

Representing a Scene of Objects

DOCUMENTATION

Graphical Processing

STUDENT: DOMȘA-STÎNĂ ANDA

GROUP: 30433

Contents

[1. Subject specification 3](#_Toc60936677)

[2. Scenario 3](#_Toc60936678)

[2.1 Scene and object description 3](#_Toc60936679)

[2.2 Functionalities 4](#_Toc60936680)

[3. Implementation details 4](#_Toc60936681)

[3.1 Functions 4](#_Toc60936682)

[3.2 Graphics model 4](#_Toc60936683)

[3.3 Data structures 4](#_Toc60936684)

[3.4 Class hierarchy 4](#_Toc60936685)

[4.Graphical user interface and user manual 5](#_Toc60936686)

[5. Conclusion and possible future developments 5](#_Toc60936687)

[6. References 5](#_Toc60936688)

# Subject specification

OpenGL (Open Graphics Library) is a cross-language, cross-platform application programming interface (API) for rendering 2D and 3D vector graphics. The API is typically used to interact with a graphics processing unit (GPU), to achieve hardware-accelerated rendering.

This project is a 3D rendering demo using the OpenGL library and various extensions such as GLFW, GLEW and GLM. The project includes features such as camera movement, object loading and rendering, lighting, and skybox rendering. Additionally, it also includes the use of shaders for advanced rendering techniques. The purpose of this project is to showcase the capabilities of the OpenGL library and its extensions in creating a visually stunning 3D environment. The project documentation will provide an in-depth look at the various components of the project, including the code structure, the implemented features, and the technical details behind them.

The user has the possibility to traverse this scene using the mouse and the well-known control keys WASD.

# Scenario

## Scene and object description

When running the project, the user is presented a typical village house, and its surrounding.

The house is surrounded by trees and behind the house there is a big lake. In front of the house there is a little flower garden. The scene also contains a well and a rotating windmill.

The static scene contains: a house, multiple trees, a lake, a duck, flowers, fences, bushes, a well and there is also a windmill which represents the dynamic object of the scene.

## Functionalities

* When entering the scene, the user can **traverse the scene** using the mouse and the WASD buttons.
* The user can **change the representation of the scene** from polygonal to wireframe and pointframe, with the 1, 2 and 3 buttons.
* Pressing the Q and E button, the user can **control the rotation of the windmill**.
* Pressing the F and G buttons, the user can **control fog appearance.**

# Implementation details

## Functions and special algorithms

3.1.1 Possible solutions:

* The project utilizes a variety of pre-existing algorithms for handling the loading and rendering of objects and textures. For example, the Model3D class is used to read the vertex and coordinate data from .obj and .mtl files, and handle the mapping of textures onto the objects.
* Animations

The user is able to navigate and explore the scene from various angles and perspectives by utilizing the camera and input from keyboard and mouse. Initially, the user is positioned in the center of the scene and can rotate the camera to view the scene from different angles, and then move the camera to different positions within the scene to gain a different perspective.

* Lighting

For the lighting we use the Phong model presented in the laboratories, which uses the three different light components (ambient, diffuse, specular), to create the whole lighting effect.

* Fog generation

The algorithm used for the fog generation in this project is the Exponential Fog algorithm. This algorithm is based on the distance between the fragment and the camera, and it uses the density of the fog to calculate the amount of fog that should be applied to the fragment. The algorithm uses the following equation to calculate the fog factor: fogFactor = exp(-pow(fragmentDistance \* fogDensity, 2)); where fragmentDistance is the distance between the fragment and the camera, fogDensity is a value that controls the density of the fog and is a user-defined value, and exp and pow are mathematical functions. The exp function calculates the exponential of the value passed to it, and the pow function calculates the power of the value passed to it. The fog factor is then used to blend the fragment color with the fog color. The resulting color is then clamped between 0 and 1 to prevent the color from exceeding the range of valid colors. The final result is a scene where the further away an object is from the camera the more it is covered by fog. This gives a sense of depth and realism to the scene.

3.1.2 The motivation of the chosen approach:

The motivation for the chosen approach in this project was to create a flexible and interactive 3D scene for the user. By implementing a camera class, the user is able to navigate and explore the scene from different perspectives, and by adding a system for object movement and rotation, the scene becomes more dynamic. Additionally, the implementation of fog and skybox effects improves the overall visual experience. The use of libraries such as GLFW, GLEW, GLM and the implementation of shaders, allows for a more efficient and optimized execution of the project. The ultimate goal was to create a visually engaging and immersive 3D environment for the user.

## Graphics model

The application uses 3D models, which are stored in files with the .obj and .mtl extensions, as the building blocks of the scene. These models are loaded into the application and processed through a pipeline. In order for the models to be rendered correctly, their normals must be calculated. This is necessary for proper lighting and shading of the objects.

