GPS Project

Dan Mitrea
Group 30431

1. **Contents**

2. Subject specification

The subject consists of a photorealistic representation of 3D objects using OpenGL library. I directly manipulate by mouse and keyboard inputs the scene of objects. My project represents a scene in a Sci-Fi city, where a fictional character tries to escape an army of enemies that is chasing him. The initial state of the scene looks like this.

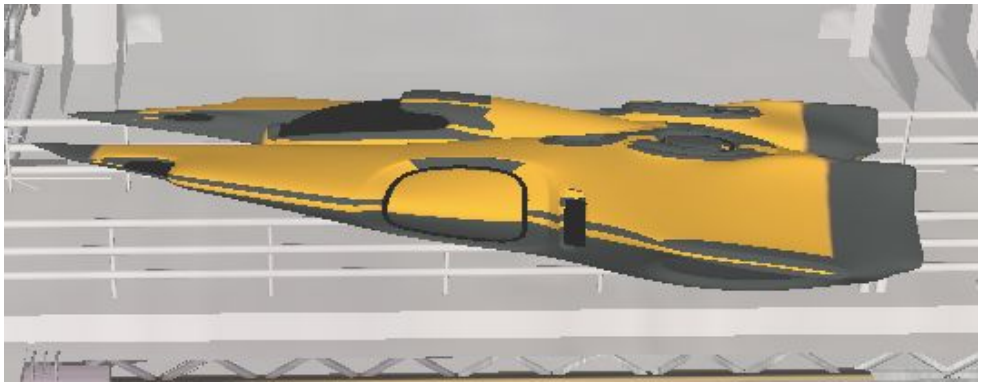
3. Scenario

3.1. Scene and Object Description

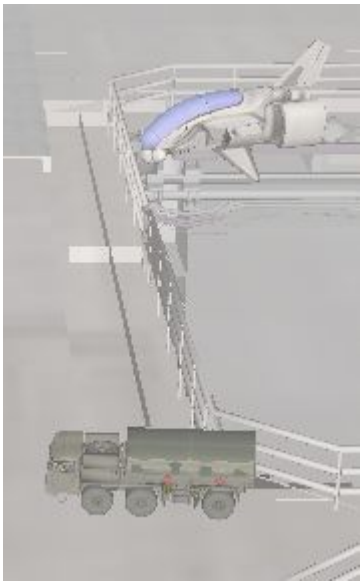
The scene consists of a big city, with skyscrapers and interesting buildings, a city that looks like one from the future. Even though the buildings are very tall and cover almost the whole eye sight, if you zoom out, you will actually see a background consisting of a sci fi scene. This background is represented using a skybox. There is a light source situated in front of the character's initial placement. By moving this light source, you can actually see the illumination of objects changing, and also the shadows of them.

I have chosen the following objects in my scene. Due to the fact that time, future and past, mean nothing here, our little character is being chased by aircrafts from the future, and German military trucks.

One of the starships chasing the character:



A truck and another aircraft going after our guy:



A big Star Wars like aircraft rotating around the scene:

The entire scene is surrounded by fog, to make it more mysterious. By zooming out, the used can actually see the scene getting grayer and more difficult to see.

3.2. Functionalities

The project has some interesting functionalities. First of all, the user can visualize the scene from different angles, by moving the camera using the keyboard, or by using the mouse. This way, he can explore the scene and see details that maybe are hard to notice from the first look. Also, by using the keys 9 and 0, he can toggle between different drawing modes.

Another functionality is that one starship is rotating all the time above our objects, making the scene look more interesting. Everything starts in the moment the character moves. This is the time when the enemies begin the chase. The character needs to run, get in the car, drive to a helicopter, and then fly to safety. The car and heli can not move until the driver gets in them, and when the enemies reach our guy, the game ends.

4. Implementation Details
4.1. Functions and special algorithms

One of the functionalities of the application is the way the user interacts with the objects. In the beginning, he can only move the character, until it reaches the truck. After that, the character disappears from the screen and gets in the truck. The truck then can be moved using other keys, to the heli. Only in the moment it reaches it, the helicopter can fly using other keys. I have done this by choosing some variables and incrementing/decrementing them upon pressing these keys, and using these variables to translate/rotate my objects.

