featuring a variety of 3D objects, employing the OpenGL library. Users will engage with the application through mouse and keyboard interactions. The depicted setting portrays a countryside vacation house.

Scenario

Scene and Object Description

The depicted setting emulates a charming rural environment, featuring a vacation house, a lakeside dock, a boat, a cat, a dog, ducks, a windmill, plants, and various amenities.

Noteworthy are automated animations, such as the boat tied by the dock gently swaying with the waves, the rope with which is tied rotating with the boat and a candle gradually melting, with its spark following it.

Illumination is provided by two sources: a directional light, situated on the house terrace, and punctiform one, represented by an LED panel near the lake.

The scene offers two distinct modes: day and night, each of them with a different level of luminosity. There are also 2 weather conditions: sunny and foggy.

For the family of ducks, it was implemented collision detection.

Functionalities

The movement through the scene is made by using the keys W, A, S and D, and the mouse for rotation (like in shooter games). Also, by using the key M, the camera will start to move on a predefined path through the scene.

The color of an outdoor light can be modified be pressing the keys: 1, 2, 3, 4 and 5. The source of light placed on the LED can be moved up and down by the keys J and L.

The sky can be of a sunny day, or of a night, the change between them being made with the keys B, respectively N. The fog is activated by the key F, while the keys T and V are used for increased, respectively decreasing the density of the fog. By pressing the key G, the weather becomes sunny again.

By pressing the key P, the scene can be visualized in the wireframe mode, the redo being made be the key O. To display only the vertices of the object, the user needs to press the key I, and to return to default mode, key U.

Using the arrow, the user can move the dog on the X-axis and Y-axis.

Implementation Details

Functions and Special Algorithms

Possible solutions

The OpenGL library encompasses a diverse range of functions, including but not limited to:

- "glViewport(...)" for configuring the Viewport transform.
- "glfwCreateWindow(...)" for window creation.
- "glGenTextures(...)"
- "glBindTexture(...)"
- "glTexImage2D(...)" for depth texture creation within Framebuffer objects.
- Numerous others.

In this project, a crucial function employed is "void renderScene()". This function is responsible for transmitting data to the shaders, computing values, creating models, calculating normal matrices, and handling rotation, translation, and scaling operations.

Inside this function are called the ones specifically oriented on the object. Each 3D model has its own render function, the differences being observed at the objects which are part of an animation.

The function "void initUniforms()", with its variants "void initLampUniforms()" and "void initPunctiformUniforms()" are responsible for details about the light, specifically, the directional light and point light.

The function "void initShaders()" instantiates shaders, contributing to the rendering pipeline.

Furthermore, the function `void initModels()` initializes all 3D models within the scene.

Several other essential functions include:

- "void processMovement()"
- "void move(gps::MOVE_DIRECTION direction, float speed)"
- "void mouseCallback(GLFWwindow* window, double xpos, double ypos)"
- "void keyboardCallback(GLFWwindow* window, int key, int scancode, int action, int mode)"

Those functions are the ones responsible for the changes applied to the camera or to the objects in the scene.

The "void mouseCallback(GLFWwindow* window, double xpos, double ypos)" function utilizes Euler angles for camera control. In particular, the variables yaw and pitch represent the rotation angles around the vertical (y-axis) and horizontal (x-axis) axes, respectively. These angles are updated based on the mouse movement in the x and y directions.

Another important library is GLM (OpenGL Mathematics), which is a C++ mathematics library designed for computer graphics and 3D applications. It provides a collection of vector and matrix operations, along with various mathematical functions, closely mirroring the functionality of OpenGL shaders.

For the collision detection, I needed to compute a Boolean function "bool checkCameraDucksCollision()", which return true if the camera position is inside the area occupied by the ducks. If yes, the camera is moved backward.

In "basic.frag" are 2 important functions. The "computeDirLight" function calculates directional lighting components (ambient, diffuse, and specular) for a given fragment in a shader.

It transforms the fragment's position and normal into eye space, computes normalized light and view directions, and then calculates ambient, diffuse, and specular lighting based on these vectors and material properties. The “computeFog” function computes a fog factor based on the fragment's distance from the viewer. It uses an exponential decay model to determine the impact of fog on the fragment, with fog density influencing the rate of decay. The result is clamped between 0 and 1, representing the intensity of fog affecting the fragment.

In “lampLightShader.frag”, there is a new element, emitted light, which is added at the final stage.

The motivation of the chosen approach

The choice for this kind of implementation was having as starting point the laboratories from this semester, where it was discussed about rendering object, light computation, animations, different photorealistic effects. Starting from this and the core of the project, I developed some ideas that fit with the chosen theme of the project.

