1337 : A First Person Shooter Designed In Unity3D

Karle Sleith

B.Sc.(Hons) in Software Development

17th April 2018

Final Year Project

Advised by: Patrick Mannion
Department of Computer Science and Applied Physics
Galway-Mayo Institute of Technology (GMIT)

Contents

About this project

Abstract 1337 is a multi-player shooter designed using the Unity Engine, The aim of the project is to create a procedually generated maze world, for the players to navigate to destroy the boss of the maze, The approach taken in this project was to make a game using C in the Unity game engine, Connect a MySQL database using XAMPP, and to Use the Unity Network Manager to connect instances of players. The SQL database will allow users to create accounts and be able to join into anothers game. Over the course of this paper, I will cover what my own opinions on Independent Game Development, how I looked at game design, How I will tackle the task, why I decided to use Unity over Unreal, System Design, my objective evaluation of my project, and finally my conclusion.We will take an indepth look at the Hunt and Kill Algorithim for maze generation and how it was implentented in Unity.

Authors My name is Karle Sleith and have a strong passion for the development of video games and their history. Ever since I was young I wanted to be a developer, with my father teaching me basic from a young age. I am creating this project for our Honors Degree in Software Development.

Chapter 1

Introduction

Video games have yet to peak as far as the technology and game design can go. As part of the growing industry, Video games have overtaking movies as the worlds most preferred form of entertainment. In September of 2013, Rockstar Games made over 700 Dollars in 24 hours with the release of "Grand Theft Auto V". And with that, Video games came to be more respected in the Software Development industry. At one time it was thought madness to sink millions into the development cost of a product, but this young, growing industry is still proving profitable despite high development costs. Major companies such as FaceBook and Microsoft are looking for new ways into the gaming scene, with Microsoft purchase of "Mojang" the creators of "MINECRAFT", and FaceBook dipping their toes in VR (Virtual Reality) with the purchase of "Oculus". Other companies have also been around with video games for many years, such as Nintendo and Sony. [1]
Virtual Reality is ever growing in the word, with HTC just announcing the "Vive Pro", we can see that the VR train is not slowing down. When we simulate a Virtual World, we want it to match reality as close as possible, we can achieve this using various equipment, such as HMDs, (Head Mounted Displays),specialized controllers, and a computer that processes the artificial world. There are also peripherals that provide the user with artificial smells and force-feedback (to give the sensation of touch) such as the humorously named "Nosulus Rift", and rumble features that have been integrated into various gaming controllers
The gaming industry is also one of the fastest growing due to its technical innovations. Facial Recognition Software, such as Intels Realsense 3D, allows the user to scan their face and display your likeness on an Avatar for you to control in a 3D world. Square Enix, the creators of hit gaming franchise "Final Fantasy", have been working on a new platform known as "Project Flare". This project will be boasting the very latest in Cloud Computing

technology, as Square aims to use the web as a massive super computer, capable of procedural generating a game world 32 Square Kilometers in size.
Live Streaming Games, is another popular trend that has been growing over the last couple of years. In 2014 Amazon acquired the web platform "Twitch" for over 980 Billion Dollars, Twitch primarily focuses on live streams of video games, including "walkthroughs" of video games by channel, broadcasts of e-sports (Electronic Sports or Competitive gaming) competitions and other gaming events. Twitch is also available on mobile, across the Android and iOS platforms. The idea behind Twitch is that people enjoy watching others play some of their favorite games, and the result of this platform has proven to bring in big revenue.
While the big blockbuster titles like Call of Duty and Far Cry will always reign supreme, the independent games scene has snuck up in the industry and taken the world by storm, they have found both critical and commercial success have proven to be profitable due to small company size, and also minus the lack of publishing cost due to digital releases now slowly becoming the norm. These companies have now been launched to the international mainstream, solving the creative fatigue within the game industry that has been around for quite some time. While big companies such as Capcom and Bethesda have shareholders to continue funding their investments, Indie Games are bound by no such limitations, it is these developers that are making the big push to have video games recognized as an art form. Independent games also have a major cultural impact, with a rally of fans ready to back their support. Crowdfunding has been an increasingly important source of funding for video games in the last 5 years. Titles such as "Mighty No 9" and "Shenmue 3", who have influential names backing them, Kenji Inafune and Yu Suzuki respectively, market their games in crowd funding campaigns on KickStarter and GoFundMe to get initial backing from the fans of their previous works. Films based on the topic of indie game development have also released to critical acclaim. Indie Game: The Movie[2], This movie shows the high level of passion from these developers, and fallows the release of 3 games "Super Meat Boy", "Braid" and "Fez". As of recent, bigger companies have now scene the benefit of indie game development, Branching out with their own divisions dedicated to smaller teams, that release a specific vision. Ubisoft's Indie Series, was founded to fund smaller companies with the resources to develop their dream games, and last year Sega, creator of pop culture icon "Sonic The Hedgehog", placed their trust in the most loyal of Sonic fans, known for developing their own "Fan Games", giving them their IP(Intellectual Property) and allowed them to make "Sonic Mania", which was released to worldwide fan fair, with many saying it was the best Sonic game to be released in the last 20 years.

Over the course of this year, I hope to make a Independent video game, this stems from the fact I believe I share the same passion for development as others in the industry.

Chapter 2

Concept

My aim is to make a Multi-player FPS (First Person Shooter) in Unity 3D using the C Object Orientated Language, I will be using multiple classes that will be used as Game Play Scripts (Shooting, Movement ect.) and creating "Prefabs", which are pre-created game objects that can be used as resources in the game.

The theme of the game will be "Cyber", being a big fan of 90's movies such as "Hackers" and "The Cube", and given the course I am attending, I feel that a computer based theme is appropriate. Additionally, as to focus on the technical side of the project over the art style, I have a idea to make the game aesthetic seem as the user is fighting "viruses" through a command line, using dark corridors and neon green walls, this will allow me to have a minimalist approach to the art design.

At the start I will be using the "Unity Network Manager Package". The "NetworkManager" is a convenience class for the HLAPI for managing networking systems.

"The High Level API (HLAPI) is a system for building multiplayer capabilities for Unity games. It is built on top of the lower level transport real-time communication layer, and handles many of the common tasks that are required for multiplayer games. While the transport layer supports any kind of network topology, the HLAPI is a server authoritative system; although it allows one of the participants to be a client and the server at the same time, so no dedicated server process is required. Working in conjunction with the internet services, this allows multiplayer games to be played over the internet with little work from developers."
For more simple network applications, the NetworkManager can be used to control the HLAPI. It provides simple ways to start and stop client and servers, to manage scenes, and has virtual functions that user code can use to implement handlers for network events. The Network Manager does have

limitations however, as it can only deal with a single client at any given time; Due to this as I want the game client to handle a max of four players, I will be developing my own custom Network Packages using the .NET Framework.
I hope for the game to have procedurally generated Maze Maps, The game is going to have a Raid like gameplay loop, where the Goal is for the players to reach the center of the maze and defeat the Boss, as every maze is a dungeon, enemies will roam and attack the players preventing progress. At the start of a "New Game", scripts will activate and create a maze of random dimension between 64 and 256 Squares, this will allow to have some short mazes and larger more difficult ones.
Lastly, I also will be using a online database that will allow users to sign into the games and save progress, such as Statistics(Games played, Virus' Eliminated ect.) Character Level, and UserID. We'll be using a SQL Database and using NAVICAT as it's editor.

2.1 Project Link

GitHub URL to the dissertation, project and Youtube Screencast

- https://github.com/karlesleith/FPSCyberAttack
- https://www.youtube.com/watch?v=RuGs9rvlAo0

2.2 Objectives

Due to being the sole developer on this project its important to stick to a schedule and meet up with my project supervisor on a regular basis to ensure I meet deadlines for each component of the project. I will be using an Agile methodology during development, incrementally updating the project and focusing on adaptability when implementing new components.

- Make a playable FPS (First Person Shooter) Using the Unity Game Engine
- Implement a Maze Generator, that will randomly generate the world each time a new instance of the game start.
- Establish a connection between the Host(Server) and Clients Using Unity's Network Manager.
- Connect to a SQL Database for Player Login
- If time allows, try and implement my own Network Packages

2.3 Chapters

In this section we will go over the different sections of the paper.

2.3.1 Gameplay Mechanics

Trying to keep a balance between the a Software Dissertation and a Game Design Document (Similar to the Doom Bible) [3] , we'll will be briefly be talking about the GamePlay Mechanics and my influences for the game.

2.3.2 Methodology

In this section we will be talking about the development methodology I used through out the project, including weekly meetings with the supervisor and planning.

2.3.3 Technology Review

In this section we will be talking about the technology we used throughout the project from beginning to end, and reasons why I chose this technology over another, drawing comparisons between the two types of software.

2.3.4 System Design

In this chapter we will talk about how all the different components were implemented and how they connect to each other, over-viewing the entire architecture of the project.

2.3.5 System Evaluation

In this chapter we will talk about how we believe the System is robust and bug free.

2.3.6 Conclusion

This final chapter summarizes the projects entire scope and will draw a conclusion about the entire development and design of the project, We will talk about why we implemented such features and if we were to do it again what would we do differently.

Chapter 3

Gameplay Mechanics

3.0.1 How Will it Play?

The game is from a FPS Perspective (First Person Shooter), The player will spawn in a generated maze and the objective is to reach the center of the maze, and defeat the dungeons boss. The game will play more ”arcade-y” like the original ”Doom”, created by John Carmack and John Romero in 1993, meaning it will be fast paced, but will have the dungeon raid-like structure, and player progression of more modern games like World of WarCraft and Destiny.

3.0.2 Targeted Platforms?

The game will be exported as a executable to be run on PCs. Since the game will be developed in Unity, it should be relatively easy to export the game to various other platforms such as UWP, Android and IOS.

3.0.3 Project Scope?

The game is made independently by myself as a Forth Year Software Development final year project, and is to be completed by 17th of April 2018.

3.0.4 Licencing?

All the Unity Assets used in the game will be made by me from scratch. I also plan on using Free Open Source music for my Title Screen and while the game is running.

3.0.5 Software

The game will use the Unity Engine 5.6 Community Edition, this is free to use software developed by Unity Technologies, I will also be using a external free programme called Blender, Blender is a professional, free and open-source 3D computer graphics software toolset used for creating animated films, visual effects, art, 3D printed models, interactive 3D applications and video games. I will be using Blender for 3D modelling to create objects for my game.

3.1 Influneces

3.1.1 Hackers

"A teenage hacker finds himself framed for the theft of millions of dollars from a major corporation. Master hacker Dade Murphy, aka Zero Cool, aka Crash Override, has been banned from touching a keyboard for seven years after crashing over 1,500 Wall Street computers at the age of 11. Now keen to get back in front of a monitor, he finds himself in more trouble than ever."

My favorite thing about this movie is the way that "Hacking" is represented, This was during a time when the use of CGI (Computer-generated imagery) was still very expensive, and the movie was able to sell tickets based on the novelty. Databases are represented as huge towers, and Virus' are Space ships that fire at the towers, Obviously this is nothing what programming is like in real life, but it provided an interesting theme for my game, we'll

be taking the idea of cleaning a virus from a system and acting it out with action orientated gameplay.

3.1.2 The Cube

"Without remembering how they got there, several strangers awaken in a prison of cubic cells, some of them booby-trapped. There's onetime cop Quentin, scientist Holloway , young math genius Leaven, master of escapes Rennes , autistic savant Kazan and architect Worth , who might have more information on the maze than he lets on. The prisoners must use their combined skills if they are to escape."
In this Science-Fiction/ Horror movie, the prisoners are trapped in this maze and have to work together to get find the exit, I have an idea to place traps around my procedural generated maze that the player will have to disarm, Also I can relate the multiple prisoners to the multiplayer aspect of the game, having the players work together to defeat the boss, I would have the boss's health scale with the amount of active players in the game.

3.1.3 Doom

In Doom, players assume the role of an unnamed space marine, who became popularly known as "Doomguy", fighting his way through hordes of invading demons from Hell. With one-third of the game (nine levels) distributed as shareware, Doom was played by an estimated 15–20 million people within two years of its release, popularizing both the business model of online distribution[5] and the mode of gameplay, and spawning a gaming subculture. In addition to popularizing the first-person shooter genre, it pioneered immersive 3D graphics, networked multiplayer gaming, and support for customized additions and modifications via packaged files in a data archive known as "WADs". As a sign of its effect on the industry, first-person shooter games from the genre's boom in the 1990s, helped in no small part by the game's release, became known simply as "Doom clones". Its graphic violence, as well as satanic imagery, made Doom the subject of considerable controversy.

The fast paced action of the Doom games is something I want to replicate, the players speed will build momentum the more he runs in one direction, allowing for a greater sense of speed through the maze. Also similar to Doom, I will be locking the rotation on the Y-axis, in Doom it was due to limitations of computers at the time, for me, personally I will be doing it as a design choice.

3.1.4 Destiny

Destiny 2 is an online-only multiplayer first-person shooter video game developed by Bungie and published by Activision. It was released for PlayStation 4 and Xbox One on September 6, 2017, followed by a Microsoft Windows version the following month. It is the sequel to 2014's Destiny and its subsequent expansions. Set in a "mythic science fiction" world, the game features a multiplayer "shared-world" environment with elements of role-playing games, such as live events.

In Destiny, their are primordially generated missions called ”Raids”, A raid is a type of mission in a video game in which a number of people attempt to defeat another number of people at a player-vs-player, a series of computer-controlled enemies in a player-vs-environment battlefield, or a very powerful boss. The term raid itself stems from the military definition of a sudden attack and/or seizure of some objective. This type of objective is most common in MMORPGs, and usually but not necessarily occurs within an Instance dungeon.I enjoyed the Raid-like structure of the games and found it interesting what was done with it , I wanted the game I created to give the same feeling of feeling lost with friends, but having a clear goal on what we have to achieve, in this case, defeating the boss and escaping the maze.

Chapter 4

Methodology

During this chapter, I will talk about the different stages of development of the 1337 Multiplayer Game, I will talk about my thought process and the my methodology approach over the course of the college year. I will talk about the C Language and technology I used in this project.

4.1 Agile

Over the course of the year, I took a Agile approach to development, due to flexibility, apposed to the Waterfall method, which would not allow me to make changes one a component was completed. The Agile software development life cycle is based upon the iterative and incremental process models, and focuses upon adaptability to changing product requirements and making sure I was satisfied with the project as a whole. This lead to me having a working prototype of the project very early on, and allowed me to fine tune different aspects one by one. The Agile approached ensured I was never stuck in a development cycle for too long a time, These short development cycles are commonly known as sprints. During a Sprint, I would typically brainstorm a new feature, design how I would implement it, program it, and test it. depending on the size of the feature, these sprints could last between 1 day to 2 weeks. The planning stage of the sprint was critical to implementing a new feature. I would give myself time to research if what I wanted to was achievable using the software I was currently using, and if it I was able to implement this feature within the given time. It was not possible to answer both at the same time, so it was better to iteratively find answers to my problems. I found that good planning lead to uncertainty and less stress while developing a new feature. When ever I reached a new milestone in the project, my progress was tracked on GitHub. Github is a great platform for

tracking development and also provides easy of use for collaboration, bug tracking and task management.

4.2 Testing

Over the course of the project I used a mixture of User Testing and Automated Testing using the Unity Test Framework, The Unity Test Runner uses a Unity integration of the NUnit library, which is an open-source unit testing library for .Net languages.
With manual testing, I would simply write a bit of code, run the game, and take notes of the changes, sometimes what would happen is that when I would add a new feature it was break code that was previously implemented. I found the best was to combat that was with Unit Testing, I was develop scripts that ran IEnumerator (Because I want these tests to be automated these were co-routines that was continuously active), that checked to see if a features behavior had changed once I edited the code. below is a pseudo-code example of making sure the Enemy Objects I spawned on the map matched the Enemy Prefab I wanted from the prefabs folder.

```
public class EnemySpawerTest{
    [UnityTest]
    public IEnumerator SpawnEnemyFromPrefab{
    var e = GameObject.FindWithTag("enemy");
    var enemySpawned = PrefabUntility.getPreFabParent(e);
    Assert.AreEqual(enemyPrefab, enemySpawned );

    }
}
```

Also thoughout my code is a series of ”Debug.Log()” fuctions, this allowed me to see if each method was being active at the expected times, and example of a Debug log is

```
int mazeRows = Random.Range(5, 64);
MaxRow = mazeRows;
int mazeColumns = Random.Range(5, 64);
MaxCol = mazeColumns;
UnityEngine.Debug.Log("Debug Rows: " + mazeRows + " Columns: "
    ↪ + mazeColumns);
```

The above code allowed me to see how many rows and collumns where generated from the Maze Algorthim

Chapter 5

Technology Review

In this section we will talking about the different technologies used trough out the development of the 1337: Multiplayer FPS (First Person Shooter), Discussing why I chose the software I did, what were the benefits of the software,and what where the alternatives.

5.1 Development Engine

During this course of this project I used a range of different programming tools to create 1337: Multiplayer Game, to finish within the deadline there was a wide range of features I had to design, program and test, which I explain below.

5.1.1 Unity

Unity is a cross platform game engine developed by Unity Technologies, it is used to develop games for PC, Consoles, Mobile Devices and Websites. Unity was designed with the idea of portability in mind, making ease of use to developed with API such as Direct3D and Vulcan. 5 major versions of Unity have been released with Unity 5 being the latest. In 2010 Nintendo started supplying Developers with free Unity Profession Edition Licences with the Software Development Kits (SDKs) in the Nintendo family(Nintendo Wii and 3DS). Unity itself was written in C and C++, but the development platform allows us to write code in C, UnityScript, and Boo. The ease of use with Unity is well documented, While the Logic of the game being developed is still needed to be coded in C scripts, which then can be attached to GameObjects, Unity allows us to have access to the Unity Store, which

supplies the Developer with assets (Paid and Free) and Scripts we can plug into our games. Although this is great for developers to use to expand on their games, the system has since been abused. In 2015 Unity Technologies drew criticism for the high volume of quickly produced video games in the PC store front Steam, these games had been using ONLY Unity Assets and monetised on the store. The CEO of Unity said while Unity is a success in making game development easier for developers, this is a unfortunate side affect.

"If I had my way, I'd like to see 50 million people using Unity – although I don't think we're going to get there any time soon. I'd like to see high school and college kids using it, people outside the core industry. I think it's sad that most people are consumers of technology and not creators. The world's a better place when people know how to create, not just consume, and that's what we're trying to promote". – John Riccitiello

Unity Vs Unreal?

When I started this project it was never my intention to development my own game engine, through the likes of a Assembly Language, I it was time for me to look else where.
As of 2018, There are currently 2 major players in development of game engines, Unity 3D and Unreal Engine. Unity 3D has 4.5 million subscriber and at 48 Percent market share whereas Unreal Engine currently is only at 13 Percent. Because of the popularity of Unity, doesn't necessarily mean it is better, infact both are quite unique to each other and carry their own advantages.
My first concern was pricing, with this being a college project, I wanted to keep cost to minimal if not none if possible. Both Engines have community editions that provide the most of the features for free, but lock others behind a pay wall, so I began to analysis what would give me the most accessibility straight out the starting gate, that also was in relation to what I was planning to develop.
After I took into consideration the programming languages, I wanted to make something the was secure and scale able, but also something I believe I could adapt to naturally with my skill set, Unreal Engine 4 uses C++ for coding and Unity 3D uses JavaScript or C. so deciding which program was better was down to my own personal preference.

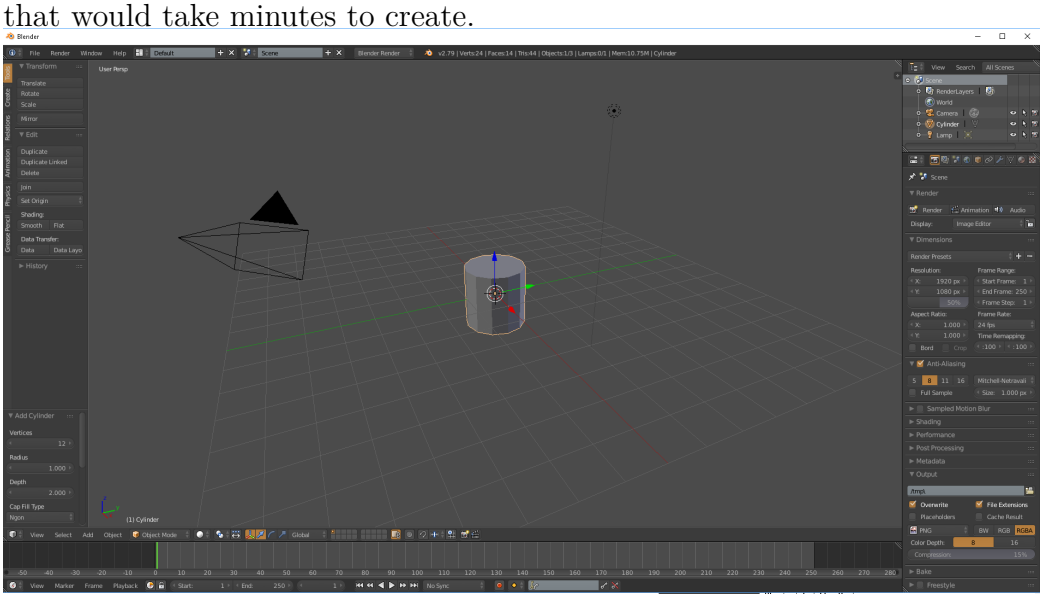
Finally was ease of use, Unity 3D is known for it's easy to use user interface where is you were new to making games, you could get started fast and easily. Though in recent updates Unreal Engine had many many improvements, they still fall short behind Unity in terms of there User Experience.

Taking these factors into consideration, I decided to go with Unity3D, Unreal Engine might have been better if I was going for a low level development process, giving me free use to script pretty much whatever I wanted. Unity with its ease of use, and C Programming language,(I believed that it wouldn't be a far jump from working on Java),which would have helped in the development process, was what led me to the decision, Due to the aesthetic of my game, and my lack of need for "Hyper Realistic Graphics", Unity matched what I needed, without me going overkill.

5.1.2 Blender

Blender is a professional, free and open-source 3D computer graphics software toolset used for creating animated films, visual effects, art, 3D printed models, interactive 3D applications and video games. Blender's features include 3D modeling, UV unwrapping, texturing, raster graphics editing, rigging and skinning, fluid and smoke simulation, particle simulation, soft body simulation, sculpting, animating, match moving, camera tracking, rendering, motion graphics, video editing and compositing. It also features an integrated game engine.

When I was trying to decide what the theme of the game would be, I wanted to focus on a Art Style that was minimalist, and would allow me to be more focused on the features that made up the game. I have had previous experience with CAD (Computer Aided Design) using programs such as SolidWorks, and intended to make low poly prefabs and enemies that would take minutes to create.

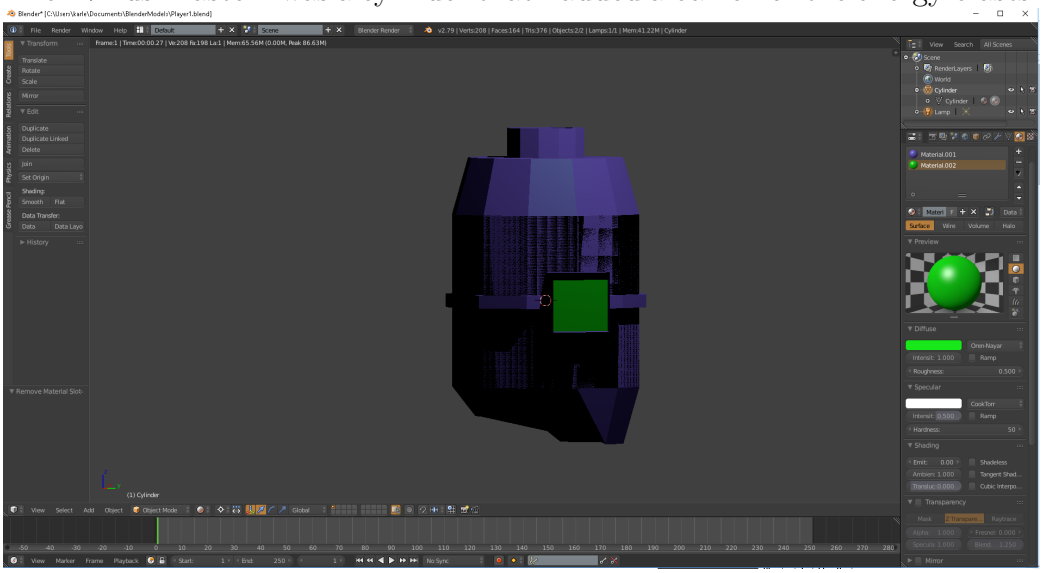


Unfortunately I found that I was unable to import objects from Mechanical CAD into Unity, the models prepare in Solidworks will have nearly all dimensions defined with Parameters. It works on features which can be edited to perform revisions. This lead to to discovering Blender, Blender is Polygon Modeling software which works on meshes also called subdivision surface. So basically you don't have to define a lot of dimensions and most of work is freeform.

Additionally Unity naively recognizes Blender's ".blend" files, Unity currently imports

- All nodes with position, rotation and scale. Pivot points and Names are also imported.
- Meshes with vertices, polygons, triangles, UVs, and normals.
- Bones
- Skinned Meshes
- Animations

I wanted to have minimal modelling, so keeping with the theme "Cyber", and having the user feel like navigating a command line, the only Objects I will be modelling with Blender are the enemies, the Walls and Floor Objects, and the players "Virus Blaster" weapon. To create the enemies I wanted to to extrude (To shape to give depth) to a font pack, and color the ASCII characters red, this gives the user the interpretation that these are enemies virus's within the world. For the walls was a simple pane extruded by 0.5, and The "Virus Blaster" was a cylinder that I added a barrel for the energy blasts.



5.1.3 MySQL

MySQL is the world's most popular open source database. With its proven performance, reliability and ease-of-use, MySQL has become the leading database choice for web-based applications, used by high profile web properties including Facebook, Twitter, YouTube, Yahoo! and many more.Oracle drives MySQL innovation, delivering new capabilities to power next generation web, cloud, mobile and embedded applications. Over The course of the project I will be using a MySQL database to take in Player Information, such as Usernames, Passwords and Statistics.

```
Basic syntax to CREATE a database:
CREATE DATABASE gameDatabase;
 DROP a database if it exists:
DROP DATABASE [ IF EXISTS ] gameDatabase;
 CREATE a new table:
CREATE TABLE table_name(
column1 Id,
column2 Username,
column3 Password,
.....
columnN datatype,
PRIMARY KEY( ID )
);
 INSERT data into a table:
INSERT INTO gameDatebase (column1, column2, column3,...columnN)
VALUES (value1, value2, value3,...valueN);
 SELECT data from a table using the AND operator with WHERE
clause:
SELECT column1, column2, columnN
FROM gameDatebase
WHERE [condition1] AND [condition2]...AND [conditionN];
```

