National College of Ireland

BSc in Computing

2016/2017

Conor Das

X11531793

conordas@gmail.com

Survive

Technical Report

TABLE OF CONTENTS

EXECUTIVE SUMMARY

The Game Survive is a first-person shooter survival game in which the player controls a character who must survive waves of enemies. All enemies will be controlled by the AI. The enemy count will increase the longer the player stays alive. The player will not be able to run far they are trapped in an arena and cannot escape. If the player dies it will be game over there is no option to save or start from where they were. This is done purposely so that the game will be more of a challenge and will become more competitive in that they must not make mistakes and must fight to survive. The objective of the game is to provide a fun competitive survival game.

1 INTRODUCTION

1.1 BACKGROUND

The competitive side of gaming is growing larger every year, some player's will compete in 'Speed Runs' to see who can complete a game fastest, other will go into a multiplayer combat game and compete against one another, I saw this market and saw an opportunity to create a game that would fit in with this market. It could be two friends playing to see who can stay alive the longest or having a global leaderboard in which streamers on Twitch.tv could compete to be top of.

1.2 AIMS

The aim of this project was to create a fun challenging and competitive game for the player. The game will begin with some basic enemies attacking the players, as time goes on the number of enemies will increase and more difficult and more powerful enemies will start to attack. The player will have a timer which will increase and see how long they can survive against the enemies.

1.3 TECHNOLOGIES

Unity 3D will be the main technology to develop my game. Unity will be the game engine in which the game's scenes and environments will be created in. I will also import assets from the Unity Asset Store to use in my project for the environments.
Visual Studio will be used for all C# coding being used within the project.
Mixamo within Adobe Fuse will be used. This is for developing the character models and obtaining character animations which will save a lot of time in the development of the game.

2 System

2.1 Requirements

There have been multiple changes to the requirements from what was described in the Requirements Specification document. These include have enemies set on a timer rather than only spawning after defeating a wave of enemies. The game was also changed from being a Third person melee combat survival game to being a First-person shooter survival game.

2.1.1 Functional requirements

Functional Requirements will outline the most important functions that the system must be able to perform.

Select Menu Options

The player must be able select any of the options on the Main Menu

Play Game

The user must be able to start the game and move freely while playing. The game will be loaded and the player will be able to move throughout the environment.

Pause Game

The user must be able to pause the game at any time while playing. They can then take a break from playing and come back to the game. This is done by pressing the ‘P’ key.

Lose Game

The player must be able to be defeated by the enemies and get sent to the ‘Death Menu’. From here they can restart the game or exit to the main menu.

Fight/Defeat Enemies

The player must be able to fight and defeat enemies. They must be able to shoot and kill the enemies. The enemies make the game challenging.

Quit Game

The player must be able to quit the game and stop playing. This is done from the Main Menu.

Player Movement

The player must be able to move around the environment. This is done using the ‘W’, ‘A’, ‘S’, ‘D’ keys to move and the Mouse to look around. The player can also sprint using the move buttons while holding down the ‘Shift’ key.

Player Shooting

The player must be able to shoot their gun. They can shoot the gun by clicking the left mouse button or the ‘Ctrl’ key.

2.1.2 Data requirements
The Data is processed in the background with the player's timer which shows time spent alive visible to the user while playing.

2.1.3 User requirements
The requirements that will be expected of the user to successfully run the game are as follows:

- The user must have a computer that is capable of running DirectX11
- The user must have a keyboard and mouse.
- The user must have the Survive application downloaded on their computer

2.1.4 Environmental requirements
For the user to be able to play the game, the users will need a windows, Linux or mac operating system environment.

2.1.5 Usability requirements
In terms of what the user will need to understand to play the game, they are as follows:

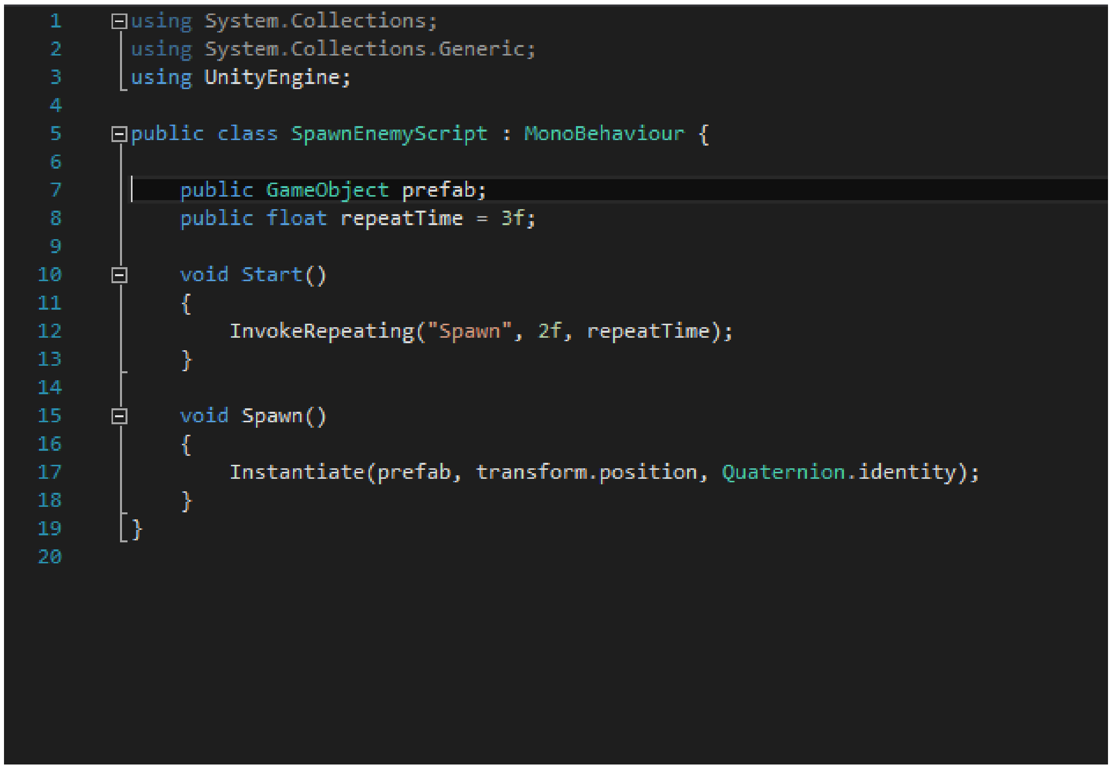
- The user must understand the games controls to play the game successfully.
- The user will need to know how to navigate a 3d world.

2.2 Design and Architecture
The application was built using the game engine Unity 3D. Unity supports the creation of games with its wide variety of function used in the building of games, these include the ability to build objects, characters and scenes. There is an option to add animation and functionality of to these, this is done by adding in animation controllers. My coding for how the project and AI will act will be done in C# through Visual Studio. I used a combination of Adobe Fuse with Mixamo and the Unity Asset Store to create my models.

2.3 Implementation
To create the project, I had many scripts. The setup of my scripts is using Master scripts to call events which will be referenced on each individual script. The main scripts are as follows:

SpawnEnemyScript.cs

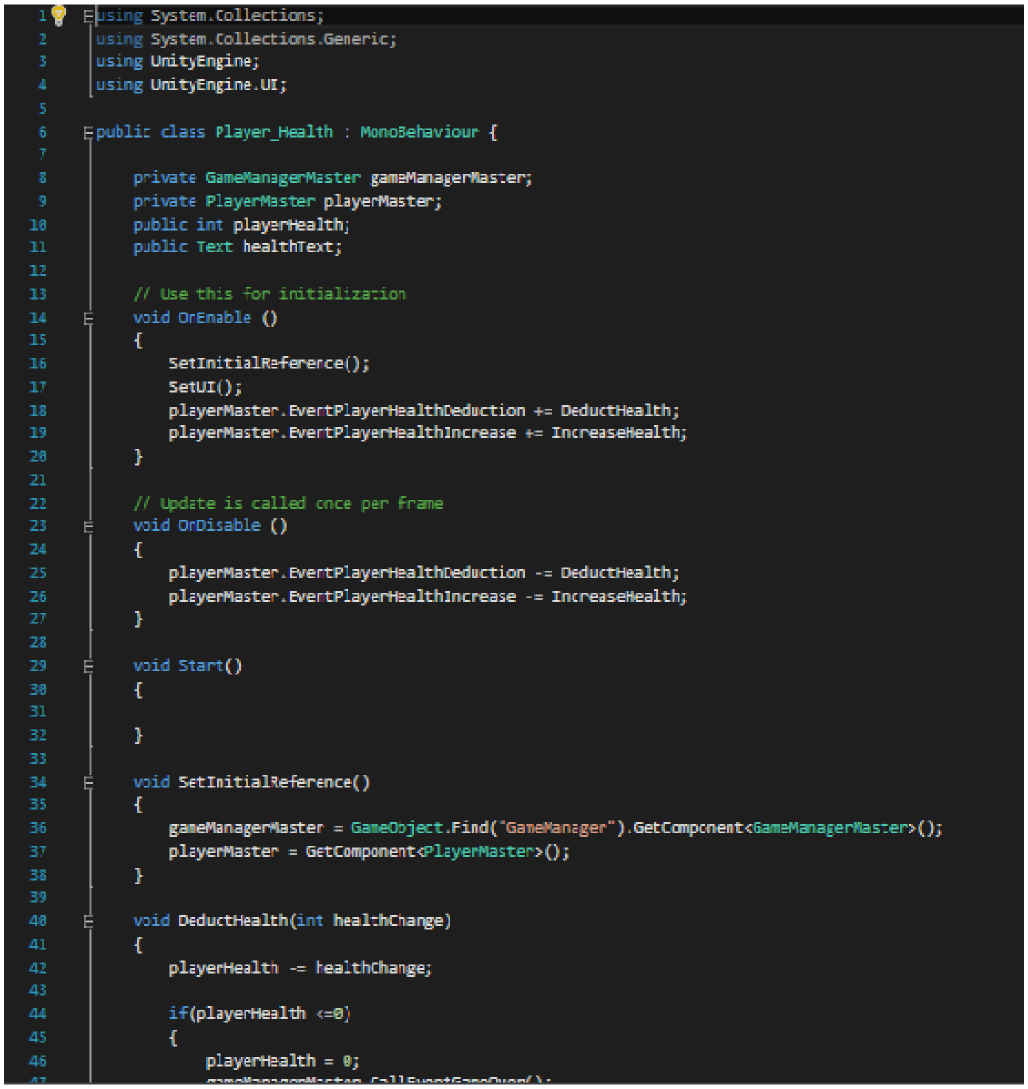
This script will spawn the enemy in to the environment. First a GameObject is declared which is a prefab that is what will load in to the environment, the repeatTime is also declared which will be how often the enemies will spawn, this is set as 3 seconds as default but is a public variable so it is editable outside the script. In the Start method we tell it to start the spawn method, to start it after 2 seconds and to repeat after the repeatTime variable we have set. In the Spawn method, you are declaring what will spawn (the enemy) where it will spawn (spawn point location) and the direction it will face.



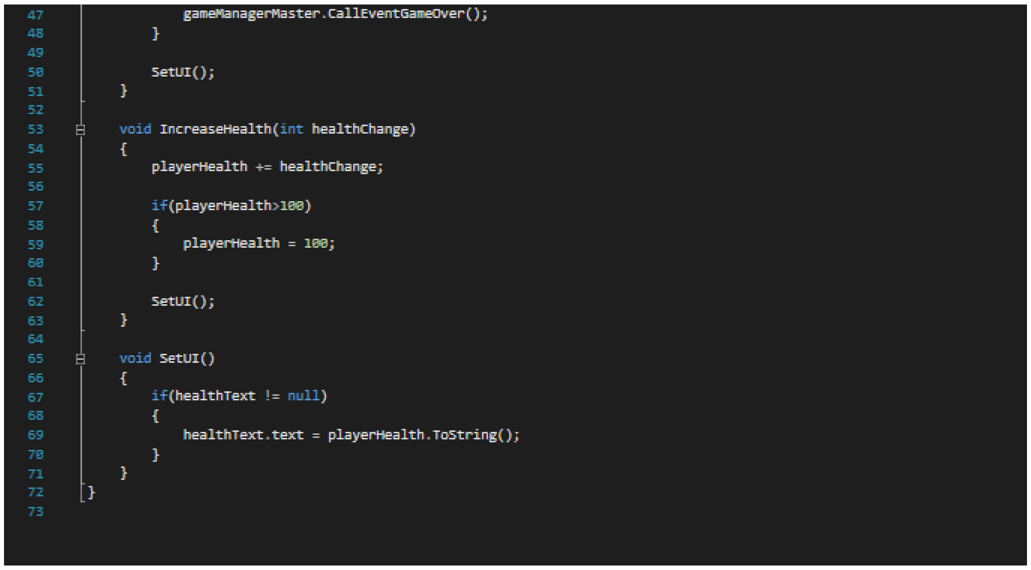
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SpawnEnemyScript : MonoBehaviour {

    public GameObject prefab;
    public float repeatTime = 3f;

    void Start()
    {
        InvokeRepeating("Spawn", 2f, repeatTime);
    }

    void Spawn()
    {
        Instantiate(prefab, transform.position, Quaternion.identity);
    }
}
```

Player_Health.cs

The Player Health script is what determines the player's health level and how what happens when it hits 0. In this script, we are calling both the Game Manager Master and the Player Master scripts, these control the events for things like player health increase and decrease and game over. First we had to do the SetInitialReferences Method this is making sure the script is referencing the Master scripts. It is a different layout for referencing the Game Manager Master script because it is not part of the same GameObject that the player_health.cs will be attached to so we must tell it to find the game object it is attached to.
In the OnEnable we are starting the SetInitialReferences method and we are setting the UI which will be the health value on our screen. We are also letting it know the events that will be used and in the OnDisable method we are telling the events to stop.
In the DeductHealth method we are declaring an integer value of how much our health will change. It also says that if the player's health is less than or equal to 0 to call the game over event from the Game Manager Master Script. Finally, we set the UI to show the relevant information.
The IncreaseHealth method is like the Increase Health method in that we declare the integer, we say the health value cannot go over 100 and then we reflect this in the UI, this method was left over from an earlier idea of putting in health pickups but I ultimately decided against this.
The last method in this script is the SetUI which is just setting the value on the user's health.



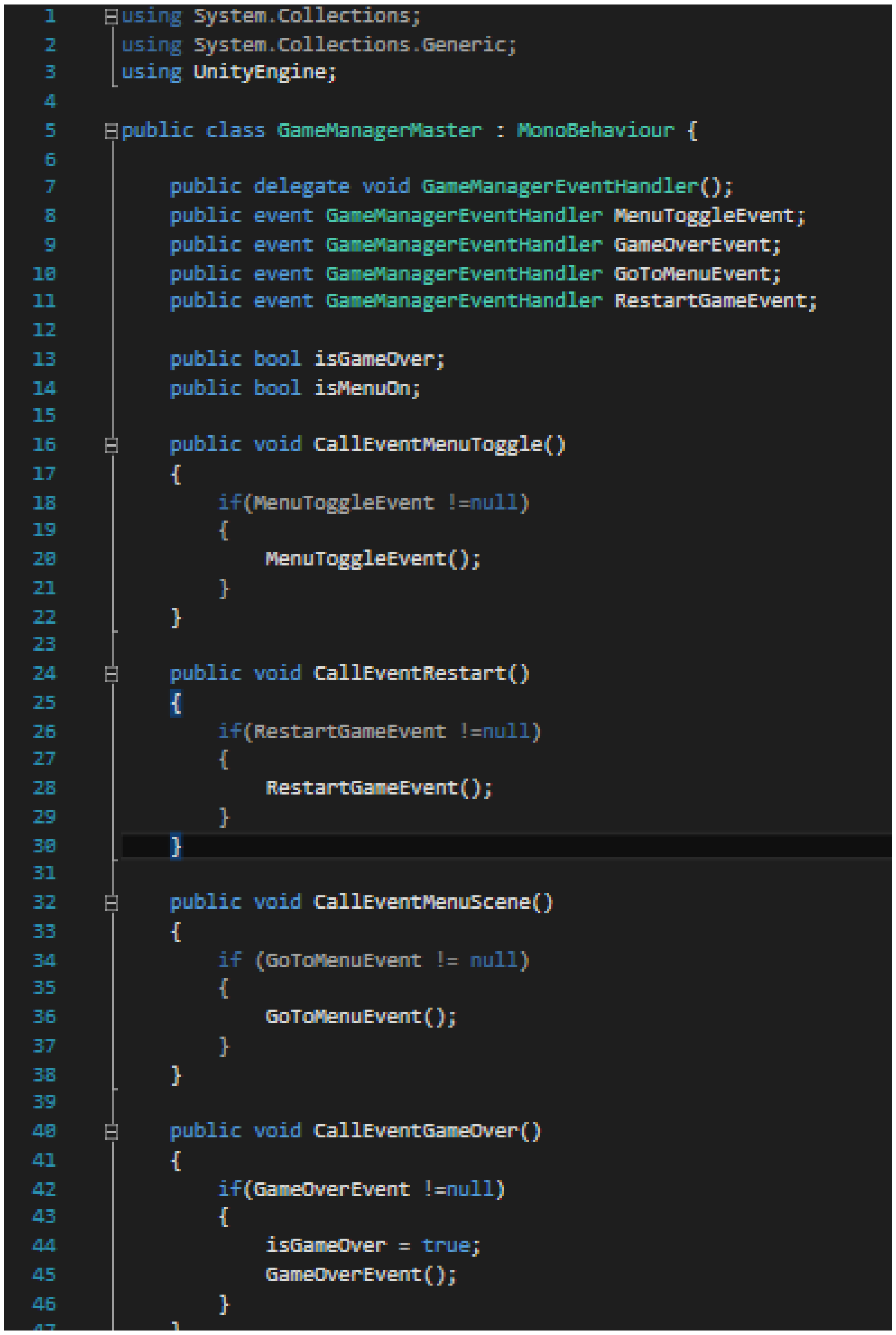
```csharp
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class Player_Health : MonoBehaviour {

    private GameManagerMaster gameManagerMaster;
    private PlayerMaster playerMaster;
    public int playerHealth;
    public Text healthText;

