Graphics Processing
Interactive OpenGL Application

Student: Elekes Péter

University: Technical University of Cluj-Napoca

Faculty: Computer Science

Year: III.

Group: 30432

Table of Contents

Subject Specification

Title: Interactive 3D Concert Application

Objective: An interactive 3D graphics application using OpenGL that allows users to view and manipulate a virtual concert environment.

Description: The project is an interactive 3D graphics application, implemented in OpenGL, which allows the user to view and manipulate a virtual environment. The virtual environment includes a number of objects and lighting sources, and the user can interact with it using keyboard and mouse controls. The application is designed to provide an immersive and interactive concert experience for the user.

Key features:

- Implemented in OpenGL
- Virtual concert environment with objects and lighting sources
- User interaction through keyboard and mouse controls
- Immersive and interactive experience

Scenario

Scene and objects description

In this interactive concert application, you are able to explore a virtual concert setting. You are able to freely move the camera around the concert stage, the speakers, a disco ball, and a variety of instruments. The stage is set up with realistic representations of instruments such as guitars, drums, and keyboards. The lighting effects change dynamically, immersing you in the concert experience. You can move around the concert stage using keyboard and mouse controls to view the stage and instruments from different angles, and you can even toggle a spotlight on the camera, and a positional light on the stage.

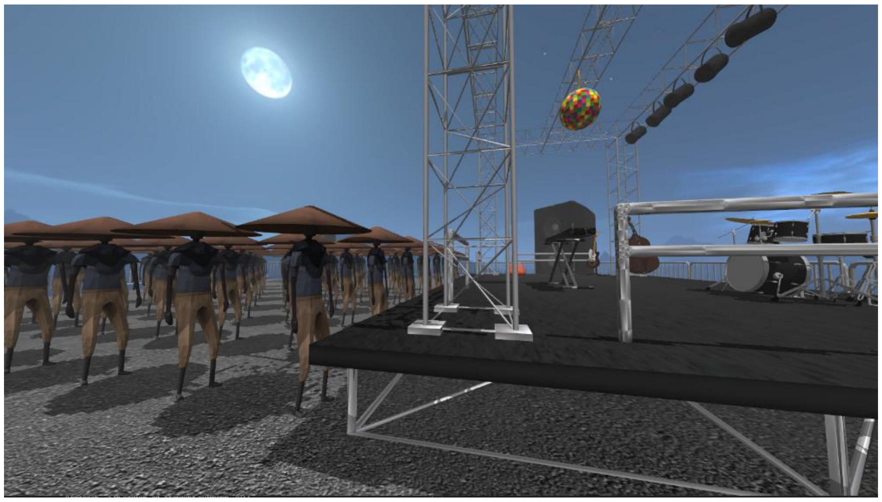

Figure 1-Screenshot of the application

Functionalities

In this interactive concert application, you have a variety of functionalities at your disposal to enhance your virtual concert experience.

- Moving around: You can move around the concert hall using the WASD keys on your keyboard. This allows you to explore the concert environment and get a better view of the stage and instruments.
- Looking around: You can use your mouse to look around and change the view angle, this allows you to see different perspectives of the concert stage.
- Viewing solid, wireframe objects, polygonal and smooth surfaces: You can toggle between viewing the objects in the concert hall in solid mode, wireframe mode, and also switch between polygonal and smooth surfaces.
- Changing shadow: You can adjust the shadow settings in the virtual concert environment, this will allow you to see the objects in different lighting conditions.
- Playing a preview animation: You can play a preview animation of a concert performance, this allows you to experience the concert as if you were watching it live.

These functionalities will allow you to fully explore and interact with the virtual concert environment, providing an immersive and realistic concert experience.

Implementation details

This interactive concert application is implemented using the Open GL graphics library. Open GL is a widely-used, cross-platform graphics API that allows for the creation of 3D graphics and interactive applications. It is designed to be portable and can be used on a variety of platforms including Windows, Linux, and MacOS.

The application is set up using Open GL's state machine model, where the state of the graphics pipeline is represented by a set of data structures. These data structures are then used to define the virtual concert environment, including the objects, lighting sources, and textures.

Open GL's rendering pipeline is used to render the virtual concert environment, which includes several stages such as the geometry stage, where the 3D objects are defined, and the rasterization stage, where the 3D objects are converted into 2D images. The pipeline also includes the fragment shading stage, where the lighting and shading effects are applied to the objects.