Graphics Model

The majority of the objects in the scene are downloaded from the Internet. Exceptions are the land, lake, sky and some minor components. All the objects were imported to the Blender, where they were placed, scaled, rotated, texturized. While some objects had existing textures, many lacked them or required modification (separate by texture, group components) for better integration into OpenGL. For the terrain, I manually sculpted the hills and the lake into a plane using Blender's tools. The sky is a textured part of a sphere (approximatively 1/3), which covers the plane. To be able to see the pattern, I had to flip the normal in the sphere.

The most time-consuming aspect involved selecting suitable coordinates and exercising creativity to craft a scene from scratch.

Data Structures

Basic data structures were essential for consolidating attributes related to various types of light, including spotlights and point lights. These structures were already incorporated into the project provided in the laboratory, proving to be highly advantageous. Moreover, I introduced extra functions to compute lights and manage fog within the project. In the shader employed for punctiform light, two new "Structs" were introduced to facilitate easier access to the components necessary for light computation.

A new implemented part of this project is about the animation of the camera, which interpolates some hardcoded points in the scene, then moves the camera along this route. The implementation employs linear interpolation. The function first checks if there are sufficient control points, returning a zero vector if not. For valid cases, the function clamps the input parameter t between 0 and 1, ensuring it lies within the valid range. It then computes the lower and upper indices based on the clamped parameter, preparing for linear interpolation. The alpha value is calculated to determine the interpolation weight between the two control points. Finally, the function uses “glm::mix” to perform linear interpolation between the control points indexed by “lowerIndex” and “upperIndex” using the computed alpha value. The result is the interpolated vector corresponding to the given parameter t along the spline defined by the control points.

Class Hierarchy

Beside the most important file, “main.cpp”, where lies the majority of the code, there are also some useful classes:

- Camera: contains the implementation for camera movement (up, down, front, back, left, right, rotate)
- Mesh: represents a 3D object; includes the following data structures:
  - Vertex (position, normal vector and texture coordinates)
  - Texture (id, type and path)
  - Material
- Model3D: contains methods for printing meshes using a specified shader program
- Shader: contains methods needed to import shader programs
- Spline Utils: contains the method used for the interpolation of points, for the camera animation
- Window: used for the creation of the visualization window

Graphical User Interface Presentation / User Manual

In the following table, the role of each key will be presented.

Key	Functionality
1	Set lamp color to red
2	Set lamp color to green
3	Set lamp color to blue
4	Set lamp color to yellow
5	Set lamp color to white
A	Move camera to left
B	Day mode
D	Move camera to right
DOWN	Move dog backward
E	Rotate camera to right
F	Enable fog effect
G	Disable fog effect
I	Enable point mode
J	Move down punctual light
L	Move up punctual light
LEFT	Move dog to left
M	Start camera animation
N	Night mode
O	Disable wireframe mode
P	Enable wireframe mode
Q	Rotate camera to left
RIGHT	Move dog to the right
S	Move camera backward
T	Increase fog density
U	Disable point mode
UP	Move dog forward
V	Decrease fog density
W	Move camera forward

Now, some print screens will be presented.

