COMP 371

SuperHypercube Game

RTX Boys

August 20, 2021

Akhilesh Warty 40056138

Alexis Bolduc 40126092

James-Samuel Lemieux-Laing 40100470

Jay Chen 40053162

Mohamed Nejjari 40056251

Pablo Arevalo Escobar 40081955

Objective

The objective of this project was to create the SuperHypercube game, a 3D interactive game complete with visuals, audio and special effects. In the design, planning and implementation of the game, several key subjects discussed in class were explored in greater detail. Through the collaborative development process, important concepts such as input and interaction, viewing and projections, building models, shaders, lighting, color and shadows were further researched and directly implemented in the game.

The project was an effective way to learn OpenGL programming. Using an iterative process, each programming assignment built on concepts discussed in class and solidified understanding of computer graphics theory and implementation. From rendering a window to creating simple objects to complex models with lighting, shadows, textures and color, the project offered a structured roadmap to familiarizing oneself and developing competence in OpenGL as a framework.

Why it interested you

Computer graphics is an interesting subject because of the diverse possibilities of its applications. The ability to create and visualize complex objects and worlds is profoundly powerful in modeling situations, building new ideas and offering a source of interactive entertainment through the form of video games, among many other possibilities. In essence, computer graphics offer the ability to extend the human imagination in a digital form, allowing better communication of ideas and perceptions to others.

This project was an exploration of OpenGL programming and the potential of computer graphics. In particular, building a video game from scratch was a compelling process because of the growing prevalence of video games in popular culture as well as in our personal lives. Since computer graphics are fundamentally important in our enjoyment of video games, computer screens, TVs and mobile phones, seeing the process of creating models, fleshing out visually appealing objects and worlds and designing a fully functional video game was a rewarding journey. In addition, interacting with our own creation was a fulfilling and engaging process. In this way, working with computer graphics is gratifying because of the ability to visually see the work on the screen.

Risks and challenges

There were several risks going into the project that manifested themselves as challenges that were overcome. First of all, the nature of the programming assignments meant that a delayed development schedule would subsequently negatively impact the remaining assignments. For example, if there were issues surrounding the creation of the complex models and walls, there would be additional development time that would need to be allocated to the following assignment (in this example, texture mapping, lighting and shadow generation would be blocked until the models were completed). An incomplete project deliverable would result in delays in the following ones as well. As such, managing time and priorities became a solution to overcoming this challenge.

Another risk and challenge was communicating remotely due to the current global situation surrounding the COVID-19 pandemic. However, this was addressed using team

communication software in order to conveniently and effectively hold regular team meetings, discuss important decisions and issues, collaborate on the coding process and post updates regarding the status of the project.

Game description

The SuperHypercube game consists of a player trying to match the orientation of a given complex object to a wall with a predetermined hole. To make the game visually appealing and engaging, texture mapping, lighting and shadow generation were implemented. The game also includes a HUD consisting of time and score. Several special effects were added to enhance the overall gameplay experience. A starfield was added to the background to embellish the game and provide a better visual experience to the user. Blurring and bloom effects were also added for that purpose. Trees and grass, as well as wind animations, were added in the background to liven the game world. Animations were implemented for miss/hit feedback to the player. In addition, the timer and score act as important gameplay elements. The player would only have a limited amount of time to make quick decisions in order to match the orientation of objects to their respective walls, and the score would reflect the decision making of the player.

Implementing the game logic was a complex process that involved trial and error in order to achieve. Since there were distinct objects with random orientations, the game would have to be able to adapt to different generated situations and correctly verify whether any given object orientation could pass through the wall. The score would then be calculated based on these elements. Audio was added as well in order to give sound feedback to the player as a way of enhancing the gameplay experience.

How you achieved your objective

Several elements were essential to effectively achieve the aforementioned project objectives. The first part of the process was communicating and collaborating as a team. Regular meetings were held to discuss important concepts and decisions that would impact the development process. Also, the lectures served as a valuable tool to learn more about the important computer graphics principles that would be vital to the understanding of OpenGL programming. Furthermore, online resources were a critical part of grasping theoretical concepts as well as implementation details.

