**Technical University of Cluj-Napoca**

**Faculty of Automation and Computer Scinece**

2021

**Farm scene**

**Implemented using Blender and OpenGL**

**Ștefan Ciuprina – group 30434**

1. Contents

[2. Subject specification 2](#_Toc61265451)

[3. Scenario 2](#_Toc61265452)

[3.1 Scene and objects description 2](#_Toc61265453)

[3.2 Functionalities 2](#_Toc61265454)

[4. Implementation details 3](#_Toc61265455)

[4.1 Functions and special algorithms 3](#_Toc61265456)

[4.1.1 Possible solutions 3](#_Toc61265457)

[4.1.2 The motivation of the chosen approach 10](#_Toc61265458)

[4.2 Graphics model 11](#_Toc61265459)

[4.3 Data structures 11](#_Toc61265460)

[4.4 Class hierarchy 12](#_Toc61265461)

[5 User manual 12](#_Toc61265462)

[6 Conclusions and further developments 13](#_Toc61265463)

[7 References 13](#_Toc61265464)

# Subject specification

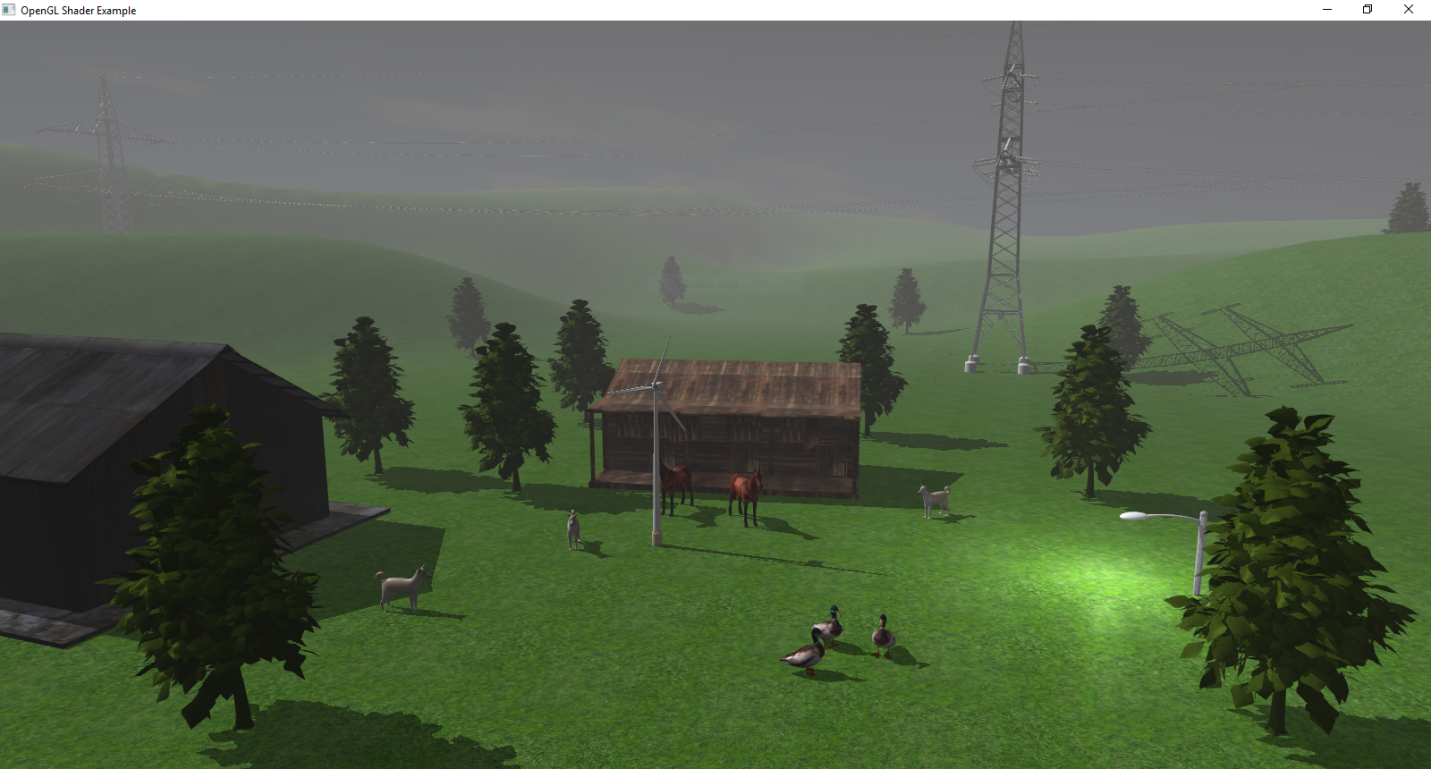
The purpose of this project is to design and implement a scene in C++ using OpenGL. Blender is used also for positioning, modeling and texturing objects to be later imported into the main project. The user is also able to interact with the scene, moving around it and triggering animations.