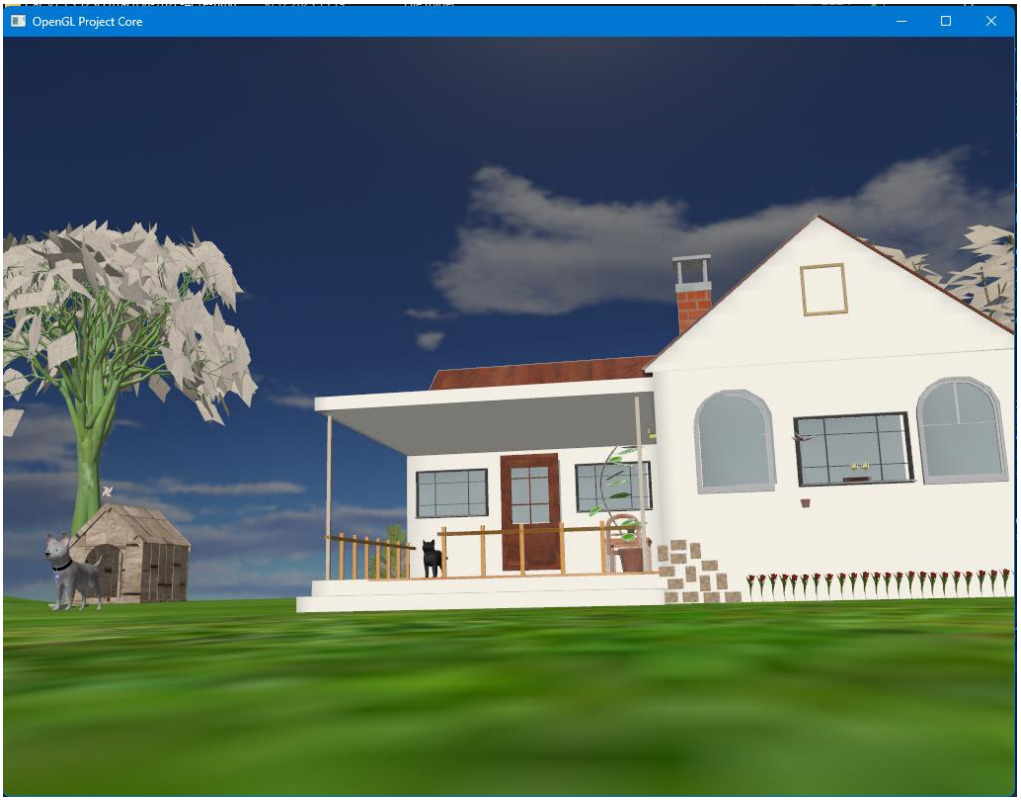

Figure 1: House, cat, dog, dog house, trees, tulips on day mode

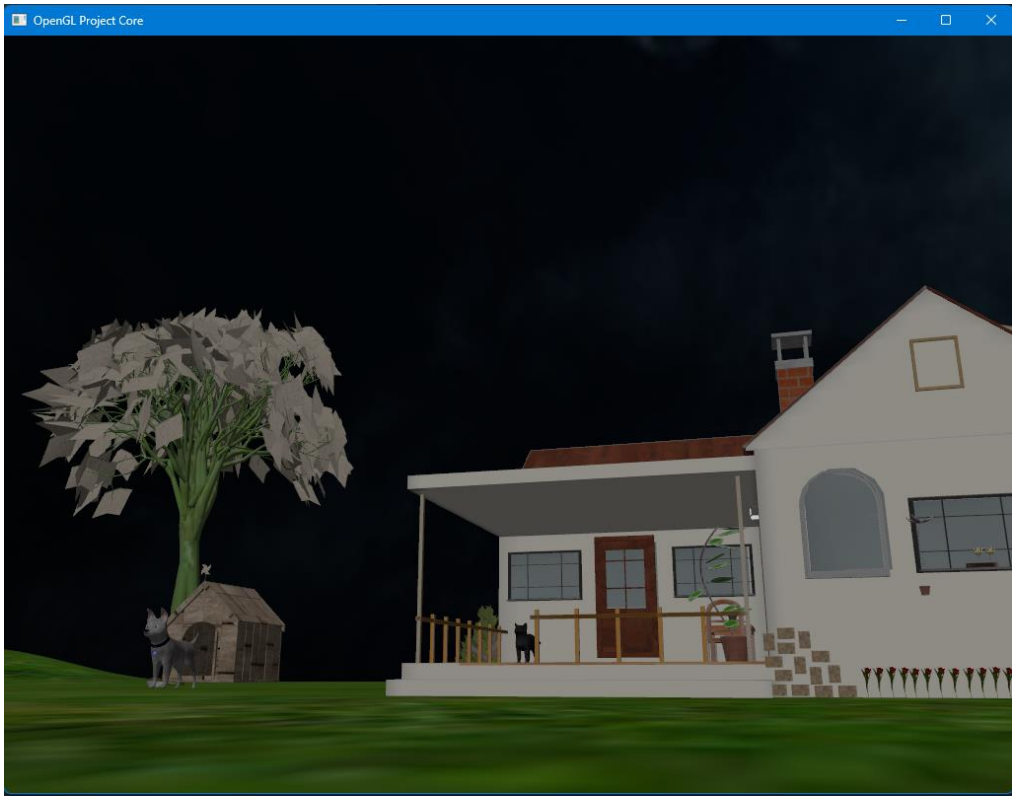

Figure 2; House, cat, dog, dog house, trees, tulips on night mode

