National College of Ireland

BSc in Computing

2014/2015

Conor Greenan

X13503827

X13503827@student.ncirl.ie

WAFT RUN

Technical Report

National College of Ireland

Table of Contents

Executive Summary

Waft Run is classed as a first person indie adventure game where the player controls a character that aims to navigate the environment collecting chimes in a limited time frame. The player will come in to contact with enemy AI and will need to defend themselves. The environment is structured in a way that supports fluid movement. The player will have the ability to boost/jetpack. The game is heavily based on movement. In my project the player will use keyboard and mouse/trackpad to control their player. For the best user experience, a mouse is recommended.

With independently-created games (Commonly known as Indie game) rapidly growing the market is very hungry for unique games like Waft Run. It is primary focused at gamers that have a strong interest in addictive/competitive gaming.

The game is built using C# and JavaScript in the MonoDevelop cross platform IDE and is aimed primarily running on Windows PCs. The game is set to be deployed on the Google play store.

1 Introduction

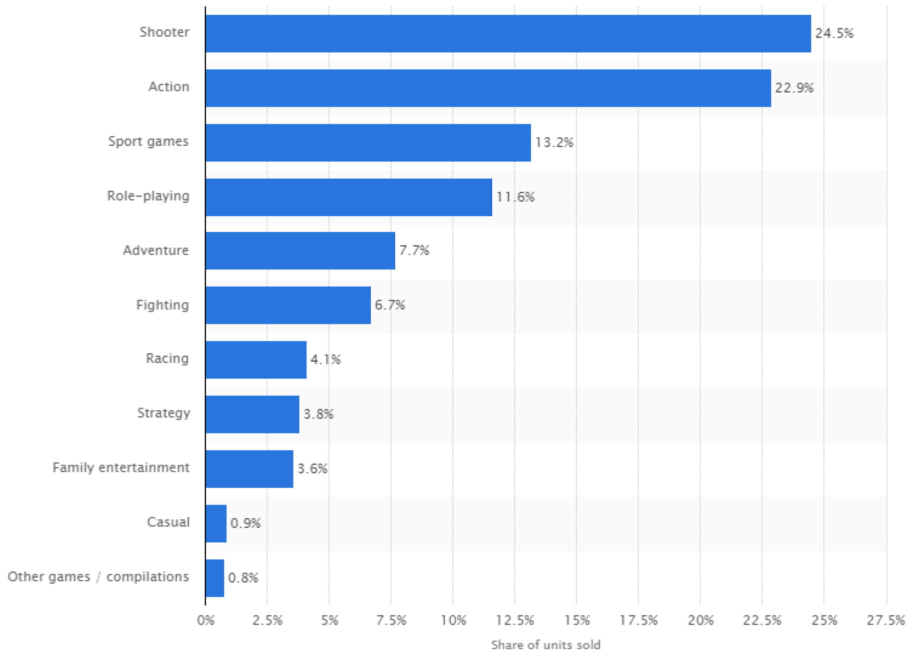
1.1 Background

My project is an 3D indie game. Unity3D was voted #2 best game engine in 2015, covers multiple industry areas and is free to download. I was debating between using Unity and Unreal Engine 4 but decided Unity was the best option as I had worked with it before.

The inspiration for this project mainly comes from a popular game named Tribes: Ascend. Tribes: Ascend is a free-to-play first-person shooter developed and published by Hi-Rez Studios The unique difference being that the objectives of the game are focused around movement.

Beginning my project, I aspired to involve a Leap Motion device to control character movement, soon after I began developed I decided the leap motion was giving too much trouble for what it was worth and it was better to focus on gameplay and functionality rather than spend all my time implementing Leap Motion Control.
The reason I wanted to involve some sort of virtual reality is because I am studying Internet of things this year and I find it very interesting. I was recommended to check out Leap Motion by a friend in a similar course to mine in DCU. In the future, I plan to implement some VR technology into the game, as it is well suited

The genres I choose for this game include adventure, racing and action. While researching, I found that very few games had been released for PC gaming in the racing genre (Only about 5% across all platforms) and thought it could be an opening in the market.



Also, you can see from this chart above that shooter and action genres are by far the most popular.

1.2 Aims

The overall aim of this project was to create a fully functional Unity 3D. The aim of the game is to use the input controls to glide your character across multiple sets of terrain in the fastest time possible, defending against any AI obstacles and collect objects that enhance the overall score.

Other functionality aims include:

- Create movements within the game that are appealing and addictive.
- Create a well-designed main menu for the user to navigate through. They will be able to adjust settings to suit their preference. Settings such as sensitivity, field of view and sound will be adjustable.
- Allow functionality for mouse + keyboard.
- The game will have a stylish GUI that will display the players score, time, health and jetpack/boost function.

- ➢ Sound effects and backtrack will be pleasing to the ear and in sync with gameplay.
- ➢ Implement characters developed in Autodesk Maya.

1.3 Technologies

- ➢ Leap Motion controller (Removed from project as explained above) - For this project I planned to use a device called Leap Motion. Leap Motion is a sensor device that supports hand and finger motions as input into unity. It requires no contact once plugged into the computer via USB.

I purchased the leap motion from www.leapmotion.com for 70 euro, this included a stand for the device.

The original aim was to allow the users to use the Leap Motion controller to control their player in the game. Different hand gestures would allow users to manipulate the camera, movement and combat.

- ➢ Unity3D - Unity is a game engine which is used for the development of both 2D and 3d games. The scene view displays everything in the game, from the terrain to the character model. The hierarchy section lists all the GameObjects currently in the scene. The project file is where the animation, audio, materials, prefab and scripts files and help. The inspector controls the details of a GameObject e.g. the gravity, weight etc.

- MonoDevelop - MonoDevelop is an IDE which is installed when you first install Unity. It allows you to create and edit C# and JavaScript files that can be used in my game. MonoDevelop has built-in functions and auto completion with coding that relates to Unity games. The Start() and Update() functions are some of the most important. Start() is called when you first launch the game and Update() is called every frame after it is launched. Accessing GameObjects through MonoDevelop is very helpful. You can interact with other scripts attached, interact with any tags that were given to the game objects and interact with the game objects co-ordinates.

1.4 Structure

Section 1 has described the background details behind Waft Run, the motivates and the technology behind the project.

Section 2 details both the functional requirements and the non-functional requirements of the project. The design and architecture diagrams are shown. The implementation (Including snippets of code), GUI and testing are also described here. The testing process is detailed at the end of this section.

Section 3 describes the conclusions I have regarding my project (including milestones and hurdles faced) and the future development opportunities.

Section 4 shows the References.

Section 5 contains the appendix of previously submitted documents and any necessary further information.

2 System

2.1 Requirements

Since the requirements specification was uploaded I have added 2 more functional requirements, pick up chime and finish race.

2.2 Functional requirements

2.2.1 Requirement 1: Play Game/Movement

Description & Priority

Describes the movement the user should be able to complete in order to play the game to its full extent. Without this requirement the game cannot function. This is top priority.