    // Use this for initialization
    void OnEnable ()
    {
        SetInitialReference();
        SetUI();
        playerMaster.EventPlayerHealthDeduction += DeductHealth;
        playerMaster.EventPlayerHealthIncrease += IncreaseHealth;
    }

    // Update is called once per frame
    void OnDisable ()
    {
        playerMaster.EventPlayerHealthDeduction -= DeductHealth;
        playerMaster.EventPlayerHealthIncrease -= IncreaseHealth;
    }

    void Start()
    {

    }

    void SetInitialReference()
    {
        gameManagerMaster = GameObject.Find("GameManager").GetComponent<GameManagerMaster>();
        playerMaster = GetComponent<PlayerMaster>();
    }

    void DeductHealth(int healthChange)
    {
        playerHealth -= healthChange;

        if(playerHealth <=0)
        {
            playerHealth = 0;
```



```csharp
            gameManagerMaster.CallEventGameOver();
        }

        SetUI();
    }

    void IncreaseHealth(int healthChange)
    {
        playerHealth += healthChange;

        if(playerHealth>100)
        {
            playerHealth = 100;
        }

        SetUI();
    }

    void SetUI()
    {
        if(healthText != null)
        {
            healthText.text = playerHealth.ToString();
        }
    }
}
```

GameManagerMaster.cs

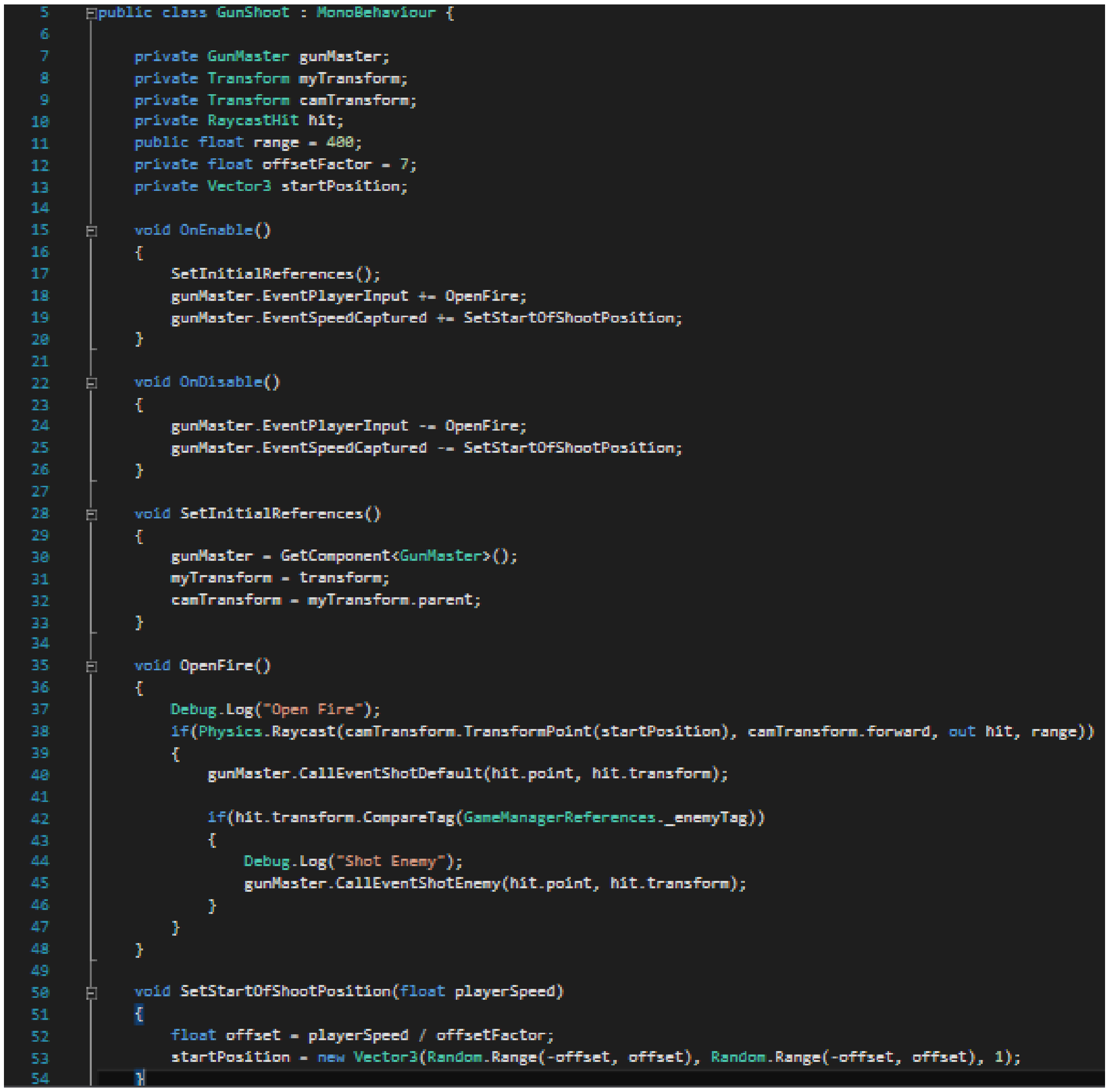
The Game Manager Master Script is where the location of the game over, restart game, main menu and pause menu events are held. This had the event handler in it and on all the other scripts that would need to call these events will call the Game Master Script at the start. There are two Boolean variables being called here ‘isMenuOn’ and ‘isGameOver’, these are used for calling the pause and game over menu respectively. You can see the ‘isGameOver’ Boolean being called in the CallEventGameOver method which checks to see if the player has died and it is game over and then if it is ‘isGameOver’ will be set to true.



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class GameManagerMaster : MonoBehaviour {

    public delegate void GameManagerEventHandler();
    public event GameManagerEventHandler MenuToggleEvent;
    public event GameManagerEventHandler GameOverEvent;
    public event GameManagerEventHandler GoToMenuEvent;
    public event GameManagerEventHandler RestartGameEvent;

    public bool isGameOver;
    public bool isMenuOn;

    public void CallEventMenuToggle()
    {
        if(MenuToggleEvent !=null)
        {
            MenuToggleEvent();
        }
    }

    public void CallEventRestart()
    {
        if(RestartGameEvent !=null)
        {
            RestartGameEvent();
        }
    }

    public void CallEventMenuScene()
    {
        if (GoToMenuEvent != null)
        {
            GoToMenuEvent();
        }
    }

    public void CallEventGameOver()
    {
        if(GameOverEvent !=null)
        {
            isGameOver = true;
            GameOverEvent();
        }
```

GunShoot.cs

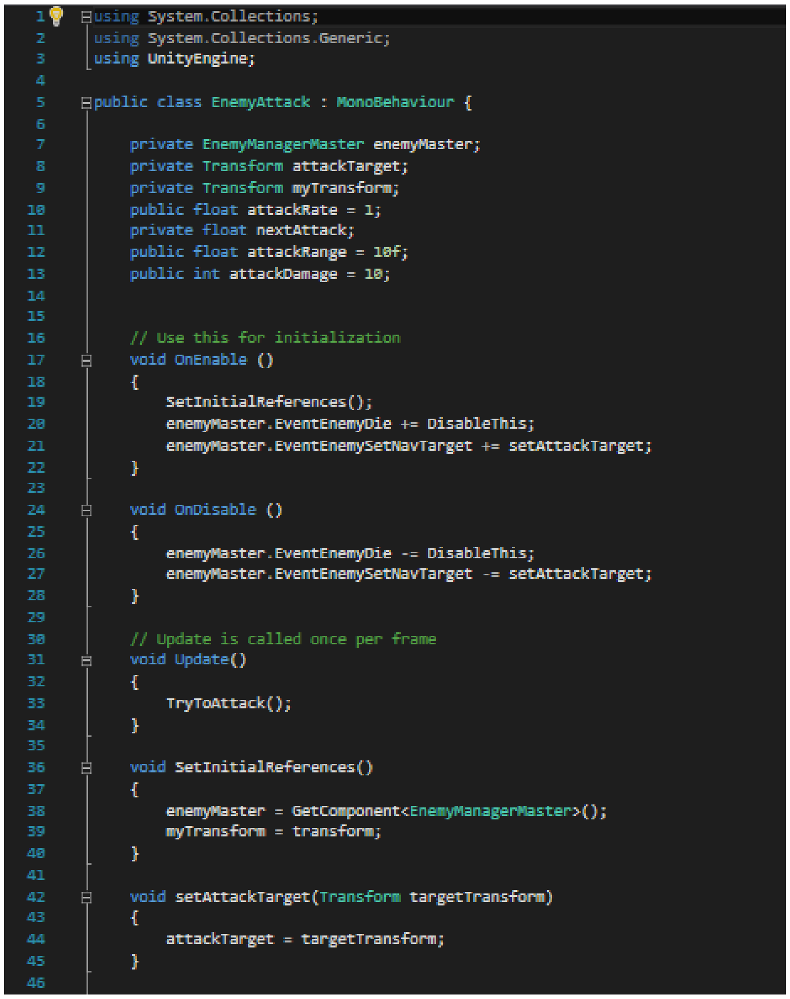
The Gun Shoot script is what allows the gun to shoot in the game. The GunShoot is quite simple, at the start we are calling multiple variables that will be used aswell as the Gun Master Script. We declare two different Transforms, our RaycastHit which is how out gun fires, the range of the gun and then you have the offsetFactor and the startPosition which are where the Raycast will fire from.
We have our SetInitialReferences method again which we reference the Gun Master script, one of our Transform variables and our camTransform which will find the camera attached to the player. Out OnEnable method we are starting the SetIntialReferences method and we are letting it know the method to start when the OpenFire method starts and the next line is doing the same with the SetStartOfShootPosition method.
In the OpenFire method we start with a Debug method which was used for testing purposes, we then set the raycast for where it will fire from, the direction, the hit detection and the range. We must check for if the raycast hit anything. We also need to check that if it hit something with the _enemy tag which we applied to our enemy and defined in the Game Manager References Script, we receive a debug to say we shot it.