Figure 3: House seen from above

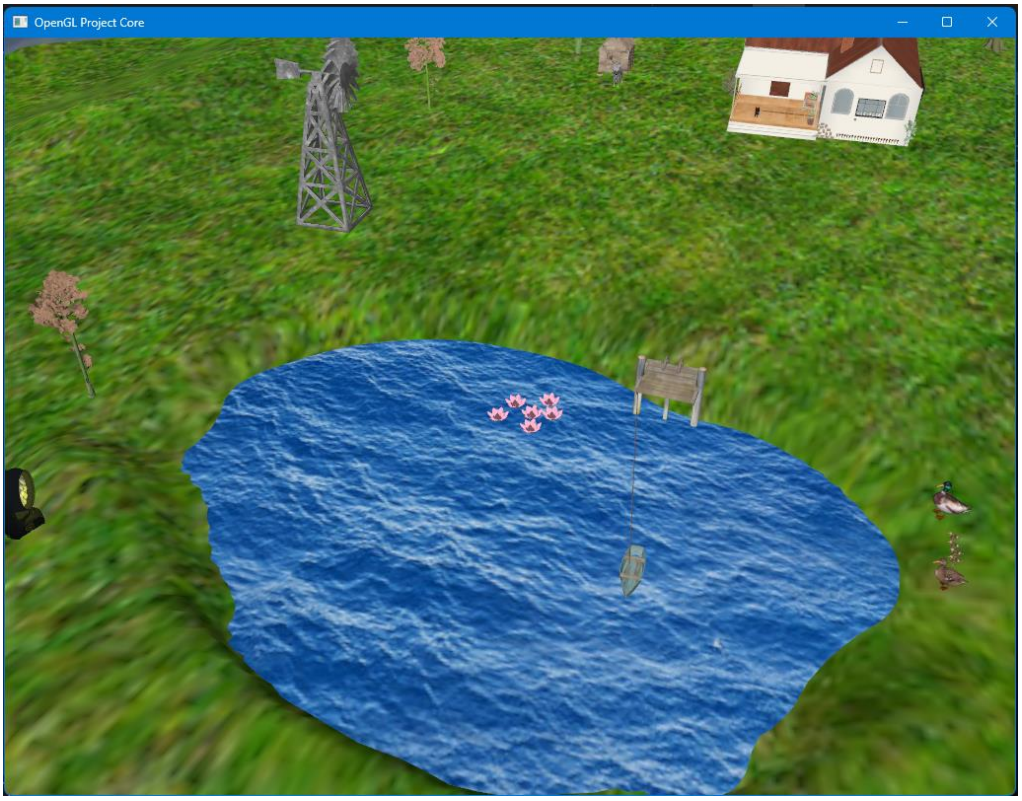

Figure 4: View from above the lake

Figure 5: From the level of terrain: ducks, dock, trees, LED light, the lake is above the camera level