## Data structures

The project utilizes several data structures, including pre-existing ones like Camera and Model3D, as well as arrays and vectors for manipulating the position and appearance of objects.

## Class hierarchy

The project utilizes a number of pre-existing classes, with some additional functionality added. These classes include:

* + SkyBox class - used for loading and displaying the skybox.
  + Shader class - utilizing shaders in various scenarios.
  + Camera class - handling camera transformations such as movement, direction, and rotation.
  + Mesh and Model3D classes - work together to provide an interface for drawing textured objects on the screen. The Model3D class reads the object while the Mesh class contains the necessary function for drawing the object.
  + Main class - controls the other classes and brings all functionalities together.

# 4.Graphical user interface presentation and user manual

1. User manual

|  |  |
| --- | --- |
| **W** | Move camera in front, bringing the objects closer |
| **A** | Move camera to left |
| **S** | Move camera to back, distancing form the scene |
| **D** | Move camera to right |
| **3** | Scene representation |
| **2** | Wireframe representation |
| **1** | Point frame representation |
| **Moving to the left** | Moves the camera direction to the left or right, up or down |
| **Moving to the left** | Moves the camera direction to the right |
| **Moving forward** | Moves the camera direction up |
| **Moving backward** | Moves the camera direction down |
| **F** | Make fog appear |
| **G** | Make fog disappear |

