**GP Documentation**

Lacatus Arpad Alex

Group 30433

**Table of contents**

1. Subject specification . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .2
2. Scenario . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .3
3. Implementation details . . . . . . . . . . . . . . . . . . . . . . . . . . . .4
4. Graphical user interface presentation / user manual .6
5. Conclusions and further developments . . . . . . . . . . . .8
6. References . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .8

**1.Subject specification**

Implement an application that consists in the photorealistic presentation of 3D objects using OpenGL library. The user directly manipulates by mouse and keyboard inputs the scene of objects.

The project must implement elements of graphic processing such as

* visualization of the scene through scaling, rotation, translation, camera movement implemented using keyboard and mouse, and animation.
* specification of light sources (minimum two different lights)
* viewing solid, wireframe objects, polygonal and smooth surfaces
* texture mapping and materials considering textures quality
* exemplification of shadow computation
* texture mapping and materials
* photo-realism, scene complexity, detailed modeling, algorithms development and implementation (objects generation, collision detection, shadow generation, fog, rain, wind), animation quality, different types of light sources (global, local, spotlights)

**2.Scenario**

**2.1 Scene and object description**

The user is presented with a forest with different elements of vegetation and a house.

The house is situated at the edge of the forest. In the forest we can find trees, bushes, rocks, and windmills that spin. These objects are generated randomly when we run our program. There are 2 types of trees used.

The static objects are:

* house
* trees
* rocks
* bushes
* windmills

The dynamic object is the nanosuit: that is able to move right/left, front/back, rotate and scaled.

**2.2 Functionalities**

* When the user enters the scene , he can traverse the forest using YGHJ buttons that move the camera and look left/right and up/down with the mouse that rotates the camera.
* The red-light source can be turned off and on with the keys O and P
* The natural light rotates naturally to give a false sense of time passing by the changes on the shadow of our objects.
* The representation of the scene can be changed between polygonal, wireframe and pointframe with keys C, V, B;
* With WASD we can move the nanosuit, QE rotate it, ZX scale it;
* The windmills have propellers that rotate continuously.
* The skybox can be seen in the distance as a cloudy regular sky
* The fog element can be seen at the first glance when looking at the elements furthest from our camera

**3.Implementation details**

**3.1 Functions and special algorithms**

For this project we have used algorithms already presented in our laboratories.

Some special functions are:

* Shadow generation – We use Shadow Mapping technique with the use of the depth map as follows: we first render the scene from the light’s point of view, and store the values in the depth map(to note here we used a combination for the view point of the camera and light in order to generate our shadows where the user is on our map and not on the whole map in order to prevent acne and/or extra memory) then we render the scene from the camera point of view and compare the depth value of each visible fragment with the values stored in the depth map in order to get the effect of shadow
* 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. We use both directional and point light source.
* Animations – Here we have the movement of windmills , nanosuit and camera that can be automated or triggered by the user. The windmills rotation is done by rotating at each render cycle of the propeller with a float, this value is in incremented at the end of rendering. The triggered modifications are similar , with the press of certain keys floats representing position on x/y axis, rotation, scale are modified. When the nanosuit is render it takes this modified values thus creating the effect of change.
* Fog – Here we have the implementation from the laboratory 12. By calculating the distance between the object and our camera we add the fog color(white) to the object’s color based on a specific fog factor.

**3.2 Graphics models**

The base components in our OpenGL project are the objects. These are represented by their .obj, .mtl and their texture files. The .obj specifies the vertices and edges of our objects and which .mtl to be used, the .mtl tells us the app how to use the textures ,while the textures are regular jpg or png files. After we make the specific transformations we compute model, modelLoc and the normal for each of our object in order to be able to draw them.

**3.3 Data structures**

We mainly use the data structures provided by openGL library or the header files from the labs. For our implementation we use Model3D, Camera, Window, Glint, GLfoat, vectors and matrices, SkyBox and Shaders. For the objects randomly generated we used a matrix of objects.