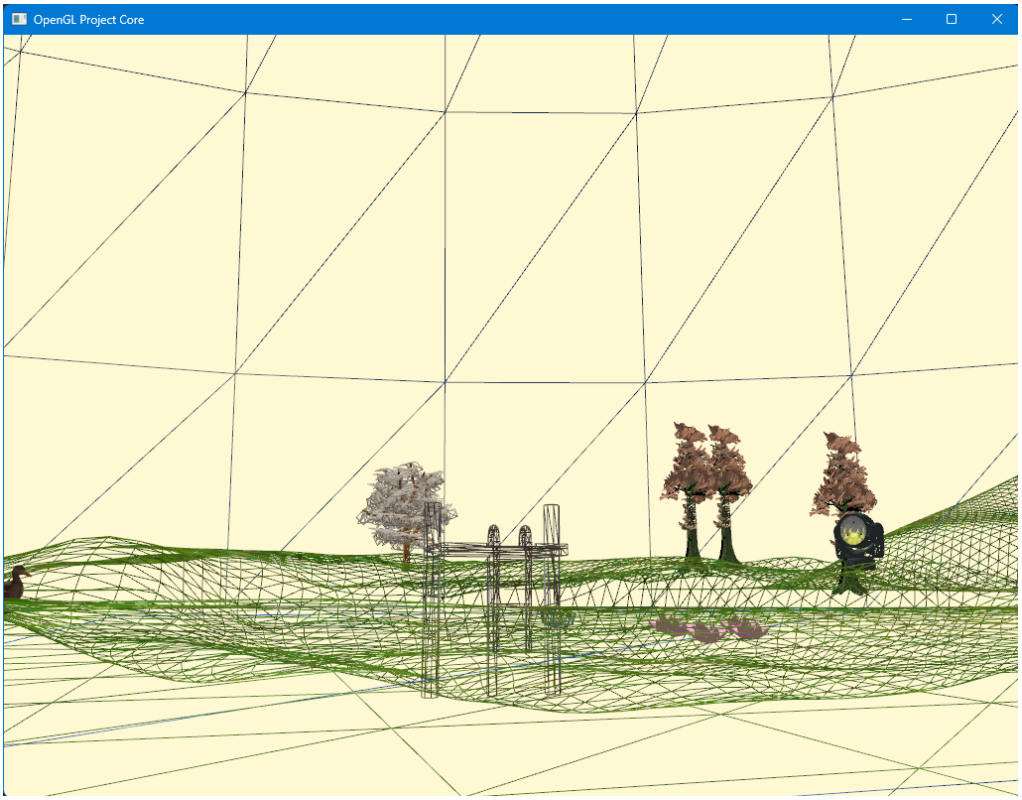

Figure 6: Lake observed in wireframe mode, from the level of land

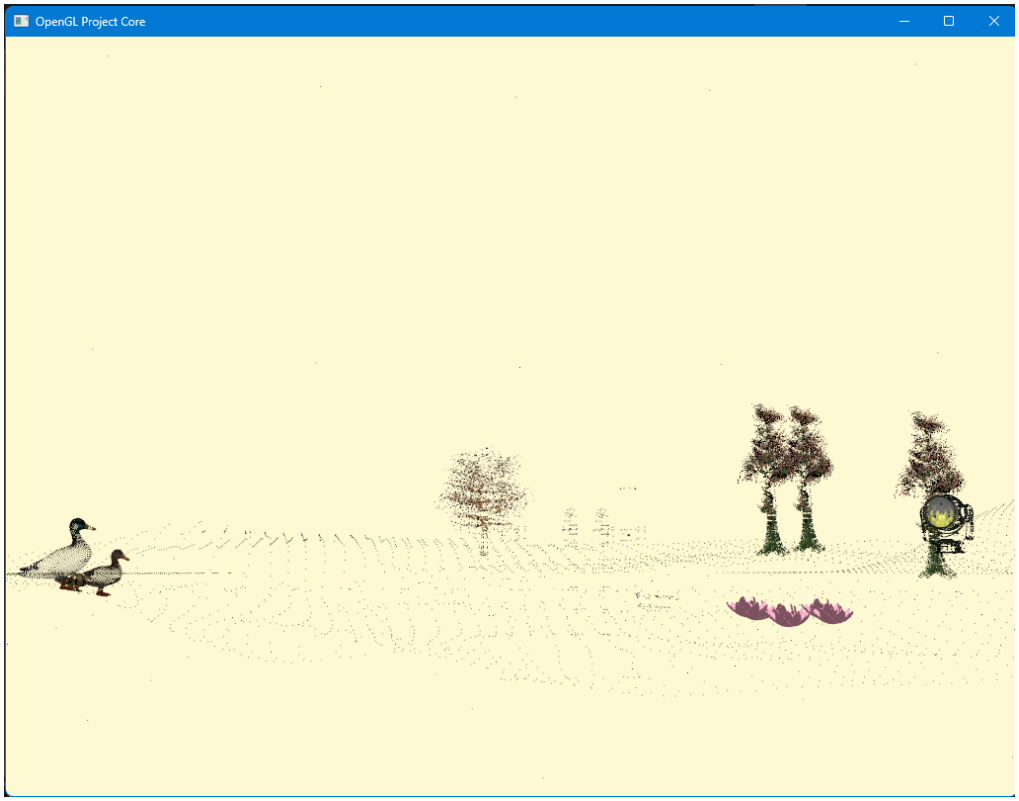

Figure 7: lake observed in point mode, from the level of land

Figure 8: Standing on the dock: lake, animated boat and rope, lotus, LED light, trees

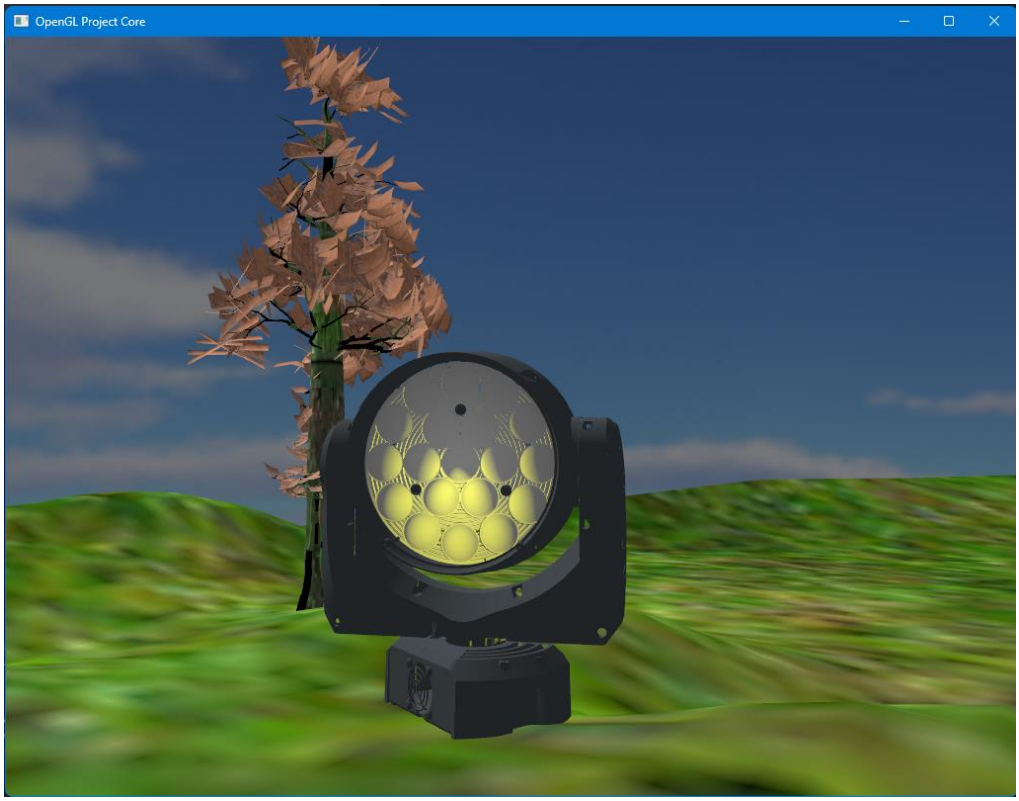

Figure 9: Close-up to the LED, it can be seen that it has a yellowish light on a white texture, source in initial position