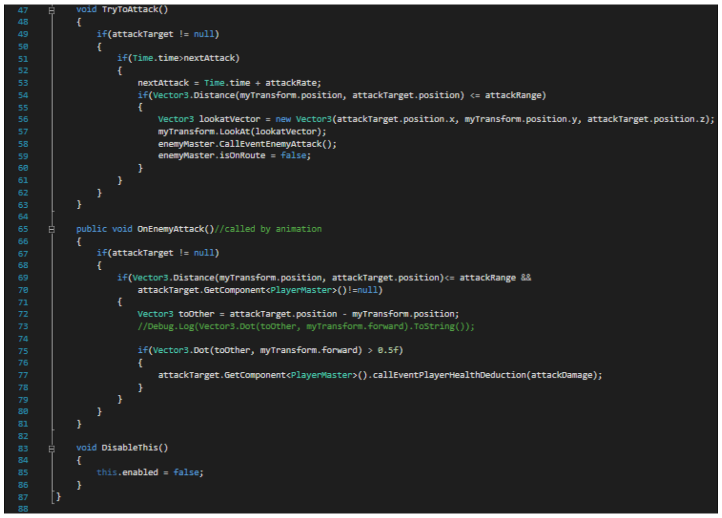


```
public class GunShoot : MonoBehaviour {

    private GunMaster gunMaster;
    private Transform myTransform;
    private Transform camTransform;
    private RaycastHit hit;
    public float range = 400;
    private float offsetFactor = 7;
    private Vector3 startPosition;

    void OnEnable()
    {
        SetInitialReferences();
        gunMaster.EventPlayerInput += OpenFire;
        gunMaster.EventSpeedCaptured += SetStartOfShootPosition;
    }

    void OnDisable()
    {
        gunMaster.EventPlayerInput -= OpenFire;
        gunMaster.EventSpeedCaptured -= SetStartOfShootPosition;
    }

    void SetInitialReferences()
    {
        gunMaster = GetComponent<GunMaster>();
        myTransform = transform;
        camTransform = myTransform.parent;
    }

    void OpenFire()
    {
        Debug.Log("Open Fire");
        if(Physics.Raycast(camTransform.TransformPoint(startPosition), camTransform.forward, out hit, range))
        {
            gunMaster.CallEventShotDefault(hit.point, hit.transform);

            if(hit.transform.CompareTag(GameManagerReferences._enemyTag))
            {
                Debug.Log("Shot Enemy");
                gunMaster.CallEventShotEnemy(hit.point, hit.transform);
            }
        }
    }

    void SetStartOfShootPosition(float playerSpeed)
    {
        float offset = playerSpeed / offsetFactor;
        startPosition = new Vector3(Random.Range(-offset, offset), Random.Range(-offset, offset), 1);
    }
```

EnemyAttack.cs

The Enemy Attack Script is one of the longer scripts in this project and is where we determine when the enemy will attack and how much damage they will do. First we are declaring our variables as per usual, we call the Enemy Manager Master script, we have our transform variables, one private float and three more public variables which are how fast the enemy will attack, what the range of the attack will be and damage that will be done by the attack.
The first thing we do is set up the SetIntialReferences script which once again will call the Enemy Manager Master Script and set the myTransform variable. The OnEnable method does the SetInitialReferences like the other scripts, the next line says that if the enemy is dead event is active to disable this script on that enemy and it also says that when the event that sets the navigation target is active that that target is also the attack target. The OnDisable method does the reverse of these two. The update method has the TryToAttack method running within. The TryToAttack method will make the attack animation play and keep the enemy focused on the player. The OnEnemyAttack method is setting the outcome of the event we applied on the animation in Unity. This says that if the enemy is close enough to the player it will attack and it will make sure that the Player Master script is attached to the player. We then set the direction so that the enemy must be facing the player when they attack. If these parameters are met, then the event for Player Health Deduction is called. Finally, you have the DisableThis method which just tells the system to stop running whatever it is attached to

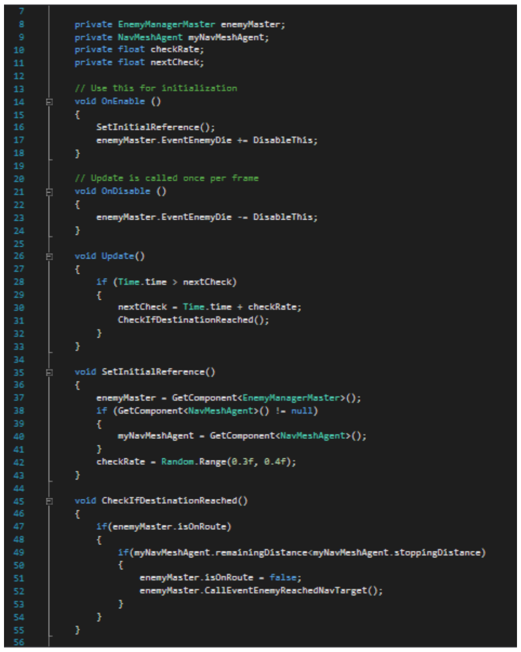

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class EnemyAttack : MonoBehaviour {

    private EnemyManagerMaster enemyMaster;
    private Transform attackTarget;
    private Transform myTransform;
    public float attackRate = 1;
    private float nextAttack;
    public float attackRange = 10f;
    public int attackDamage = 10;


    // Use this for initialization
    void OnEnable ()
    {
        SetInitialReferences();
        enemyMaster.EventEnemyDie += DisableThis;
        enemyMaster.EventEnemySetNavTarget += setAttackTarget;
    }

    void OnDisable ()
    {
        enemyMaster.EventEnemyDie -= DisableThis;
        enemyMaster.EventEnemySetNavTarget -= setAttackTarget;
    }

    // Update is called once per frame
    void Update()
    {
        TryToAttack();
    }

    void SetInitialReferences()
    {
        enemyMaster = GetComponent<EnemyManagerMaster>();
        myTransform = transform;
    }

    void setAttackTarget(Transform targetTransform)
    {
        attackTarget = targetTransform;
    }

```



```
    void TryToAttack()
    {
        if(attackTarget != null)
        {
            if(Time.time>nextAttack)
            {
                nextAttack = Time.time + attackRate;
                if(Vector3.Distance(myTransform.position, attackTarget.position) <= attackRange)
                {
                    Vector3 lookatVector = new Vector3(attackTarget.position.x, myTransform.position.y, attackTarget.position.z);
                    myTransform.LookAt(lookatVector);
                    enemyMaster.CallEventEnemyAttack();
                    enemyMaster.isOnRoute = false;
                }
            }
        }
    }

    public void OnEnemyAttack()//called by animation
    {
        if(attackTarget != null)
        {
            if(Vector3.Distance(myTransform.position, attackTarget.position)<= attackRange &&
                attackTarget.GetComponent<PlayerMaster>()!=null)
            {
                Vector3 toOther = attackTarget.position - myTransform.position;
                //Debug.Log(Vector3.Dot(toOther, myTransform.forward).ToString());

                if(Vector3.Dot(toOther, myTransform.forward) > 0.5f)
                {
                    attackTarget.GetComponent<PlayerMaster>().callEventPlayerHealthDeduction(attackDamage);
                }
            }
        }
    }

    void DisableThis()
    {
        this.enabled = false;
    }
}
```

EnemyNavDestination.cs

The Enemy Nav Destination scripts tells the AI to move towards its target. At the beginning of the script we are declaring our variables as with all our previous scripts we call the master script which for this is Enemy Manager Master, we then define our NavMeshAgent which when created in unity decided how fast your AI may go and its stopping distance before the player. We must declare two float variables for the checkRate and the nextCheck. Our SetInitialReferences is slightly different this time, we do the same as we did previously in the first line in that we reference the Enemy Manager Master script. In out OnEnable method we start the SetIntialReferences Method and we tell the script that if the enemy dies to kill this script. The update method is telling the script to check if the destination is reached, this ensures that the enemy will know if the player has moved away and if it needs to revert to pursuing. The last part of this script will check to see if the enemy has reached the target by comparing the remaining distance of the NavMeshAgent again the stopping distance set on the NavMeshAgent, if this is correct and the enemy is not moving it will call the event EnemyReachedNavTarget.