Use Case

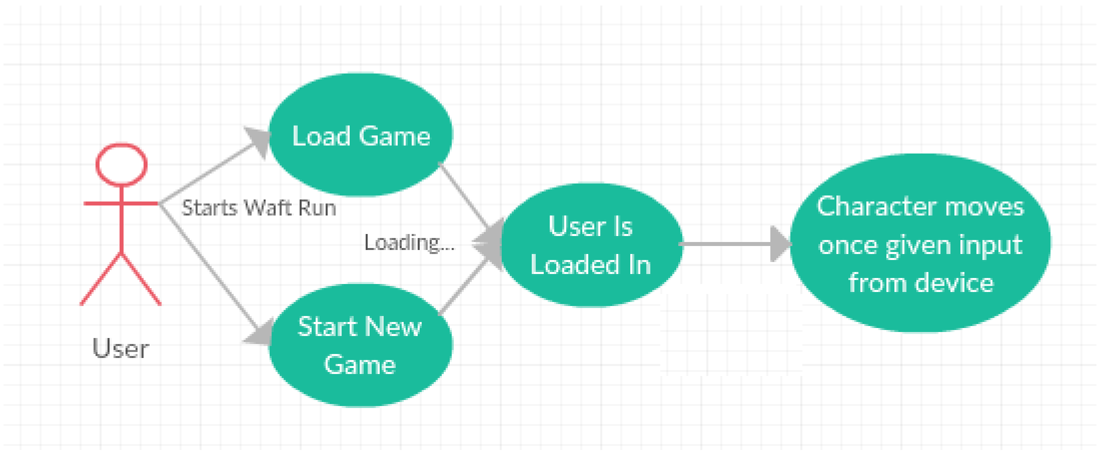
Scope

The scope of this use case is to let the user play the game e.g. move around the terrain freely.

Description

This use case describes the movement that the player must be able to do to play the game.

Use Case Diagram



Flow Description

Precondition

The game is installed on the desktop and the desktop meets the minimum requirements. The Keyboard + Mouse must be available for use.

Activation

This use case starts when the user uses the input device (keyboard and mouse) to make the player move in any desired direction.

Main flow

A1: Input from keyboard and mouse

1. The user selects start/load game.
2. The game loads a previously saved or new level for the user.
3. The user sends input using the keyboard and mouse.
4. The game takes that input, uses it in the playerMovement script and moves the character.

Termination

The process stops when the users stops giving input otherwise the requirement continues.

Post condition

There in so post condition as movement and input from the devices will always be mandatory to play the game.

2.2.2 Requirement 2: Health System

Description & Priority

The player should have a health system shown to the player via the GUI. The player can lose health and if health is zero the player should die.

Use Case

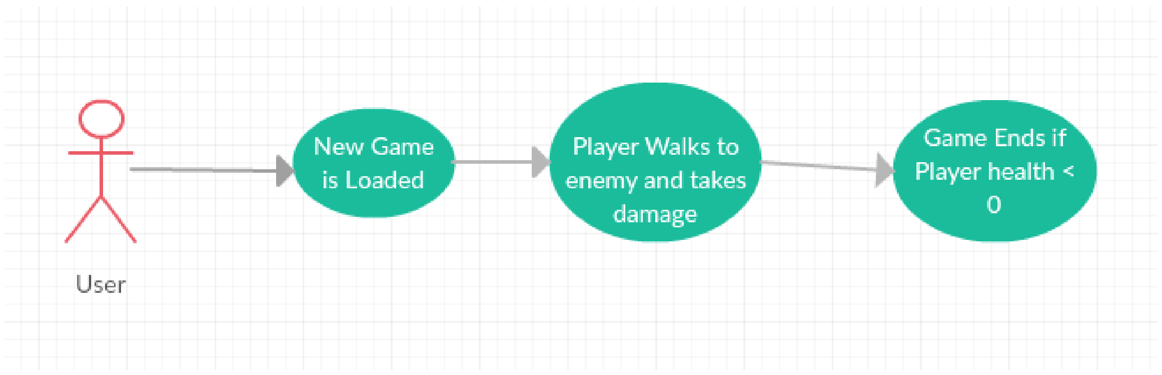
Scope

The scope of this use case allows the user to lose health/die.

Description

This use case describes the process in which player takes damage.

Use Case Diagram


User
New Game is Loaded
Player Walks to enemy and takes damage
Game Ends if Player health < 0

Flow Description

Precondition

The game is installed on the desktop and the desktop meets the minimum requirements. Keyboard + Mouse must be available for use.

Activation

This use case starts when the user has started the game.

Main flow

1. The user starts Waft Run
2. The player walks up to enemy and takes damage
3. The player loses part of health
4. The player continues as health is still greater than 0

Alternate flow

A1: Health = 0

1. The user starts Waft Run
2. The player takes damage from the enemy
3. The

Termination

The system takes the player to the main menu to continue the game.

Post condition

The use case is finished and the game is idle

2.2.3 Requirement 3: Restart Game

Description & Priority

The user restarts the game. This is used in the game that the user does not want to save their progress but wants to restart a fresh game.

Use Case

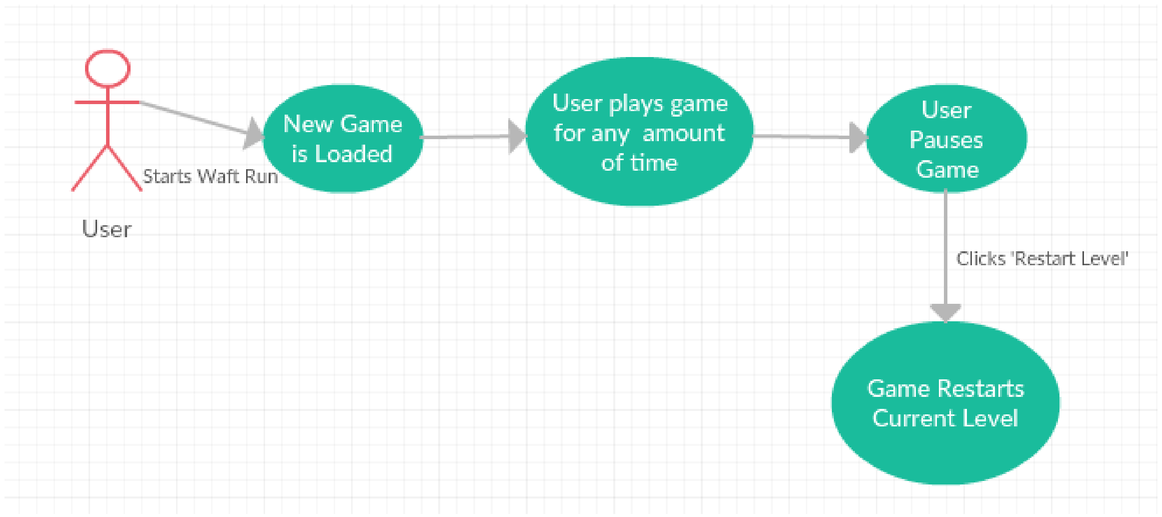
Scope

The scope of this use case is to allow the user to restart their game

Description

This use case describes the process by which the player restarts the game

Use Case Diagram



Flow Description

Precondition

The game is installed on the desktop and the desktop meets the minimum requirements. Either or both the Keyboard + Mouse must be available for use.

Activation

This use case starts when the user starts the game

Main flow

1. The user starts the game
2. The user creates a new save
3. The user plays for any amount of time
4. The user pauses and hits restart
5. The system takes the user to the start point and resets their score.

Alternate flow

A1: Restart after 1-hour play time

1. The user starts the game
2. The user begins a new says and plays for > 1 hour
3. The user pauses and hits restart
4. The user hits 'No'

Termination

The system presents the next screen to the user.

Post condition

The game has restarted with no faults.

2.2.4 Requirement 4: Interact with Game System

Description & Priority

The user picks up a boost. The boost function in the game is integral to the overall experience.

