

Computer Games Development

Project Report

Year IV

[Student Name]

[Student Number]

[Date of Submission]

[Declaration form to be attached]

**Contents**

// problems solutions, issues unsolved even

[Acknowledgements 2](#_Toc54714373)

[Project Abstract 2](#_Toc54714374)

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

// reference to algorithms

[Literature Review 2](#_Toc54714376)

[Evaluation and Discussion 3](#_Toc54714377)

[Conclusions 3](#_Toc54714378)

[References 4](#_Toc54714379)

[Appendices 4](#_Toc54714380)

# 

# Introduction

For this report will outline the development of my final year project. I will begin my discussing my initial direction I went with my project and why I ended up changing it and where I ended up going with it instead, what aspects of this new direction I wanted to pursue and why, and then with this e prospects where I actually ended up

# Project Introduction

I had my mind set on doing some sort of survival game that would include various passive systems that happened around you such as weather, resource growth like trees and crops and entities that lived in the world that went about their day to day. This grabbed my interest with the opportunity to design systems for entities that would have a cycle and a schedule. Like different types of animals living in the world either hunting or grazing with their own schedules and behaviours. Similar with resources like food and building resources renewing after a time and the balancing of that.

A collage of images

AI-generated content may be incorrect.

My original inspiration being from survival game which I enjoyed playing my self along with other games which had systems I though were cool. Games With player driven building took my focus as I liked the idea of creating something that gave the user the ability to create something that effected the game. For graphics library I wanted to use Raylib, I thought it would be interesting to do some work in 3D using a library instead of an engine.

After doing some work on my project in the couple of weeks and getting some simple systems running in Raylib. I found after working on the project for some time and thinking more about my idea that I found that I with my initial idea that I planned on making game features that I like playing in my own time but not one that I found very interesting to make. Having come to that conclusion I went back to and rethought what I wanted to from my project.

Upon rethinking my project idea I decided to go down a different direction with my project, I went back on Raylib and decided to use SFML as I thought if I wanted to do anything complicated, I wanted to spend my time solving the technical problem of the feature and not my unfamiliarity with the library I was using.

With this change of library, I also to change my goals to some degree. I still wanted to make something that was modifiable by the user like base building but not focus on the sort of NPCS systems and scheduled systems as I mentioned before, instead I wanted to do something procedurally generated as this is not something I had any experience with a thought would be a very interesting.

Upon deciding on player modifiable structures and procedural generation being my focus I decided it would be space themed, and the player would build their own ship and space station and would be enemy space statins that would be procedurally generated. But how would I go about creating a generated space station in c++ using SFML in a way that supports NPC navigation and gameplay interactions?

# Literature review

My main point of research for this project was the dungeon generation as this was probably the most unfamiliar aspect of the project I delved into. While other parts I had experience in before were still difficult to implement and took time I was able to rely on my own prior experience with these topics to get them implemented

In researching procedural generation I found a lot of reference to what people where calling the tiny keep algorithm which was a dungeon generation algorithm implemented for a game called TinyKepp upon seeing it some steps to implementation it at (*Procedural Dungeon Generation Algorithm*, Game Developer) I found myself interested in pursing this method as it had a diverse range of techniques used in stages to create the final product of a generated dungeon. This involved a Delauney triangulation and a minimum spanning tree which I had no experience with.

What attracted me to this method was the how the randomness was created by the initial spawning of rooms in radius and then applying a separation force to them until they were all separated, following then a culling of rooms of a particular size which would leave you with a unique structure each time.

In this description of the generation a game engine was used but and many different libraries to perform the Delauney triangulation and minimum spanning tree. These I would myself in C++ as it gave me the opportunity to learn even more outside my comfort zone instead of delegating it to a library.

I did some research into the concepts of Delauney triangulations at (*Introduction to Delaunay Triangulation*, Tinfour Documentation) to better understand the concept of what it was I was going to implement, but also what its use cases were outside of mine, which I found very interesting to find that it was used for creating meshes of non-overlapping triangle from a set of points which get commonly used in terrain genaration, 3D modelling and mesh generation.

The minimum spanning I also research at (GeeksforGeeks, *What is Minimum Spanning Tree (MST)?*,)which provided some good explanations and visualisation on how the algorithm worked along with some variations on it. I ended up going with Prims which is a greedy search algorithm. This research helped me understand the algorithm and its purpose for my project on cutting down the edges created by the triangulation so that we had naviagatbale hallways that weren’t excessive cluttered with connections between rooms.

These sources and techniques helped guide the design of my procedural generation but also helped improved my own skills in implementing pure ideas and not relying on working from code examples as having to implement these features from understanding the ideas really improved my own skills as a programmer.

# Evaluation and Discussion

## Procedural Generation

My procedural generation of the space station was very successful. The algorithm consistently produced varied room layouts with each room properly spaced and connected with through a logical network of hallways. When creating the perfect spacing though I had to make some changes as I found with the default algorithm it often placed rooms quite close together, so I adjusted the collider of each room to be twice the size, so it gave it this buffering room which solved my problem and made for better layouts. The use of the Delauney triangle and minimum spanning tree made sure all rooms are reachable.

A key benefit of how I designed the dungeon generator is its flexibility. Through some adjustable parameters, such as the number of rooms you want spawned, their max/ min room size and hallway dimension make the system adaptable and easy to fine tune.

Most importantly the resulting dungeon worked well with other existing system I made for the project like the NPCS pathfinding which worked seamlessly due to it being built with my already implemented grids.

One are I would like to improve would be its visual diversity and general tuning of visuals, so they were in general better and also rooms where more diverse than they are. Introducing a simple decoration system would greatly improve the feel of the generated dungeon.

The speed of the generation varies dependent on the number of rooms you spawn but in general it quick enough that it can be done at runtime but would require a small wait behind a loading screen. The final generated grid takes up some memory but due to the optimisation of only rendering what the camera seeing it doesn’t affect performance in any meaningful way.

## Pathfinding

The pathfinding system in this project was implemented using a grid-based version of the A\* algorithm. This system is responsible for NPCs pathfinding from their current node to their goal node. Due to the bases not having any dynamically moving obstacles the path gets calculated all at once at the start of when a goal is assigned.

My main goal for this agent was for it to function similarly to the Unity NavMeshAgent where a goal position gets assigned and the agent handles the movement logic independently. Which I believe I successfully achieved, as in my implementation you need only set a goal for the agent after which it manages the path that needs following and updates it position over time. For the developer the only thing you need to do other than update it is set your character position to the agents.

While it preforms well one area, I would have liked to improve would have been to add a hierarchal A\* where each room also had a node. This higher level of nodes per room would have improved the NPCs pathfinding making it more efficient and realistic for movement between rooms.

## Decision Tree

The decision trees are easily contructed with their various nodes and make for a great way to test behaviour. The trees design makes it easy to give different NPCs different decision trees made from different nodes, making a tree up of a few simple behaviours gives the NPCS reactive interactiosn with the player when it comes to combat.

Though not hard to make your self I would liked to crea e

# Technical Achievements

# Project Review

# Conclusion

# Future Prospects

# Acknowledgements

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

John Doe of ACME who kindly agreed to …

I would also like to thank ICME for use of ….

Use this template when writing your research report. As a rule of thumb, the report should be of the order of 10 pages (about 250 words/page).

# Project Abstract

Replace this text with an appropriate Project Abstract.

This section should introduce the problem domain and clearly identify, justify and explain the solution(s) chosen. Care should be taken to ensure that the summary clearly demonstrates the writer’s expert understanding of the problem domain.

# Project Introduction and/or Research Question

Replace this text with an appropriate Project Introduction.

Present relevant background or contextual material and define any terms or concepts when necessary.

Here you present to the audience what you are doing and why it is important. In essence, please provide an introduction to the project, why was it chosen, the potential impact of this research. You should state a research question (if any) and present the project objectives. This will most likely be a concrete question probably from one specific area, such as AI, Networking, Graphics etc.

E.g., Research Question Example (Networking): What is the effect of threshold size in the dead reckoning approach on player performance and player experience?

Summarize the main contributions of the project.

# Literature Review

Replace this text with an appropriate Literature Review.

The literature review places your research in context. You aren’t the first person to investigate or research a particular topic. Present a short literature review with the following goals:

* 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

Replace this text with Results and Discussion.

Describe the results using diagrams such as graphs etc. as appropriate, and discuss what the results mean.

Example: Results indicate that once the threshold gets over a certain point it significantly reduces player performance and player experience

**Project Milestones**

Replace this text with Project Milestones.

Key project milestone dates and measurement on schedule, was project schedule adhered to, effectively planned for delivery on-time or ahead of schedule if appropriate.

**Major Technical Achievements**

What are your major technical achievements?

**Project Review**

What went right? What went wrong? What (if anything) is still outstanding/missing (i.e., still left to do)? If starting again, how would you approach this project differently? What advice would you have for someone attempting a similar project in the future? Were your technology choices the right or wrong ones? If you chose the wrong technology, provide justifications for why you think this. What were the implications of your technology choices?

# Conclusions

summarise your work and findings.

**Future Work**

Indicate what might be some next steps to try (if a student next year was going to undertake a project in this area what might be an interesting thing for him/her to examine?).

# References

# Johnson, D., *Procedural Dungeon Generation Algorithm*, Game Developer, Available at: <https://www.gamedeveloper.com/programming/procedural-dungeon-generation-algorithm> [Accessed 2025].https://gwlucastrig.github.io/TinfourDocs/DelaunayIntro/index.html

# Triggs, G.W.L., *Introduction to Delaunay Triangulation*, Tinfour Documentation, Available at: <https://gwlucastrig.github.io/TinfourDocs/DelaunayIntro/index.html> [Accessed].Appendices

GeeksforGeeks, *What is Minimum Spanning Tree (MST)?*, Available at: <https://www.geeksforgeeks.org/what-is-minimum-spanning-tree-mst/> [Accessed].This might include ethics application and other relevant material e.g. copy of any questionnaires used.