**Graphical Processing Project**

**Antonescu Maria-Cristina**

**Group 30431**

**CONTENTS**

1. Subject specification

2. Scenario

2.1. scene and objects description

2.2. functionalities

3. Implementation details

3.1. functions and special algorithms

3.1.1. possible solutions

3.1.2. the motivation of the chosen approach

3.2. graphics model

3.3. data structures

3.4. class hierarchy

4. Graphical user interface presentation / user manual

5. Conclusions and further developments

6. References

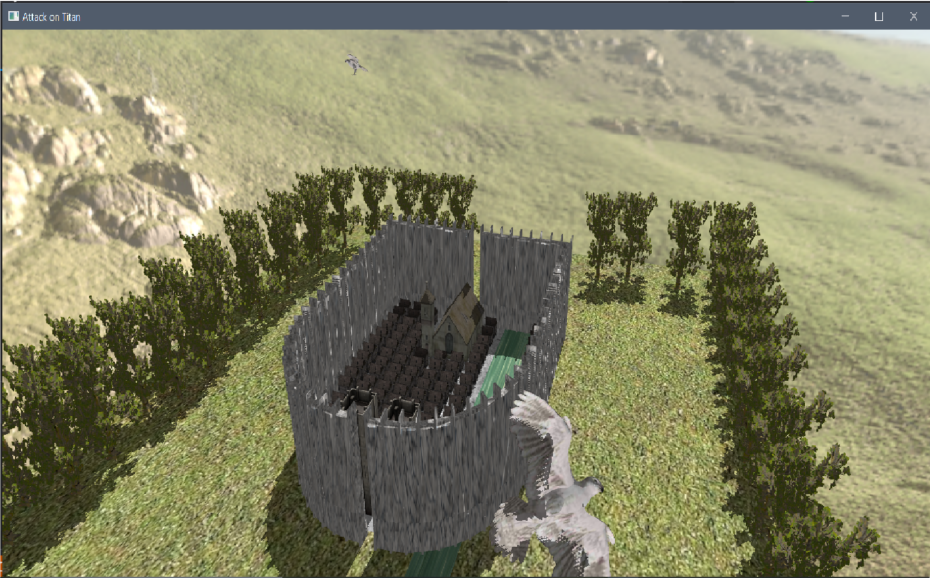
1. **Subject specification**

The goal of this project was to implement a photorealistic scene in OpenGL., using the knowledge we have accumulated this semester at the Graphical Processing laboratory. 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.