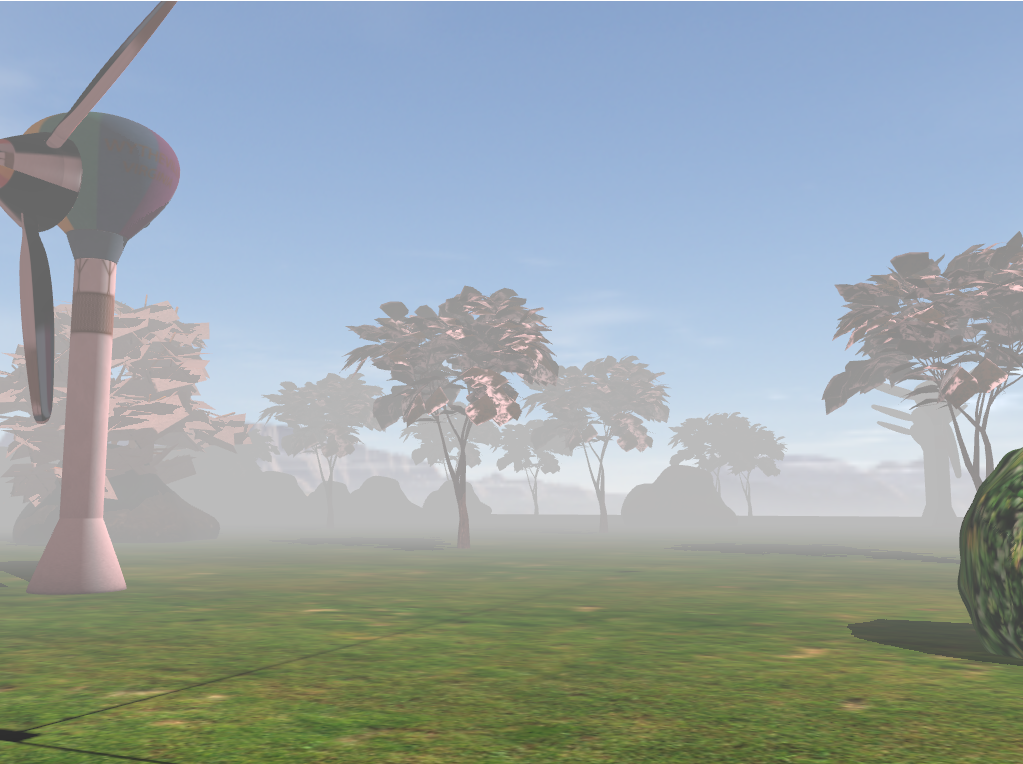
**3.4 Class hierarchy**

The used classes are the ones already included in the starting project, with the mention, that I implemented some additional functionalities. Below I will provide a brief explanation of each class:

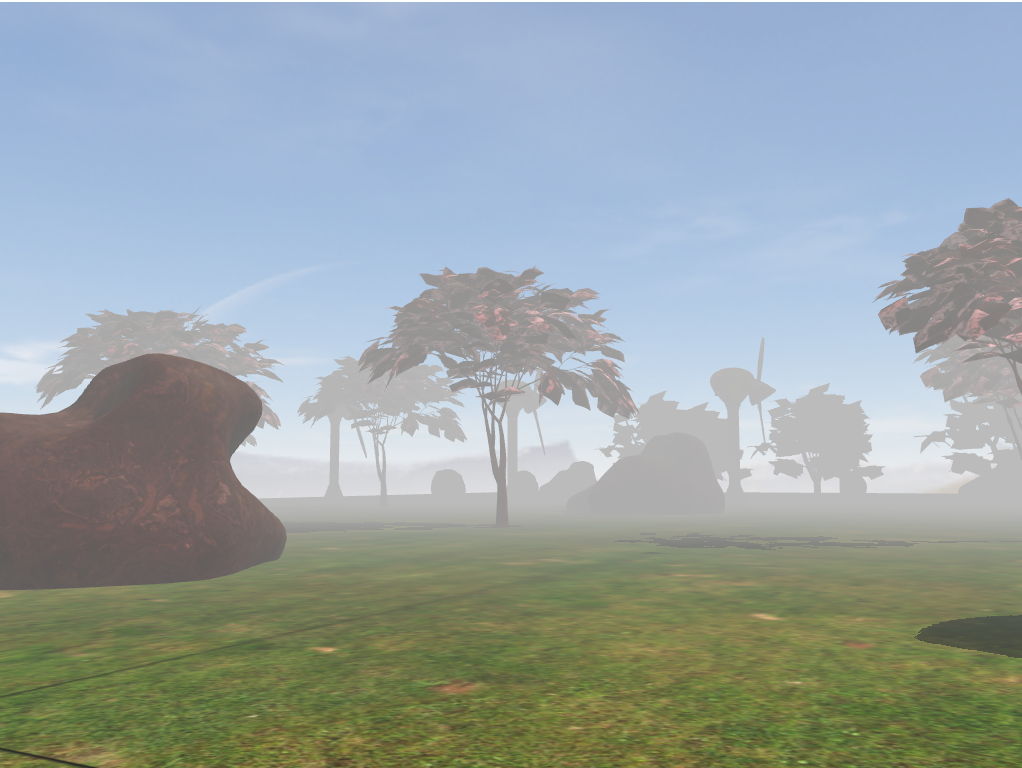
* **SkyBox** class – we use this class for loading and drawing the skybox.
* **Shader** class – we use this class for loading and using specific shaders depending on situation.
* **Camera** class – this class is used for transformations of the camera, such as moving and changing the direction, rotation.
* **Mesh & Model3D** we use them together to create our objects on the screen. While Model3D creates the image from the textures, obj, mtl files the mesh is responsible for drawing the actual object on the screen .
* **Main** class – makes use of all the classes. Here we have implemented all the functionalities of our project.

**3.Graphical User Interface and user manual**

Here we have some representative screenshots:

****

****

****

Here we have the user manual:

|  |  |
| --- | --- |
| WASD | Moves nanosuit forward/left/backward/right |
| QE | Rotates nanosuit |
| ZX | Scale nanosuit up/down |
| YGHJ | Moves camera forward/left/backward/right |
| CVB | Changes view to line/point/polygonal |
| OP | Turns off/on the red-light |

**5.** **Conclusion and possible future developments**

I consider that this project helped me better develop my understanding of computer graphics and provided an opportunity to better myself and learn how to create a scene with a realistic setup such as fog, shadow, animation, and sky.

If I were to chose an improvement first I’d pick the collision detection as it would bring more realism to this project. Secondly the movement of the nanosuit could have been better done with the use of animation.

**6.References:**

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

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

3. https://learnopengl.com/

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