Figure 2-OpenGL logo

Functions and special algorithms

Possible solutions

In the application many of the OpenGL's [1] built-in functions were used to set up the environment and run the application, such as: glfwPollEvents(); glfwSwapBuffers(myWindow.getWindow()); glCheckError(); glUniformMatrix4fv(); glActiveTexture(); glBindTexture(); etc…

However, this interactive concert application uses a variety of custom functions and algorithms to handle the different elements of the virtual concert environment. [2]

- processKeys (): This function handles the user input from the keyboard.
- initOpenGLState(): This function sets up the initial state of the Open GL pipeline, including the clear color, depth test, and blending.
- initModels(): This function loads and initializes the 3D models used in the concert stage, including the objects, instruments, and stage.
- initShaders(): This function loads and compiles the shaders used to render the virtual concert environment. Shaders are small programs that run on the GPU, they are used to handle the lighting and shading effects.
- initUniforms(): This function sets up the uniform variables used in the shaders, including the lighting and shadow settings.
- initFBO(): This function sets up the Framebuffer Object (FBO) used to render the shadow depth map. The FBO is used to render the concert environment from the light's point of view and stores the depth information in a texture, which can be used to calculate the shadows in the scene.

These functions and algorithms work together to handle the different elements of the virtual concert environment, and provide the user with an immersive and interactive concert experience.

Motivation of the chosen approach

The chosen approach for this interactive concert application is to use Open GL, functions and algorithms such as processKeys(), initOpenGLState(), initModels(), initShaders(), initUniforms(), and initFBO(). These were chosen because they are examples provided in the laboratory sessions and are simple solutions for implementing the application.

[1] https://learnopengl.com/
[2] Graphics Processing Laboratories

Graphics model

The graphics model used in this interactive concert application was created using Blender, a popular 3D modeling software. The scene was set up in Blender [3]and objects were downloaded from Sketchfab.com[4], a website that provides free 3D models. The objects were then processed in Blender to make sure they were suitable for use in the application. Once the scene was set up, the objects were exported as OBJ files along with their associated materials. These files were then imported and used in the Open GL application to create the virtual concert environment.

The use of Blender and Sketchfab.com allowed for a wide variety of objects to be incorporated into the virtual concert environment, adding realism and immersion to the experience. The process of exporting the OBJ files and materials also ensured that the objects were properly formatted for use in the Open GL application. This approach allowed for a high level of flexibility and control over the virtual concert environment, resulting in a more realistic and immersive experience for the user.

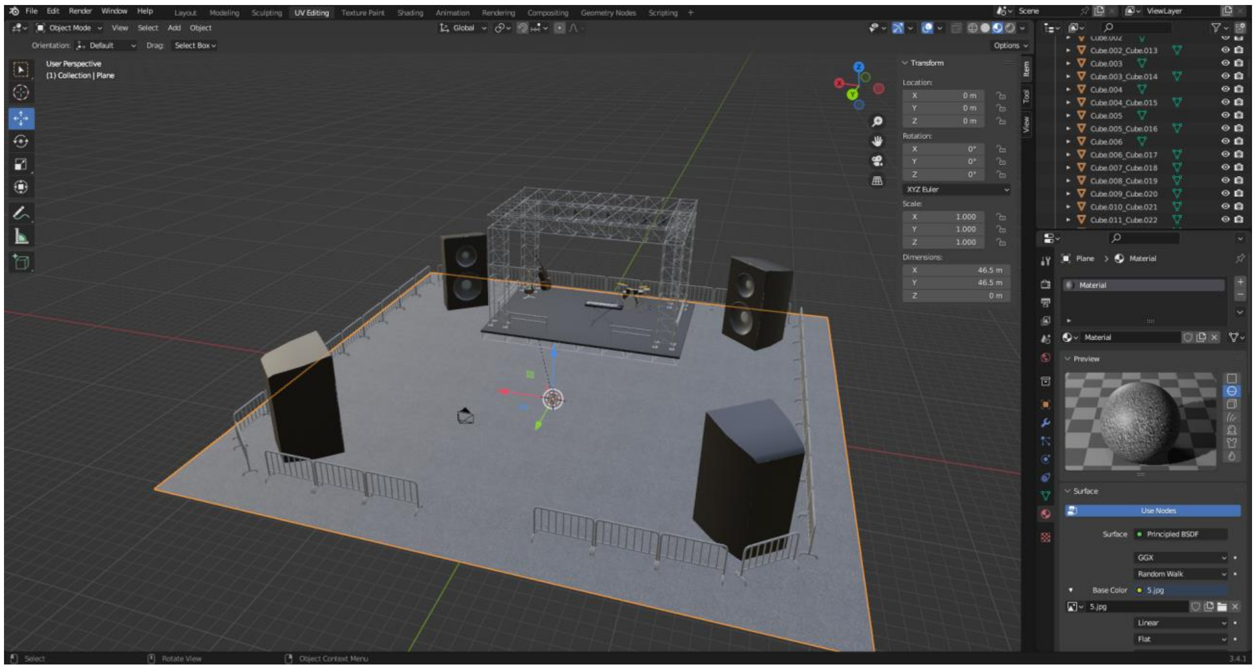
File Edit Render Window Help
Layout Modeling Sculpting UV Editing Texture Paint Shading Animation Rendering Compositing Geometry Nodes Scripting
Object Mode View Select Add Object
User Perspective
(1) Collection | Plane
Transform
Material
Preview
Surface
Use Nodes
Principled BSDF