The iterative nature of the project also allowed for a smoother development process. Since each assignment extended on the work of the previous one, there was natural growth involved in the building and implementation of the game.

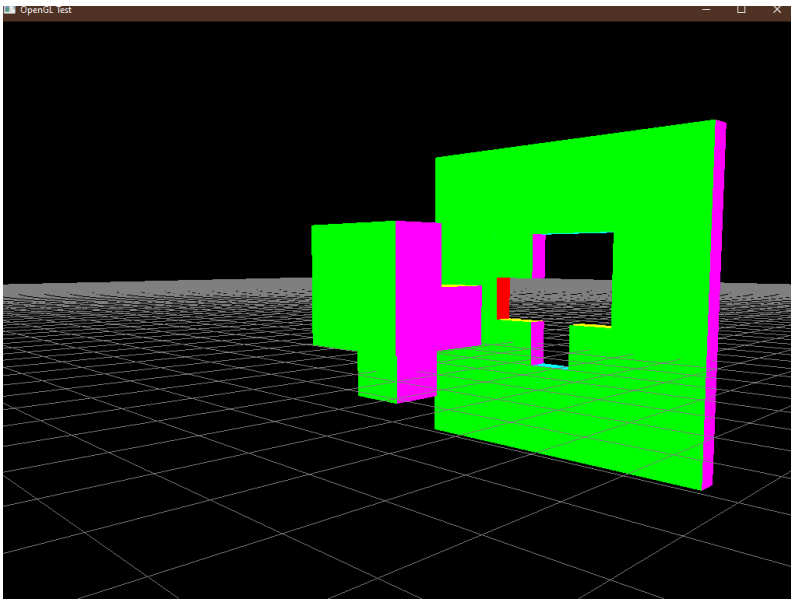

Figure 1: Building a complex model and corresponding wall

The first major milestone in the project was being able to create a 3D world using a grid and populating it with complex objects and walls. As shown in Figure 1, a complex model consisting of several cubes was implemented along with its corresponding wall (with a matching hole).

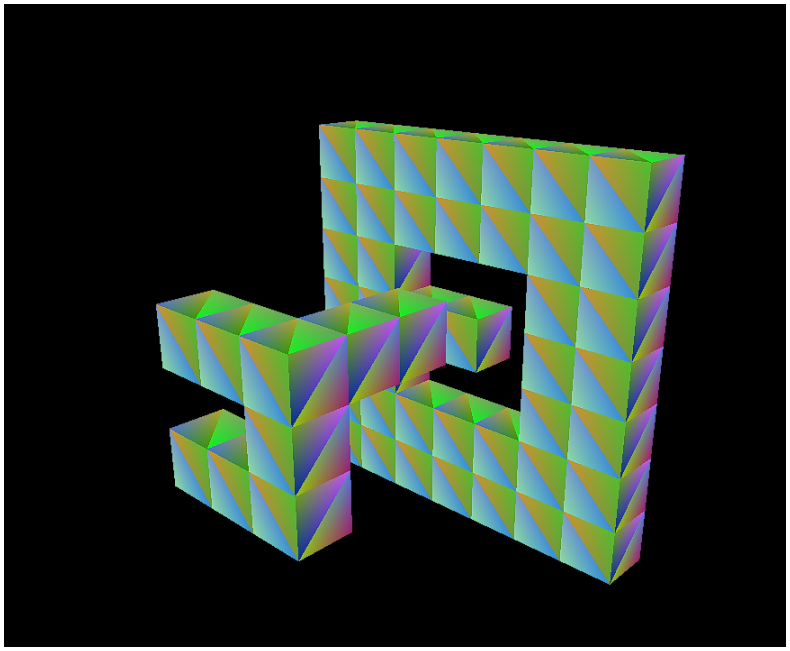

Figure 2: Experimenting with vertices and colors

After the creation of these 3D models came the tinkering of the shaders for each complex model. This created uniqueness in the different models and allowed for a thorough understanding of using shaders and creating different effects with them, which in turn helped in understanding the GLSL language better and served as a solid foundation for implementing more features down the line.