# Scenario

## 3.1 Scene and objects description

The scene represents a farm, hosting a number of domestic animals. We have two horses, three dogs and three ducks. The horses also have a stable, being present behind them. Other than those objects, we have an old house, 27 trees, a street lamp, a windmill and three electricity towers. Also, a ground, a skydome and a sun (represented as a sphere) are present, completing the decoration of the scene.

An overview of the scene can be seen below.



## 3.2 Functionalities

As mentioned in *section 2*, the user will not only view the scene, but also interact with it. The functionalities include movement around the scene using the mouse and the keyboard, triggering the windmill animation or the camera cinematic, viewing the scene with either solid or wireframe objects, moving the sun (changing the lightning this way) and displaying the depth map (used for shadow computation).

# Implementation details

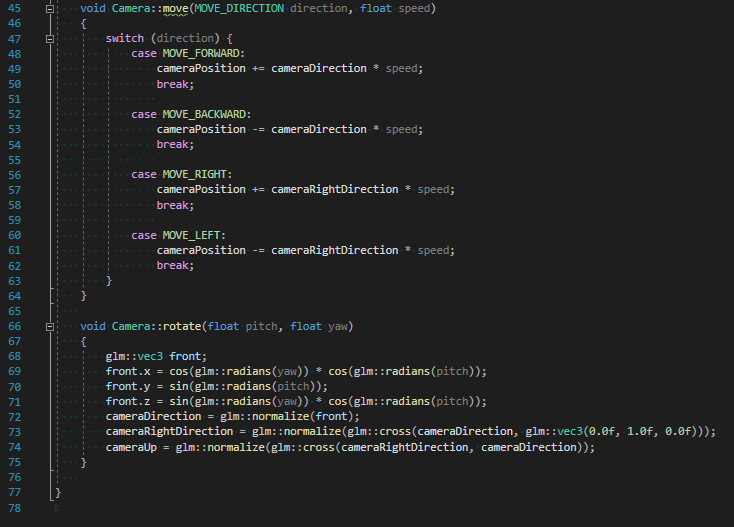
## Functions and special algorithms

### Possible solutions

#### 4.1.1.1 Camera movement

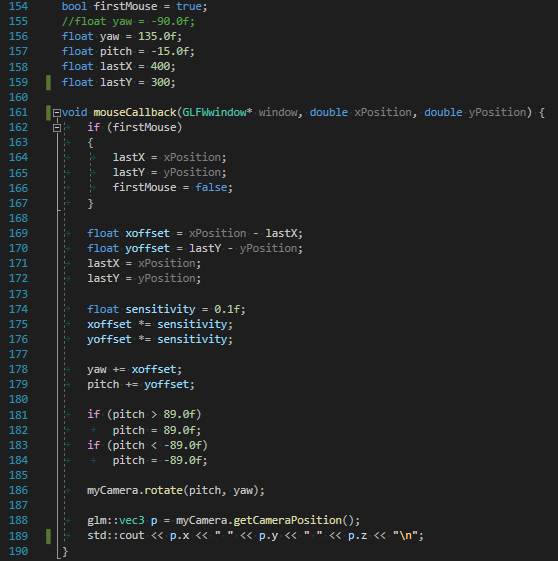
The camera movement has been implemented by following the tutorial on the Learn OpenGL website[1]. Using the move and rotate functions, we are able to use the mouse and the keyboard to move around the scene, and also display a cinematic, moving the camera automatically around the scene.

