Final Project Report

Written by

Luiza Nogueira Costa

ID 40124771

COMP 371, Section BL, Winter 2022

Due April 5th, 2022

I certify that this submission is my original work and meets the Faculty's Expectations of Originality.

INTRODUCTION

This project falls under the category of real time rendering using OpenGL. Its objective is to render an interactive scene depicting a vampire's study under theme-appropriate lighting. The user is able to employ the first-person camera and keyboard controls to walk around the room and explore items belonging to this mysterious character. Across from this space is high-ceilinged window from which sunrays filter through, casting shadows onto the floor, as well as candles floating above to provide additional dynamic lighting befitting to the fictional setting. The user can change between lighting modes to explore how illumination can influence atmospheric storytelling by turning a lighthearted summer morning into a crimson-covered medieval castle at dusk or a dim, claustrophobic ancient study at night.

OBJECTIVES AND COMPLETION STATUS

The proposed objectives concluded for the project are as follows:

1. Set up of the OpenGL environment for the rendering of the 3D scene. Creation of a room with walls and floors, followed by the inclusion of base objects to populate the scene and their respective 2D textures.

   Status: complete.

   The creation of the base scene was done in two steps. First, individual objects such as a desk and bookshelves were obtained from different common-source assets libraries available for download online and then loaded into Blender. There they were arranged into the final room configuration by means of scaling and transformations, and textures were added manually for the assets that were missing them with UV unwrapping and baking the settings directly into texture files. The complete scene was then exported into a single .obj file that could be loaded into the OpenGL scene with assistance of the ASSIMP library [2]. A modified Blinn-Phong GLSL shader was written to draw the assets into the scene by mapping the fragment color to the diffuse and specular texture at the correct coordinates.

   Challenges found for this objective: although I had originally not intended to create the scene on Blender myself, having multiple objects that had to be scaled and arranged in proportion to one another made it the more sensible choice comparing to hardcoding it all into OpenGL. Learning a new software was, as expected, a generous curve. Another issue was that the main room model came without any embedded textures, which meant I had to manually add them in Blender before exporting the final .obj. Although I was quite pleased with the results obtained, the baking of the textures including the normal maps and scaling with respect to the texture coordinates did not go as planned due to the complexity of the UV unwrapping, and so details such as the tiles on the walls and stones on the floor were lost.

   Further improvements: I would like to improve the definition of the textures by properly baking them into an UV unwrapping of the room, as well as to include the normal maps, roughness, and other components already configured in Blender when rendering the scene in my project for added realism. Given additional time, I would also populate the scene with more objects to better illustrate the horror atmosphere, as well as implement some form of body collision to keep the camera from freely crossing these assets.

2. Implementation of a first-person camera that allows the scene to be explored using cursor and keyboard input.

   Status: complete.

   To allow for user movement within the scene, listeners for keyboard and mouse input were implemented alongside a camera class responsible for performing the correct rotation and translation operations with respect to the I/O input provided. Parameters such as yaw, pitch, sensitivity, and zoom were added to ensure that the camera moves in a visually pleasing manner. Keyboard movement is converted into the scene by multiplying the camera position by either the right or the front vectors, plus the correspondent direction and velocity multiplier, whereas the mouse movement will influence the yaw and the pitch of the camera [4]. After every input, the camera update() method is called and the camera's right, up, and front coordinate vectors are changed accordingly. To ensure that the camera remains in first-person mode, the y-coordinate is restricted to a value of 0.2f.

   Challenges found for this objective: none to note.

   Further improvements: to make the project interactive, I would ideally apply a mechanism that allows the user to view and interact with objects individually, such as inspecting a book or opening the doors to the next room.

3. Implementation of atmospheric lighting in the form of rays passing through the glass window.

   Status: complete, but may be further expanded.

   The preliminary stages of lighting included the definition of a directional light source located outside of the scene's window, whose parameters are then passed on to the Blinn-Phong shader to calculate the adequate ambient, diffuse, and specular components. Then, a skybox object was defined, which encompasses the entire scene. To simulate atmospheric lighting under different conditions, three different lighting and skybox configurations where defined – early morning, dusk, and night. By pressing the key "Q" the user may alternate between these three modes and see the lighting conditions change dynamically.

   Then, shadow-mapping [3] was applied by first rendering the scene into an orthogonal projection of the depth map as seen by the main light source. The default shader then receives this map of the shadows and calculates the degree to which each fragment is to be shaded. This shading configuration can be toggled on and off by pressing the "Z" key.

   Volumetric light scattering as a post-process [1], also known as the implementation of god-rays, is to be done in three stages [5] [6]. First, the scene is to be rendered offscreen to a framebuffer object so that it includes the target light source and the occluding objects rendered in black. The scene is then rendered normally to the framebuffer, including the use

of shaders. Lastly, a switch is to be made into the orthogonal projection so that the values from the FBO and framebuffer are then passed onto the specialized God Ray post-process pixel shader. This shader samples the light over the ray from the pixel location to the screen-space light position based on the occlusion pass, which are then scaled by the weight and decay constants to create the desired effect, after which the textures are blended. Density of rays may also be adjusted, and the final color is scaled by the exposure coefficient. Lastly, gamma correction is applied.

The volumetric light scattering can be toggled on or off using the "R" key.

Challenges found for this objective: understanding the intricacies of multi-pass rendering has been the most challenging element to this step thus far. Figuring out how to render to each framebuffer and combine all images into the correct final result was quite the puzzle at the later project stages, particularly because the use of this technique expands lighting possibilities so much. One notable issue is how the direction of the scattered rays depends on the view plane position of the light, which changes with the camera's movement. The rays also point towards the right of the scene, when ideally they would shine downwards onto the ground. These are both bugs that must be fixed in due time.

Further improvements: a more intricate shadow map that accounts for the shadows coming from each individual candle would make the scene seem more realistic. Likewise, referring the normal maps when calculating the lighting may improve the appearance of the textures and allow for a more intricate rendering of the scene.

4. Addition of further cinematic lighting through candlelight simulation.

   Status: complete, but may be further expanded.

   The simplified rendering of candlelight without the implementation of a full particle-simulation procedure was achieved by creating a point light inside each candle object. This set of lights was then passed onto the default shader as an array parameter so that each fragment may compute the influence coming from multiple lights in the scene. The distance from the fragment to the light is also calculated and used as a modifier that accounts for the fact that the closer to the candle an object is, the more intense the illumination it receives. To mimic the flickering effect natural to flames, the light's intensity is changed according to a multiplier that is a function of the system time. A simple improvement that may be done is to add a post-process bloom effect atop the candle objects, to better simulate the influence of the emitted light on the candle object itself.

   Challenges found for this objective: none to note.

   Further improvements: as previously mentioned, implementing a post-process bloom would likely improve the quality of the candlelight simulation. A more complex, though more realistic alternative is to render an already animated candle frame-by-frame and apply the lighting effects over it, or to develop a simple particle simulator that more accurately describes the state of each light in relation to time.

5. The original proposal included a goal related to the real-time rendering of blood using GLSL shaders. This was the objective dropped in response to the new project deadline, and so it would be natural that the implementation of this feature would be the next step. Employing normal mapping to render fresh blood on the floor and walls would bring this scene back to its horror origins, and particle simulation of dripping fluids would be an interesting challenge to tackle moving forward.

BIBLIOGRAPHY

[1] "Rays of light through stained glass windows - github pages." [Online]. Available: https://mdzahidh.github.io/cs148/assets/best/visuals/01-report.pdf. [Accessed: 06-Feb-2022].

[2] "Assimp," *LearnOpenGL*. [Online]. Available: https://learnopengl.com/Model-Loading/Assimp. [Accessed: 03-Apr-2022].

[3] "Shadow mapping," *LearnOpenGL*. [Online]. Available: https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping. [Accessed: 03-Apr-2022].

[4] "Camera," *LearnOpenGL*. [Online]. Available: https://learnopengl.com/Getting-started/Camera. [Accessed: 03-Apr-2022].

[5] "Chapter 13. Volumetric light scattering as a post-process," *NVIDIA Developer*. [Online]. Available: https://developer.nvidia.com/gpugems/gpugems3/part-ii-light-and-shadows/chapter-13-volumetric-light-scattering-post-process. [Accessed: 03-Apr-2022].

[6] F. Sanglard, "Fabien Sanglard's website," *Fabien Sanglard*, 25-Nov-2008. [Online]. Available: https://fabiensanglard.net/lightScattering/. [Accessed: 03-Apr-2022].

"Welcome to OpenGL," *Learn OpenGL, extensive tutorial resource for learning Modern OpenGL*. [Online]. Available: https://learnopengl.com/. [Accessed: 04-Apr-2022].

Objects and textures used:

M. StashkoFollow, "Throne room - download free 3D model by Maria Stashko (@maria_stashko) [21b28d1]," *Sketchfab*, [Online]. Available: https://sketchfab.com/3d-models/throne-room-21b28d1e278d498c820f2a4c63d5f20f. [Accessed: 03-Apr-2022].

AmatsukastFollow, "Antique desk - download free 3D model by Amatsukast (@amatsukast) [7418653]," *Sketchfab*, 01-Jan-1968. [Online]. Available: https://sketchfab.com/3d-models/antique-desk-7418653c1f53481c978bd96b6f19585e. [Accessed: 03-Apr-2022].

I, "Candle light: 3D model," *CGTrader*. [Online]. Available: https://www.cgtrader.com/free-3d-models/furniture/other/candle-light. [Accessed: 03-Apr-2022].

R, "Wingback Chair: 3D model," *CGTrader*, 07-Jul-2021. [Online]. Available: https://www.cgtrader.com/free-3d-models/furniture/chair/wingback-chair-1617561c-53c6-404c-b4bf-2028cc16f63b. [Accessed: 03-Apr-2022].

A, "4K Sky Series HDRI free: Texture," *CGTrader*, 12-Apr-2018. [Online]. Available: https://www.cgtrader.com/free-3d-models/textures/natural-textures/hdri-freebie-series. [Accessed: 03-Apr-2022].

Katsukagi, “Tiles terracotta 007,” *3D TEXTURES*, 22-Jun-2021. [Online]. Available: https://3dtextures.me/2021/06/22/tiles-terracotta-007/. [Accessed: 03-Apr-2022].

I. U. P. U. I. U. LibrarypremiumFollow, “Caroline Harrison Piano - Download free 3D model by IUPUI University Library (@iupuiul) [b88a396],” *Sketchfab*. [Online]. Available: https://sketchfab.com/3d-models/caroline-harrison-piano-b88a396e583d4da9951ed2ae73aa287c. [Accessed: 04-Apr-2022].

mohamedhussienproFollow, “Sofa 03 4K - download free 3D model by Mohamedhussien (@mohamedhussien) [7d4bddd],” *Sketchfab,*. [Online]. Available: https://sketchfab.com/3d-models/sofa-03-4k-7d4bdddb547241aa9cfaacf04186184e. [Accessed: 04-Apr-2022].

N. RecordFollow, “Persian rug - download free 3D model by Nicholas Record (@nrecord) [fe13eec],” *Sketchfab*. [Online]. Available: https://sketchfab.com/3d-models/persian-rug-fe13eecfc9b8405dab1fe76c5dd28a4c. [Accessed: 04-Apr-2022].