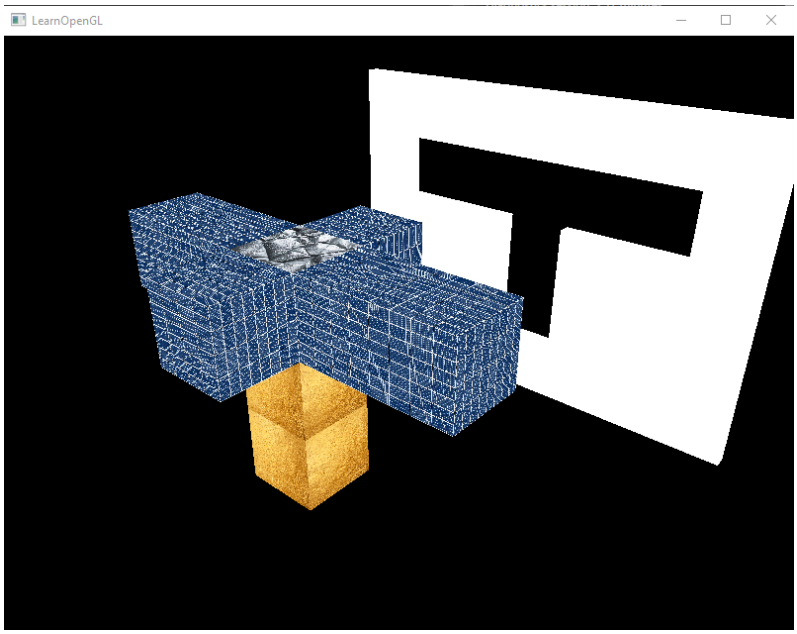

Figure 3: Experimenting with textures

The ability to experiment and visualize changes in OpenGL was an entertaining experience. It solidified core concepts and was useful in optimizing the program to meet design requirements. It also introduced the concepts of textures and how to manipulate said textures to achieve a desired effect. These textures need to be treated in their unique way in the shaders for this project so the required information could be manipulated to meet the requirements of this project.

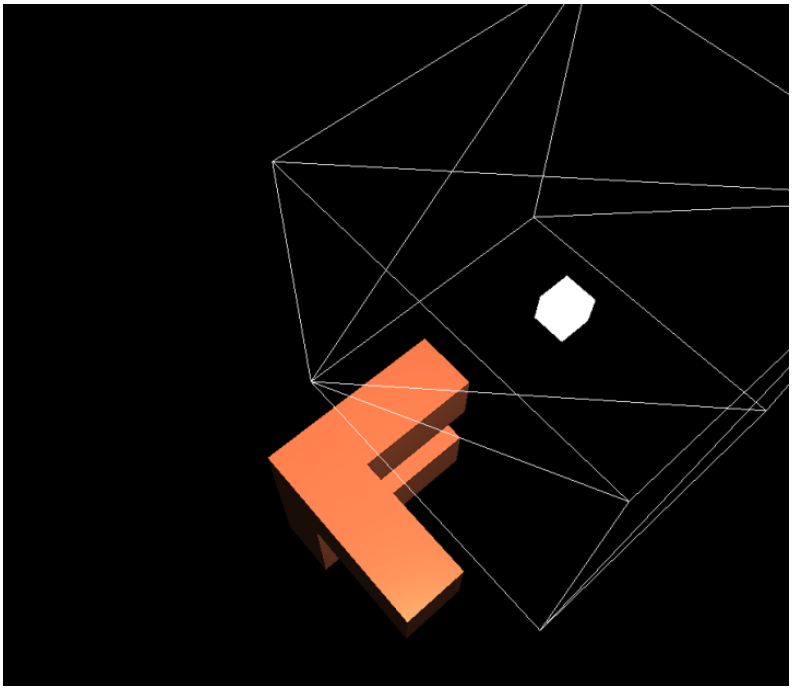

Figure 4: Creating the Phong Model

After the implementation of textures on the models, understanding how lighting worked in OpenGL was the next major criteria. Understanding the way different types of lighting worked and how to use them to create the Phong model was the main aspect of the next iteration to create a realistically lit model. Understanding how the light interacts with different materials was necessary to implement the lighting model. Creating a point light source and simulating the way it worked was the first step towards completing the goal. The specular lighting and its different interactions with various materials (metal, dirt, etc.) was quintessential in creating the Phong lighting model.