```

    private EnemyManagerMaster enemyMaster;
    private NavMeshAgent myNavMeshAgent;
    private float checkRate;
    private float nextCheck;

    // Use this for initialization
    void OnEnable ()
    {
        SetInitialReference();
        enemyMaster.EventEnemyDie += DisableThis;
    }

    // Update is called once per frame
    void OnDisable ()
    {
        enemyMaster.EventEnemyDie -= DisableThis;
    }

    void Update()
    {
        if (Time.time > nextCheck)
        {
            nextCheck = Time.time + checkRate;
            CheckIfDestinationReached();
        }
    }

    void SetInitialReference()
    {
        enemyMaster = GetComponent<EnemyManagerMaster>();
        if (GetComponent<NavMeshAgent>() != null)
        {
            myNavMeshAgent = GetComponent<NavMeshAgent>();
        }
        checkRate = Random.Range(0.3f, 0.4f);
    }

    void CheckIfDestinationReached()
    {
        if(enemyMaster.isOnRoute)
        {
            if(myNavMeshAgent.remainingDistance<myNavMeshAgent.stoppingDistance)
            {
                enemyMaster.isOnRoute = false;
                enemyMaster.CallEventEnemyReachedNavTarget();
            }
        }
    }

```

2.4 Testing

Software testing was a vital step in the development of my game, I needed to make sure each aspect was functioning correctly before moving on to other areas.

Unit Testing

I performed unit testing on my game after I had implemented each function. This allowed me to make sure everything was functioning as intended before moving on, which is vital because if you implement multiple features at one time without testing them individually you could discover an error upon running the game without being able to tell where the error is originating from. I did most testing my using the built-in player in Unity and checking the console to make sure my Debug.Log was displaying the correct information. If the game would not run due to a major error the console would usually point towards the error.

Black-Box testing

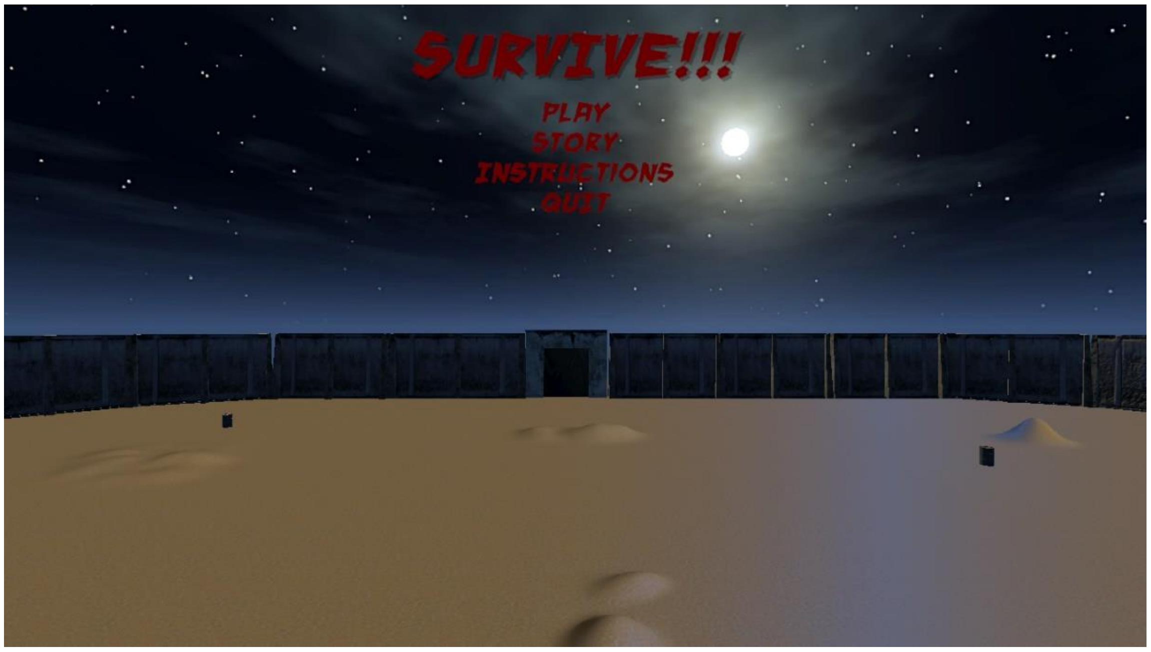
Black –Box testing or Functional testing check what the software that is being tested is currently doing. I used this method of testing closer to the end of my development as I wanted to make sure that I had everything functioning as intended and I hadn't noticed any graphical errors. I gave a beta version of my game to my testers. They had a basic understanding of what is required of the game. When I received the feedback from my testers, I discovered that if the player died and selected restart game it would load but would not allow any movement without the player pausing and unpausing the game. Black box testing helped me find minor errors that I had overlooked upon my own initial testing.

2.5 Graphical User Interface (GUI) Layout

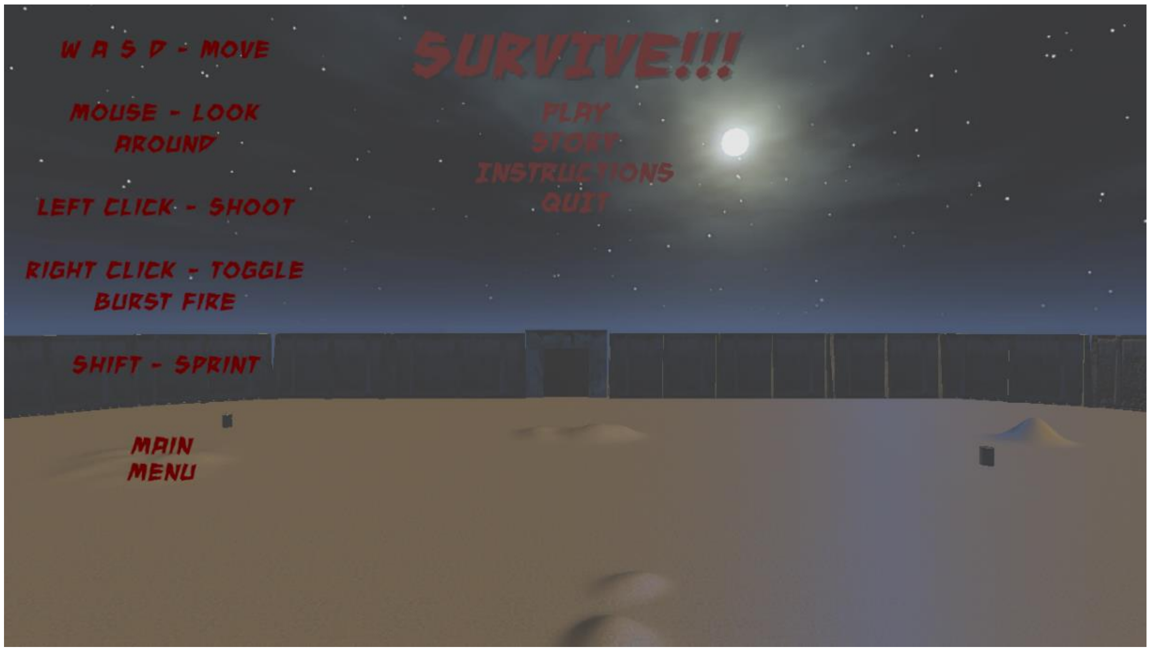
I wanted to make a simple but helpful interface, compared to earlier version which had a large health bar across the top of the screen and a timer in the bottom left corner.

I created a user interface which I created by creating a scene and inserting a terrain and a skybox. I download a package of Skyboxes that I found on the Unity Asset Store called 'Sky 5X', I used the Skybox in this called 'sky5X5'. The player had the ability to move the camera using the FPS controller which meant any mouse movement would adjust the camera. The player can move the mouse around and the crosshair in the center of the screen will direct where the enemy will shoot. There is a gun to the right of the screen to create the visual effect of the user holding a weapon. All text within the game is used using the 'Maneaterbb_bold' font which I downloaded from Fontspace.

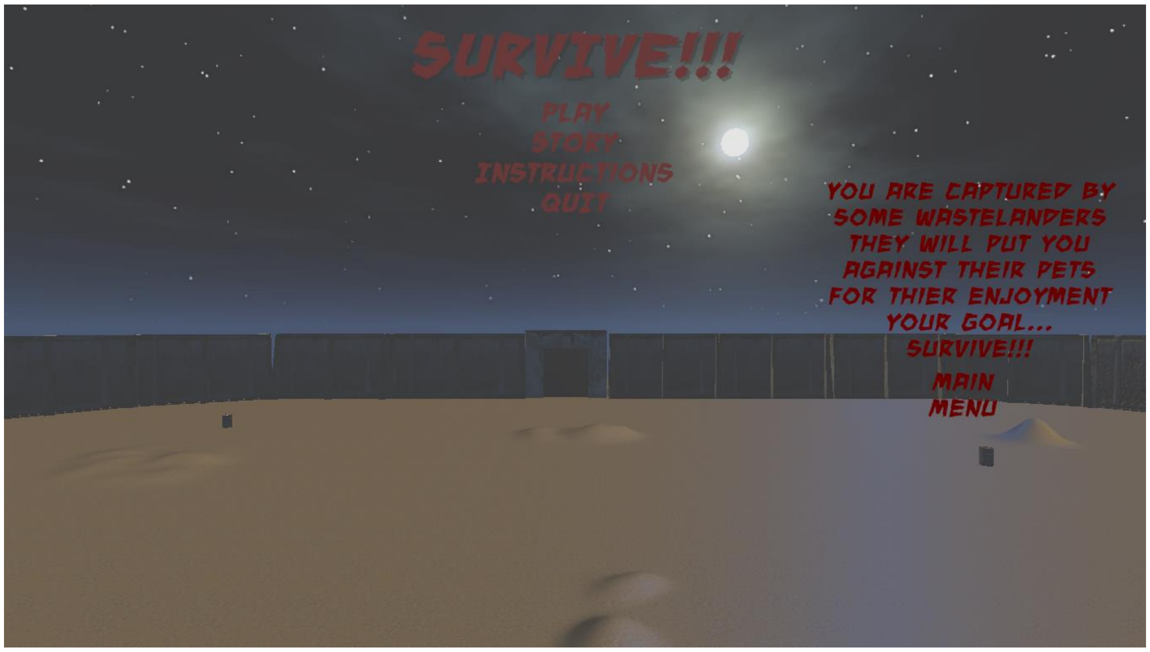
Main Menu



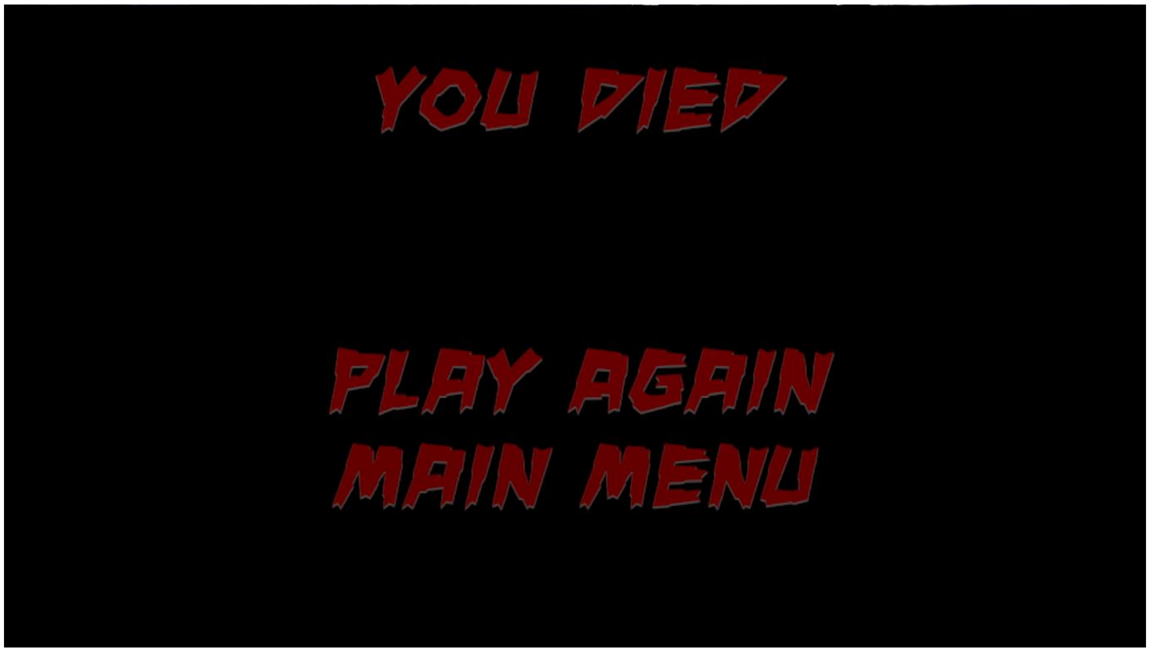
Instructions Menu



Story Menu



Death Menu



2.6 Customer testing

During my testing, I needed feedback on my game from outside testers. I gave a beta version of my game to two main groups of testers. People with programing experience, people without programing knowledge.

Version 1 (This version had basic enemies with a single variant, silent single fire weapons and a low-quality GUI)

The comments suggested that the game was a bit too boring and repetitive. I agreed with this view and I tried to create a more enjoyable experience by adding more enemies of different variants.

The enemy was walking through objects in the environment and through one another. I adjusted the environment design and added Character controllers to the enemies so that they could no longer do this.

Design needed to be improved overall as the silence in the game along with a bright environment did not immerse the player in the game and the bright environment made the game seem too vibrant and positive.

The overall evaluation of this testing was that the game had potential but needed improvement. Adjustments were suggested and what needed to be changed.

Version 2 (some improvements, single fire gun and inconsistencies)

I took the suggestions from the first wave of testing in to account and began to adjust the game and improve it. I added another variant of enemy as a weaker model, this meant I had a weak enemy and a stronger enemy. I also slightly adjusted the lighting so it was darker.

I was given some more feedback after testing on this version of the game. I was told that the Skybox needed an adjustment and that the lighting between the Menu and the Game were not consistent. I found a Skybox that made it look like it was night and I made sure that the environment in both the Menu and in the game were the same.

An error was discovered when the player dies, if the player selected 'Restart Game' the game would reload the level but would not allow the player to move or the enemies would not spawn. I looked into this and found that when the game transitioned from the Death Menu to the Game again it was staying in a pause state. I had to remove a line of code that was causing this error

One of the testers said that the stronger enemies were too difficult to kill and that with the single fire gun they could be not be defeated easily, also that it was too easy to just run from the enemies and not have to fight them.

Version 3 (major improvements)

I took the notes and suggestions from version 2 and I changed the gun so it fires fully automatic and I adjusted the enemy's health so that they could be defeated easier.

The suggestions I received at this stage of the game were that the enemy spawn rates should be adjusted so that they are not spawning at one time and to speed the enemies up as they were much slower than the player, I also added a Special enemy that will spawn every 90 seconds that is easy to defeat but can kill the player in one hit. I did adjust the spawn rate so that the enemies would spawn at different times it also added some unpredictability to the game. I have received good feedback from the testers that they are enjoying the game and find it both fun and challenging.

2.7 Evaluation

I have been constantly testing my game by running it in unity and making sure that there were no errors showing up in the console. This has resulted in a final product that I am happy with.

I had multiple testers that I had instructed to try and 'break the game' by pushing it to force errors and thankfully there was nothing major found. Overall I received good feedback that it was an enjoyable game and they would recommend it.

3 Conclusions

Overall I really enjoyed developing this project, I gained a greater understanding of C# and using Unity which I had not previously used before developing this project. I enjoyed seeing the project come together and designing the different aspects of the game. Design and User Experience is something I enjoy doing in my spare time so getting to do it for a game was an enjoyable experience. My biggest regret was that I did not get as much functionality in to the game as I would have liked. I had 2 very busy semesters in college and then got stuck on a certain game breaking bug which consumed more time than I would have liked.