Figure 10: Close-up to the LED, source of light moved up

Figure 11: LED light, on foggy weather, light can be observed better

Figure 12: Windmill

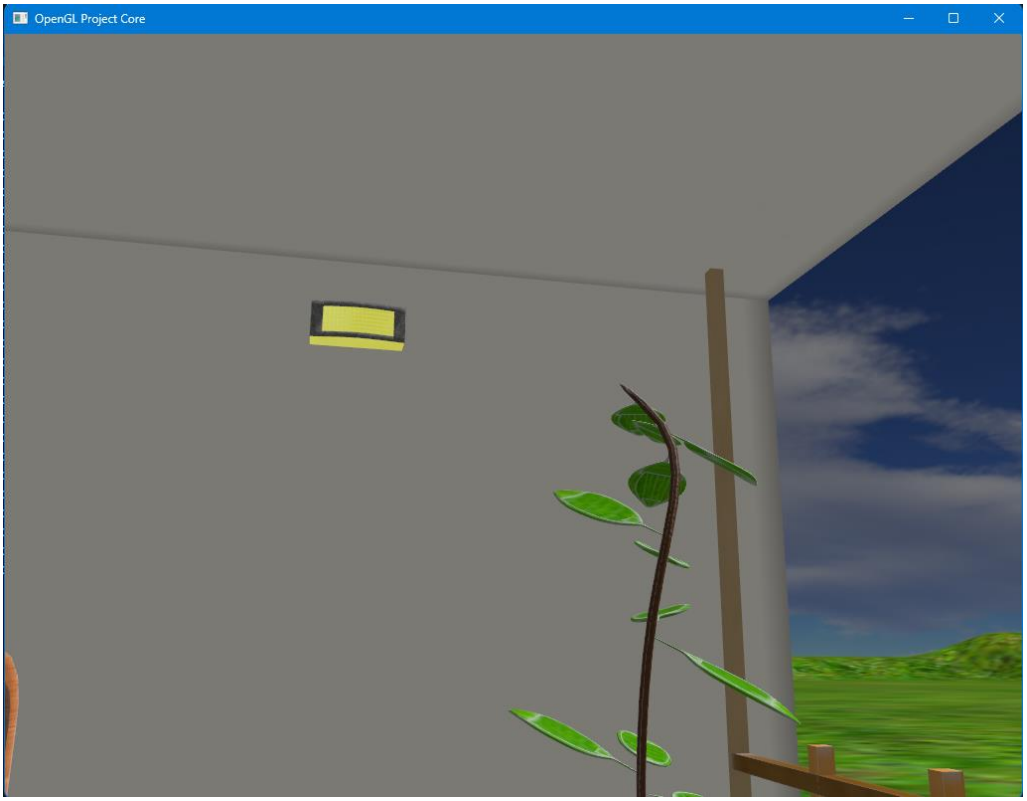

Figure 13: Directional source of light, colored in yellow

Figure 14: Directional source of light, colored in blue

Figure 15: Directional source of light, colored in blue, with fog activated

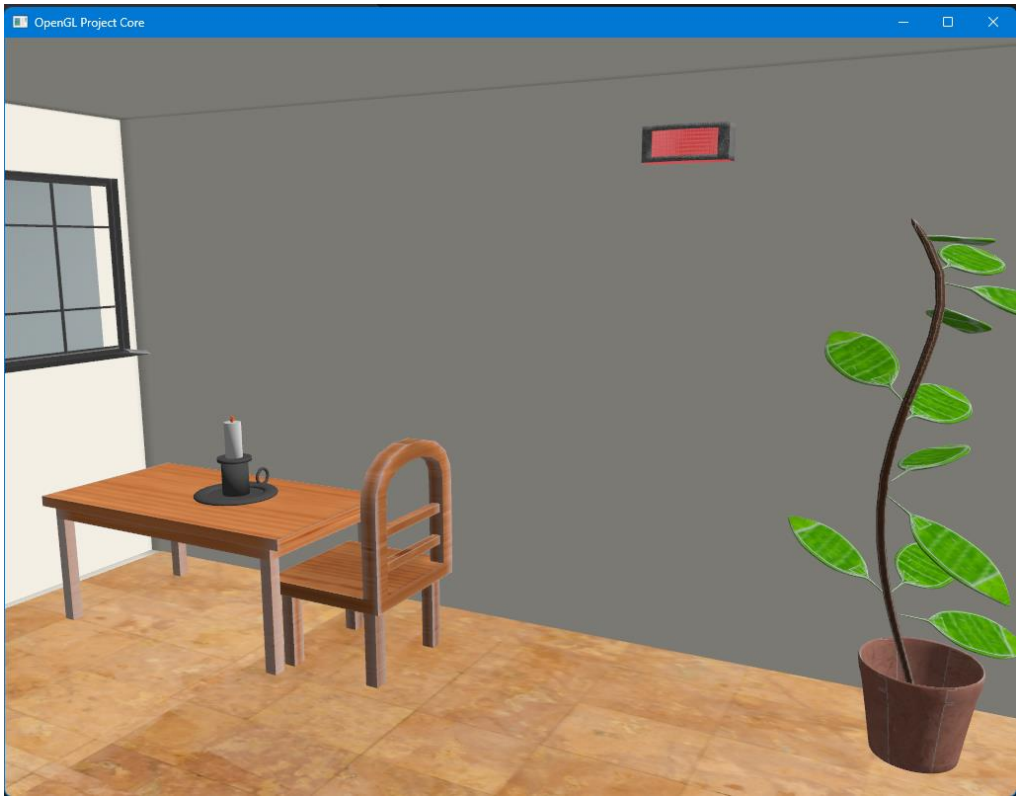

Figure 16: Directional source of light, colored in red and animation of melting candle