Use Case

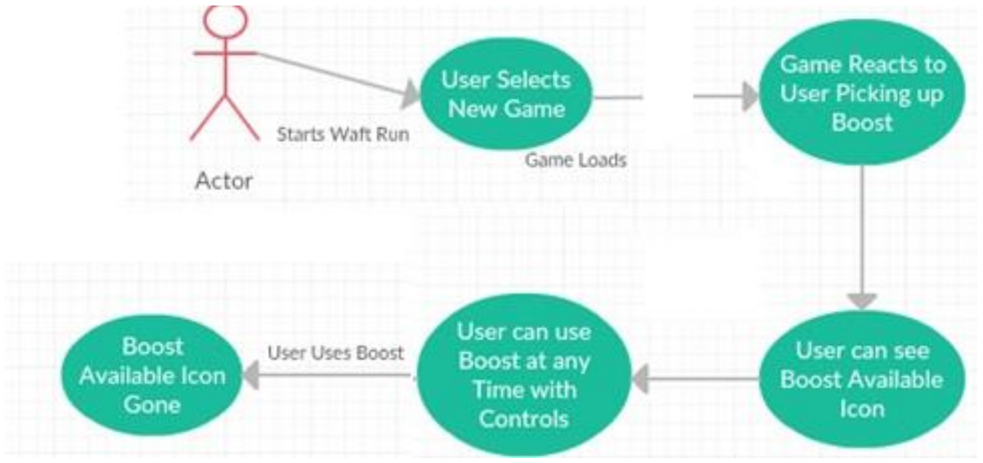
Scope

The scope of this use case is to allow the player to pick up and use the boost

Description

This use case describes the process of picking up and using the boost.

Use Case Diagram


Actor
Starts Waft Run
User Selects New Game
Game Loads
Game Reacts to User Picking up Boost
User can see Boost Available Icon
User can use Boost at any Time with Controls
User Uses Boost
Boost Available Icon Gone

Flow Description

Precondition

The game is installed on the desktop and the desktop meets the minimum requirements. Either or both the Keyboard + Mouse must be available for use.

Activation

This use case starts when the user starts the game

Main flow

1. The user starts the game
2. The user loads a new/previously saved level.
3. The user navigates to a boost and picks it up
4. The user uses the boost
5. The system applies speed boost

Alternate flow

A1: No boost remaining

1. The boost picked up runs out of energy
2. The user cannot boost or gain additional score anymore
3. The game continues until the user picks up another boost

Termination

The system presents the next screen to the user

Post condition

User is back to its original state.

2.2.5 Requirement 5: Exit Game

Description & Priority

The user exits the game. This is used either in-game or on the main menu. This is important for the user to close the game.

Use Case

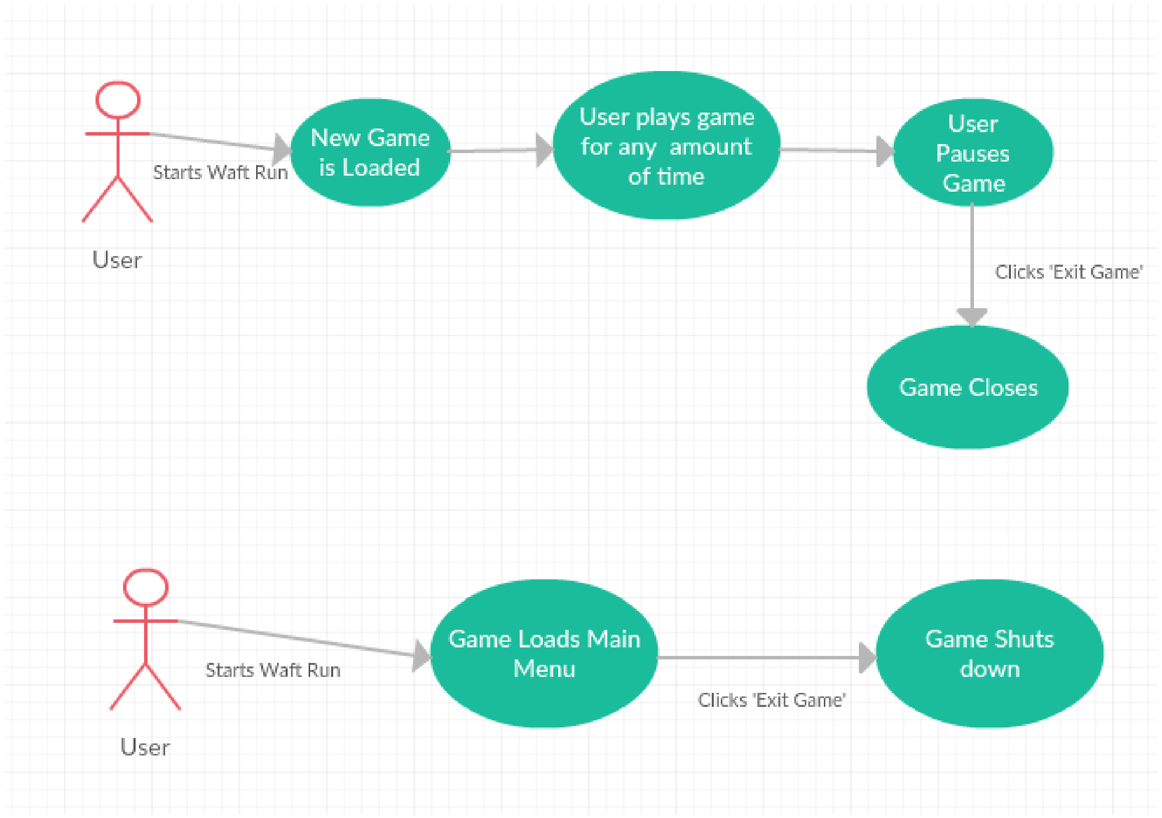
Scope

The scope of this use case is to allow the user to exit the game

Description

This use case describes the process in which the user exits the game

Use Case Diagram



Flow Description

Precondition

The game is installed on the desktop and the desktop meets the minimum requirements. Either or both the Keyboard + Mouse must be available for use.

Activation

This use case starts when the user starts the game.

Main flow

1. The user starts the game
2. The user clicks ‘Exit Game’
3. The game closes.

Alternate flow

A1: In-Game Exit

1. The user starts the game
2. The user creates new game
3. The user pauses the game
4. The user clicks ‘Exit Game’

5. The game closes

Termination

The system closes.

Post condition

The system closes.