Despite this, I do feel like I have learned a lot about time management and splitting my time between projects and other work. I have developed a huge appreciation for all game developers as this project has made me realise how difficult it can be.

4 Further development and research

I would like to develop the game further to have a built-in Cloud Leaderboard and I would also like to port the game over to a console. I feel like the game would work well on an Xbox or PlayStation as a fun arcade game. This would improve the game and increase the player base of the game.

The code used in the development of this game could be ported between many different games I could develop in the future. It is code that could be universally useful

5 References

Xenosmash Games. (2013). YouTube channel. Available: https://www.youtube.com/channel/UCKkQs1aEpGoboaTjoxyvGnA.

Gamer to Game Developer. YouTube Channel. Available: https://www.youtube.com/user/GamerToGameDeveloper.

BurgZerg Arcade. YouTube Channel. Available: https://www.youtube.com/channel/UCOIcO8Lsk1MERedhOnew73A.

6 Appendix

6.1 Project Proposal

6.1.1 Objectives

The game will be an arena based combat game. The goal of the game is to have the player play it and see how far they can get without dying. It will have the player pitted against a variety of enemies all with different levels of health and strength. The aim is for it to be a competition to see how far they can get in the game without dying. The game will get progressively more difficult with each wave of enemies that is completed with the lower weaker enemies being replaced by stronger more powerful enemies.

The player's character, the enemies and the environment will all be 3d. The environment will be an arena so the player has nowhere to escape and must fight the enemies, there will be health regeneration if the player's character does not get hit for a certain time, this means there will be times when the player must flee so they can recover, because of this I will place obstacles so that the player would have to alter their path to survive and recover. It will stick to what is expected of a combat game with fluid fighting and the need to use different tactics against certain enemies.

6.1.2 Background

The background for me doing this project is that I have a huge interest in all aspects of gaming. I have been playing Video Games since I was very young. I have played on all consoles from the early consoles like the Atari to PC and more recently the Xbox One. I enjoy most types of games but my personal favourite type of games would be anything combat related.

My idea came from watching the likes of Twitch and reading articles about people being competitive in gaming, be it Speed Running, Player vs. Player combat or just to see who can get the furthest in a game without losing. The point of this is not to be competitive for money or anything like that just for bragging rights. There are many people who would enjoy this time of game they would enjoy the competitive nature of trying to get the furthest through the game without dying.

I have played many games that would have something somewhat similar in their game, but it would be a side quest/mission to get a better reward to help you advance further in the main game. The big difference between something like that and my own game is that mine is never-ending it will just keep generating new waves of enemies more difficult than the last and will test the hardcore players when it gets to the later levels and the increasing difficulty. The more intense the game gets the more the player must work to survive and employ different tactics to make sure they survive and defeat the enemies.

I will be developing the game to run on Windows as PC gaming is becoming more and more popular and it one of the most accessible platforms to develop on. I may consider progressing the game on to console but I will see how I find the PC development and result of my work before I commit to developing on a different platform. The game will not be a 'heavy' game, the player's PC would not need to be a top gaming PC to run this game successfully, my decision to do this is based on the fact there have been many games I have been unable to play because I did not have a PC or laptop capable of running high intensity games without losing frames, I want my game to be fun and playable by almost anyone.

6.1.3 Technical Approach

My approach will be to research the development engines available for me to develop on (i.e. Unreal Engine 4, Unity). I will need to find out which one would be best suited to developing the game I intend to make.

I will then need to do some research on using the programs. I have never used Unity or Unreal Engine 4 before so this will be a completely new experience for me. I will follow tutorials I find online to learn about the different tools available in each program and how to incorporate certain aspects that I would need. I have already bookmarked a few in my browser so I have easy access to them in the future. Once I have learned these tutorials I can pick and choose what aspects would be most beneficial to me when I am developing my own game.

I will make sure the Hardware I am working on can match the requirements I need for the development program and for running the game I will develop.

6.1.4 Special Resources Required

The special resources I will need for development of this project are a capable laptop that can develop the game and run it.

I will need access to guides on using the program I choose to develop my project on, thankfully there are more than enough available for free online.

6.1.5 Technical Details

I will be using either C++ or C# as my development language. Unreal Engine uses C++ as its primary development, while Unity uses C# as its development language.

6.1.6 Evaluation

I will be continuously testing the game myself, Unity has a built-in simulation function so I can test it to make sure there is no bugs and make sure the game is running smoothly.

When I am content that my game is up to my standards I will get testers to play the game and receive their feedback. I will get testers who are familiar with games and testers who would not play many games. I will take the feedback and make any improvements and fix any bugs that have been found.

6.2 REQUIREMENT SPECIFICATION

6.2.1 INTRODUCTION

PURPOSE

The purpose of this document is to set out the requirements for the development of my Arena Survival game for my final year project. My game is an arena survival game where the player must try and survive and defeat as many enemies as possible without dying.

The intended customers are anyone who enjoys games with a challenge. The game could be used in a competitive sense where friends or gaming communities could compete against one another to try and kill as many enemies as possible without being defeated to try and stay alive longest.

PROJECT SCOPE

The scope of the project is to develop a functional arena combat game. The system shall have a gaming engine with 3D Graphics, Animations, and AI System and logic. Most character and environment models will be developed using Unity and assets downloaded from the Unity Asset Store.

I had conversations with potential players and Dr. Anu Sahni to receive the following requirements:

- Game Environment - The game play area must ensure the player cannot leave the arena.
- User Controlled Character – The player must be able to control the character from a first-person perspective.
- Dynamic Lighting – The arena must be lit by the Sun with Shadows being cast by the objects and Characters.
- Sound Engineering – The game will require sound effects from the player weapons and footsteps. This will make the player feel more immersed in the game.
- Random Elements – The game will need to have random elements in which the opponents will do different random movements upon each play.
- Event Triggered Animations – There will be event triggered animations such as when the enemy changes from being idle to walking or from idle to attack.
- Event Triggers – There will be certain events triggered when something happens in the game such as the Start Menu or the Death Menu.

Definitions, Acronyms, and Abbreviations

UN	Unity
