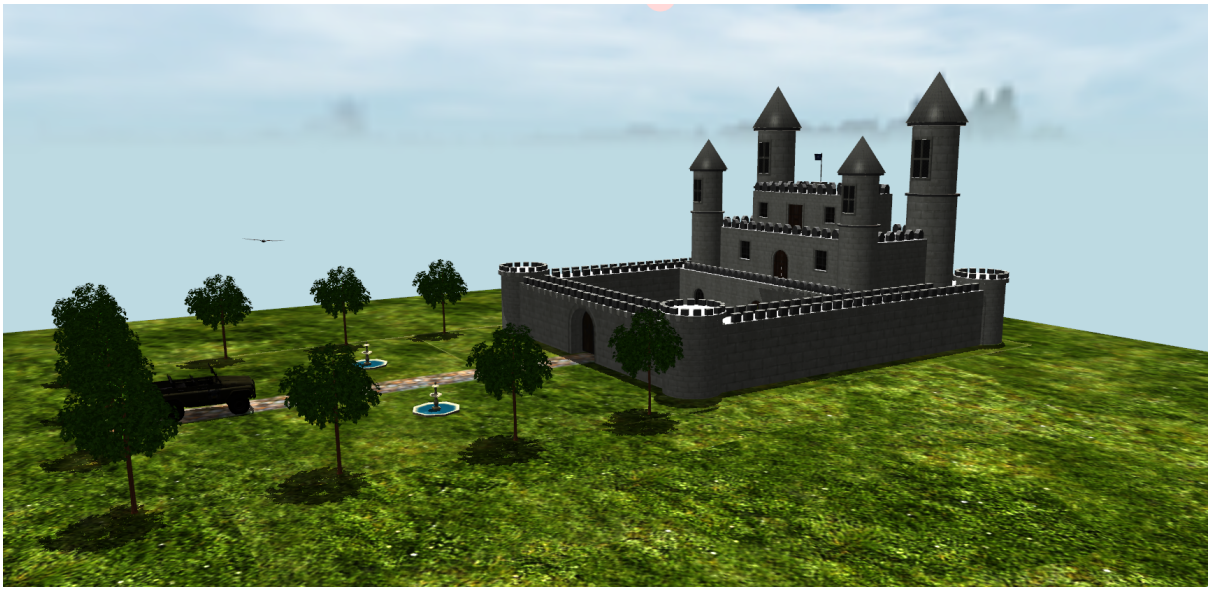
Technical University of Cluj-Napoca

Graphic processing



Castle - scene

Teacher: Professor,Eng. Adrian Sabou

Group: 30432_1
Name: Giurgiu Răzvan

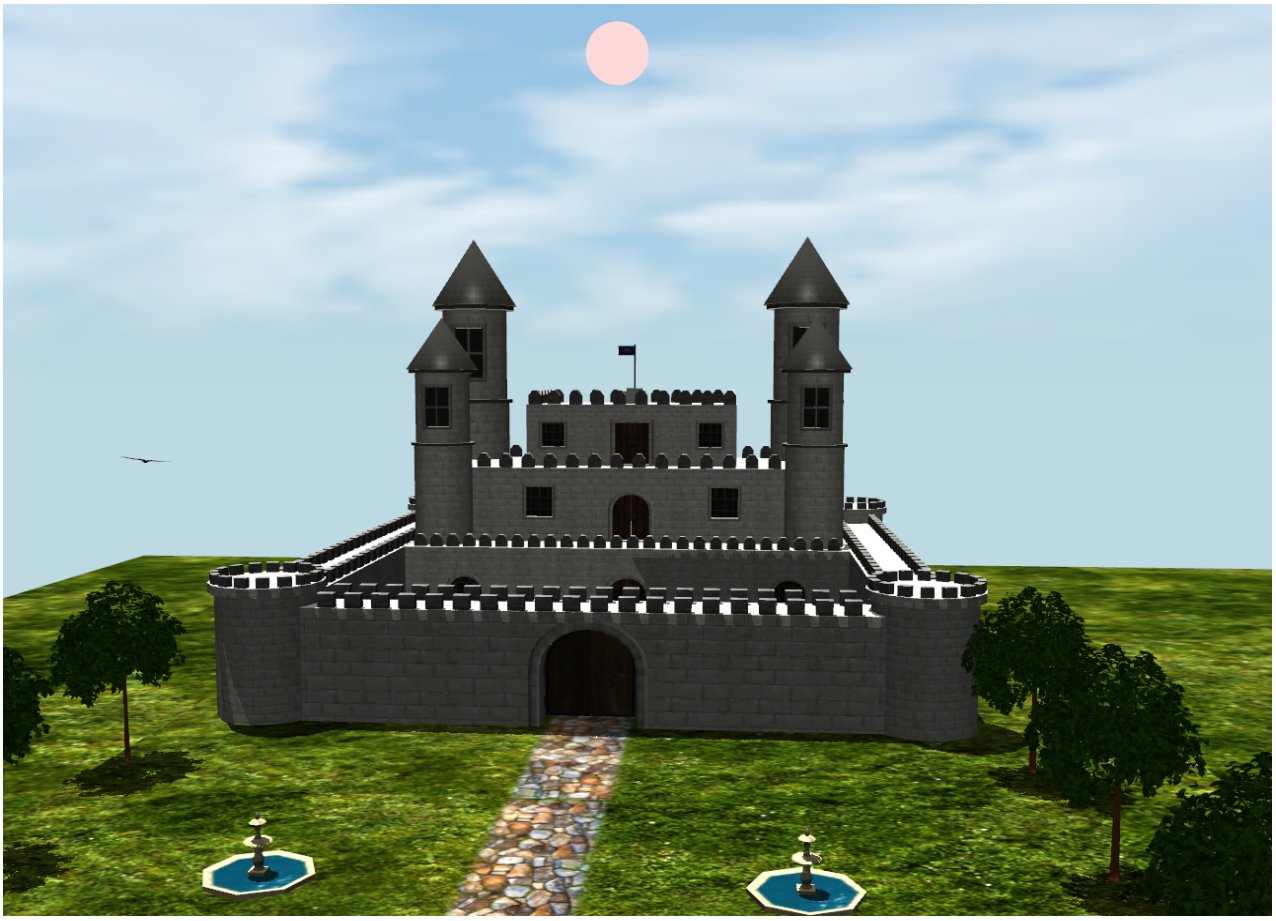
Subject Specification

The subject of the project consists in the photorealistic presentation of 3D objects using OpenGL library. The user directly manipulates by mouse and keyboard inputs the scene of objects. 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 chose to implement a scene where you can go and visit an old castle in the skies, move an old military car around the scene and watch how wind blows in some of the tree's leaves. You can choose to move the camera in any wanted direction so you can discover every element from this scene.

Scenario

Scene and objects description

This scene was built using multiple objects on which there were applied textures in order to be more realistic. First object of the scene is the “sun”, in the middle of the scene, which can be moved around to produce shadows on other objects. Main background scene uses multiple objects exported as one, the castle, a stone road, two small fountains and the grass that’s surrounding the castle.



An old military style car has been introduced which can be moved around the scene and the stone road.

In this scene I've also implemented some trees which are built from two separate objects. First object represents the trunk of the tree and the second object represents the leaves of the tree. Leaves are moving, simulating the effect of wind. Another moving object from the scene is represented by a pterodactyl that is moving around the castle.

Functionalities

Functionalities implemented are **visualization** of the scene using a camera, **multiple view frames** of the same scene, **wind and fog** effect, **multiple sources of light**, **animated and moving objects**.

- **Visualization** - When we're talking about camera/view space we're talking about all the vertex coordinates as seen from the camera's perspective as the origin of the scene: the view matrix transforms all the world coordinates into view coordinates that are relative to the camera's position and direction.

- **Multiple view frames** - Representation of the scene switching between Solid View Wireframe View and Polygonal view using keyboard keys.

- **Wind and fog effect** - We can increase the realism of a 3D scene by adding fog or wind effect which will create a particular atmosphere. By manipulating the attributes associated with the fog effect we can personalize the atmosphere and also we enhance the perception of depth.

- **Multiple sources of light** - Multiple sources of light have been implemented such as directional light and point light, that can be toggled on and off by pressing keyboard keys.

- **Animated and moving objects** - We have two moving objects in the scene that can be individually controlled using the keyboard, the sun and the military truck. An animation effect is created by the pterodactyl movement revolving around our main scene, the castle.

Implementation details

Functions and special algorithms

Possible solutions

OpenGL (Open Graphics Library) is a software interface to graphics hardware. The interface consists of over 250 different function calls which can be used to draw complex two and three-dimensional scenes from simple geometric primitives such as points, lines, and polygons. There are also routines for rendering the scenes with control over lighting, object surface properties, transparency, anti-aliasing and texture mapping.

