Graphic Processing

Project Documentation

Terente Ioana-Emilia
Group:30432

Computer Science

Assistant Teacher: Diana Cristina Enoiu

Content

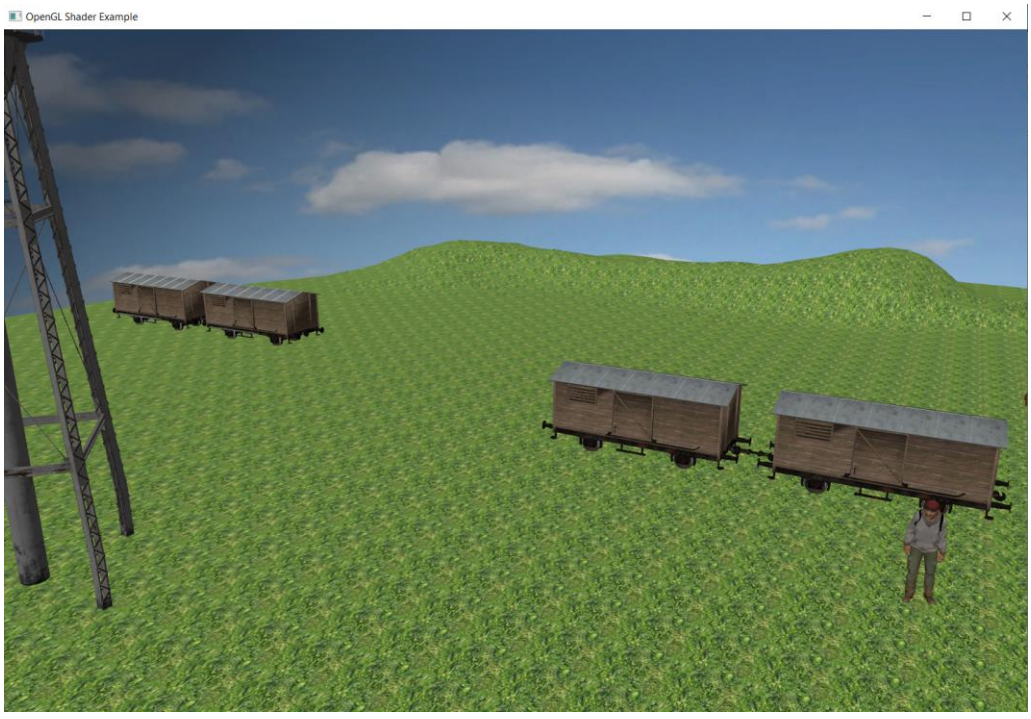
1. Subject specification

The scene depicts a rustic rural setting with a digital representation of a farm environment. The main structures include a wooden water tower, a log cabin-style farmhouse, an outhouse, and three wagons, possibly for hay or grain transportation.
Additional elements include wooden crates, a well with a crank, and a small body of water that resembles a pond.
The landscape is composed of rolling green hills with a simple texture to simulate grass and the sky is a clear blue with scattered clouds.
A humanoid figure stands near the center of the scene, providing a sense of scale.
The scene has animations which only appear when a specific key is pressed.
This is a 3D rendering window of a development environment with debugging and shader code visible on the left side, a test or a development scenario.