UAS	Unity Asset Store
GUI	Graphical User Interface
AI	Artificial Intelligence
AFM	Adobe Fuse/Mixamo

6.2.2 User Requirements Definition

The requirements I have previously listed outline are what I was told would be some important features that the people I consulted would like to see in the game. For the user to be able to play the game they must have:

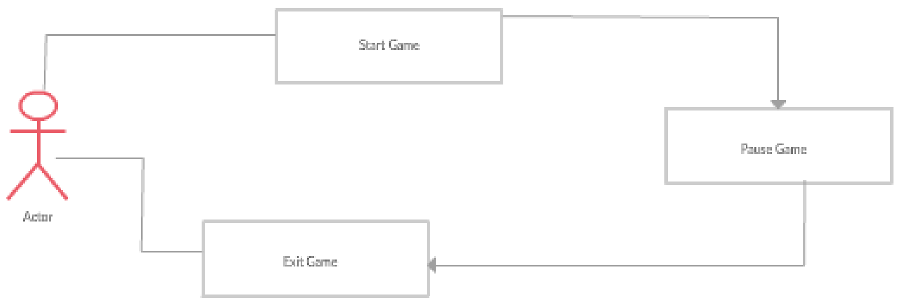
- A functional computer, keyboard and mouse.
- A computer running Windows.
- The game installed on their system.
- Understand the game controls.
- Have a computer that can run DirectX 11

6.2.3 Requirements Specification

6.2.3.1 Functional requirements

Use Case Diagram

This is the Use Case Diagram for the player starting the game, pausing the game and exiting the game.



Requirement 1 <Graphics and Sound Effects>

Description and Priority

The game will need to load the arena. This requirement will make sure the game will load the environment, the user and the enemies. The animations for the AI enemies will load. The sound effects for the players and weapons will also be loaded.

Requirements Activation

The games graphics, AI enemy animations and sound effects will load once the player starts the game.

Technical Issues

Problems could occur with the loading of the graphics and sound effects if the player's system cannot manage the specs of the game.

Risk

Any problems that may occur will be because the hardware of the user computer is not capable of running the game.

Requirement 2 (Gameplay)

Description and Priority

The player will be able to control their character with the use of the keyboard and mouse to move their character and get them to attack. The player will be able to move, run and attack. The user may also pause the game or exit whenever they like.

Requirements Activation

The gameplay requirement will activate once the player starts the game.

Technical Issues

Problems could occur if the player does not move properly or there is input delay on pressing the keys.

Risk

Any problems that may occur throughout the game will be down to an error in the code.

Requirement 3 (AI)

Description and Priority

The game will need to have many different enemies trying to fight the player and defeat them. The enemies will be stronger and more aggressive as the player survives longer in the game. The game must consistently put in more enemies to try and attack the enemy.

Requirements Activation

The enemies will begin to spawn after the player has loaded in to the environment. The delay of the spawn of the next wave of enemies will occur after a set duration of time.

Technical Issues

Problems could occur with the loading of the enemies. The enemies could load in a forbidden location so they are not accessible by the player. The enemies could also spawn in greater numbers than intended.

Risk

Any problems that may occur throughout the game will be down to an error in the code.

6.2.3.2 Non-Functional Requirements

Performance/Response time requirement

When creating a game, the game must not suffer from any graphical or performance problems. The frame rate needs to be consistent and not show any lag. The game must have a good response time so there is no input lag when the player tries to make a move.

Platform requirement

The game must be able to run on a Windows based computer. This will be tested by running the game on my own computer which runs Windows.

Reliability requirement

The game must be reliable and not lag or crash on any system that meets the Hardware Requirements of the game.

Maintainability requirement

The game must be able to be improved after it is released. This means I will be able to patch any bugs or errors that may have been missed during user testing.

Portability requirement

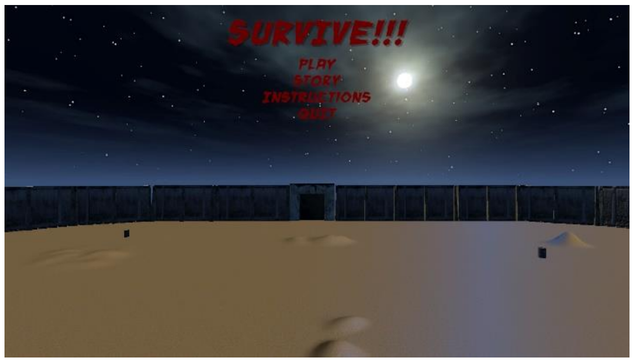
Developing the game using Unity means that in the future it can be cross developed as a web app, Android game or even developed so it is playable on console.

Reusability requirement

Developing a game in the genre or any combat based game means that much of what I used here could be used in that same project. The collision detection, melee attacks, changing weapons could easily be ported across to a similar game.

INTERFACE REQUIREMENTS

This game will require a start menu which will contain instructions on what buttons will do and a pause menu for if the player wants to take a break or exit the game.



6.2.4 System Architecture

Will complete at later stage not sure of exact architecture of game to create a class diagram just yet.

6.2.5 System Evolution

The game could constantly be developed further. This could be done by adding in new environments for the user to play in. I could also add in new enemy types or port it so it is possible to be playable on console.

6.3 MONTHLY JOURNALS

6.3.1 Month: September

My Achievements

This month, I settled on what my idea would be for my final year project, my idea is to create an arena game where the user is a combatant fighting waves of enemies, on every 5-10 round there will be a boss type enemy, the goal of the game is to see how many waves of enemies you can get through without dying as each wave gets progressively harder. I had my Project Pitch. I met with the lecturers to give my proposal of my Gaming Application for my final year project.

My contributions to the projects this month were that I started my Project Proposal and did some research on what I would need to do to complete the project. I have not fully decided whether I will use Unity or Unreal Engine to develop at the moment but I am currently leaning more towards Unreal Engine as it has the built-in Blueprint visual scripting. This will help for prototyping and can make it quite quick to make a visual environment so more time can be spent on the gameplay.

My Reflection

I felt, it worked well to put my ideas in my project pitch slides as it helped to formulate my thoughts and structure my idea more in my own head.

However, I was not successful in beginning my project, I put it off until I got approval from the lecturers when I should've started even just to learn how to use the development programs.

Intended Changes

Next month, I will start the development of my project and create the visual environment.

I realised that I need to put more work into learning about the program I will be using, using Unity or Unreal Engine is a completely new experience to me.

6.3.2 Month: October

My Achievements

This month, I had my Project Pitch, I met with three of my lecturers to go through what I intended on creating for my 4th year project. I explained my idea and how I intended to go about creating it. My project was approved which meant I could finalise my Project Proposal and begin to work on the actual project. I was given Anu Sahni as my Supervisor which I was very happy with as that is who I wanted. I had my first supervisor meeting with Anu and she asked would I think about developing the game in VR. I explained the game would not work as a VR game, I talked about how if the development goes well how I would like to expand it to be able to be played on Xbox One as an arcade style game.

My contributions to the project this month were that I finished my Project Proposal document and submitted it, this may still be altered as the submission is not final.
I also worked through a couple of tutorials for my projects. They covered a lot of what I want to incorporate into my project and gave me new ideas on what I can add to my game. I started to develop some small pieces in Automax 3DS max that I can import into Unity when I need.

My Reflection

I felt, that working on the tutorials gave me a huge insight into what I will need to do and the difficulty of developing my project. They also gave new ideas and have contributed to the skills I will need for development. I also created some objects boxes, barrels, etc. in Autodesk 3DS Max that I will need further into the UI development for my game.

Intended Changes

Next month, I will work on my Project Requirements and work more on the UI for my project and by the end of November I want to have the Project Prototype almost complete with only minor design changes needed.