2.2.6 Requirement 6: Pick up Chime

Description & Priority

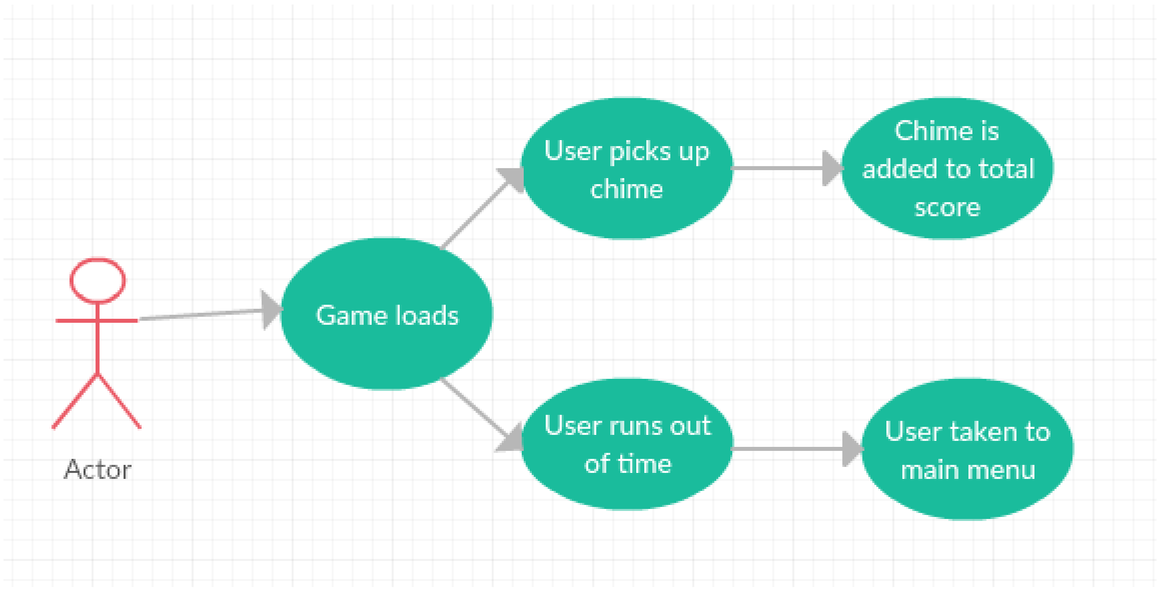
This use case describes the process in which users can collect chimes in which will add to their score.

Use Case

Scope

The scope of this use case is to allow the user to pick up a chime. The chime is added to the users score

Use Case Diagram



Flow Description

Precondition

The user has entered the game.

Activation

This use case starts when the user enters the game

Main flow

1. The user starts the game
2. The user navigates to a chime
3. The user collects the chime
4. The chime is added to the users score

Alternate flow

1. The user starts the game
2. The is too slow and the time runs out
3. The user is taken to the main menu

Termination

The user collects a chime or the time runs out

Post condition

There is no post condition

2.2.7 Requirement 7: Finish

Description & Priority

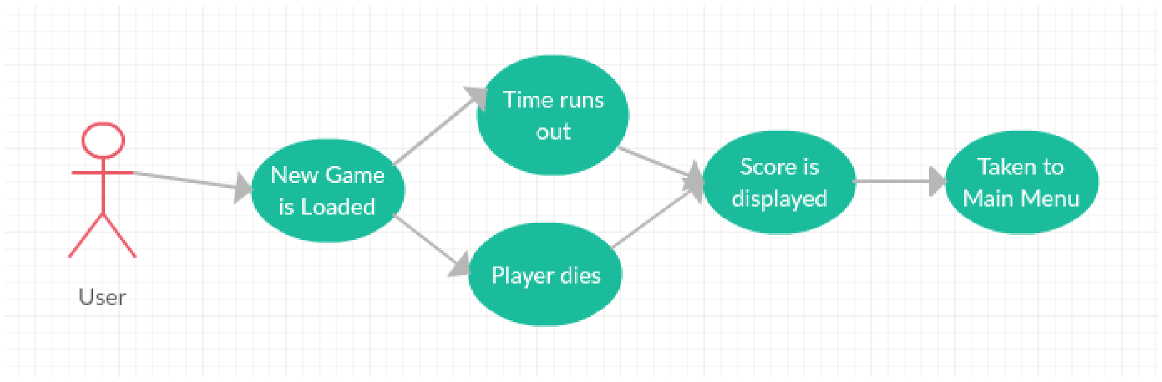
The user runs out of time or dies, the game will end and proceed to the main menu.

Use Case

Scope

The scope of this use case is to allow the user to finish the race.

Use Case Diagram



Flow Description

Precondition

The user has started the game.

Activation

The use case starts when the user is in the game

Main flow

1. The user starts the game
2. The user navigates to collect chimes
3. The users time runs out
4. The users score is displayed
5. The game takes the user to the Main menu

Alternate flow

1. The user starts the game
2. The user dies from enemy character
3. The game takes the user to the main menu

Termination

User is taken to the main menu

Post condition

User is at the main menu

2.3 Data requirements

To download, the game requires a maximum of 300mb of storage space from the users computer.

The in-game data is passed in the background of the game These variables keep information such as the player score, health, time, and position. They also keep information about other GameObjects in the scene, all until the game ends.

2.4 User requirements

The user requirements are as follows:

- The user's desktop machine must meet the minimum requirements for the Unity 5 engine. These are:
    - ➢ Windows 7, 8, 10; Mac OS X 10.8+. Windows XP and Vista are not supported.

    - Graphics card: DX9(shader model 3.0)
    - CPU: Dual-Core Intel or AMD, 2.0GHz or faster.
    - 6GB RAM.
    - Desktop, Mac or laptop

- For the best experience the user's desktop machine should meet the recommended requirements for Unity 5. These are:
    - Windows 7, 8, 10; Mac OS X 10.8+. Windows XP and Vista are not supported.
    - CPU: Quad-Core Intel or AMD, 2.5GHz or faster.
    - 8GB RAM.
    - Desktop
- The user must have a keyboard and mouse.
- The user must have a fully built version of the game.
- The user must have sufficient space on their hard drive to download the game.

2.5 Environmental requirements

To create this project I require a game engine, the game engine I have chosen is Unity 3D. MonoDevelop was installed along with Unity. This is the IDE I will be using to write the scripts for the game. They will be written in C#.

Blender will be used to create simple in-game objects such as the chime pick-up.

Internet access was necessary to access specific resources such as assets, tutorials and to download the software stated above.

2.6 Non-functional Requirements

2.6.1 Performance/Response time requirement

Usually when creating a game performance and game response are very important so that the user does not experience any graphical or gameplay delay. The game will need to run at a consistent frame rate of 30-60fps to ensure smoothness. The user should be able to modify these settings to suit themselves.

2.6.2 Availability requirement

The game will be developed on the Desktop. Due to this there is a possibility that it can be realised on Google play store. The user is limited to using just their mouse and keyboard. Google play has a very large customer base in the PC market.

Access to the game download will only be unavailable if maintenance is being done.

2.6.3 Recover requirement

In the event of a crash only the current game data will be lost, since this is at most 45 seconds of game time it should not be a big deal.

2.6.4 Security requirement

The user will need to be authorised to use the proper security privileges that will allow them to first download the game and then to run the game.

2.6.5 Reliability requirement

Once the game has been downloaded on the users machine it is expected to be available at all times.

2.6.6 Maintainability requirement

If released on Google play the game will be subject to online patches that will fix and bugs and update game graphics and functionality. Players can report bugs on their Google account.

2.6.7 Portability requirement

The game is to be developed as a desktop application. Unity also gives the possibility to create a web player that can run on any site. In the future the game is subject to be developed on Android devices or even console devices.