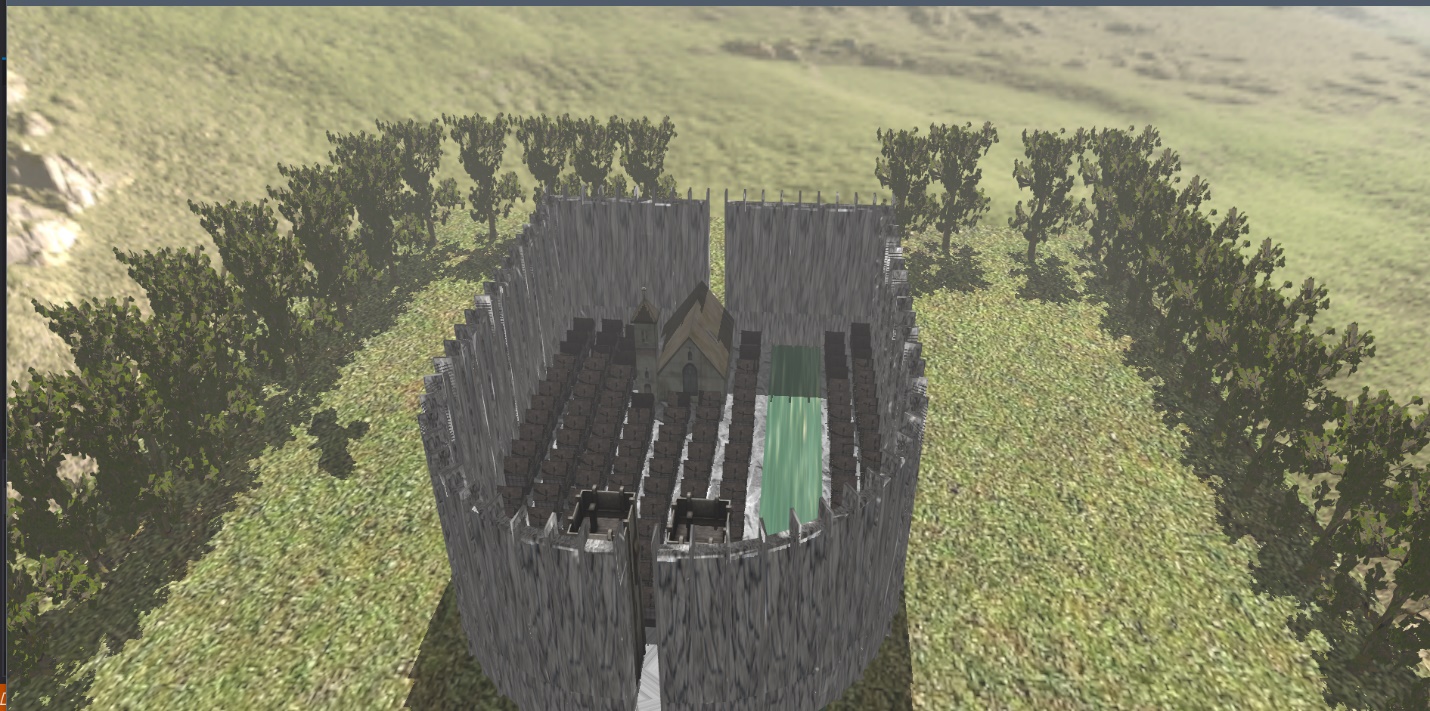
I have chosen to implement a scene inspired by the very popular anime Attack on Titan, with which the user can directly interact by using the mouse and pressing certain keys on the keyboard.

1. **Scenario**
   1. **Scene and objects description**

My project is a representation of the Trost district in Attack on Titan. The city itself has houses, a river passing through it, a church and two defense towers right at the entrance. There are 4 falcons flying above it in a circle and it is surrounded by trees.







****

Another object worth mentioning is the titan which appears behind the Trost, which is supposed to mimic the first scene of the anime.



* 1. **Functionalities**

Like I have mentioned previously, the user can interact with this scene using the mouse and the keyboard. The visualization of the scene can be done in wireframe mode, point mode or polygonal mode. The special effects implemented in this project are the rain and the fog, both of which appear at the press of the mouse. There are 2 types of camera animations: translation and rotation. The rotation gives a tour of the city and the translation makes the camera move forward, towards a titan which has been animated. There are 4 birds which continuously rotate around the city. Another key also lights off a yellow point light in front of the city.

1. **Implementation details**
   1. **Functions and special algorithms**
      1. Possible solutions

This project is done only using the GLM library, which has already implemented data structures such as matrices and operations such as translation and rotation and the GL and GLFW libraries, which enable the programmer to draw 3D objects with many in-built functions.

I mainly worked in the drawObjects(), processMovement() and renderScene() functions from the main program. In the drawObjects() function, I drew the objects on the screen and I performed operations such as translation and rotation. In the renderScene(), I modified the camera position for the rotation. In the processMovement(), I assigned to each key a specific action. I also used the initUniforms() function to send the lightPosition and color of the point light source to the fragment shader.

I also wrote a separate function in Camera.cpp called animate(), which rotates the camera around the city. It performs a translation and a rotation and then updates the other parameters of the class.