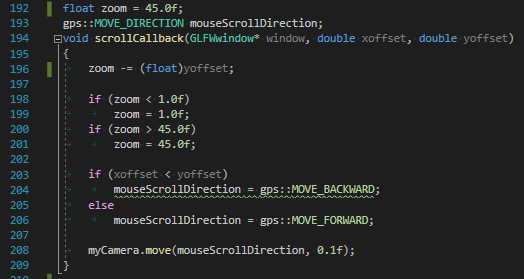
*Camera.cpp*



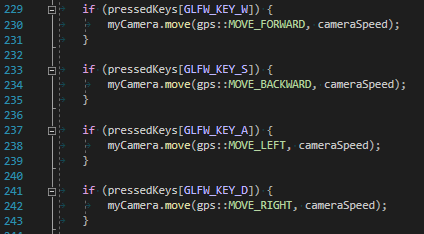
For mouse and keyboard movement:

*Main.cpp*



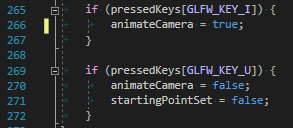


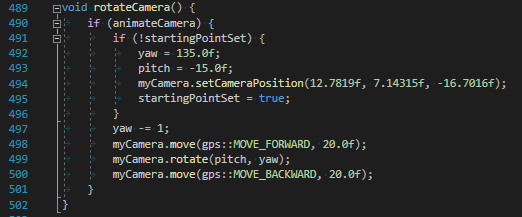
*Main.cpp, processMovement function:*



The cinematic is a rotation around the scene, triggered by the user using keyboard imput. The idea used to create this is to first set the camera in a specific position, looking at the center of the scene; then, at each frame, the camera will move to the center, rotate around itself, and then move back, all in one step, the user seeing only the rotation around the point.

