# Graphic Processing

Project Documentation


Student: Tudor Roxana Ioana

Group: 30434

Specialization : Computer Science

Year : 3rd

Teacher : Victor Bacu

# Contents

# 1. Subject specification

The subject represents the usage of OpenGL in order to create a 3D scene that create a photorealistic effect.

The 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. Applications use it extensively in the fields of computer-aided design (CAD), virtual reality, scientific visualization, information visualization, flight simulation, and video games.

In order to take advantage of the library I choose to implement a medieval village placed on a main commercial route.

# 2. Scenario

## 2.1 Scene and objects description

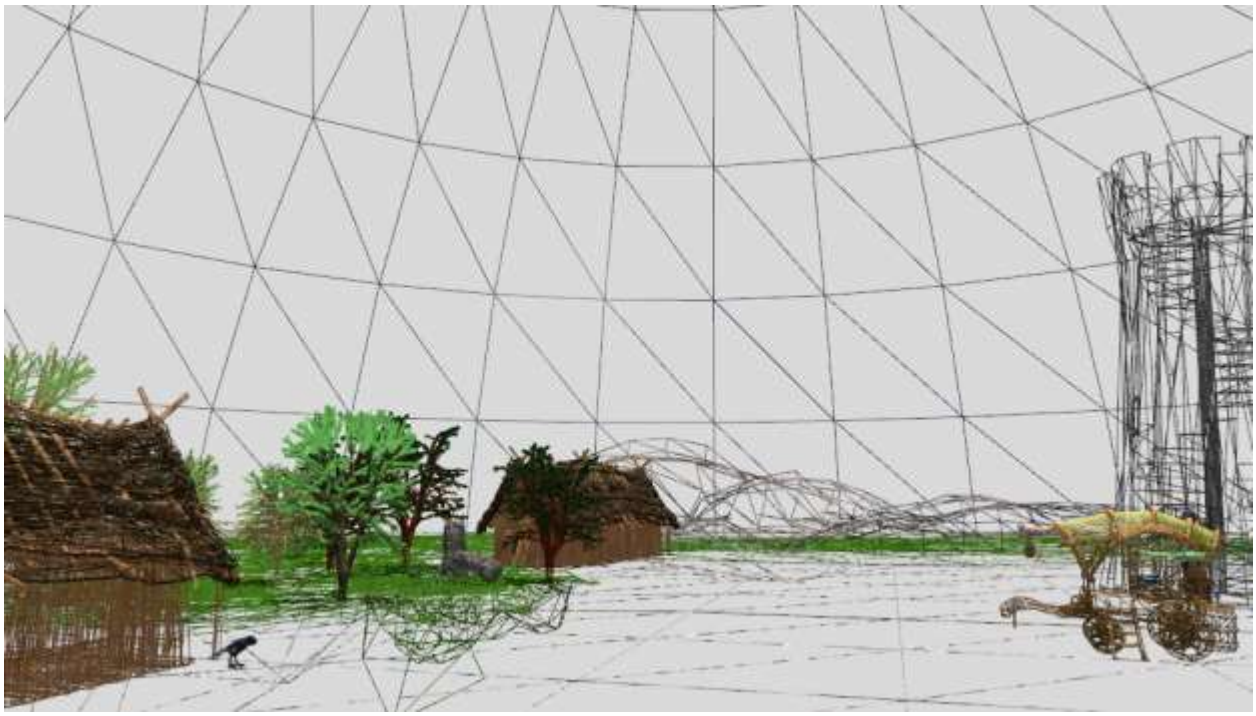
The scenario present a small medieval village located on a main commercial route. There little meadow could be noticed in the background with a thirsty deer that came by the lake to drink some water. The mountains could be noticed in the background. In this village we can notice a knight that guards by tower, he watches over the village. There is little a dog, three houses, a trading cart and a little impatient crow that begs for food.

I choose to put such objects in the scene in order to make it more seem more alive.

There could be seen the deer the little crow and an apple .

When the fogs come in the village

Wireferame view



## 2.2 Functionalities

The program start with a camera animation, where the camera moves forward and then perform a little rotation in order to have a panoramic view. The user can have a visualization of the entire scene by moving from keyboard and changing direction through the mouse movement. There area keys that let the user access the solid, wireframe and point view. Fog will appear if you press the button and also the night will come over the town, having a little light from the lamp placed on the trading cart.