Figure 17: Animation of melting candle

Figure 18: Dog in original position

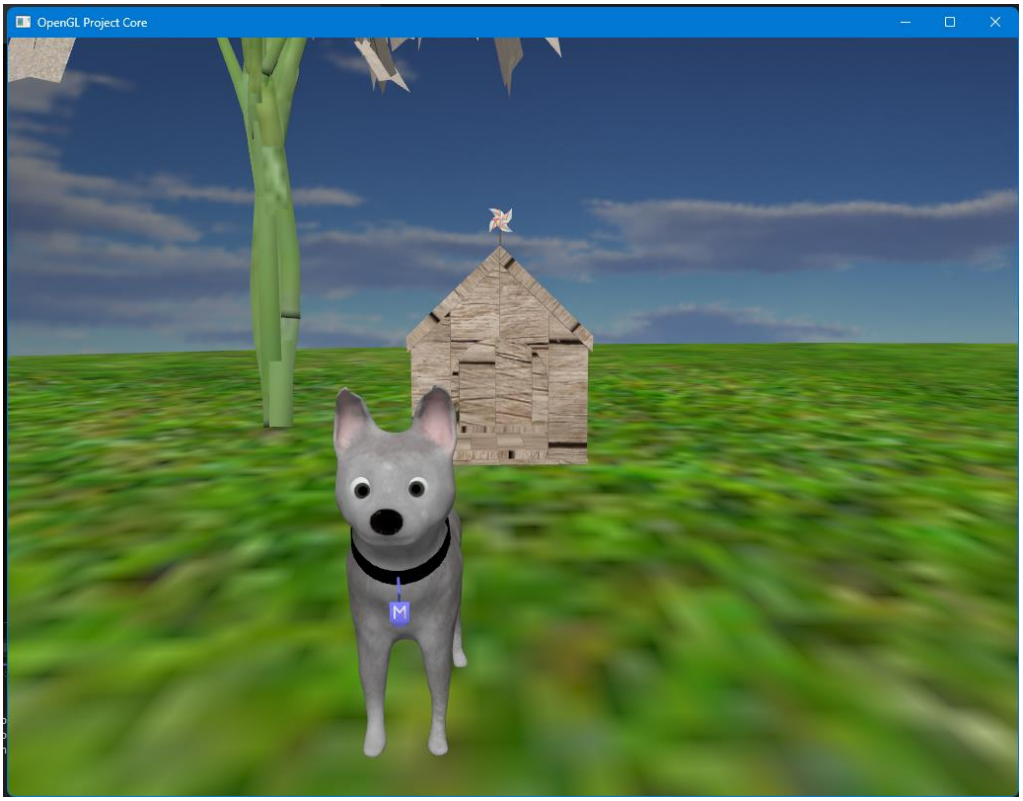

Figure 19: Dog after it was moved using the arrows

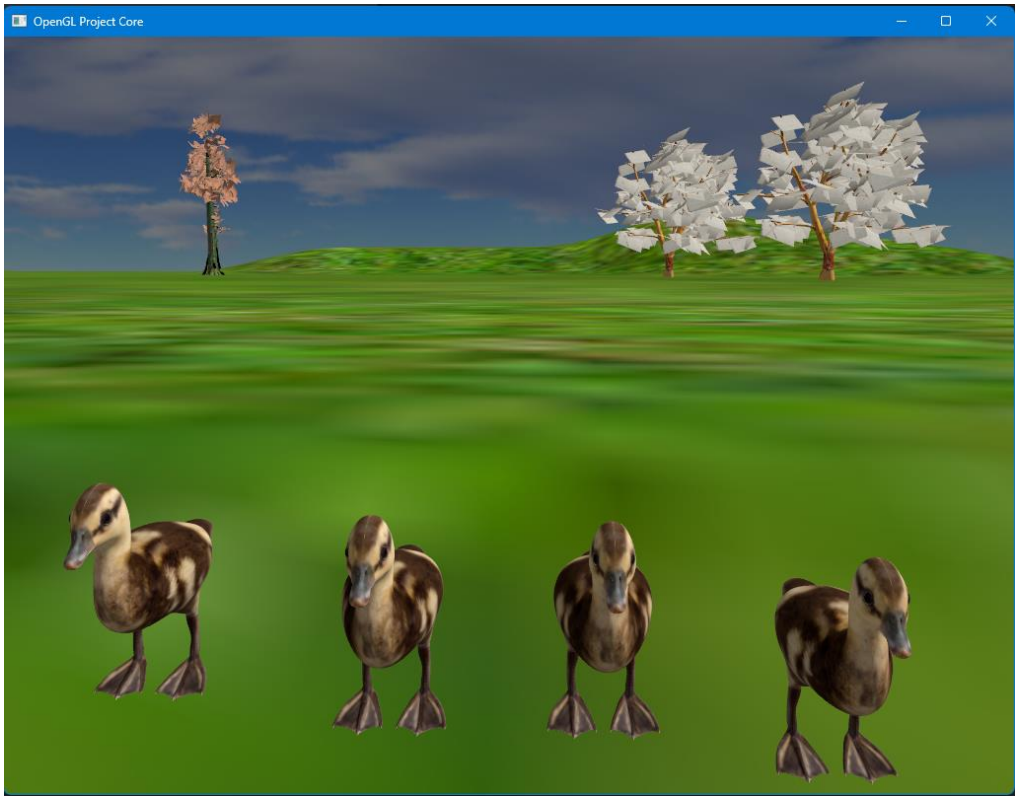

Figure 20: Border for collision for ducks

Conclusions and Further Developments

In summary, this project involved working with two programs: Microsoft Visual Studio and Blender, the latter being a new experience for me. Crafting the scene in Blender consumed a significant amount of time as I aimed for visual appeal.

On the other part, the coding part was a little more difficult, because even if there were the laboratories works as a starting point, putting all together and combining them was quite complicated.

For further developments, I have the following ideas:

- Adding shadow computation
- Introducing new animations
- Including an avatar for the used
- Incorporating new weather effects: rain, snow etc.
- Implementing collision detection for more objects
- Enabling actions upon the objects in the scene (e.g. grab, drop)

References

1. "Multiple lights", "https://learnopengl.com/Lighting/Multiple-lights"
2. "Camera", "https://learnopengl.com/Getting-started/Camera"
3. "OpenGL", "https://en.wikipedia.org/wiki/OpenGL"
4. "OpenGL Mathematics (GLM)", "https://www.opengl.org/sdk/libs/GLM/"
5. "Free3D", "https://free3d.com/"
6. "TurboSquid", "https://www.turbosquid.com/"
7. "Tutoriale Blender", "https://www.youtube.com/playlist?list=PLrgcDEgRZ_kndoWmRkAK4Y7ToJdOf-OSM"
8. "Linear interpolation", "https://en.wikipedia.org/wiki/Linear_interpolation"
9. "Mix", "https://registry.khronos.org/OpenGL-Refpages/gl4/html/mix.xhtml"