*Main.cpp*

*-processMovement function*

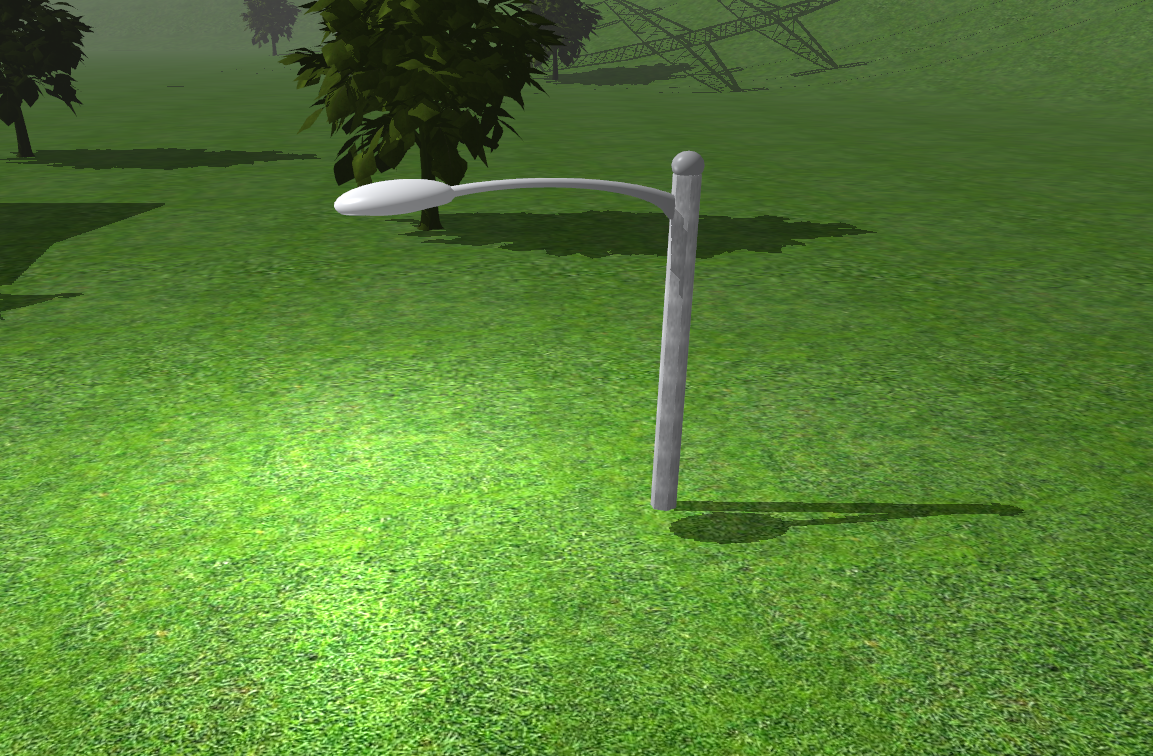


#### Light sources

The scene has a total of 2 light sources: one directional and a point light.

The directional light is used to simualte the light coming from the sun. Phong lightning is used, combining ambient, diffuse and specular light components. It has been implemented by following the tutorial on the Learn OpenGL website[2].

As mentioned earlier, there is also a point light, implemented using the resources from the Learn Open GL website[3] as well as the laboratory work for the Graphic Processing course[4]. The point light has been located at the street lamp object, symbolizing the light coming from the lamp.



#### Shadows

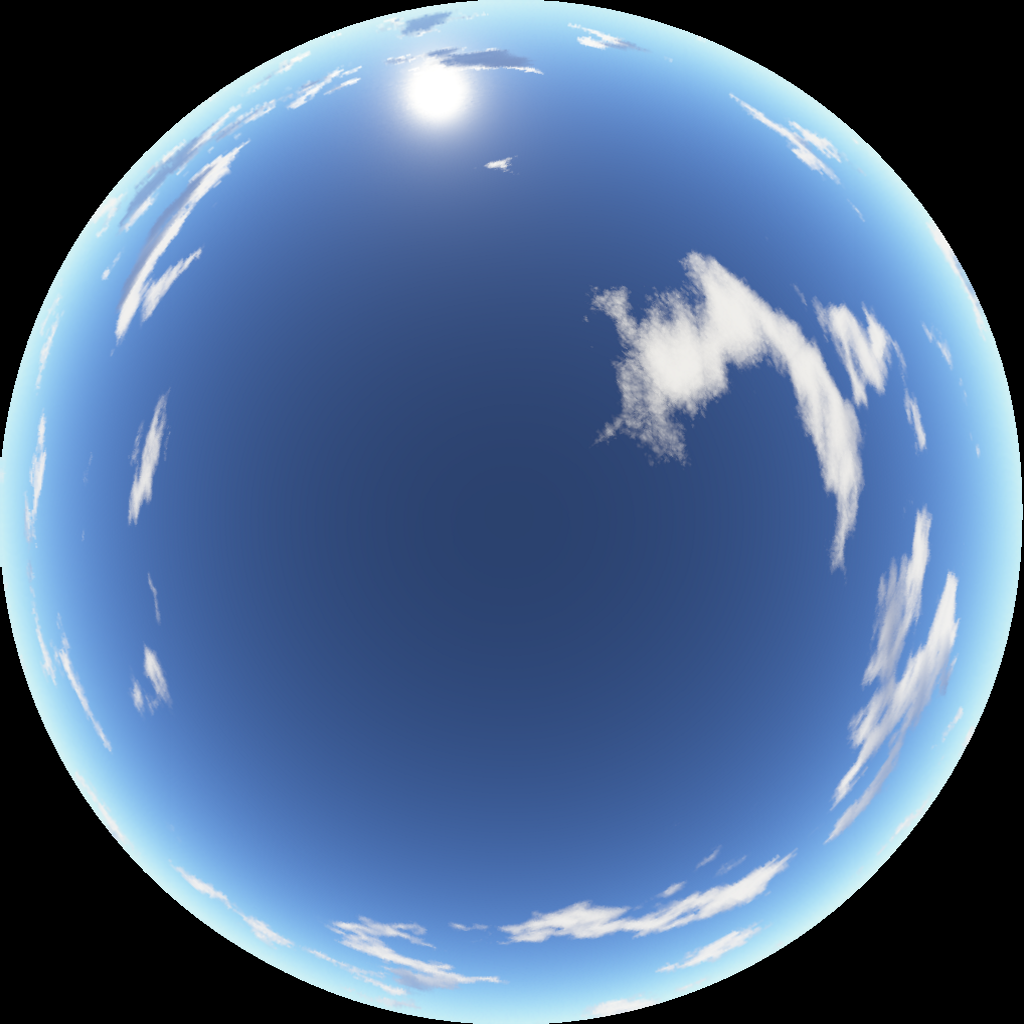
All of the objects present in the scene have shadows rendered, respective to the directional light. As mentioned in the earlier chapters, the directional light source (i. e. the sun) can be rotated around the scene by the user using the keyboard input, changing the shadows as well. An overview of the shadow (along with the directional light) can be seen in the picture showing a tree below:



The shadows have been implemented using the resources from the Learn Open GL website[5] as well as the laboratory work for the Graphic Processing course[6].

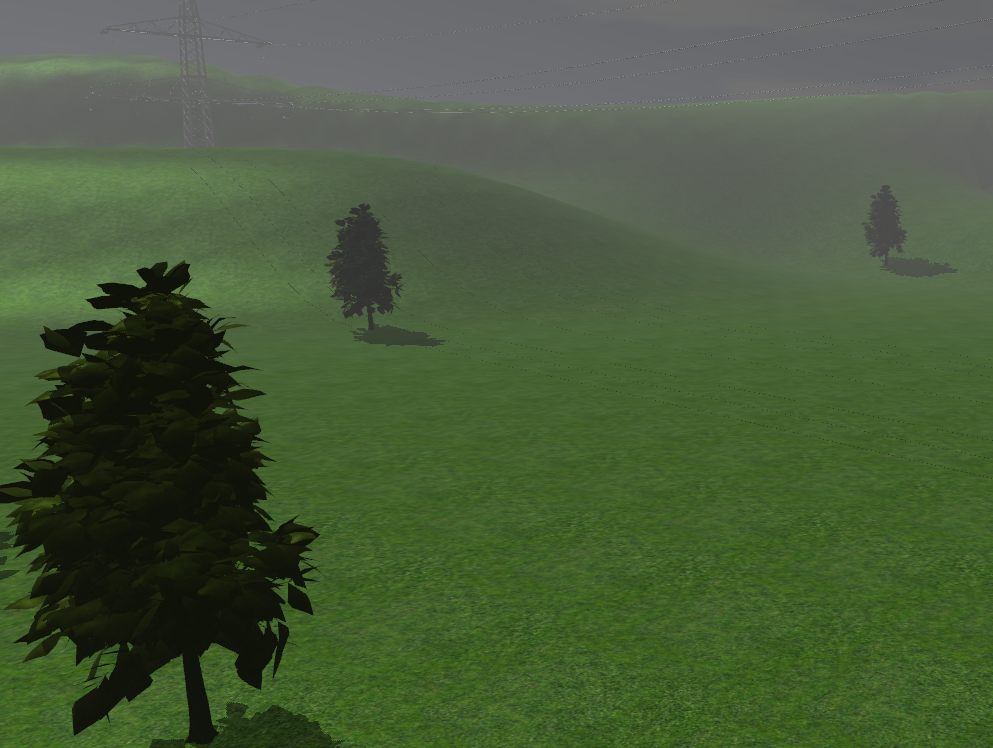
#### Textures

The objects used have been downloaded from the Free 3D Models website[7], being positioned in the scene and textured using Blender. An interesting example of texturing used is on the skydome, a dome placed over the whole scene with a specific texture symbolizing the sky. The image used as texture is shown below.