2. Scenario and functionalities

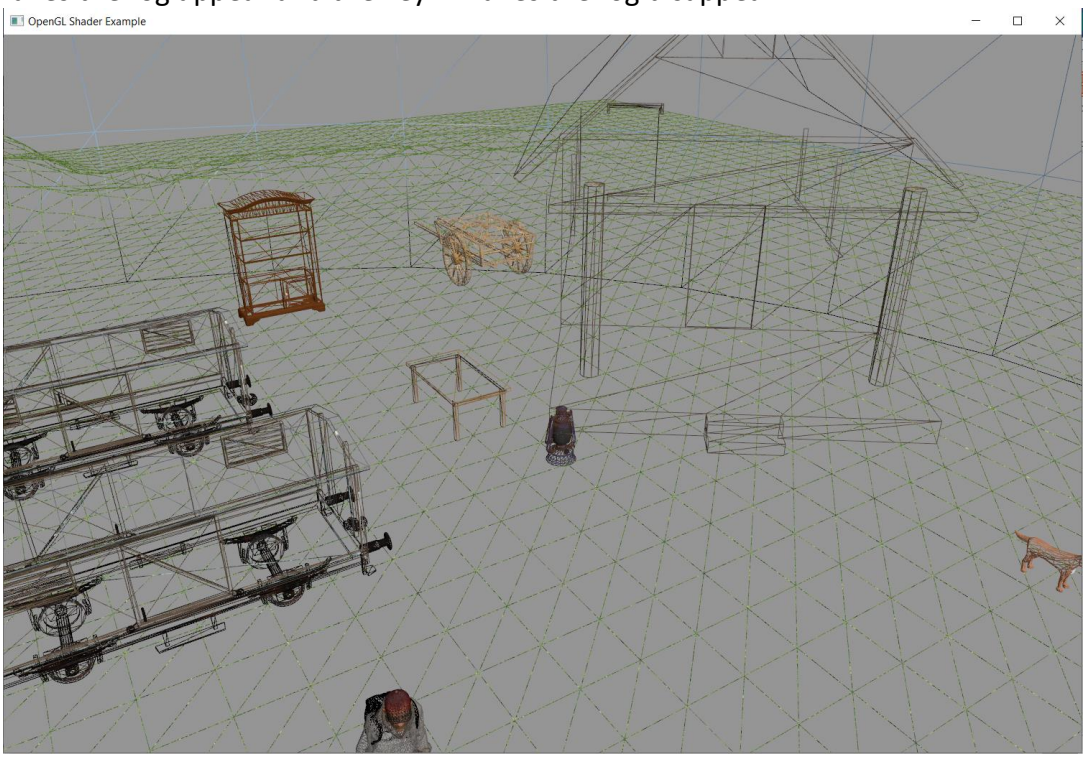
The scenario represents a stage in developing a 3D application where the user is testing the OpenGL project.
The elements in the scene are positioned to demonstrate spatial relations and the effects of lighting and texturing within the engine.
The human figure is to give a sense of immersion and interaction with the close objects.
The water source is next to the house and a boat is floating on it.
The presence of the debugging and shader code indicates that the troubleshooting, optimizing, or experimenting with graphical effects like shading and texture mapping.
This is a simplified representation aimed at test the basic functionality of an OpenGL scene like: directional light, point light, fog, moving objects or moving parts of an object.

Here you can see the pond and next to it the two picnic tables. I placed there a haystack and in from of the house it can be seen a small brown dog. On the left are four wagons. The ones on the right do not move, they are still, but the ones on the right can be moved. It can be seen the picture below how I moved the wagon by pressing a key(M) and I can move it back by pressing another key(N).



On the right side of the house a gas lamp, which lit up when key O is pressed and shuts down when key P is pressed, is found on the ground, next to it a table and in the back a dray with hay. The piece of furniture has a special movement, when the user presses key K the door opens and the key J closes the door. These movements can be seen n the picture below.

Key Q makes the fog appear and the key Z makes the fog disappear.



By pressing key E the user can see the wireFrame and key V shows the pointFrame. Key F takes them back to normal.

3. Implementation details

a. Functions and special algorithms

i. Possible solutions

OpenGL Initialization Functions:

- **initOpenGLWindow()**: Sets up GLFW window and OpenGL context options, initializes GLEW, and prints renderer and version info.
- **initOpenGLState()**: Initializes various OpenGL states like clear color, viewport, depth test, culling options, and framebuffer sRGB.

Shader and Object Loading Functions:

- **initObjects()**: Loads 3D models from files into the **house**, **vagon**, **dulap**, and **usa** objects.
- **initShaders()**: Loads vertex and fragment shaders from files and compiles them into the **myCustomShader** program.
- **initUniforms()**: Initializes uniform variables like model, view, projection matrices, light direction, light color, and light position in the shader program.