Chapter 6

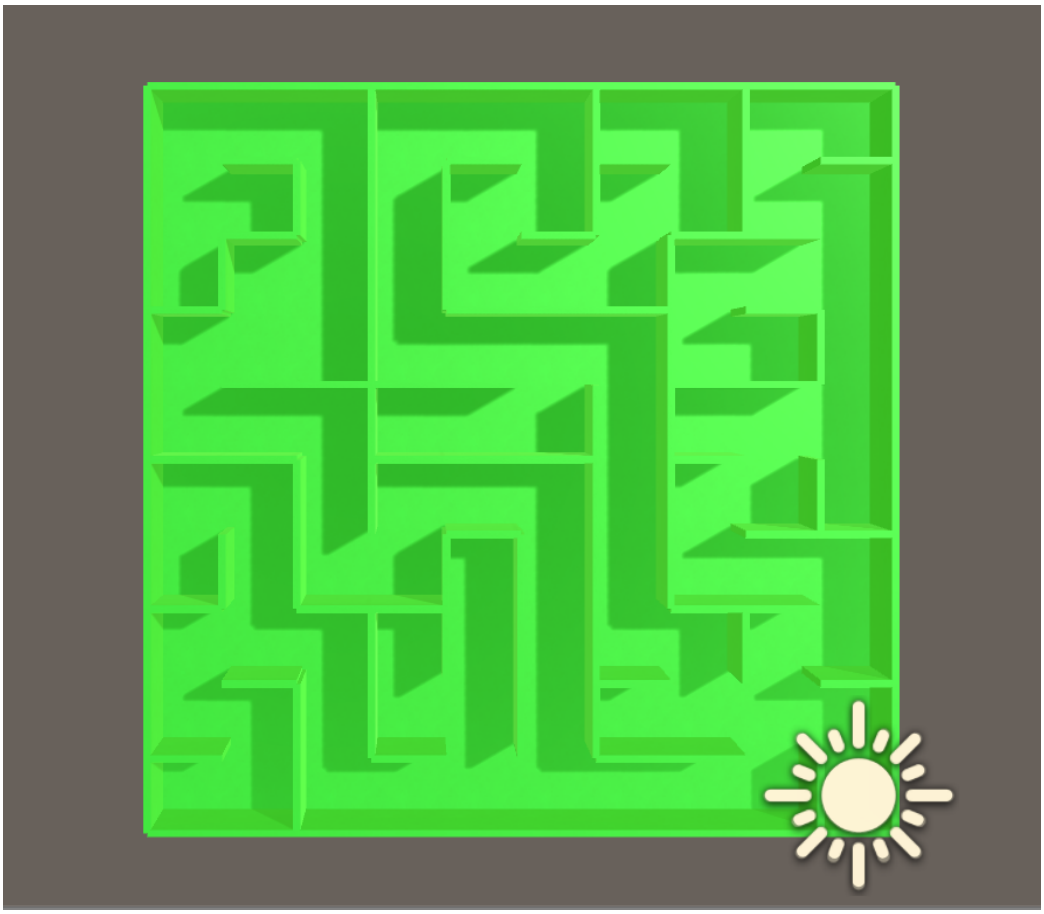
System Design

In this chapter we will discuss the overall implementation of the design of the game, included is a UML Diagram that will help visualize the software structure.

6.1 Game Design

The game starts with 3 primary scenes, The Title Screen, The Maze Builder and the Game, When the player selects the the start new game option, it creates a new instance of the game, and drops the player in a random point in the maze to defeat the boss.
The Game will Randomly Generate a Maze between 32 Cells Squared (1024 Cells) and 64 (Cells 4096), a, drop the player into the game, the objective is to file the Enemy Boss and that will complete the Game. I will begin with creating an enemy spawner and generate the enemies around that spawnPoint, making the purpose of the Players is to destroy the Enemy Spawner.

6.1.1 Maze Generation



In the game I created a procedurally generated Maze that will be used as the basis for my level, based of the ”Hunt and Kill Maze Algorithm”. The Hunt and Kill Algoritim works as such.

- 1. Choose a Starting Location
- 2.Perform a random walk, carving tunnels to unvisited cells, until the current cell has no unvisited neighbouring cells.
- 3. Begin the Hunt! We scan the cells looking for an unvisited cell that is next to a visited cell. If found, carve a tunnel between the two and let the formerly unvisited cell be the new start point.
- 4. Repeat steps 2 and 3 until the hunt mode scans the all cells and finds no unvisited cells.

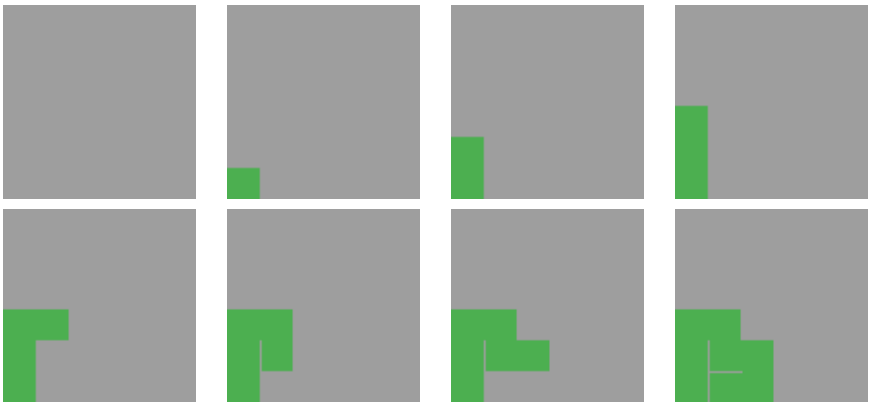
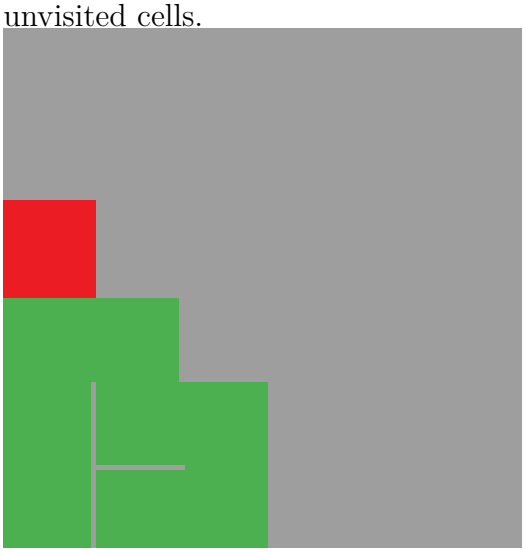


Figure 6.1: Representation of 5 x 5 Maze Gen

I used the site "BucksBlog" to get me started understanding Maze Generation. [4]
I had also ha access to a JavaScript maze builder to help me visualize the procedure step by step [5]

At this stage the Maze Generation would come to an end, so we implement recursive back tracking to go back to the first row and look for previously unvisited cells.



As you can see, the new red cell is our new Starting location, and we repeat the same algotithim as before, Once we connect unvisted cells to a visited neighbour cell, our Maze Generation is complete.

6.1.2 Code Implementation

The First Script to define is the Cell Element, each cell is has the same properties so we can define it in its own script.

```
//Class Interface Object for Each cell, each cell has a boolean
   ↪ to see if it was checked, and has GameObjects Connected
   ↪ to it, that represent 4 walls and a floor
public class MazeCell {
       public bool visited = false;
    public GameObject nWall, sWall, eWall, wWall, floor;
}
```

After we have to Initialize the maze, what I have done here is only put walls on the east and the south side of every cell, I did this because if I created Cells with all four wall, I will be creating overlapping objects, that have Box Colliders, and will spend memory, cause a clipping effect, and create potential issues for the player navigating the maze, The issue is with this set up the first row and column will be lacking walls at the boundaries of the maze. I used a simple If statement to account for this, if the c = 0, add a wall to the west of the cell, if the r = 0, add a wall to the north. Finally for simplicity. I use the same panel for the was as I do the floor.

```
private void GenerateMaze() {
       mazeCells = new MazeCell[mazeRows,mazeColumns];

       for (int r = 0; r < mazeRows; r++) {
              for (int c = 0; c < mazeColumns; c++) {
                     mazeCells [r, c] = new MazeCell ();

                     mazeCells [r, c] .floor = Instantiate (
                        ↪ wall, new Vector3 (r*size, -(size/2
                        ↪ f), c*size), Quaternion.identity)
                        ↪ as GameObject;
                     mazeCells [r, c] .floor.name = "f " + r +
                        ↪  "," + c;
                     mazeCells [r, c] .floor.transform.Rotate
                        ↪ (Vector3.right, 90f);

                     if (c == 0) {
                            mazeCells[r,c].wWall = Instantiate
                               ↪  (wall, new Vector3 (r*size,
                               ↪  0, (c*size) - (size/2f)),
```

```
                                    ↪ Quaternion.identity) as
                                    ↪ GameObject;
                            mazeCells [r, c].wWall.name = "
                                    ↪ wWall " + r + "," + c;
                    }

                    mazeCells [r, c].eWall = Instantiate (
                        ↪ wall, new Vector3 (r*size, 0, (c*
                        ↪ size) + (size/2f)), Quaternion.
                        ↪ identity) as GameObject;
                    mazeCells [r, c].eWall.name = "eWall " +
                        ↪ r + "," + c;

                    if (r == 0) {
                            mazeCells [r, c].nWall =
                                    ↪ Instantiate (wall, new
                                    ↪ Vector3 ((r*size) - (size/2f
                                    ↪ ), 0, c*size), Quaternion.
                                    ↪ identity) as GameObject;
                            mazeCells [r, c].nWall.name = "
                                    ↪ eWall " + r + "," + c;
                            mazeCells [r, c].nWall.transform.
                                    ↪ Rotate (Vector3.up * 90f);
                    }

                    mazeCells[r,c].sWall = Instantiate (wall,
                        ↪  new Vector3 ((r*size) + (size/2f),
                        ↪  0, c*size), Quaternion.identity)
                        ↪ as GameObject;
                    mazeCells [r, c].sWall.name = "sWall " +
                        ↪ r + "," + c;
                    mazeCells [r, c].sWall.transform.Rotate (
                        ↪ Vector3.up * 90f);
            }
        }
}
```

Below I show the kill Method, After will first initialize the Maze with a random cell count, we have to run our ”Hunt and Kill agorithim” to create it. The Algorithim goes on a random stroll, and wont go back on itself until it hits a wall.

```
private void Kill() {
       while (AvailbleRoute(currentRow, currentCol)) {

              int direction = NumGen.GetNextNumber ();
       Debug.Log("Direction for KillMethod " + direction);

              if (direction == 1 && CellIsAvailable (
                  ↪ currentRow - 1, currentCol)) {
                      // North
                      DestroyWallIfItExists (mazeCells [
                          ↪ currentRow, currentCol].nWall);
                      DestroyWallIfItExists (mazeCells [
                          ↪ currentRow - 1, currentCol].sWall);
                      currentRow--;
              } else if (direction == 2 && CellIsAvailable (
                  ↪ currentRow + 1, currentCol)) {
                      // South
                      DestroyWallIfItExists (mazeCells [
                          ↪ currentRow, currentCol].sWall);
                      DestroyWallIfItExists (mazeCells [
                          ↪ currentRow + 1, currentCol].nWall);
                      currentRow++;
              } else if (direction == 3 && CellIsAvailable (
                  ↪ currentRow, currentCol + 1)) {
                      // east
                      DestroyWallIfItExists (mazeCells [
                          ↪ currentRow, currentCol].eWall);
                      DestroyWallIfItExists (mazeCells [
                          ↪ currentRow, currentCol + 1].wWall);
          currentCol++;
              } else if (direction == 4 && CellIsAvailable (
                  ↪ currentRow, currentCol - 1)) {
                      // west
                      DestroyWallIfItExists (mazeCells [
                          ↪ currentRow, currentCol].wWall);
                      DestroyWallIfItExists (mazeCells [
                          ↪ currentRow, currentCol - 1].eWall);
          currentCol--;
              }
```

```
                mazeCells [currentRow, currentCol].visited =
                    ↪ true;
        }
```

At the end of the ”Kill” phase of the algorithm will will assume that the maze unless was can prove otherwise,With the ”hunt” method we are actively looking for a new starting location, with an cell next to it that has not been visited, if it hasn't been visited we we go through the Kill again from the new starting location, until we can no longer find a empty cell in the Hunt phase.

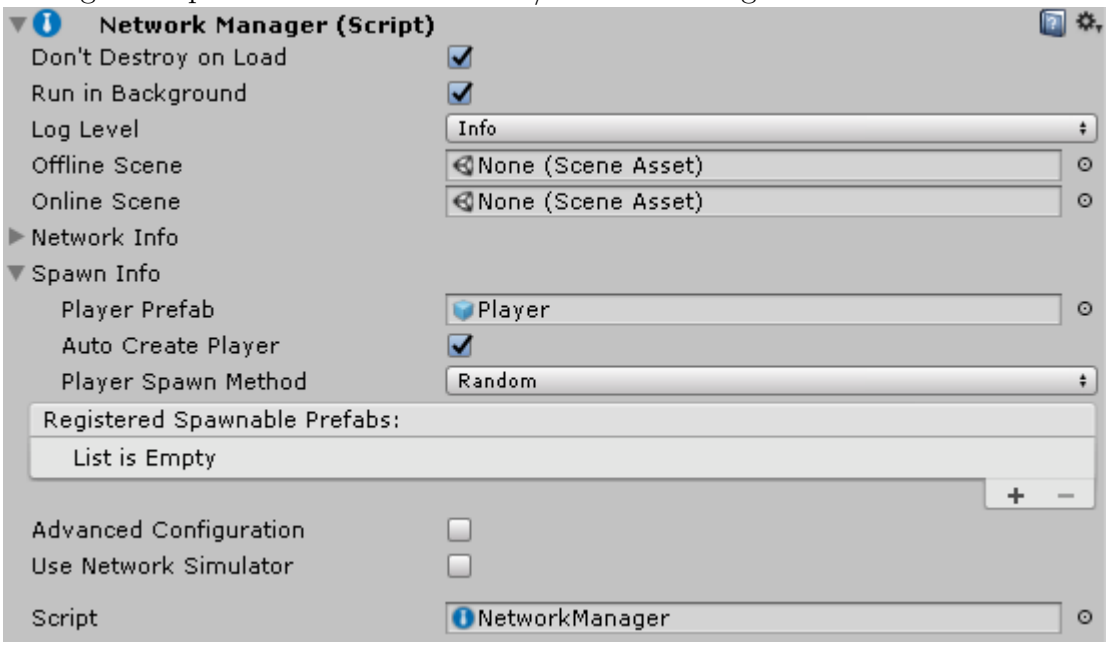
```
private void Hunt() {
        mazeDone = true; // We will assume the maze is complete
            ↪  if we fine a cell that is not, we revert the
            ↪ Bool back to false

        for (int r = 0; r < mazeRows; r++) {
                for (int c = 0; c < mazeColumns; c++) {
                        if (!mazeCells [r, c].visited &&
                            ↪ CellNextVisited(r,c)) {
                                mazeDone = false;
                                currentRow = r;
                currentCol = c;
                                DestroyWall (currentRow,
                                    ↪ currentCol);
                                mazeCells [currentRow, currentCol
                                    ↪ ].visited = true;
                                return; //break


                        }
                }
        }
}
```