The rain implementation was quite easy, I imported a drop object in Blender and then multiplied it. I imported all the 500 drops as a single object. I initialized a variable rainOk, which I changed from the keyboard. If it is 1, then the rain object starts translating with rainSpeed on the Oy axis. The fog is rendered in a similar manner, since it is computed at the press of a key, but it is computed in the fragment shader.

The camera animation has been done in the Camera class, I initialized the cameraPosition with a value given by me, then between each frame I translate and rotate it. I also update the rest of the parameters in that function.

* + 1. **The motivation of the chosen approach**

I have chosen to implement this project in the specific manner for two reasons. Firstly, it was a good way to practice all the concepts learnt at the laboratory this semester (transformations, types of lights, shadows, etc.). Secondly, I am fairly new to computer graphics and this is the easiest way for me to implement this project, since it is well documented and I have access to many resources and help from my teachers.

* 1. **Graphics model**

The scene has been created in Blender. Many of the objects have been imported from the websites I will mention in the references, mainly the ones that have been provided to us in the laboratory resources. The ground of the city has been sculpted by me in Blender, using the tools demonstrated in the tutorial. This was my favourite part of the project, because I really enjoyed working with Blender and creating my scene.

* 1. **Data structures**

I did not create any new relevant data structures, as the ones provided in the code for the lab were enough to implement the functionalities of my project.

* 1. **Class hierarchy**

Like I mentioned above, the classes I used for this project are the ones provided on Moodle:

1. **Camera** – defines the cameraPosition, cameraTarget and certain functions such as move(), which are useful in the application;
2. **Mesh** – used to define the polygons which make up the objects by their indices, vertexes and textures;
3. **Model3D** – used to define the objects made up of meshes;
4. **Shader** – used to render the objects;
5. **SkyBox** – used to define the skybox, which is the textured cube that contains the whole scene.
6. **Graphical user interface presentation / user manual**

The user can directly interact with the screen by using:

1. **The mouse** – moving the mouse makes the target of the camera change
2. **The keyboard** – different keys have assigned different actions to them:
   1. **UP** – moves the camera upward;
   2. **DOWN** – moves the camera downward;
   3. **LEFT** – moves the camera to the left;
   4. **RIGHT** – moves the camera to the right;
   5. **Q, E** – rotate the model to the left or right;
   6. **J, L** – rotate the directional light sources (light cube);
   7. **W** – view the scene in wireframe mode;
   8. **F** – view the scene in fill mode;
   9. **P** – view the scene in point mode;
   10. **R** – starts the rain;
   11. **I** – starts the fog;
   12. **A** – starts an animation, the titan is rendered on the screen and the camera starts moving towards it;
   13. **B** – turns on the point light source, which is yellow, to make it more noticeable;
   14. **H** – the camera starts rotating around the city, to preview it;
   15. **S** – it stops all of the above actions.
3. **Conclusions and further developments**

Overall this has been one of my favourite projects so far in university. I really enjoyed letting my creativity flow when drawing the scene in Blender and I liked the way most of these operations performed on the objects are based on geometry.

There are many things that could be added to the project to improve it:

* Sound when the animation of the titan starts (I actually couldn’t a find a way on the Internet unfortunately);
* The birds can move their wings as they are flying in a circle above the city, to make it look more realistic;
* The city can have more objects, such as streets, maybe some street lamps (these can become spotlights);
* A better skybox to make the scene more realistic(I can make my own skybox for this purpose), etc.

1. **References**
2. [Graphic Processing, Fall 2022](https://moodle.cs.utcluj.ro/course/view.php?id=526)
3. [TurboSquid](https://www.turbosquid.com/3d-model/free/church?file_type=159%2C119)
4. [Sketchfab](https://sketchfab.com/focagamer87/collections/aot-a4f6e34b0a954437a5970c0224da0a96)
5. [Free3D](https://free3d.com/)
6. [LearnOpenGL](https://learnopengl.com/Introduction)
7. [BlenderTutorial](https://youtube.com/playlist?list=PLrgcDEgRZ_kndoWmRkAK4Y7ToJdOf-OSM)