#### Fog

The scene presents a small fog, not too dense, objects being as faded as further distance they are. As the image below shows, the closest tree could be seen almost clearly, the first one on its left being grayer, while the leftmost one being even more faded. The fog effect could also be seen on the hills.

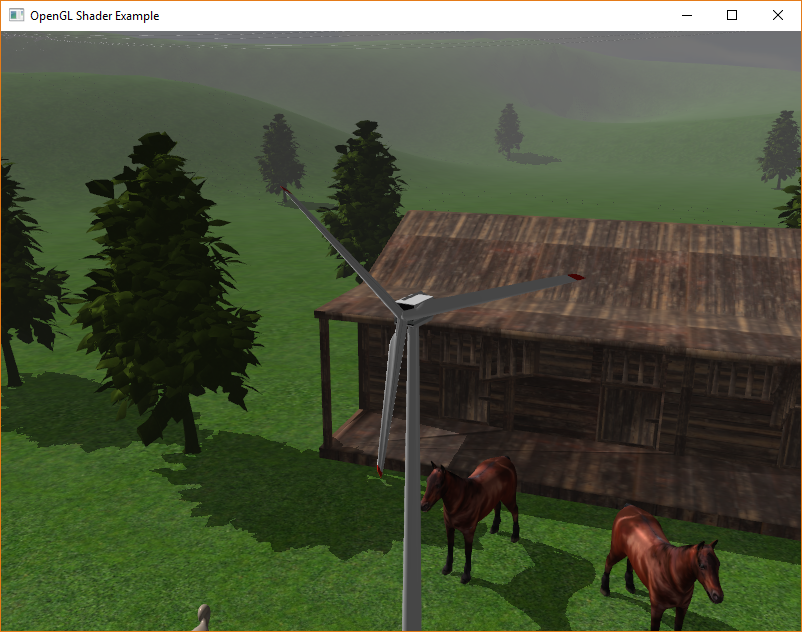
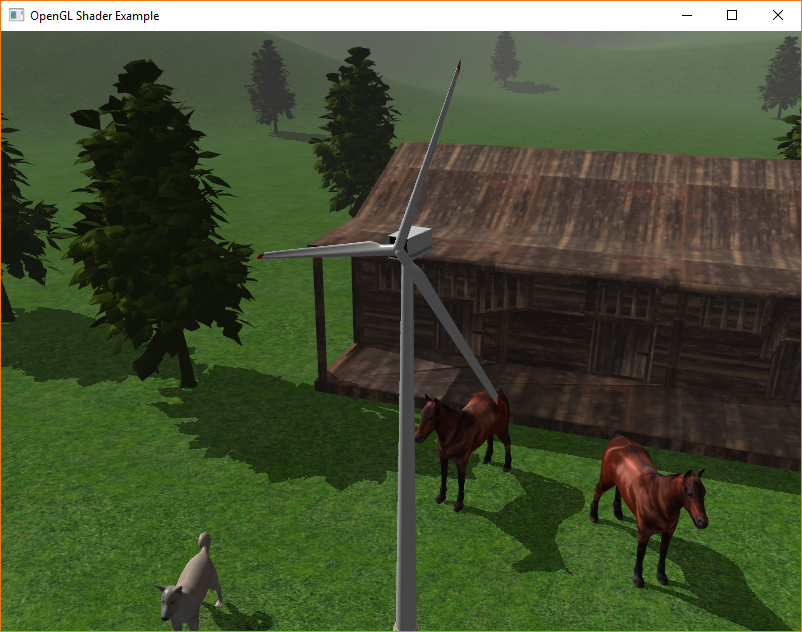


The fog has been implemented using the resources from the laboratory work for the Graphic Processing course[8].

#### Animations

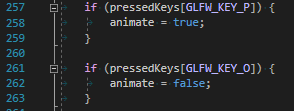
There are two animations implemented in the scene.

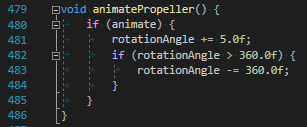
The first one is the windmill animation, the propeller rotating around its center, simulating it being blown by the wind.

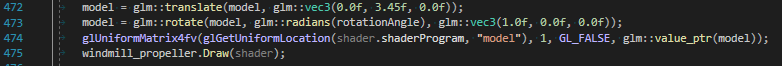


The object is placed at the origin of the scene. To bring it to its desired place, it is translated at each frame. When the user triggers the animation with keyboard input, the propeller is first rotated, then translated, by an angle which is incremented at each frame.

*Main.cpp*

*-processMovement function*

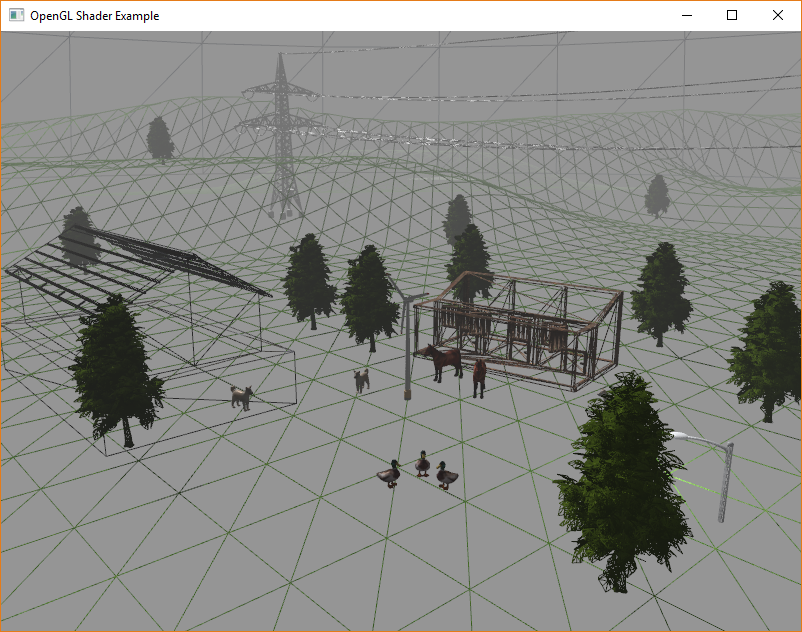


*-drawObjects function*

The second animation is the camera cinematic, which was previously presented in **section 4.1.1.1**.

#### Solid and wireframe objects view

The objects of the scene can be rendered in either solid or wireframe style, this choice being the user’s, using the keyboard input. A preview of the scene with solid objects can be seen in **section 2.1**. The one with wireframe objects is shown below.



The animals and the trees are also in wireframe style, they can’t be seen because of the detail they have.

### The motivation of the chosen approach

The approach presented in the previous section was motivated by the desire to create the farm scene as realistic as possible, with the proper ambient and decorations. The windmill animation has the purpose to make the scene more dynamic, while the camera cinematic helps the user have a faster overview of the scene, without moving manually through it.

## Graphics model

The objects present in the scene use polygonal representation. As the 4th course[9] of the Graphic Processing course says, this representation has the following advantages and issues:

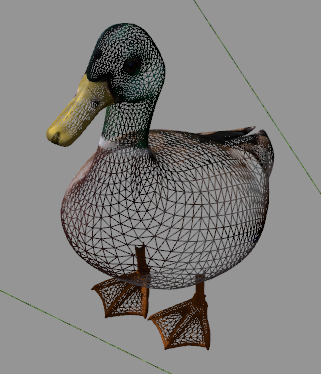
***Advantages:***

* **Ubiquitous in computer graphics**
* **Simple and straightforward**
* **Used as intermediate form for many other data structures**

***Issues:***

* **Approximation of curved surfaces**
* **Accuracy depends extremely on the number of polygons; e.g. zoom, scale**
* **Texture mapping**
* **Smooth surfaces**
* **Shading algorithms**

The polygonal representation can be seen using the wireframe view of the scene. Let’s take, for example, the object of a duck. If we zoom close enough to it, we can see the polygons that form the object.