Figure 3-The scene in Blender

[3] https://www.blender.org/support/tutorials/
[4] https://sketchfab.com/feed

Data structures

In this interactive concert application, the data structures used include:

- Matrices to store the model, projection, view, etc.
- Vectors to store the directions of the light, the color, the faces of the skybox, the different translations of the objects, etc.
- Models to store the objects
- Shaders to store the shaders
- Uniforms to store the variables that can be sent to the shaders
- and many more

Class hierarchy

In this interactive concert application, there are different classes that handle numerous aspects of the functionalities. The classes are:

- shaders: This class is responsible for loading, compiling and linking the shaders used in the application. It also stores the uniform locations and handles the passing of data to the shaders.
- camera: This class stores the position, direction and target of the camera, and also handles the camera movement and rotation.
- model3d: This class is responsible for loading, initializing and rendering 3D models in the scene. It also stores the model, view and projection matrices, as well as the model's textures and materials.
- skybox: This class is responsible for loading, initializing, and rendering the skybox. It also stores the skybox's textures and handles the passing of data to the shaders.
- window: This class is responsible for creating and managing the application window. It also handles the input events and the main loop of the application.
- main: This is the main class that controls the overall flow of the application. It creates instances of the other classes, initializes the OpenGL context, and handles the main loop.

These classes work together to create the interactive concert application, with each class responsible for a specific aspect of the application.

User Manual

You can control the camera and manipulate the virtual environment using keyboard and mouse controls.

The following are the keyboard controls for the application:

- Press "W" to move the camera forward.
- Press "S" to move the camera backward.
- Press "A" to move the camera to the left.
- Press "D" to move the camera to the right.
- Press "Q" to start the animations
- Press "E" to stop the animations
- Press "Z" to enable the spotlight[5]
- Press "X" to disable the spotlight
- Press "C" to enable the point light[6]
- Press "V" to disable the point light
- Press "F" to increase the intensity of fog
- Press "G" to decrease the intensity of fog
- Press "J" to rotate the shadows [7]clockwise
- Press "L" to rotate the shadows counter-clockwise
- Press "M" to show/hide the depth map
- Press "ENTER" to play the preview animation
- Press "BACKSPACE" to stop the preview animation
- Press "ESC" to exit the application

You can also move the camera by simply moving the mouse

In the application, you can also see the gates opening and closing, disco ball rotating, and audience jumping (even if they jump a bit too fast)

[5] https://learnopengl.com/Lighting/Light-casters and laboratories

[6] https://learnopengl.com/Lighting/Multiple-lights and laboratories

[7] https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping and laboratories

Conclusions and further developments

In conclusion, this interactive concert application project was a great opportunity to learn about programming graphics and 3D modeling using OpenGL and Blender. The project required the implementation of various algorithms for lighting and shadows, which helped to enhance the realism of the virtual environment.

Working with OpenGL and Blender allowed me to gain a deeper understanding of the concepts and techniques involved in creating 3D graphics and animation. Additionally, the use of keyboard and mouse controls to navigate the virtual environment and manipulate the objects and lighting sources provided a fun and engaging experience.

Although the project has been completed, there are still many possibilities for further development. One potential direction would be to implement object detection, which would allow the user to interact with specific objects in the scene. Another possibility would be to add a first-person perspective, where the user can experience the concert as a member of the audience.

Overall, this project has been a great learning experience and I hope that others will find it as enjoyable and educational as I did.

References

1. Dorian Gorgan, & Laboratory Assistants. (n.d.). Graphics Processing Courses and Laboratories.
2. https://www.blender.org/support/tutorials/
3. https://sketchfab.com/feed
4. https://learnopengl.com/
5. *Light Casters*. https://learnopengl.com/Lighting/Light-caster
6. *Point lights*. https://learnopengl.com/Lighting/Multiple-lights
7. *Shadows*. https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping