

Computer Games Development

Project Report

Year IV

[Daniel Carroll]

[C00272187]

[11/05/2025]

[Declaration form to be attached]

**Contents**

[Acknowledgements 2](#_Toc54714373)

[Project Abstract 2](#_Toc54714374)

[Project Introduction and/or Research Question 2](#_Toc54714375)

[Literature Review 2](#_Toc54714376)

[Evaluation and Discussion 3](#_Toc54714377)

[Conclusions 3](#_Toc54714378)

[References 4](#_Toc54714379)

[Appendices 4](#_Toc54714380)

# Acknowledgements

I would like to thank the following people / organisations who assisted in completing this project including;

Id Software for the inspiration of this projects setting, playstyle and realesing their old assets freely to the public.

Realm667 who supplies the free art assets from old Doom games in modern file formats and in an easily downloadable manner.

I would also like to thank the creator and maintainers of raylib used heavily to bring this game to life and for their form discussions on features and bugs found in the library.

# Project Abstract

I focused on tackling the task of crafting an enjoyable retro-modern style gaming experience. Specifically, creating a 3D high speed first-person shooter (FPS) game from the ground up without relying on any preexisting game engine. The decision was made to utilize a graphics library to handle the low level rendering, Raylib, in conjunction with C++, to ensure maximum adaptability and artistic freedom while still getting the experience to manage the lower level nature of creating a game with no engine. This undertaking yielded valuable insights into 3D programming, the importance of abstracting code and ensuring its manageability and readability. Also giving me the opportunity to navigate the complexities of the software development process from inception to finalization gave me insights and valuable lessons regarding the combination of various game elements, either enhancing each another or resulting in a mismatch in the gameplay and an incohesive feeling.

# Project Introduction and/or Research Question

**Background:**  
The game Soul Chain draws heavy inspiration from the classic FPS genre, specifically games like the original Doom, noted for their impactful gameplay and immersive experiences. In Soul Chain, players battle through waves of enemies attempting a demonic invasion. The game merges nostalgic retro elements with modern gameplay features, making it appealing and intuitive for modern audiences while preserving the style of its predecessors.

**Significance:**  
I have loved game development since I was a child, and there was a story that always stuck with me. John Carmac of id Software showing his colleague the first prototype of his raycasting 3D graphics engine. After picking up a key to unlock a door a big monster appeared and they literally fell out of their seat. It is important to note that this was being rendered at what is now a laughably low resolution and this monster was a static pixel image, yet the impact was imesurable and the gaming landscape was about to change forever.

id Software had just established what a modern game is, specifically 3D games. This significantly influenced the trajectory of gaming as a whole driving technical innovation forward years into the future. This project revisits these roots, attempting to recreate and modernize the intense reactions experienced by early players an era when even low resolution graphics profoundly immersed players.

**Objectives:**

* Develop a fun and engaging 3D FPS game, by focusing on fast paced gameplay and dire consequences for mistakes adding challenge and excitement yet maintaing fairness and preserving what gaming is all about, fun.
* Create proprietary game development tools, including a custom particle system and level editor to allow the game to expand faster than it could ever be programmed by code alone.
* Allow for extensive customization options for both developers and players using these systems, nearly infinite different custom maps should be creatable and playable while maintaining all core game functionality.

**Project Focus Areas:**

* Artificial Intelligence (AI) and advanced 3D A\* pathfinding algorithms.
* Tools and customization features enabling developers and users to design and edit their own game maps and assets.
* Advanced difficulty scaling and diverse, unique enemy behaviors.
* Comprehensive game physics and detailed texture mapping techniques.

# 

# Literature Review

The primary technical foundation of the project involved the following tools and technologies:

* **Raylib:** Chosen for its flexibility and capability in rendering 3D graphics. Initially this was chosen just to handle the low level graphics rendering, later more features such as its built in bounding boxes and raycasting features were also used to help speed up development.
* **C++:** Selected for its powerful capabilities, performance advantages, and vast opensource libraries, allowing unrestricted development and optimizations.
* **GitHub:** Used for version control, project management, and documentation of the development process.
* **Photopea:** A free on the web Photoshop alternative. Utilized extensively for creating custom game textures, user interface elements, and graphical assets.
* **Audacity:** A free audio editor used to split up and create audio files for the game sounds increasing overall imersion.

Though Raylib’s documentation and syntax examples provided an initial reference and aided in my understanding of how the rendering livrary functioned, the creative and technical design process was largely independent, with minimal direct influence from existing literature.

Give the reader a good overview of the key concepts;

* Describe the most relevant work (in your own words) that other people have done in this area;
* Use proper academic writing with references.
* Show how the existing work influenced your project.

# Evaluation and Discussion

The primary technical foundation of the project involved the following tools and technologies:

* **Raylib:** Chosen for its flexibility and capability in rendering 3D graphics. Although Raylib provides robust examples and syntax resources, all elements including textures, tiles, enemy models, weapon interfaces, and particle systems were custom-developed from the ground up.
* **C++:** Selected for its powerful capabilities, performance advantages, and breadth of libraries, allowing unrestricted development and optimization.
* **GitHub:** Used for effective version control, project management, and documentation of the development process.
* **Photoshop:** Utilized extensively for creating custom game textures, user interface elements, and graphical assets.

Though Raylib's documentation and syntax examples provided an initial reference, the creative and technical design process was largely independent, with minimal direct influence from existing literature.

**Project Milestones**

**Inception:**

I started this project with project1, a complete copy of Raylibs 3D rendering example, very little was changed as I focused on understanding the syntax and tools this library had to offer to see if it was a good fit for my use case.

**Confirmation:**  
Once I decided that Raylib was for me I began tooling around with it increasing my understanding by altering game elements more severaly to help me visualize what my project would look like in the end.

**Industry day:**

By industry day key design choices were implemented but it was lacking… a game and was only sudo 3D and was really only a 2D environment posing as 3D. My next focus was to fix that.

**Final weeks:**  
With only a few weeks left I finished the 3D level editor and added player physics to create a real 3D environment, my enemy pathfinding was initially designed for 2D with 3D in mind down the line so its adaptation was simple with of course some problems incountered but ultimately resolved.

**Final hours:**

The finalisation of adding new enemies and their unique abilities, spawning of enemies into the game world based on a difficulty setting finally implemented to give the game more game!

Clearing of unused assets for the repo and completion of reports and documentation.

Creating an example map for the final draft of the demonstration video.

**Major Technical Achievements**

1. **Custom 3D Level Editor**: Developed a proprietary 3D level editor from scratch, significantly enhancing level creation and providing the flexibility for nearly limitless custom maps.
2. **Advanced AI and Pathfinding**: Implemented an advanced 3D A\* pathfinding algorithm, ensuring intelligent enemy navigation across complex terrains and any range of crazy custom maps. Running asynchronously to maintain smooth fluid frame updates and so it introduces no stuttering even with a large gameworld and many enemies.
3. **Realistic Entity Physics**: Successfully implemented physics for player and enemy interactions with eachother and the world, enabling realistic movement, jumping, and collision detection within a fully 3D environment.
4. **Custom Particle System**: Created an optimized particle system capable of generating various visual effects dynamically, significantly speeding up the development of custom assets used to enhancing gameplay immersion and aesthetics.
5. **Dynamic Difficulty and Enemy Behavior**: Developed sophisticated enemy spawning mechanics and behaviors that are dynamically adjustable based on selected difficulty levels getting increasingly difficult over time while maintaing a certain level of randomness withing the given ranges, greatly improving replayability and challenge.
6. **Decal Projector**: Created a decal system that projects images onto world elements such as bullet holes on walls, doors and stairs, creating dynamic changing environments based on the players actions.
7. **Systems Abstraction**: Everything built custom from the ground up such as the Level editor, Particle system, difficulty, enemy spawner and behaviours down to every last bullet hole, shooting and amount of damage taken is abstracted in a manner allowing to quickly and completely alter or overhaul practically any gameplay feature to match any desired gameplay style which greatly sped up development and allowed testing and tweeking of gameplay features where it otherwise wouldn’t of been possible.
8. **Custom Shader** : Created a custom transparancy shader as Raylib supprisingly does not support correctly culling and layering objects, such as billboards in my case, based on distance and visability to the camera in the correct order.

**Project Review**

**What went right:**

* Successfully delivered a fully functional 3D FPS game without relying on any existing game engine.
* Achieved significant technical milestones, such as the proprietary level editor and advanced pathfinding, considerably expanding future development possibilities.
* Efficient integration of player physics and realistic enemy behaviors greatly enhanced gameplay realism and enjoyment.
* Shaders have always been the boogyman of games programming for me, this project out of necessity I was forced to develop my own which turned out to be simple yet incredibly effective to create and to enhance the visuals of the game.

**What went wrong:**

* Initial approach relied heavily on pseudo 3D methods, requiring significant redevelopment in later stages to achieve full 3D capabilities. This took away a lot of the final remaining time as we approached the development deadline.
* Early versions lacked complete gameplay mechanics, causing delays and rushed features closer to project deadlines such as industry day and final submission.
* Pickups were dropped, they were the final feature I wished to add, including health, ammo, speed boosts, armour, special limited use minigun, special ammo types and even the SOULS the theme of the game. The underestimated time to convert the 2D pathfinding to 3D meant this feature had to be dropped. I had a prototype working and it was obvious it enhanced gameplay greatly to the point where after this project is submitted and demoed I will return to add it as its that great of a loss in my eyes.
* I wish there were audio assets in the game, I have a big folder of all the enemy sounds that theres no time left to implement, it makes it really easy to get snuck up on and quickly lose a lot of health that due to no pickups is unrecoverable.
* 3D models were created and even imported into the game and used but the technicall challenge of getting them to animate was too great and close deadlines lead to me scrapping it in favor of billboards.

**Outstanding Issues:**

* More New enemy vairents created from my spawner hadn’t enough time to be tested. Smaller enemies in conjunction with pathfinding and world bounding boxes means when a varient is created quite small it will struggle to climb stairs, they used to struggle with all gravity but that was fixed. They remain to struggle with stair pathfinding as it was only discovered in final testing hours before the deadline.
* I added a death screen an hour from the deadline that returned you to the main menu but it broke and I couldn’t find a solution in time.

**Approach for Future Projects:**

* Prioritize establishing fully functional 3D prototypes earlier in the project to prevent late stage redevelopments of entire core systems.
* Make the game fun to play as early as possible, the focus on technologies over gameplay for so long left certain features that make up core gameplay loops out such as the pickups. An example of this is the particle system, its great and incredibly powerful, but particles are used twice in the entire game, enemy blood splatter and bullet shell casing, while taking a long time to develop. If instead the particle system was simple, not abstracted or easy to use but still worked and there was an entire pickup system, the gameplay would be a lot more indept and fun.

**Advice for Future Students:**

* Begin with a clear and manageable project scope, focus on gameplay and only develop technologies as they are needed and only if they are needed.
* Invest time early on in learning your chosen tools and technologies to see if they are right for you.
* Start as soon as possible and get a working prototype going!

**Technology Choices Review:**

* Overall, Raylib and C++ proved highly suitable, providing necessary flexibility and power. Photopea and Audacity were also more than capable of handling my workflow.

# Conclusions

This project effectively demonstrated the feasibility and challenges of developing a sophisticated 3D FPS from the ground up without relying on a preexisting game engine. Significant insights were gained into game design, AI implementation, real time physics integration, and the critical importance of scalable tools like level editors. Overall, Soul Chain achieved its goal of recreating and modernizing classic FPS gameplay, balancing nostalgic appeal with contemporary innovations.

# References

* Raylib official documentation: <https://www.raylib.com>
* Realm667 asset repository: <https://www.realm667.com>
* Photopea: <https://www.photopea.com>

# Appendices

[DanCarrollSp/Project-G4-SEM-1](https://github.com/DanCarrollSp/Project-G4-SEM-1)