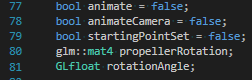


## Data structures

In order to store the details of different components creating the scene, different variables and data structures are being used. A data structure is used for encapsulating the values for calculating the attenuation of the point light, namely the struct data structure.

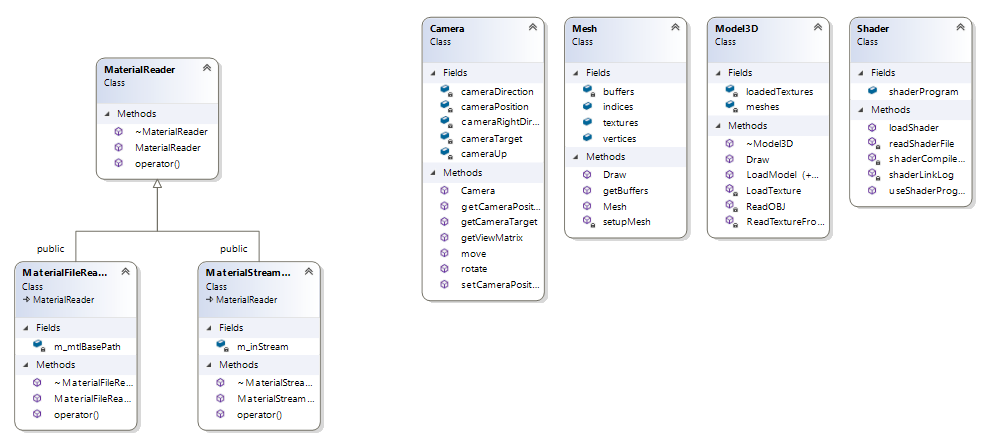


For triggering the animation of the propeller and the camera cinematic, boolean variables have been used. Also, a mat4 and a GLfloat are used for rotating the propeller.



## Class hierarchy

The UML Diagram for the classes present in this project is shown below.



# User manual

As mentioned in the subject specification, the user can interact with the scene using the mouse and the keyboard. Each feature with its respective command is presented below.

*Mouse:*

* Movement of the mouse: rotation of the camera
* Mouse scroll: zoom in / zoom out

*Keyboard:*

* W/A/S/D: movement around the scene
* J/L: rotation of the sun (directional light)
* I: play camera cinematic
* U: stop camera cinematic
* P: play windmill animation
* O: stop windmill animation
* M: show/hide depth map
* G: show wireframe objects
* H: show solid objects

# Conclusions and further developments

The goal of the project of creating a farm scene having with which the user is also able to interact was reached and the application is ready to use. Further developments could be done from this point, such as other weather effects (such as rain or snow), animating the animals or even adding a playable character, turning the scene into a game, the possibilities are so many. Nevertheless, the project helped me familiarize with OpenGL and has improved my overall programming knowledge.

# References

* [1]: <https://learnopengl.com/Getting-started/Camera>
* [2]: <https://learnopengl.com/Lighting/Basic-Lighting>
* [3]: <https://learnopengl.com/Lighting/Multiple-lights>
* [4]: <https://moodle.cs.utcluj.ro/pluginfile.php/93897/mod_resource/content/3/Laboratory_work_8.pdf>
* [5]: <https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping>
* [6]: <https://moodle.cs.utcluj.ro/pluginfile.php/93904/mod_resource/content/4/Laboratory%20work%209.pdf>
* [7]: <https://free3d.com/>
* [8]: <https://moodle.cs.utcluj.ro/pluginfile.php/93917/mod_resource/content/2/Laboratory_work_11.pdf>
* [9]: <https://moodle.cs.utcluj.ro/pluginfile.php/101841/mod_resource/content/0/gps-04-3D%20ObjRep%20(P1).pdf>