2.6.8 Reusability requirement

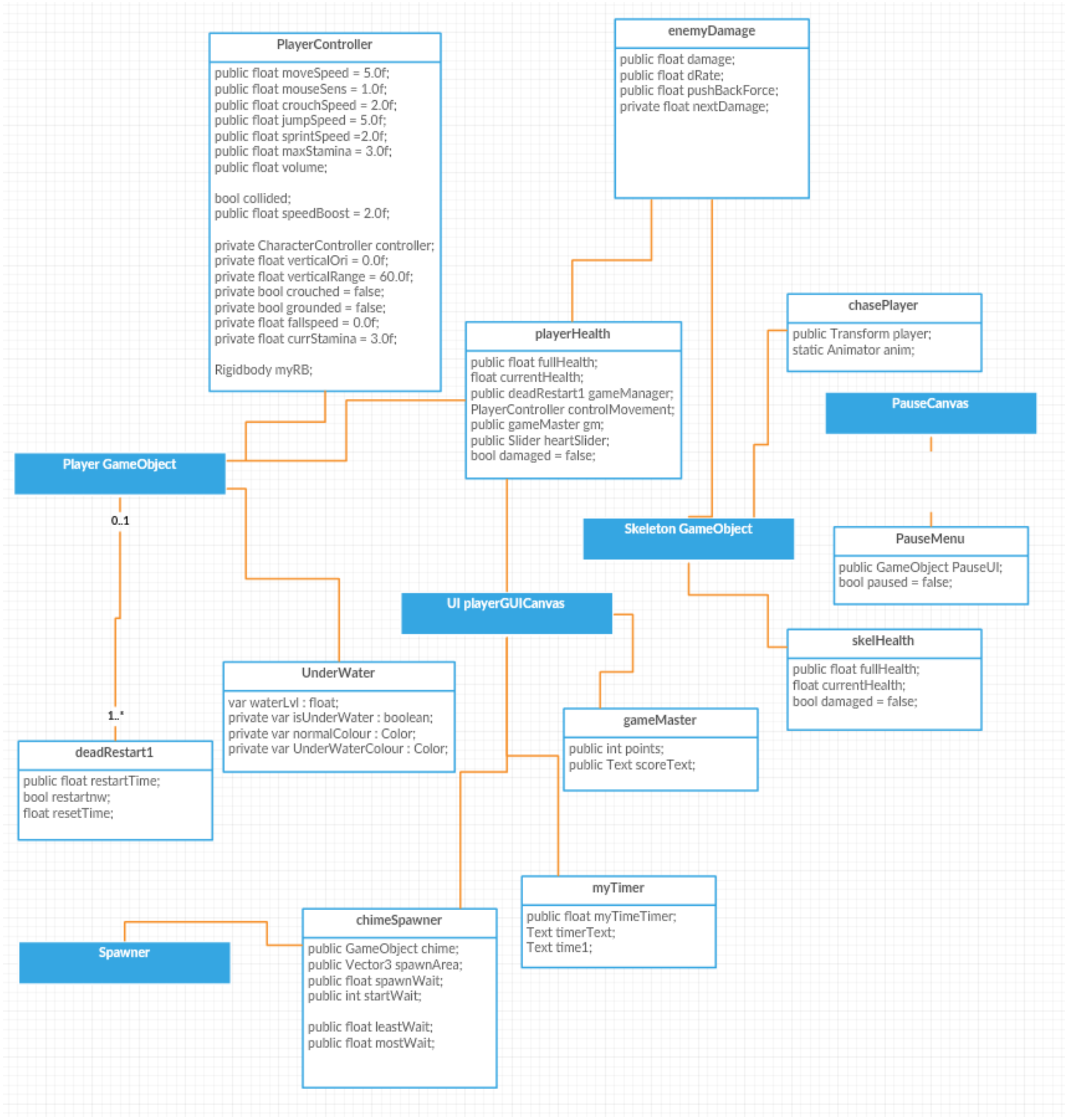
Many features in this game will be reusable in other 3D games that can be developed. Many of the scripts such as player movement, camera movement and terrain scripts can be reused.

2.6.9 Resource utilization requirement

The game will use as much resources as it requires to run at optimal performance, unless the user has specified differently in-game. This will ensure stability.

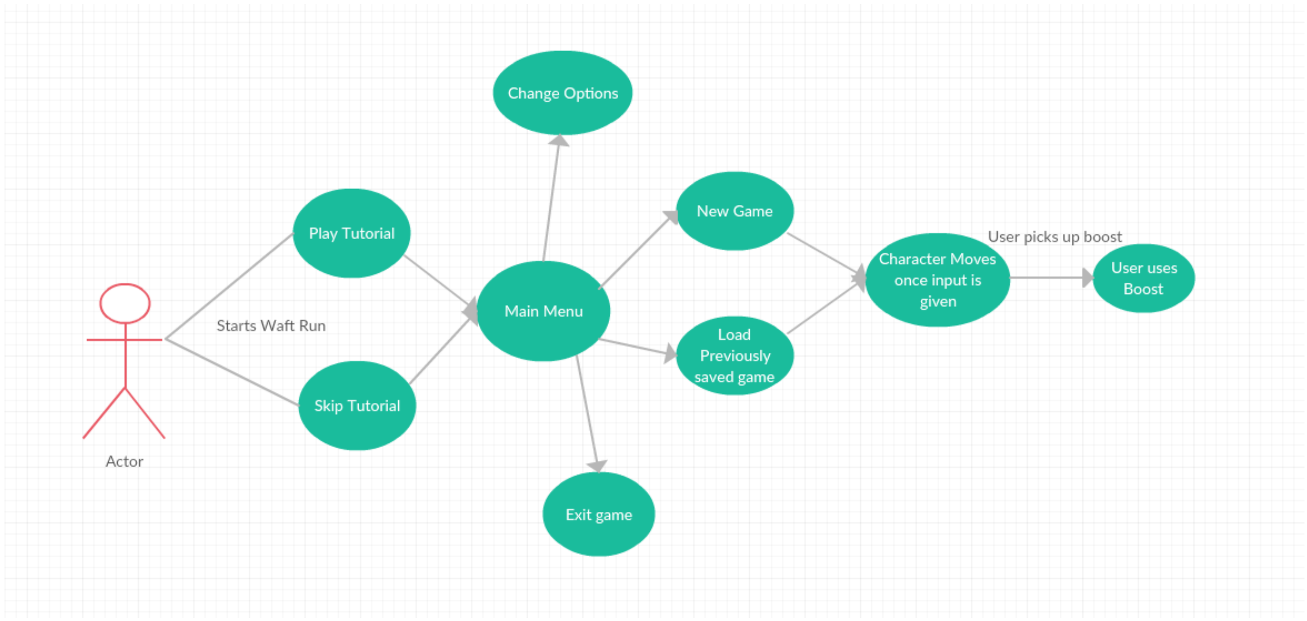
2.7 *Design and Architecture*

2.7.1 Class Diagram


PlayerController
public float moveSpeed = 5.0f;
public float mouseSens = 1.0f;
public float crouchSpeed = 2.0f;
public float jumpSpeed = 5.0f;
public float sprintSpeed =2.0f;
public float maxStamina = 3.0f;
public float volume;
bool collided;
public float speedBoost = 2.0f;
private CharacterController controller;
private float verticalOri = 0.0f;
private float verticalRange = 60.0f;
private bool crouched = false;
private bool grounded = false;
private float fallspeed = 0.0f;
private float currStamina = 3.0f;
Rigidbody myRB;
enemyDamage
public float damage;
public float dRate;
public float pushBackForce;
private float nextDamage;
chasePlayer
public Transform player;
static Animator anim;
playerHealth
public float fullHealth;
float currentHealth;
public deadRestart1 gameManager;
PlayerController controlMovement;
public gameMaster gm;
public Slider heartSlider;
bool damaged = false;
PauseCanvas
Player GameObject
0..1
Skeleton GameObject
PauseMenu
public GameObject PauseUI;
bool paused = false;
UI playerGUICanvas
skelHealth
public float fullHealth;
float currentHealth;
bool damaged = false;
UnderWater
var waterLvl : float;
private var isUnderWater : boolean;
private var normalColour : Color;
private var UnderWaterColour : Color;
1..*
deadRestart1
public float restartTime;
bool restartnw;
float resetTime;
gameMaster
public int points;
public Text scoreText;
myTimer
public float myTimeTimer;
Text timerText;
Text time1;
Spawner
chimeSpawner
public GameObject chime;
public Vector3 spawnArea;
public float spawnWait;
public int startWait;
public float leastWait;
public float mostWait;

This is my class diagram; it outlines the interact between the developed scripts in my project.