GLFW Callback Functions:

- **windowResizeCallback(GLFWwindow* window, int width, int height)**: Handles window resize events.
- **keyboardCallback(GLFWwindow* window, int key, int scancode, int action, int mode)**: Handles keyboard input events.
- **mouseCallback(GLFWwindow* window, double xPos, double yPos)**: Handles mouse movement events.

Rendering and Animation Functions:

- **processMovement()**: Processes user input for moving the camera, toggling render modes, and enabling features like fog and point light. It also handles door and wagon movement.

- **renderScene()**: Clears the screen and renders the 3D models using the custom shader program, applies transformations, and sets the uniforms for each object.

Utility Functions:

- **glCheckError_()**: Checks for OpenGL errors and prints them to the console.
- **cleanup()**: Cleans up GLFW resources and destroys the window.

Main Loop:

- **main()**: The entry point of the program that initializes the window, OpenGL state, objects, shaders, and uniforms. It contains the main application loop that processes input, renders the scene, and swaps the buffers.

Special algorithms and techniques:

- Camera Dynamics: Utilization of the `gps::Camera` class to manage camera positioning and viewing angles. User inputs are translated into camera movements, which are smoothed out using the delta time between frames, ensuring fluid motion.

- Shader-Driven Graphics: Deployment of custom shader programs facilitates the rendering process. This enables the computation of intricate lighting and visual effects directly by the GPU, enhancing the scene's realism.

- Geometric Transformations: The application of transformation matrices to the models for altering their position, rotation, and scale within the 3D space, allowing for dynamic scene composition.

- OpenGL Error Surveillance: Implementation of the `glCheckError()` macro function serves as a vigilant system to detect and report OpenGL errors, ensuring the robustness of the rendering process.

- Illumination Control: Management of lighting within the scene is achieved through a directional light that simulates sunlight, complemented by a point light whose activation can be toggled, providing nuanced lighting effects on the objects.

- Atmospheric Fog Simulation: Integration of a switchable fog mechanism that, when activated, introduces a layer of atmospheric haze, contributing depth and ambiance to the virtual environment.

ii. The chosen approach

I implemented this scene with vintage and rustic components. Implemented two types of lights, moving objects, fog and skyDome because I found this way of implementing easier.

b. Graphics model

Objects and textures have been downloaded from the internet. Most objects were imported into the Blender to be edited and placed in the final scene.

This part seemed to me quite difficult, because some textures were not displayed on the object or, moreover, the object was not displayed. Therefore, I had a longer search for objects/textures.

c. Data structures

The data structures that I have used, at least some of them were already implemented at the lab. I added just the fog, spotlight, moving objects and the scene form Blender where I placed first all the object with textures.

d. Class hierarchy

- **Camera**: contains the implementation for camera movement (up, down, front, back, left, right)
- **Mesh**: represents a 3D object; includes the following data structures:
  - Vertex (position, normal vector and texture coordinates)
  - Texture (id, type and path); Material
- **Model3D**: contains methods for printing meshes using a specified shader program
- **Shader**: contains methods for creating and activating shader programs

4. User manual

The user can interact with the scene using the following keys on the keyboard:

- Q – fog
- Z - erase fog
- W – front
- S – back
- A – left
- D – right
- E – wireframe
- V – pointView
- Y- up
- H -down
- O – lights point light
- P – turns of the point light
- N - moves wagon to the right
- M -moves wagon to the left
- K – opens the door
- J - closes the door
- F – goes back from wireFrame and pointView

5. Conclusions and further developments

The OpenGL Advanced Lighting project successfully demonstrates the application of sophisticated 3D graphics techniques, offering an interactive and visually rich environment. Through the implementation of dynamic lighting, fog effects, and user interactivity, the project not only validates the robust capabilities of OpenGL but also serves as a testament to the developer's technical proficiency and problem-solving skills. While successful, the project also presents opportunities for further enhancements and optimizations, laying a solid foundation for future development and learning in the field of computer graphics.

As further developments, they can be:

- shadow
- wind
- rain
- addition of new elements in the scene
- animation of several objects
- etc.

6. References

- https://free3d.com/
- Modern OpenGL Guide test, Alexander Overvoorde (January 2019)
- https://turbosquid.com
- https://open.gl/
- https://learnopengl.com/