6.3.3 Month: November

My Achievements

This month, I did not get to work too much on my project until later in the month as I had multiple projects and Continuous Assessments due throughout November and December, one of my projects for Computer Graphics and Animation was useful for my final year project. In the module, our project was to create a game that must incorporate gameplay, a leaderboard and use multiple input devices. I will need a leaderboard in my final year project so this was a useful feature to learn about to add to my final year project.

When I had completed some of the projects for my other modules and I began to work on my final year project. I used Adobe Fuse with Mixamo to create some character models for my Prototype and possibly to use in my final project. I also downloaded a Unity Asset for the landscape for the environment design.

The models being used may be changed for the final project they are just added in for the prototype to demo the concept of the game.

My contributions to the project this month were that I created some character models and the environment for the project. I also added an animation controller for my character models so they can move around the environment. I put some more work into my Mid-Point Technical Report which was due in early December.

My Reflection
I felt, that I did not get to put in the work into my final project that I would have liked, as my workload for project and CA's was extremely high and required a lot of my free time which I would have spent on my final project. I was pleased I got some of my prototype complete and had some character models created through Mixamo.

Intended Changes
Next month, I will focus primarily on my finishing my prototype and Technical Report in time for the Mid-Point Presentation. Once I complete my final exam I will put all my efforts in to the final design of my project and my final report.

6.3.4 Month: December

My Achievements
This month, I worked on my Mid-Point presentation which was due on the 19th of December. I worked on the Mid-Point Technical Report which required some information from my previous Project Requirements Specification document and some new information which I had to add. This is a living document so it may change over the course of the development of my project.
I added in an environment and character animations. I also created my Start menu of my project which gives an option to 'Start Game' and 'Exit Game', I found a tutorial online to assist me in doing this as I was unsure of what process I needed to follow.
On the 19th of December I had my Mid-Point Presentation with my Supervisor Anu Sahni and Catherine Mulwa. I had created a PowerPoint Presentation with all the information contained for my Mid-Point Presentation, during this presentation I showed a prototype for my project.

My contributions to the project this month were that I added in Character Animations and had my character walking under control of the user. I created a Start Menu for the game which allowed the user to 'Start Game' or 'Exit Game'. I also finished my Mid-Point Technical Report and uploaded it, this document may change before its final upload as it is a living document.

My Reflection
I felt, that my Mid-Point Presentation went well, I got some positive feedback which was good to hear. I was told that the prototype of the game looks well and the overall game has some good potential. I was glad I got my Mid-Point Technical Report completed and uploaded on time. I was glad with some of the work I got done on my project, I was slightly disappointed I did not get any work done on my project after the completion of the Mid-Point Presentation, I had a high workload with other projects being due and my exams starting on the 5th of January.

Intended Changes

Next month, after I complete my final exam I will focus on improving and putting new features in my project and working on the final document.

6.3.5 Month: January

My Achievements

For January, I didn't get to do much work on my final project, I had my exams from the 5th of January to the 12th of January so I did not do any work during that time. I took a break for a few days after I finished my exams to relax after a busy Semester in college before the new Semester began. I was back in college on the 23rd of January and began working on new assignments and on my project, again.
I mostly just went through some more tutorials this month to learn how add in other features I feel like would improve my project. I also did some problem solving on the project why the character is falling through the environment, this took a while to solve but I eventually solved it.

My contributions to the project this month were nothing substantial apart from solving the character collider bug. I did learn some more on how to add new features I feel would improve the overall project.

My Reflection

I wish I had of got more done on the development of my project but I felt like the break I took after my exams was needed and made me feel a lot more relaxed coming in to the final stretch of 4th year. I am glad that I could fix the Character Collider bug as I found that to be extremely frustrating when I first encountered it.

Intended Changes

Next month, I will start to program the AI to be focused on the User's character and actively try to attack it.

6.3.6 Month: February

My Achievements

For February, I made some good progress on my project. I adjusted the title of my game from "Survival" to "Survive", I felt this small change looked better on the Menu Screen. I got the environment created to look apocalyptic, I added colliders into all obstacles and objects in the game so that the player cannot walk through any of them. I added in a story option into the menu along with instruction both which give the user an insight in to how the game is played and why your character is in that situation.
I got my character running, turning and striking with both kicks and punches which is essential for the end product of my game. I created one of my enemy zombie characters which will actively hunt down and attack the player.

My contributions to the project this month made up for all the missed time to work on the project in the past. I got a lot of stuff added from Menu options and character controls to animations and my environment.

I had two supervisor meetings with Anu this month, In the first meeting I showed her what progress I had made on my project to that date, I was told to focus on getting my AI functioning in the next week.

My Reflection
I feel like I made up for some lost time during the last couple of months but I need to keep my motivation to keep working away on my project to hit my deadline successfully. I do feel however I made some good progress which has reduced the required workload I had.

Intended Changes
Next month, I will get my project up to a testing stage so that I can use some product testers to see what bugs are encountered and solve said issues.

6.3.7 Month: March

My Achievements
For March, I didn’t get too much work done on my project. It was my final month of classes and I had a lot of Continuous Assessments projects and in class tests. When I got to the end of the month when I had finish all the tests and projects I broke my ankle and required surgery on it, this unfortunately meant I had another week and a half where I could not do any project work.

My contributions to the project this month were minimal I did however decide to change my game from a Third Person melee combat to a First-Person Shooter, I felt that this change would enhance the overall immersion and experience within the game.

I had one supervisor meeting with Anu this month, I was scheduled to have a second but unfortunately due to me being in hospital I was unable to attend.

My Reflection
Unfortunately, I got little to no work done on my project this month as I had a high workload with module work and then my injury towards the end of the month.

Intended Changes
Next month, I will get my project completed, I must focus and put in a lot of hard work to get my project to an acceptable standard but I feel like I can do it.