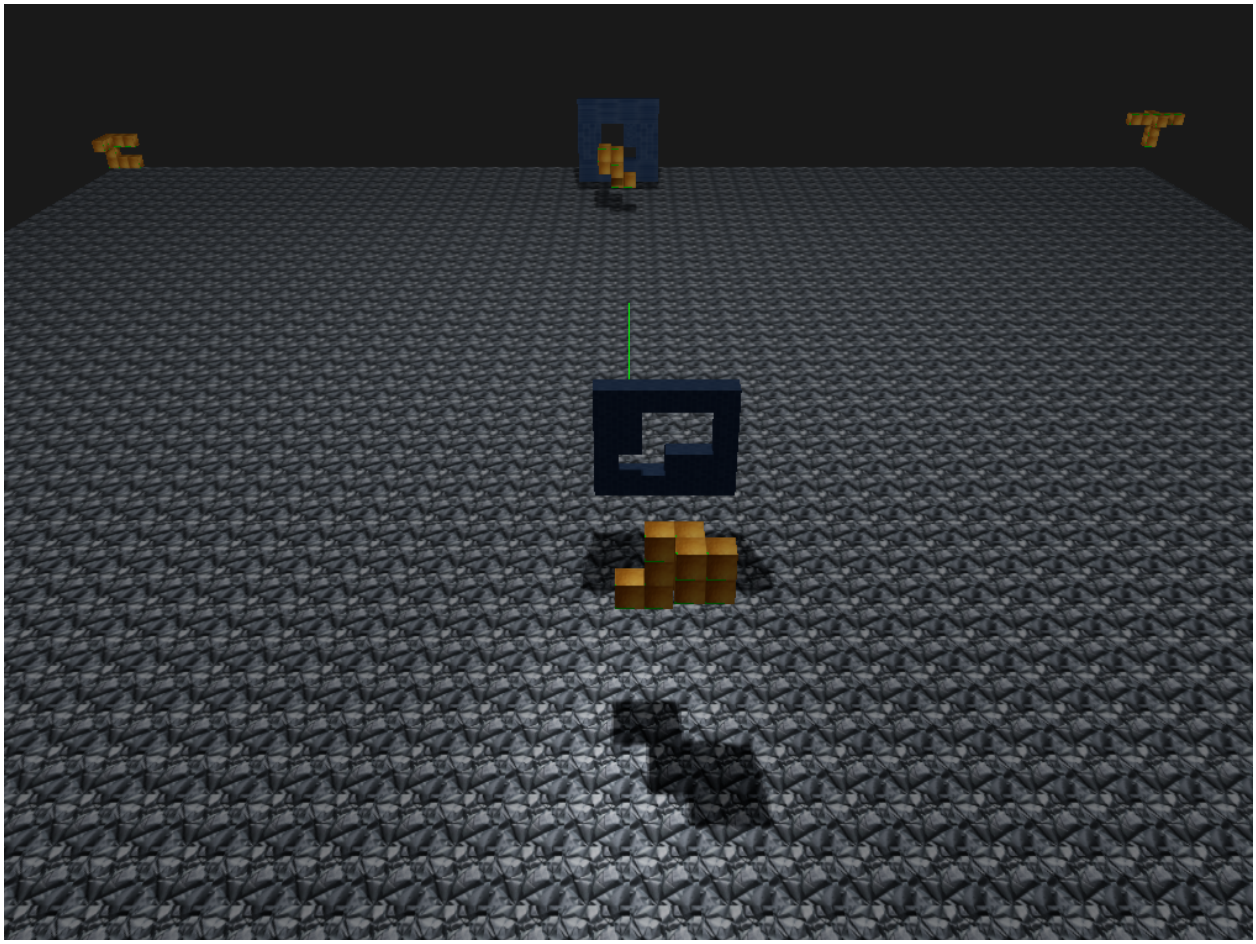

Figure 5: Implementing textures, lighting and shadows

The shadow mapping was a particularly challenging aspect of the project. The use of a depth map and the logic behind calculating shadows was important to know before it could be implemented. An orthographic projection was necessary and acted as the backbone to calculate all the shadows for this project. Online resources, lectures and labs helped tackle implementation details and gain a better understanding of the underlying theoretical concepts.