6.2 Network Connection

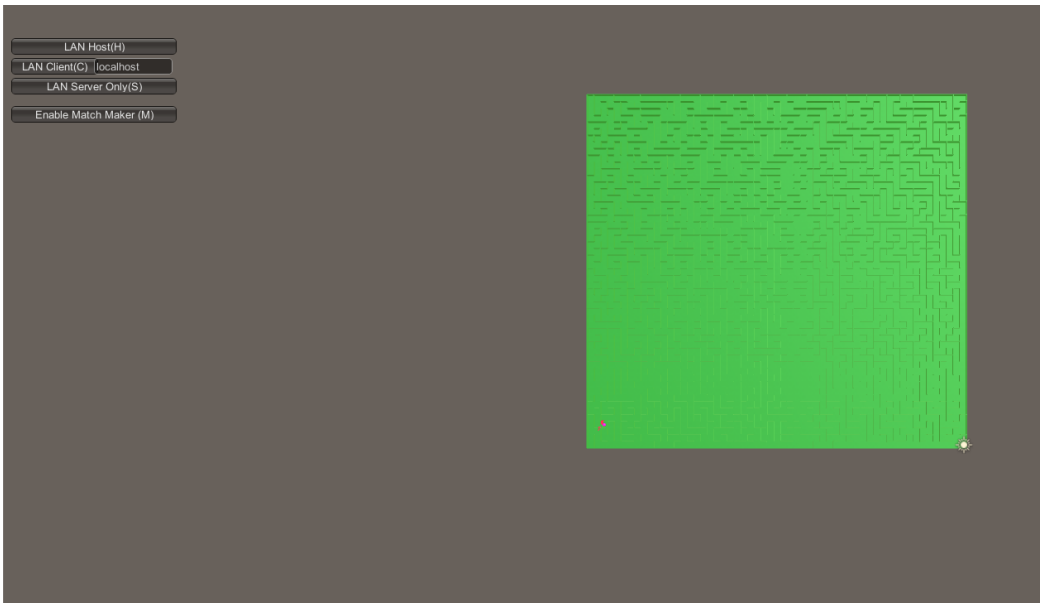
During the project the plan was to make the Network Packages on my own and implement them on my own, During my research I discovered the Unity Network Manager to help get me started.

6.2.1 Unity Network Manager

For Network Applications the NetworkManager can be used to control the HLAPI (Higher Level API), The Network Manager also comes with a component to provide a HUD interface, It provides a simple way to start and stop client and servers, to manage scenes, and has virtual functions that user code can use to implement handlers for network events. The NetworkManager deals with one client at a time.
To get started with the Network Manager, I must first create an empty GameObject to host the `NetworkManager`, Script, Then add the NetworkManager component from the Network/NetworkManager menu item.



The Network Manager HUD is another component that works with the Network Manager. It gives you a simple user interface when the game is running to control the network state. This is good for getting started with a network project.



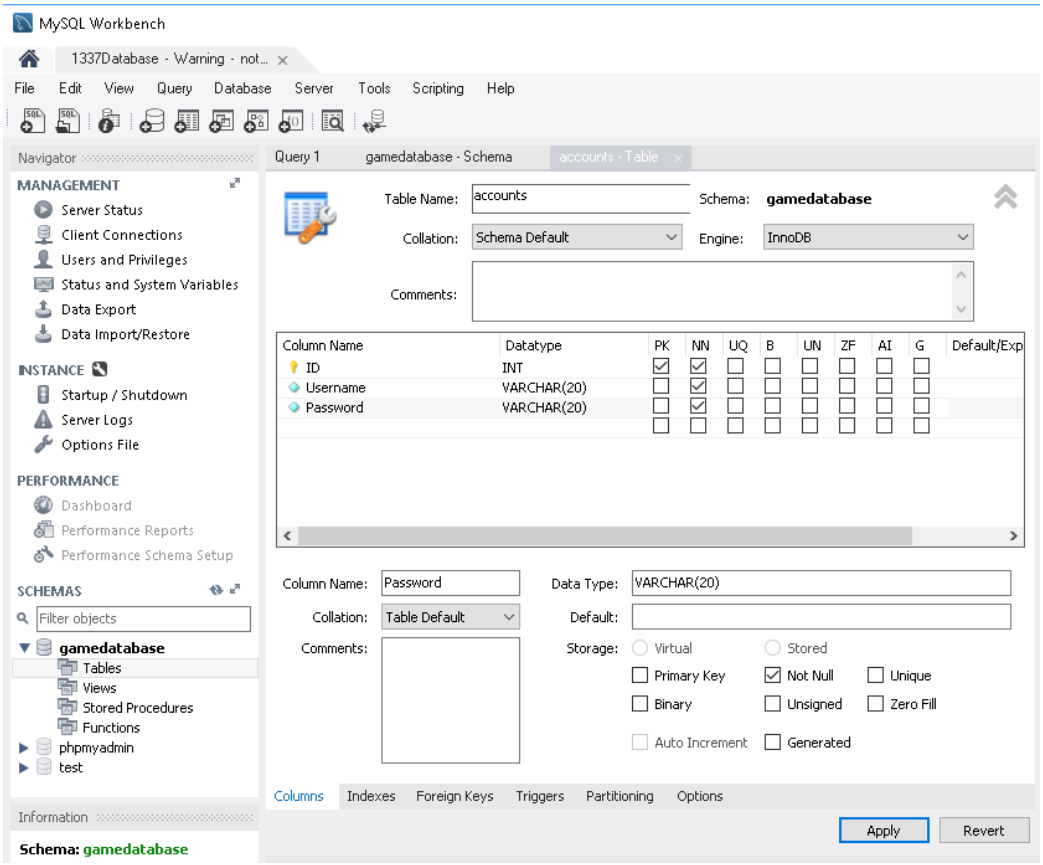
Once I added these scripts, you have to assign Network Identity to the Player Prefabs, Anything that is being sent over the internet must be given a Network Identity.

6.3 Database Connection

I made a connection to the a MySql Database through the uses of XAMPP and MySQL Workbench, MySQL is an open-source relational database management system (RDBMS). Its name is a combination of ”My”, the name of co-founder Michael Widenius’s daughter, and ”SQL”, the abbreviation for Structured Query Language.

```
1  CREATE TABLE `gamedatabase`.`accounts` (
2    `ID` INT NOT NULL,
3    `Username` VARCHAR(20) NOT NULL,
4    `Password` VARCHAR(20) NOT NULL,
5    PRIMARY KEY (`ID`));
6
```

When I started the SQL Database, I intended for it to be a login feature for the game, I downloaded XAMMP, which is a free cross platform Web Server interface that will host a SQL Client. I combined it with MySQL WorkBench and Database Interface to create tables and accounts for the game.



I made a couple of test accounts to see it I could access the MySQL DataBase in C, and then parse the information into my game.

Chapter 7

System Evaluation

In this section I will evaluate the system based on the following topics

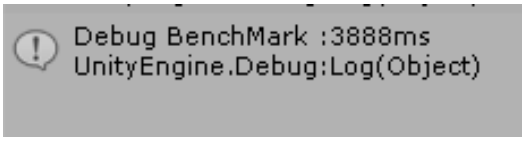
- 1. Speed, Benchmarking the the Space and Time complexity for the Maze Generation Algorithm
- 2. Scalibity, Can the Application Scale?
- 3. Robustness, Did I properly test the scripts to make sure I won't run in to unexpected outputs.
- 4. Mantainablity, Using C did I use proper use of the Object Orientated language, can the software be easily updated in the future?

7.1 Speed

```
void Start () {
    //BenchMarking
    Stopwatch sw = Stopwatch.StartNew();

    GenerateMaze();
    MazeGen mg = new HuntAndKillMazeAlgorithm(mazeCells);
    mg.BuildMaze();

    sw.Stop();
    UnityEngine.Debug.Log("Debug BenchMark :" + sw.
        ↪ ElapsedMilliseconds+ "ms");

}
```

Above I am testing how fast in terms of milliseconds it takes to generate a maze, my first test will be 10 X 10 Cells,then 32 X 32 Cells, and 128 X 128 Cells.
What I am tring to work out here is called Algorithmic Efficiency, for the maximum efficiency what we want to do is to minimize the ammount of resources used. However, the various resources (e.g. Space vs Time) cannot be compared directly to each other, so which of two algorithms is considered to be more efficient often depends on which measure of efficiency is considered the most important, e.g. the requirement for high speed, minimum memory usage or some other performance benchmark. To me currently I want to find out what is the biggest maze I can create in the shortest amount of time, I will then take that number and set that to the maxium size of the maze. In my case I am happy with 64 x 64.