2.7.2 Use Case Diagram



Above is my use case diagram. When the player starts the game the tutorial is launched, unless the user has already completed the tutorial previously. The user is then taken to the main menu where they have the following options: New Game, Change Options, Load previously saved game and Exit game. If new game or load previously saved game is pressed the user is loaded into the game. Exit game will close the application and change options will take the user to another screen where they will have multiple game objects.

2.7.3 Implementation

In this section I will outline the implementation process involved in creating Waft Run in Unity3D.

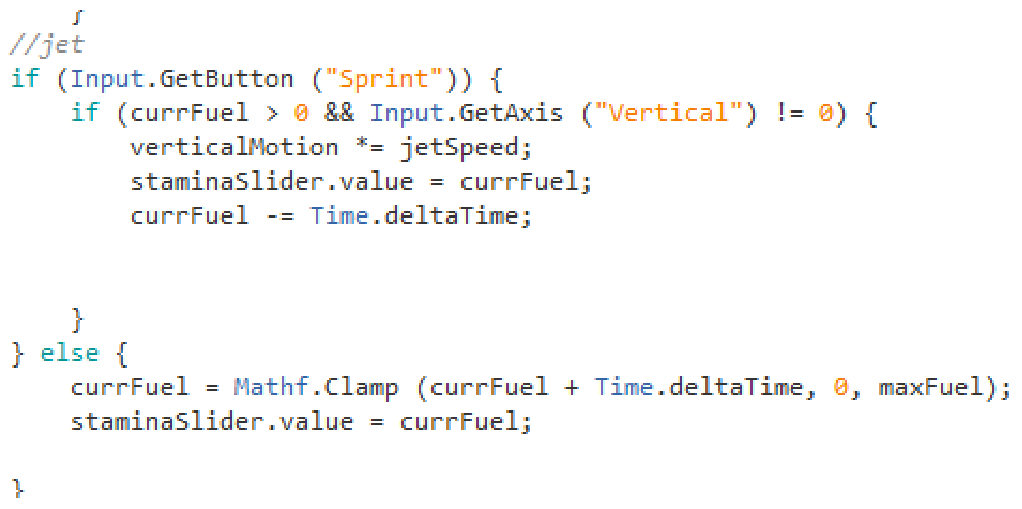
To create the project in Unity I first created the project name. When you select to create a project an option is given to import any Unity assets to aid in development. The only option I choose here was the Environment package. The environment package allowed me to use various already created terrain and object textures. An additional skybox and skeleton character was added from the asset store to aid in development of the game's look.

The main GameObjects involved within the game will now be explained along with some code details.

PlayerController.cs:

This player controller script controls the movement of the main character in the game. It also controls the available mouse movements for the player. The speed when the character is spawned is the moveSpeed.

jetSpeed is used as the jetpack function in game. Below is the code controlling the speed of the jetpack and the jet stamina slider. If the fuel is greater than 0 and the motion is vertical the player will increase speed. When not pressing the sprint button the fuel will regen, up to a max capacity of 10. The stamina slider is attached to display the current stamina in the top right corner below the healthbar.



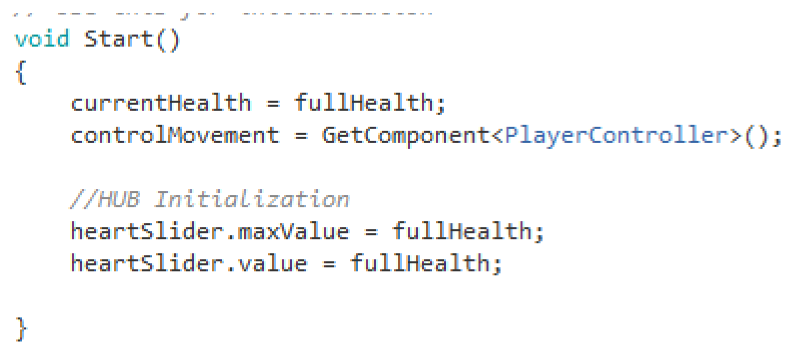
```
    }
//jet
if (Input.GetButton ("Sprint")) {
    if (currFuel > 0 && Input.GetAxis ("Vertical") != 0) {
        verticalMotion *= jetSpeed;
        staminaSlider.value = currFuel;
        currFuel -= Time.deltaTime;


    }
} else {
    currFuel = Mathf.Clamp (currFuel + Time.deltaTime, 0, maxFuel);
    staminaSlider.value = currFuel;

}
```

The jumpSpeed controls how high the character can jump; he can only jump when he on the ground. Once in the air the character falls with increasing gravity. The snapDownToGround variable constantly pulls the character to the ground. If this was not implemented the character would not slide down hills freely, he would walk out and fall down, giving a jagged look to the game.

playerHealth.cs:

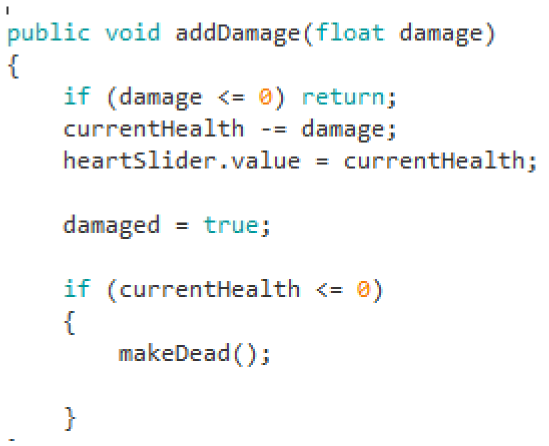
The player health script involves displayed the character's health and allowing the character to take damage. If the character's damage is less than or equal to 0 then the character will be made dead.

In the start function I set the character's health and initialise the health slider.



```
void Start()
{
    currentHealth = fullHealth;
    controlMovement = GetComponent<PlayerController>();

    //HUB Initialization
    heartSlider.maxValue = fullHealth;
    heartSlider.value = fullHealth;

}
```