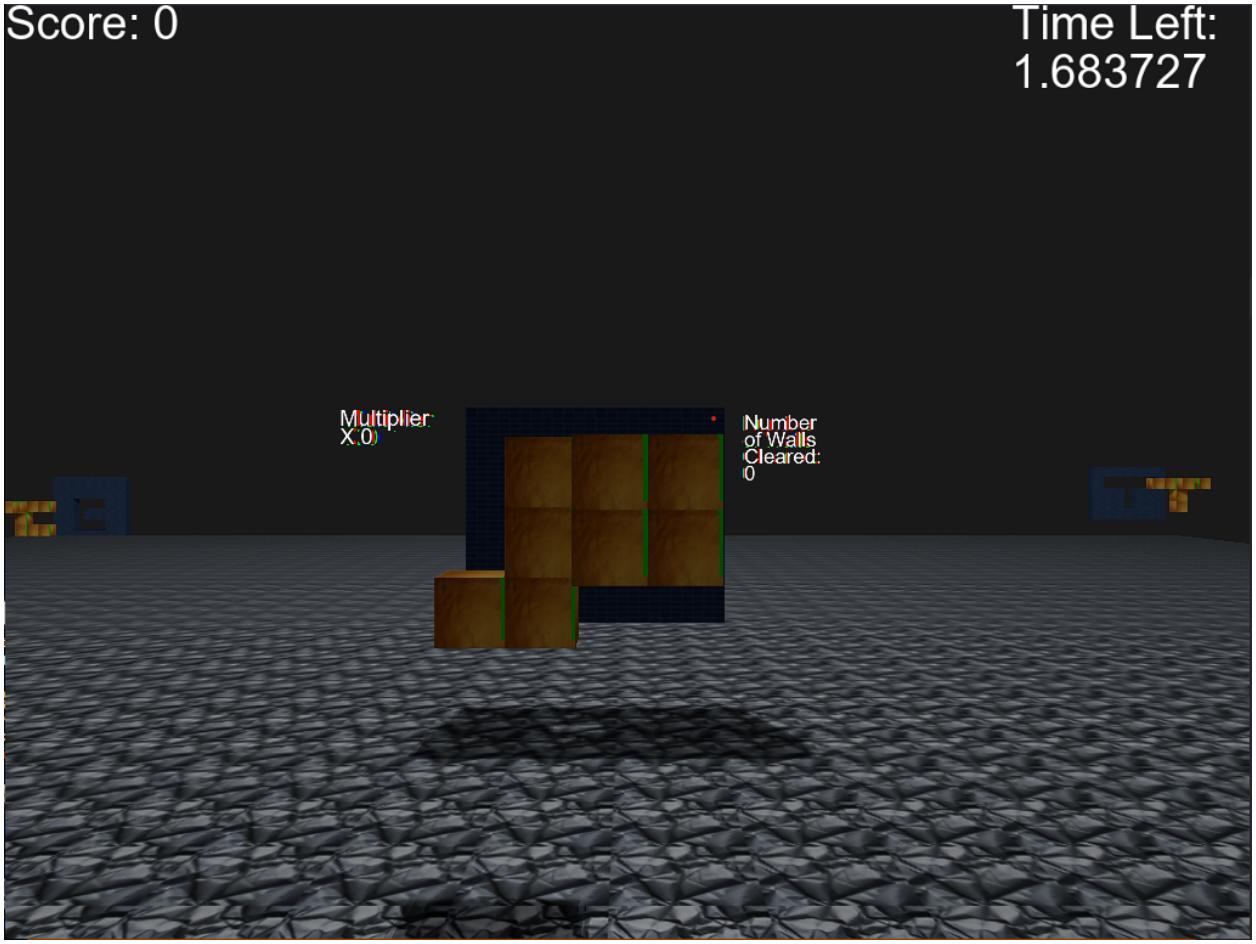

Figure 6: User Interface

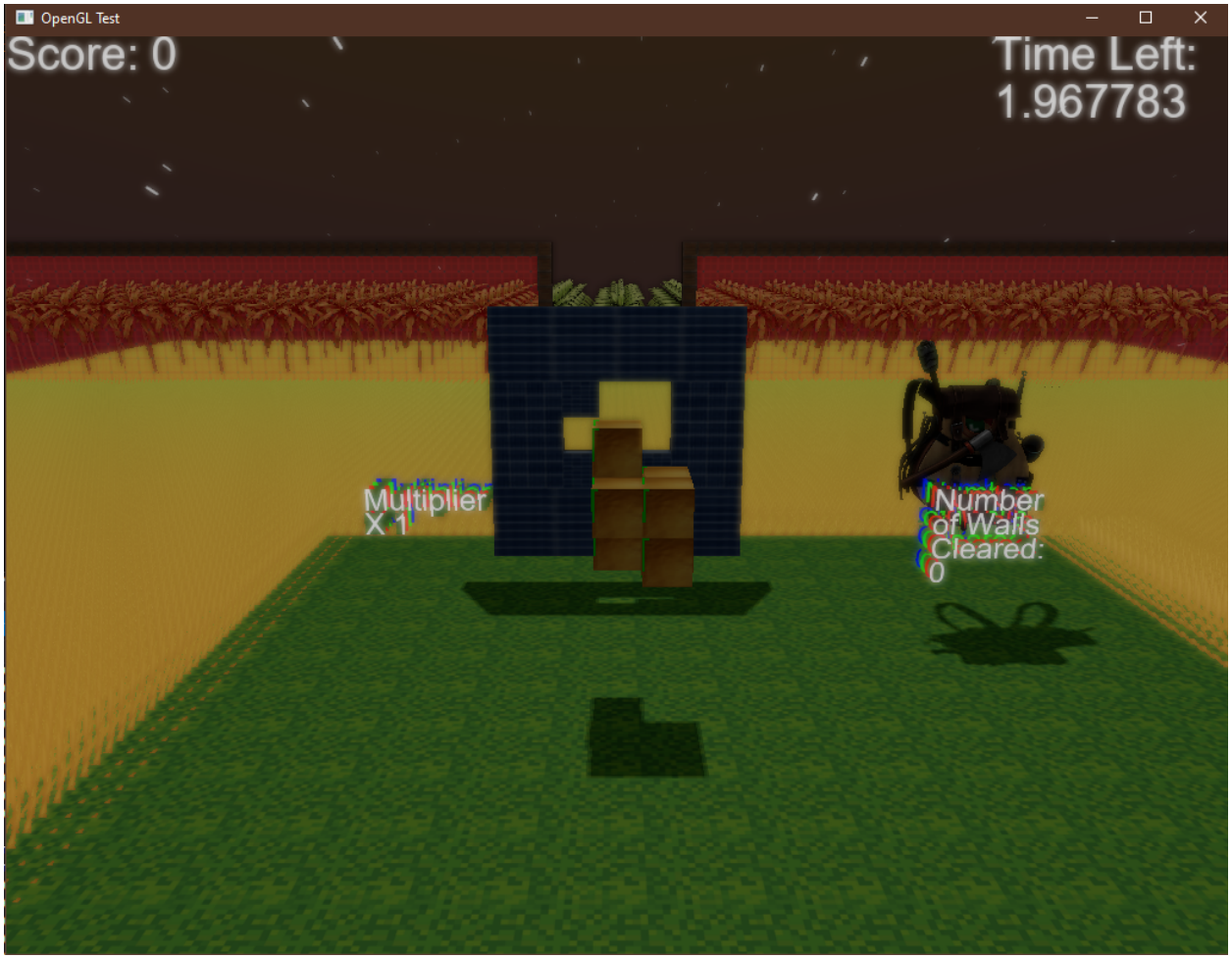

Figure 7: Final product

The combination of audio, visuals, user interface (HUD), timer, and scorekeeping came together as a simple, appealing video game. The development process was challenging but rewarding, and the final result is a success. The HUD was separated into two different parts: the overlay and the 3D elements. The overlay was created by superimposing a 2D orthographic plane on top of the already existing scene. The 3D elements were created by using the same projection and view that the entire scene was using to show the progress in the game. A video demo can be found at https://www.youtube.com/watch?v=bRVAwCjC19w.

What you learned as a result

All in all, the objectives outlined for this project were met. The creation of the game proved to be a very practical way of learning OpenGL programming, instilling the course material and demonstrating core concepts in a way that solely theory would not be able to communicate.

The project covered a variety of important computer graphics topics in a natural, iterative way: geometric objects and transformations, viewing, building models, shaders, colors, lighting, shadows, projections, curves and surfaces, and texture mapping. As mentioned, since each programming assignment iterated on the previous, there was an organic learning process that was both insightful and pragmatic.

Consulting online resources offered valuable information about implementation details. As a team, it became clear that it would be necessary to apply critical thinking in order to tackle complex development issues in a compressed time frame owing to the shortened summer session. As a result, time management and task prioritization were paramount in the success of this project, even more so than in a regular course schedule.

From creating simple 2D shapes to a complete 3D game, this project has been a team accomplishment and a lot of progress has been made on learning OpenGL programming and computer graphics principles. This has been an interesting introduction to computer graphics, and there remain many possibilities in this area to explore.

Works Cited

De Vries, Joey. “Basic Lighting.” *LearnOpenGL*, https://learnopengl.com/Lighting/Basic-Lighting.

De Vries, Joey. “Light casters.” *LearnOpenGL*, https://learnopengl.com/Lighting/Light-casters

De Vries, Joey. “Point Shadows.” *LearnOpenGL*, https://learnopengl.com/Advanced-Lighting/Shadows/Point-Shadows.

De Vries, Joey. “Shadow Mapping.” *LearnOpenGL*, https://learnopengl.com/Advanced-Lighting/Shadows/Shadow-Mapping.

point_shadows_soft.cpp. https://learnopengl.com/code_viewer_gh.php?code=src%2F5.advanced_lighting%2F3.2.2.point_shadows_soft%2Fpoint_shadows_soft.cpp, LearnOpenGL.

“Retro Future Nights - Synthwave - Royalty Free Music.” *YouTube*, uploaded by TeknoAXE’s Royalty Free Music, 28 October 2017, https://www.youtube.com/watch?v=D_jQLR6zq30.

Shadertoy. (2020). ShaderToy.Com. https://www.shadertoy.com/view/MdlXWr

shadow_mapping.cpp. https://learnopengl.com/code_viewer_gh.php?code=src/5.advanced_lighting/3.1.3.shadow_mapping/shadow_mapping.cpp, LearnOpenGL.

shadow_mapping_depth.fs. https://learnopengl.com/code_viewer_gh.php?code=src/5.advanced_lighting/3.1.3.shadow_mapping/3.1.3.shadow_mapping_depth.fs, LearnOpenGL.

shadow_mapping_depth.vs.
https://learnopengl.com/code_viewer_gh.php?code=src/5.advanced_lighting/3.1.3.shadow_mapping/3.1.3.shadow_mapping_depth.vs, LearnOpenGL.

“SuperHypercube Game by RTX Boys.” *YouTube*, uploaded by Alexis Bolduc, 20 August 2021, https://www.youtube.com/watch?v=bRVAwCjC19w

text_rendering.cpp.
https://learnopengl.com/code_viewer_gh.php?code=src/7.in_practice/2.text_rendering/text_rendering.cpp, LearnOpenGL.