I have multiple sources of light: directional light, spotlight (comes from the lamp of the trading cart) and the point light. The spotlight and the point light can be turned on/off by pressing the corresponding keys.

# 3. Implementation details

## 3.1 Functions and special algorithms

### 3.1.1 Possible solutions

To implement the scene I took advantage of the main functions of the OpenGL library, an the functions found in the template provided for the project.

glfwCreateWindow(..) – creates the window

glViewport(..) – set the viewport

glGenTexture() – creates the texture

glBindTexture() – used to bind the texture

glUniform..() – used to send data from the main application to the shader

etc..

In order to be able to move the camera direction according to the mouse pointer I have used the mouseCallback() function. In order to compute the change of direction I have used the rotate function found in the Camera class. For the keyboard interpretation there exists the keyboardCallback(), which uses the function move() written in Camera class. The move function has the following directions (forward, backward, up, down, left, right).

In order to draw and compute my objects I have created renderEXMPL () that calls my shader, compute the transformations and send the new matrix model back to the shader.

The matrix model and the normal matrix model are one of the most important components that are used.

In function renderScene() we compute the shadows and the scene .In order to do that I have to use the shadowDepthMap shader and the myBasicShader.

The initUniforms() function is used in order to set the uniform variables and send them to the shader.

The initShaders() function is used in order to load the Shaders used in my project.

The initModels() function is used in order to load all the objects used in the project.

For the spotlight computation I have used an algorithm that takes in account the angle of a cone and computes the intensity of light accordingly. Changing the inner cone and outer cone values to a lower value it will make it seem more like a point light. Because the spotlight is a particular case of point light.

### 3.1.2 The motivation of the chosen approach

Using the materials provided in the laboratory about light computation, shadows, performing different transformation like translation, scaling , rotation, adding fog I was able to have a well defined sketch to start from. After that all I needed to do was to play around a little with functions and the values in order to adapt it for may project.

## 3.2 Graphics model

The models were downloaded from specific websites, that had free 3D objects. Most of them had textures. For the others I have looked for textures online. Then I have imported them in Blender and edited them as I will

It was a challenge to find good textures or to map the one's I had, on the object in order to create a high quality effect.

## 3.3 Data structures

The only data structures used were vectors, matrices or primitive type like int or float required for uniform variables. I have added some functions in order to be able to send data object or calculate the lights or fog.

## 3.4 Class hierarchy

**Camera**: The camera class is the most important one, because it provides the movement for the user

**Mesh:** represents class that setup the arrays and buffers for a particular object or model and then draws it.

**Model3D:** contains the meshes used for one object. Loops over faces and applies the texture or color to each one of them and then draws the object

**Shader:** contains the shader operation for creating and loading the shader programs

**Window:** contains the methods used to create delete or set dimensions of the window

**Stb_image:** contains the methods used to parse an image and interpret it

**Tiny_obj_loader**: contains method that help parse an obj/mtl file and load it

# 4. Graphical user interface presentation / user manual

This are the main keyboards use in order to interact with the scene:

W – move forward
A – move backward
S – move left
D – move right
T – move up
Y – move down
Q – rotate to left
E- rotate to right
G – increase fog density
J - turn on the spotlight
H – turn off spotlight
O – turn on wireframe view
N – turn on solid view
P – turn on point view
B- turn on/off camera animation
V -turn on apple falling animation

# 5. Conclusions and further developments

In conclusion, this provided a good challenge to work with Blender and OpenGL, in order to deepen my knowledge and gain experience.

The code part was a bit difficult to understand at first until I fully understood the pipeline and how the communication between shaders is done.

As further developments I take in consideration:  adding rain, addition of more light sources, addition of more characters in scene, sound.

# 6. References


- Graphic Processing, courses
- Graphics Processing , labs
- Youtube
- https://learnopengl.com/
- https://www.youtube.com/watch?v=45MIykWJ-C4&t=3618s
- https://www.turbosquid.com
- https://www.cgtrader.com/
- https://free3d.com/