Another functionality is the way the enemies move. I have precomputed some fixed coordinates to use them as references: their coordinates in relation to the initial coordinate of the character and truck. Then, using some mathematical relations, I am able to tell when any enemy has reached my character/truck if the character is in it, and stop the game if this happens.

```
//enemies chasing
if ((moveCharacter == true || moveCar == true) && gameOver == false)
{
    lr1 -= 0.1f;
    lrVehicle -= 0.05f;
    lrAircraft -= 0.1f;

    //detect of enemies caught our guy
    if (moveCharacter == true) {
        if ((lrAircraft + 21.0f - characterX <= 0.0f) || (lrVehicle + 13.0f - characterX <= 0.0f) ||
            (lr1 + 16.0f - characterX <= 0.0f)) {
            gameOver = true;
        }
    } else if (moveCar == true) {
        if ((lrAircraft + 35.4f - carX <= 0.0f) || (lrVehicle + 29.4f - carX <= 0.0f)
            || (lr1 + 32.4f - carX <= 0.0f)) {
            gameOver = true;
        }
    }
}
```

For example, I found out that the aircraft's X coordinate would be 21.0 in the moment it reaches the initial position of the character, and by using it as reference, I can signal if the enemy reached our guy anytime. Same can be said for the car.

4.2. Graphics model

All models are .obj files, which I have downloaded from the internet, namely from the website Free3D . Each objects has a separate folder in the project file. Most of these objects were imported, then exported using Blender, which I consider helped me a lot and enabled me to write less code. I am aware that it changes their coordinates, but by "trial and error", I managed to place them exactly where I wanted in the scene. I also used glm functionalities as translate, scale, and rotate.

4.3. Data Structures

My application has OpenGL specific data structures, like matrices and vectors, all specific to the glm library. It also contains vertex arrays and more complex data structures, used by the framework to generate its

objects and know exactly all the information it needs about them, like shape and position in the scene.

4.4. Class Hierarchy

My project's class hierarchy follows the exact hierarchy as in the laboratory provided code. It contains many classes that are used to model objects, the scene, the camera movement, the shaders, and even the background image("Skybox"). All of these classes are then imported in the main class and, combined together, create the representation of the scene as it can be seen above. Also, their functionality is of great importance, because they create an intuitive representation of the world we represent. They also make the whole application easier to debug and improve in the future.

5. GUI Presentation / User Manual

The graphical user interface is exactly as I have previously presented it. The whole action takes place in a small portion of a big Sci-Fi city. Using W,A,S,D, the user can move the camera to obtain a better look of the scene. Also, by moving the mouse, the same action can be obtained, as it aids in the camera movement. The following keys are used to further manipulate the objects and the scene: 9 - activate wireframe drawing mode, 0 - activate normal drawing, LEFT - move the character left, RIGHT - move the character to the right, E - rotate the character right, Q - rotate the character right, J - move the light source left, L - move the light source to the right. UP - move the truck to the left, DOWN - move the character to the right(both of these are possible only if the character is inside the truck), Z - lift up the helicopter, X - descend with the heli, T - move the helicopter to the right, G - move the helicopter to the left, D and G - rotate the helicopter. Also, in the moment the character reaches the helicopter, the enemies stop moving, as they realize that our guy has escaped them.

6. Conclusions and further developments

In conclusion, this application models a scene where a fictional character has to escape an army chasing him. The whole world can be manipulated and viewed differently, using specific keys, or the mouse. The objects interact between them, and their movement is dependent on other objects. The used is responsible for the final state of the game, which can be: WIN or LOSE.

There are several improvements that could be made. There could be more objects in the scene, so that it would look more impactful. Also, the objects that the user controls, could also rotate(namely the truck, the character can rotate) so that the game would be more interesting to play. What is more, another light source could be added, so that the scene would look more realistic.

7. References

I have used the source code provided in the lab as main reference and as a starting point. Everything has been built on this. Besides this, I has used some websites to provide me extra knowledge and ideas where I needed them. I used Learn OpenGL and Stack Overflow to seek answers to questions I could not find a solution on my own. I have also used Free3D to get my objects with textures applied to them.