This project was created using a core OpenGL project provided by our laboratory teacher which included to following functions :

- **GLenum glCheckError_(const char *file, int line)** - Used to detect and display in console possible errors that may occur in project's implementation.
- **void windowResizeCallback(GLFWwindow* window, int width, int height)** - Which was implemented during work for the project, useful in case the user resizes the provided OpenGL application.
- **void keyboardCallback(GLFWwindow* window, int key, int scancode, int action, int mode)** - User can close the application by pressing ESC key on the keyboard.
- **void mouseCallback(GLFWwindow* window, double xpos, double ypos)** - Application can change point of view using the mouse, to look around the scene in the wanted direction.
- **void processMovement()** - Checks the user's input from the keyboard and changes events in the ongoing application such as moving the camera, turning on point light, and moving certain objects from the scene.
- **void initModels()** - Used to load models that are going to be used in our project.
- **void render{objectname}(gps::Shader shader)** - Useful component for rendering the objects with any included shader. It is one of the main functions of the project since it is reusable and it draws the element on the screen.
- **void renderScene()** - Function used for sending all the data to our created shaders and computing all the needed values. First iteration is responsible for drawing all the shadows and the second iteration is used to display the elements on our current scene. We also use this function for similar aspects such as initializing the skybox and the "sun".
- **int main(int argc, const char* argv[])** - Predefined main that uses all our already created functions.

The motivation of the chosen approach

This scene is trying to simulate the aspect of an old castle which is surrounded by green grass and tall trees that are slightly blown by the wind. We can also interact with the current scene, not only visualize it, by controlling, on the stone road from the edge of the map until castel's gate, a military style car. Another moving object from the scene that can be observed is the pterodactyl that's revolving around the castle, in a circular motion.
I chose to implement this scene because because I liked the aspect of the castle and wanted to implement secondary objects that match this castle's style

Graphics Model

Models used in this project are free open source models, downloaded from the internet. All the models were downloaded in an .obj format along with the textures and an .mtl file which binds the object to the texture. All the models were imported in blender and modified so that they all have the same size and a relative position one to another. Objects were then parsed using OpenGL and displayed on the screen, minimum transformation was used because objects exported from blender had a relative position in the scene.

Class hierarchy

1. **Camera** - Because OpenGL does not provide a camera by default, we simulate the camera's behaviour using this class. Movement is simulated through predefined classes which were implemented during the work for this project.
2. **Mesh** - Represents a 3D object in space using it's position, normal vector and bind textures to the object.
3. **Model 3D** - Provide functions to import and parse objects and use a specified shader program to print the meshes.
4. **SkyBox** - Load SkyBox Textures and place them in the desired position, forming a cube around our scene.
5. **Windows** - Maintains Window calls such as Initiating the window in which our program is rendered and displayed on the screen, getting window dimensions and closing the main windows at the end of the execution.

User manual

Users can directly interact with this scene, using the keyboard and the mouse, in order to view the scene from multiple angles, try multiple light sources, observe shadows and move elements. The following table shows each element’s assigned key.

Assigned key	Effect
E	Used to rotate all objects from the scene to the RIGHT.
Q	Used to rotate all objects from the scene to the LEFT.
W	Change the camera's view point and move it towards the scene.
A	Change the camera's view point and move it away from the scene.
S	Change the camera's view point and move it LEFT from the scene.
D	Change the camera's view point and move it RIGHT from the scene.
J	Move the sun in a circular motion to the LEFT.
L	Move the sun in a circular motion to the RIGHT.
5	Decrease fog coefficient.
6	Increase fog coefficient.
ARROW_LEFT	Move the car to the LEFT.
ARROW_RIGHT	Move the car to the RIGHT
ARROW_UP	Move the car to the FRONT.
ARROW_DOWN	Move the car to the BACK.
3	Start pointlight.
4	Stop pointlight.
7	Toggle Line View.
8	Toggle Point View
9	Toggle Solid View

Conclusions and further developments

As hard as this program seemed at first sight, I enjoyed working in OpenGL and other tools to modify the objects or place them in our scene. This project helped me gain knowledge about graphic processing and how every element from a “videogame” is represented through mathematical formulas and transformations. This project can be developed in a lot of new ways such as expanding the map and building levels, adding more quality shadows, implementing object collision and other A.I. characters or other players that you can interact with.

References

1. https://clara.io/
2. https://free3d.com/
3. https://open3dmodel.com/
4. https://3dexport.com/
5. https://docs.blender.org/
6. https://docs.gl/
7. Labs
8. Courses