Figure 7.1: Benchmarks from Maze Generation, 64 x64 (4096 Cells)



As this number is currently under 5 seconds, I am quite happy to have this be the maximum size of the level for the game. In theory the algorithm currently should be liner to time and space, but in reality the more objects we are tring to generate to the world, which is 5 per cell the Maze Generation will get slower, for a comparison between how many objects were generated with a 64 x 64 maze and a 128 x 128 maze.

$$(64^2) * 5s$$

= 20480 GameObjects in 3.8 seconds

$$(128^2) * 5$$

= 81920 GameObjects in 17.5 seconds

7.2 Scalability

Scalability is the process of adding more resources to the program. There are two different types of scaling. **Horizontal Scaling** or to scale in/out. This is the process of adding more nodes (or removing nodes) to an already established system. **Vertical Scaling** or to scale up/down, is means to add

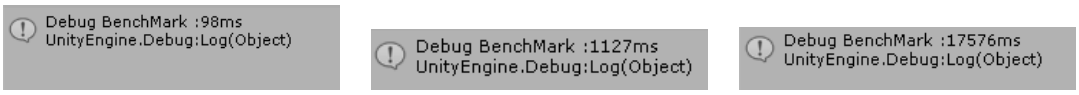


Figure 7.2: Benchmarks from Maze Generation, 10 x 10, 32 x 32 and 128 x 128

resources to (or remove resources from) a single node in a system, typically involving the addition of CPUs or memory to a single computer to make it more powerful. Vertical scaling of existing systems also enables them to use virtualization software more effectively
The project is highly scalable, with the Unity Network Manager can take in as many players as the Host's computer can handle, Also with the MySQL database is easily Scalability through the use of additional thread pools, In the context of the project this was not needed.
The Maze Building algorithim is designed to be scalable, and its space time complexity will be down to what the hosts computer can handle.

7.3 Robustness

Robustness testing is any quality assurance method that is focused on the testing the robustness, which is the strength of the code of the software. Robustness testing has also been used to describe the process of verifying the robustness (of how correct) the test cases were in a test process.
Over the course of the project I used a mixture of User Testing and Automated Testing using the Unity Test Framework, The Unity Test Runner uses a Unity integration of the NUnit library.

```
[Test]
public void GameObject_CreatedPlayer_WillHaveTheNamePlayer()
{
    var a = new GameObject("Player");
    Assert.AreEqual(Player, a.name);
}
```

The biggest thing during testing is to worry about if the correct objects are being spawned in the world at the right time. Some Objects can get lost while being created. An example is that when I am procedurally generating each cell, that each cell must keep track of their four walls and floor, considering each wall will have its on number (E.g w(23,23)).
Over the course of the project, we maintained consistent robustness, and

continuously and accurately tested for various problems that may have arose. The application is very robust and works with any known issues. During the projects development we conducted various test, to measure the strength of the application.

7.4 Maintainability

Maintainability is defines that how easy it is to maintain the system. This means that how easy it is to analyze, change the code and test the application. There are various ways to test the maintainability of the Software Maintainability test will be defined on the time it takes and difficulty on editing code, so that changes to the software can take effect

Corrective maintenance is the process of correcting problems. The maintainability of a system can be measured in terms of the time taken to diagnose and fix problems that have arisen within the Software System.

Perfective maintenance is the process of making enhancements. The maintainability of a system can also be measured in terms of difficulty that it to to make required enhancements to that system. This can be tested by recording the time taken to achieve a new piece of identifiable functionality such as a change to the database,We can determine how difficult it is to make these changes by running a series of test case, all of which relate to enhancements to the same database, and the resulted difficulty is the average time it took to make the changes between tests, We can then compare this against the target time will have project, and weather or not the target enchantments are met.

Adaptive maintenance is the process of adapting to changes in environment. The maintainability of a system can also be measured in terms on the effort required to make required adaptations to that system. This can be measured in the same way described above for perfective maintainability.

Preventive Mainteance is the process of preventing further development costs on a project, It is important to identify what actions/steps can be taken to make ease of development and testing in the future.

With `C#` being a Object Orientated Language, The class system as lead to my design with cohesion and loose coupling. The project is well structured, and easily maintainable.

Chapter 8

Conclusion

To finish up this project, Over the last 9 months I have created a Multiplayer FPS(First Person Shooter), with a procedually generated world, and functionality to connect to a MySQL Server. I have learned alot about Maze building algorithms and the Unity Engine and its limitations. It was my intention to make a game that I would have enjoyed growing up based on my interests as a child, but also balance being able to show my skills in programming, Algorithmic efficiency and Object Orientation, and I believe that I have achieve that.

8.1 Outcomes

At the beginning of this project I had set out objectives to tackle that had resulted in the following outcomes

- I have created a Unity 3D First Person Shooter, with Multiplayer Components
- I have created a Procedually Generated Maze, That will be the world for the Players to Explore
- I have established a connection between Unity instances and a SQL database.

8.2 Future Development

Although we are happy with how we developed the game, there is still room to improve, from a gameplay standpoint the World is very bland, although is is due to design and simplicity, given more time, we would've added "GamePlay Modifiers" that would have changed the behavior of weapons and enemies. we would've added a larger variety of enemy types, using a random generator that will choose enemies types from a "pool" to place in the maze, Weapon variety would be the top of the list, with different fire speeds, damage outputs and cosmetic variations.
At the moment the world is not very scalable, and that most of the resources go into generating a Maze, I am unable to conclusively determine that whether or not it is due to hardware limitations or do to with the "Hunt and Kill" Algorithim I have currently in place, While I have Initially wanted to have HUGE dungeon designs between 64 Cells(4096 Cells) Squared and 256 Cells Squared(65536 Cells), I learned very fast it was not feasible, both from gameplay (With the map just being to large to be fun to explore) and from memory limitations.
The SQL Database currently only takes in User name and nothing else, we would've liked to have been able to track different player statistics.
While we have started the creating custom network packages for the game, I have fallen back on the Unity Network Manager, Given more time I feel I would've been able to complete this feature.
Finally the players are unable to communicate with each other, I would've like to added a messaging system between players to help with communication facing the boss.

Bibliography

[1] A. Beattie, “Power Up Your Portfolio With Video Game Stocks, howpublished = ” "https://www.investopedia.com/articles/stocks/08/video-games-investing.asp”, year = 2018,.”

[2] J. Swirsky and L. Pajot, “Indie Game: The Movie, howpublished = ” "http://buy.indiegamethemovie.com/”, year = 2012,.”

[3] T. Hall, “Doom Bible, howpublished = ” "http://5years.doomworld.com/doombible/doombible.pdf”, year = 1992,.”

[4] J. Buck, “Maze Generator.” ""http://weblog.jamisbuck.org/2011/1/24/maze-generation-hunt-and-kill-algorithm/, 2018.

[5] N. Lakhotia, “Maze Generator.” "https://codepen.io/neeraj08/pen/NbzPmE/, 2018.