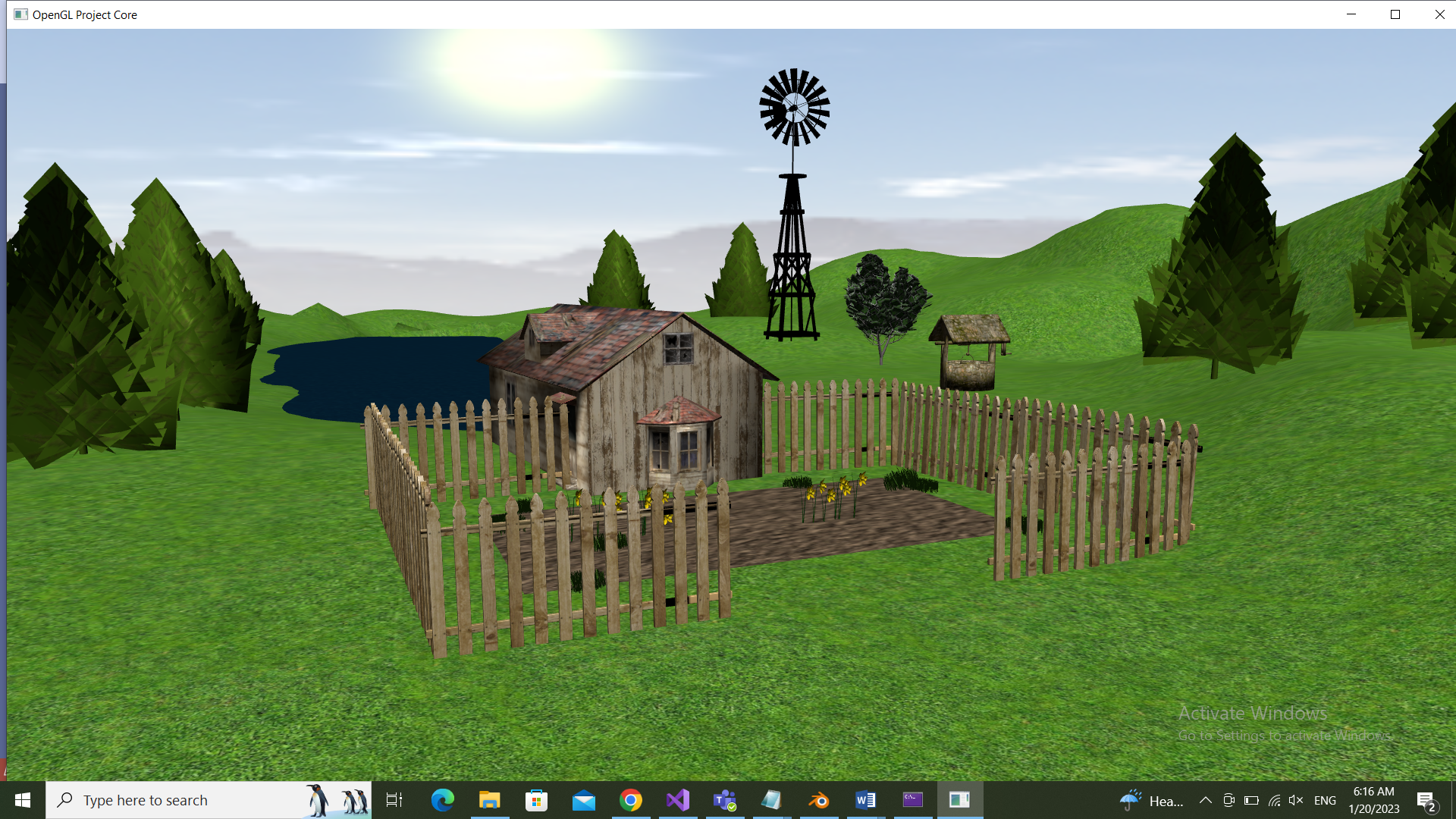


Figure 1 Scene map

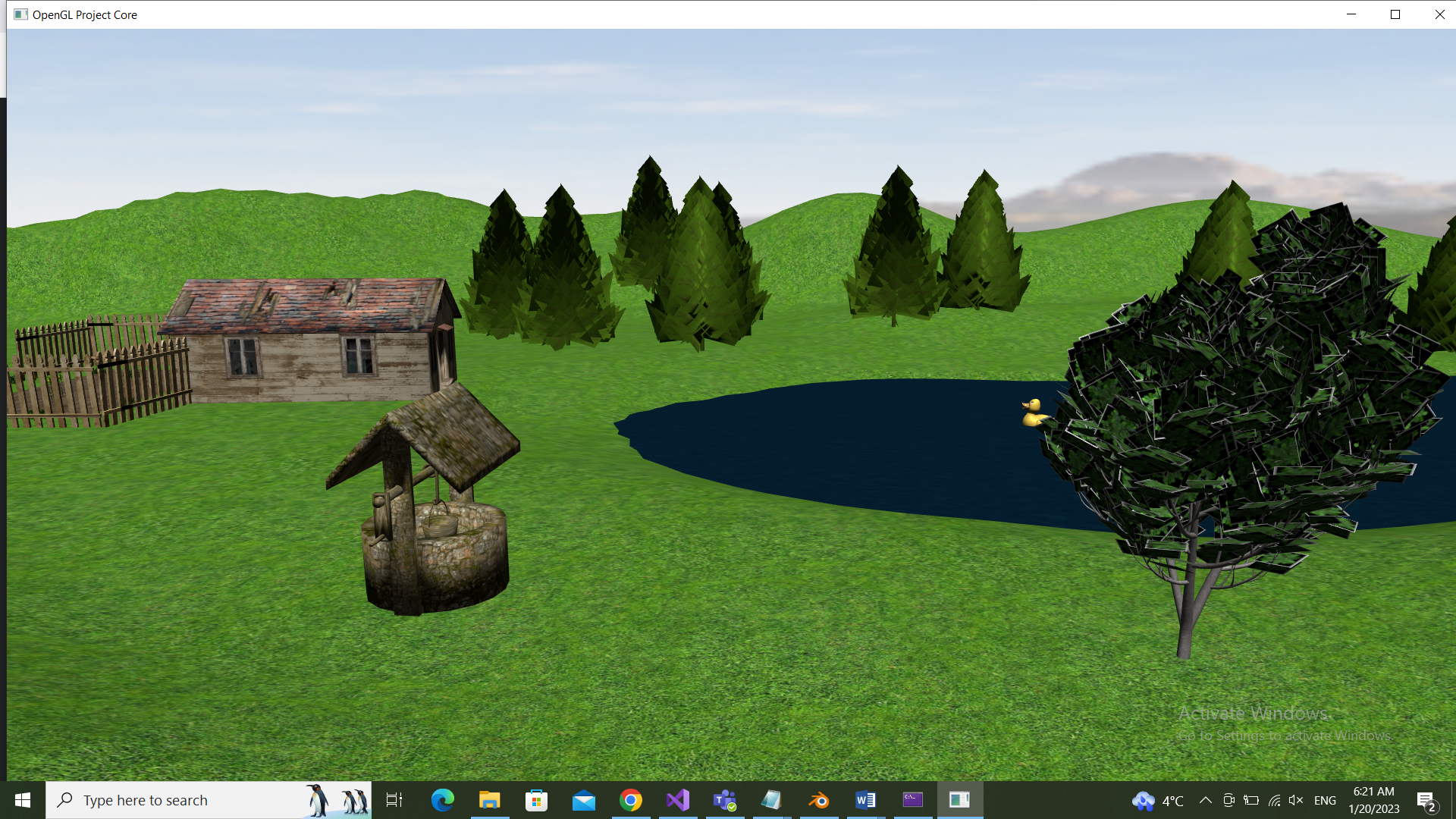


Figure 2 Behind the house: lake and well

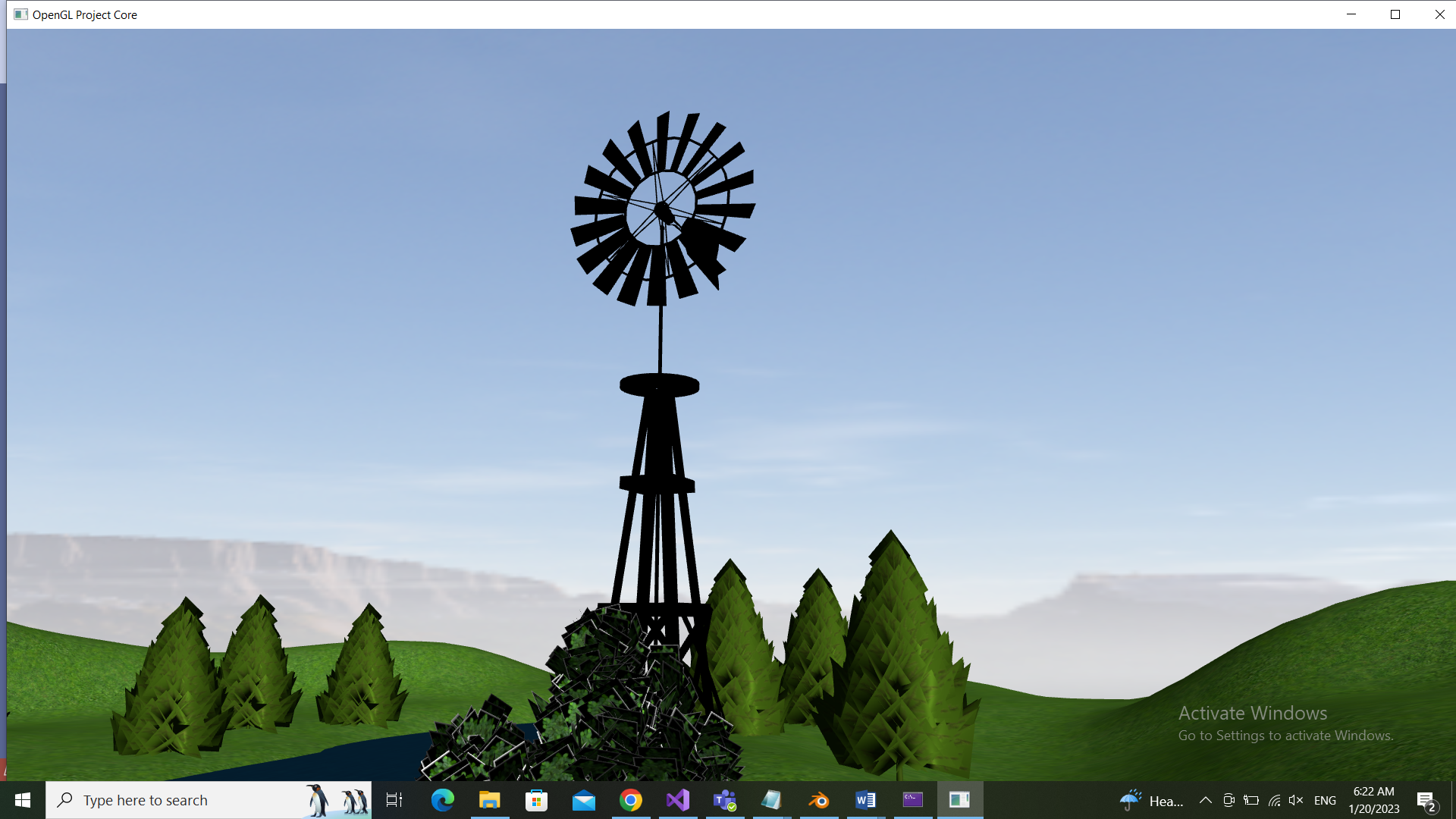


Figure 3 Rotating windwill

# 5. Conclusion and possible future developments

In conclusion, this project provided a basic implementation of a 3D scene with objects, textures, lighting, and camera movement. The implementation used several pre-existing classes such as the SkyBox, Shader, Camera, Mesh, and Model3D classes to handle different aspects of the scene.

For future development, some possibilities could include adding more complex lighting models, shadows, implementing a physics engine, or adding more interactive elements to the scene such as objects that can be manipulated by the user. Additionally, the implementation of more advanced techniques such as physically-based rendering or global illumination could enhance the overall realism of the scene. Furthermore, implementing a GUI would make the application more user-friendly. Overall, this project served as a good starting point for more advanced 3D graphics programming.

# 6. References

1. https://www.turbosquid.com/Search/3D-Models/free

2. <https://free3d.com/>

4. https://www.cgtrader.com/free-3d-models/

5. https://learnopengl.com/

6. https://moodle.cs.utcluj.ro