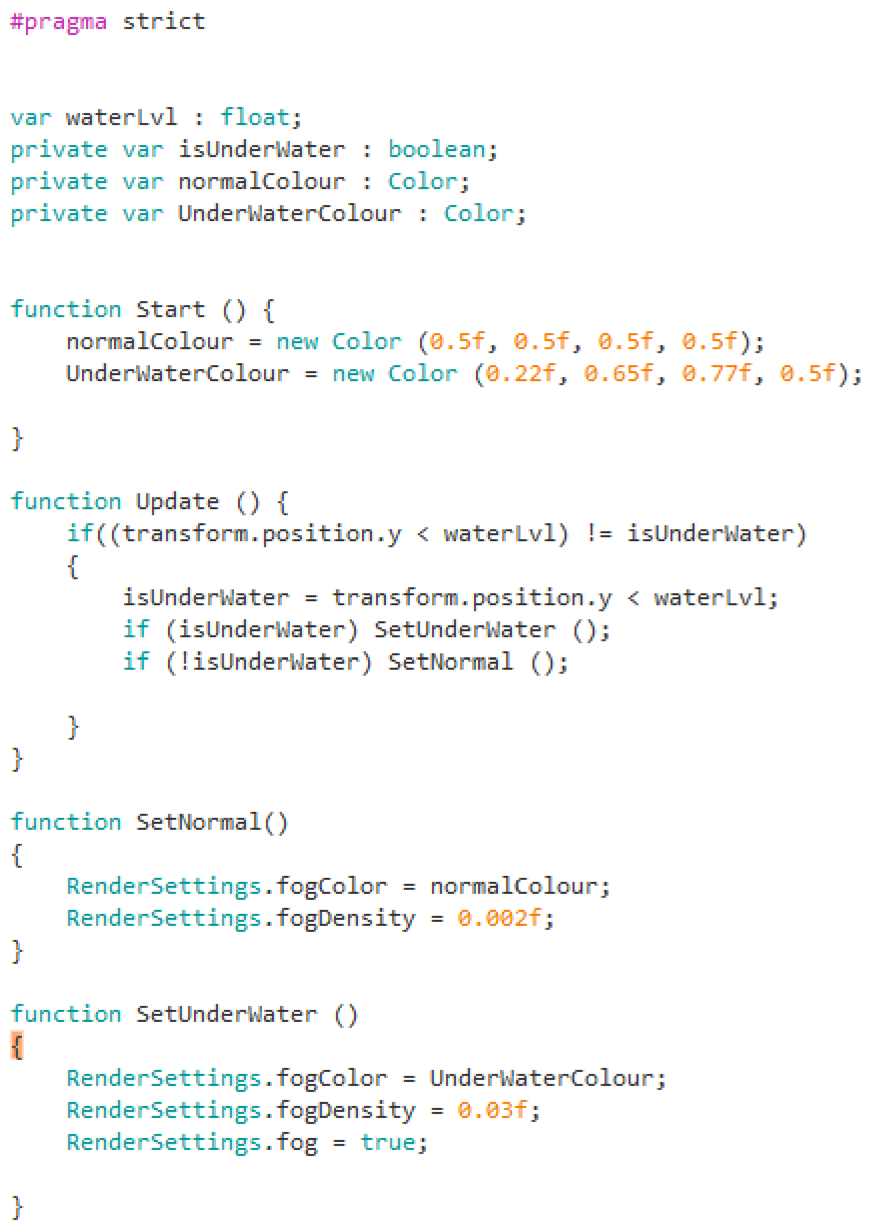
If damage is taken from the enemyDamage script the health bar is updated. The health reaches 0 the makeDead function is called.



```
public void addDamage(float damage)
{
    if (damage <= 0) return;
    currentHealth -= damage;
    heartSlider.value = currentHealth;

    damaged = true;

    if (currentHealth <= 0)
    {
        makeDead();

    }
```

UnderWater.js:

The UnderWater script is the only JavaScript file used in the project. I followed a tutorial online that showed how to make it look as if the character was under water. The code sets the fog to true if the character is below a certain level on the map(i.e. below the water) and tints the fog to a blue colour, giving the under water effect. Since I did not want fog enabled above water I disabled.



```
#pragma strict


var waterLvl : float;
private var isUnderWater : boolean;
private var normalColour : Color;
private var UnderWaterColour : Color;


function Start () {
    normalColour = new Color (0.5f, 0.5f, 0.5f, 0.5f);
    UnderWaterColour = new Color (0.22f, 0.65f, 0.77f, 0.5f);

}

function Update () {
    if((transform.position.y < waterLvl) != isUnderWater)
    {
        isUnderWater = transform.position.y < waterLvl;
        if (isUnderWater) SetUnderWater ();
        if (!isUnderWater) SetNormal ();

    }
}

function SetNormal()
{
    RenderSettings.fogColor = normalColour;
    RenderSettings.fogDensity = 0.002f;
}

function SetUnderWater ()
{
    RenderSettings.fogColor = UnderWaterColour;
    RenderSettings.fogDensity = 0.03f;
    RenderSettings.fog = true;

}
```

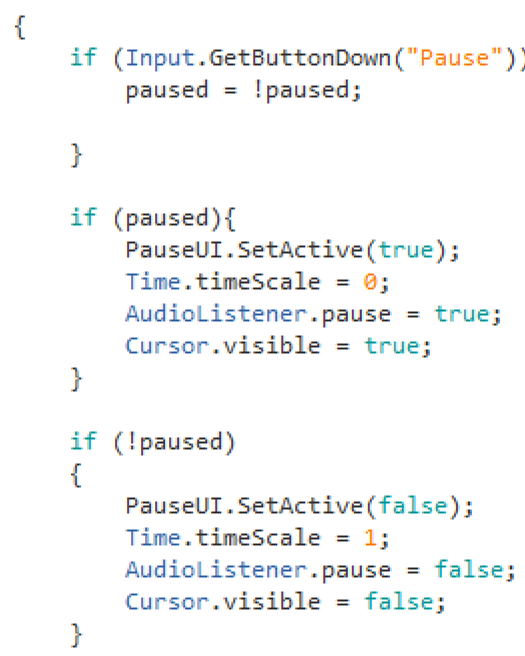
Here is an example of what the fog under water looks like:

deadRestart1.cs:

This script simply does what the title says, if the player is dead restart the game. Restarting the game will reset the health, jetpack fuel, score and time.

PauseMenu.cs:

On starting the game the pause menu is set to false, if the input button for pause is pressed by the user the game is paused. Pausing will set the pause UI equal to true, set the time to stop, stop listening for any audio sounds (as I do not want any in game sounds to play) and make the mouse curser invisible. Is un-paused the game will resume back to the state it was before it was paused. The User also has the option to restart the game and quit the application.



```
{
    if (Input.GetButtonDown("Pause")){
        paused = !paused;

    }

    if (paused){
        PauseUI.SetActive(true);
        Time.timeScale = 0;
        AudioListener.pause = true;
        Cursor.visible = true;
    }

    if (!paused)
    {
        PauseUI.SetActive(false);
        Time.timeScale = 1;
        AudioListener.pause = false;
        Cursor.visible = false;
    }
```

myTimer.cs:

The timer script simply controls the timer on the GUI, counting down from 60 seconds.

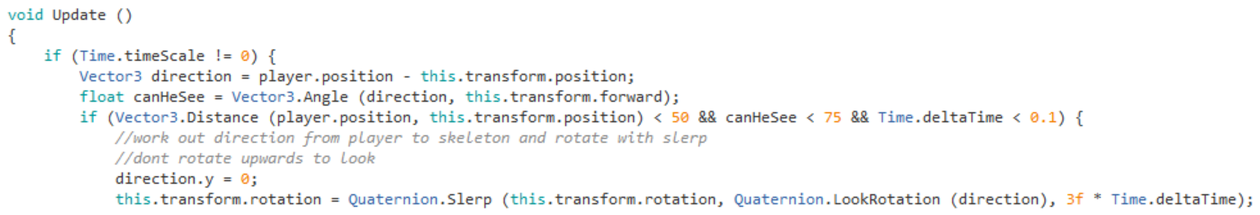
gameMaster.cs:

The game master script takes and updates the points. Then displays it on the scoreText on the GUI.

chasePlayer.cs:

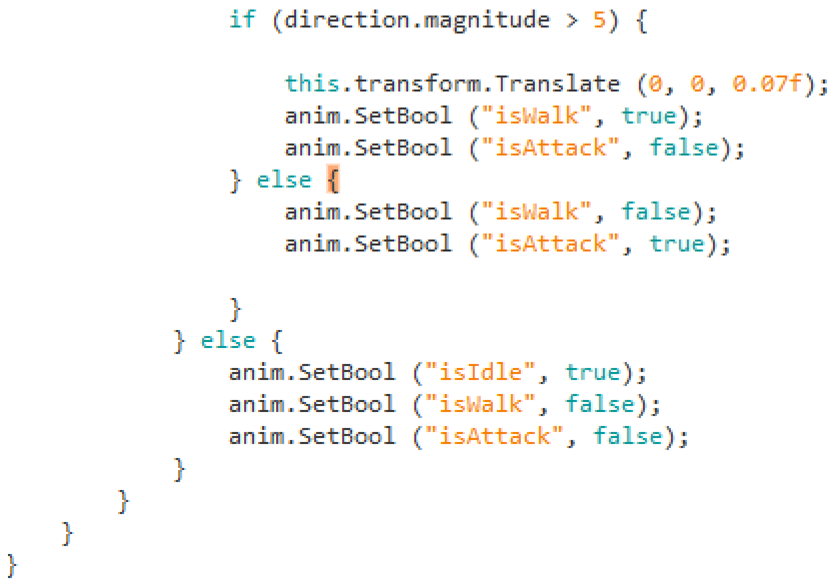
The chase player script is the script that controls the AI. An online tutorial was followed to complete the basic AI script and was added to/altered to for the feel I wanted within my game.

The Code below says if time is not 0 (I.e. if the game is not paused) get the player position. If the player is less than 50 from the AI and the angle the AI can see is greater than 75 then the AI will walk towards the player at a speed of 3. The AI will not look in the Y direction. This is to stop the AI rotating upwards and downwards.



```
void Update ()
{
    if (Time.timeScale != 0) {
        Vector3 direction = player.position - this.transform.position;
        float canHeSee = Vector3.Angle (direction, this.transform.forward);
        if (Vector3.Distance (player.position, this.transform.position) < 50 && canHeSee < 75 && Time.deltaTime < 0.1) {
            //work out direction from player to skeleton and rotate with slerp
            //dont rotate upwards to look
            direction.y = 0;
            this.transform.rotation = Quaternion.Slerp (this.transform.rotation, Quaternion.LookRotation (direction), 3f * Time.deltaTime);
```

The animation is set to idle as default. If the AI more than 5 magnitude from the player and moving the animation will be walk. If the Ai is less than 5 from the player the attack animation will begin.



```
                if (direction.magnitude > 5) {

                    this.transform.Translate (0, 0, 0.07f);
                    anim.SetBool ("isWalk", true);
                    anim.SetBool ("isAttack", false);
                } else {
                    anim.SetBool ("isWalk", false);
                    anim.SetBool ("isAttack", true);

                }
            } else {
                anim.SetBool ("isIdle", true);
                anim.SetBool ("isWalk", false);
                anim.SetBool ("isAttack", false);
            }
        }
    }
}
```

enemyDamage:

The enemy damage script will allow the AI character to damage the player. This will be done with a collider. If the player character collider and the AI collider collide the addDamage function from our player health script will be called. When this function is called, the player loses a percent of their health. The heartSlider is updated accordingly.

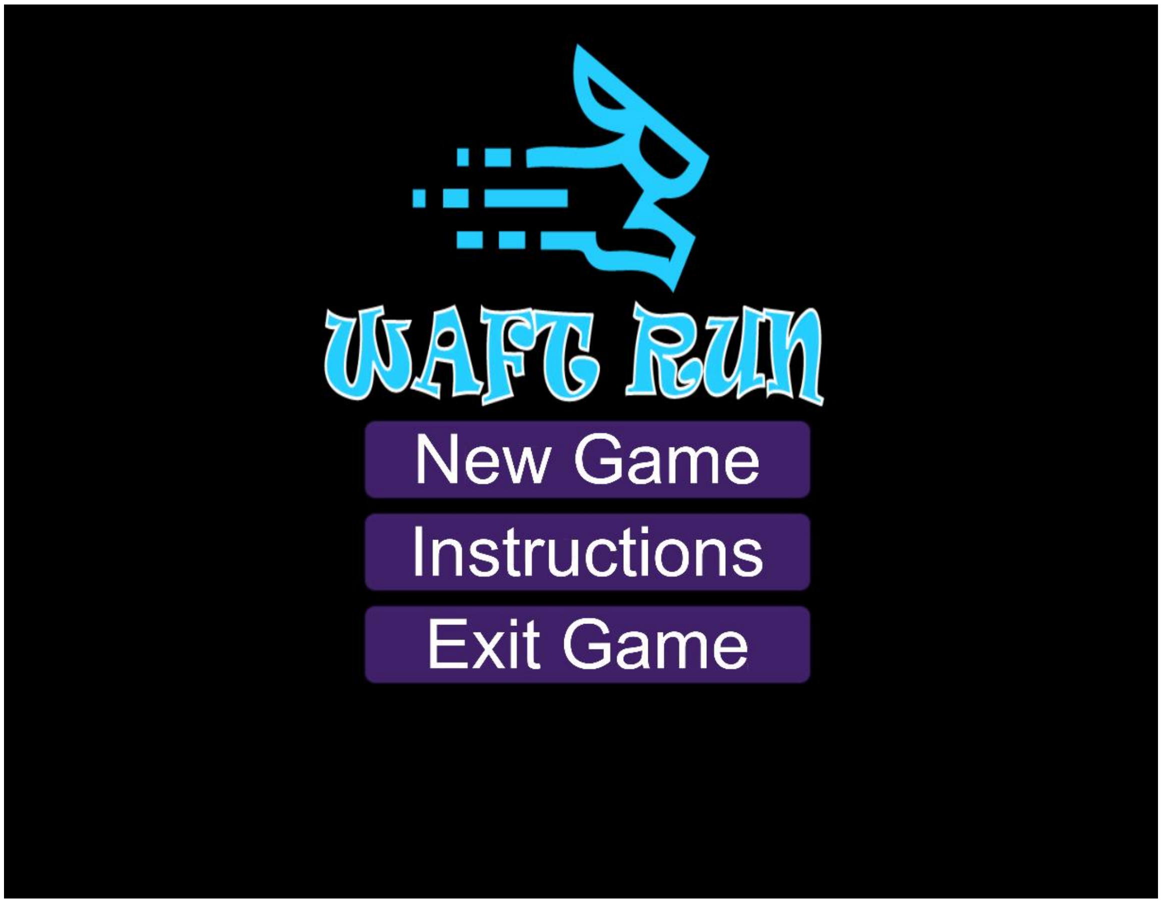
chimeSpawner:

The chime spawner script spawns a chime every few seconds around the inputted x, y and z co-ordinates. The chime will be spawned at a random time between the leastWait and mostWait. Some chimes may be out of reach to the player. The objective is to gather as much of the available chimes in time given.

```
// Update is called once per frame
void Update () {

    //make spawn rate random
    spawnWait = Random.Range (leastWait, mostWait);

}

IEnumerator Spawner(){


    //Wait for an amount of time
    yield return new WaitForSeconds (startWait);

    while (true) {
        //Spawn random at x & z, but Y = 1
        Vector3 spawnPosition = new Vector3 (Random.Range (-spawnArea.x, spawnArea.x), Random.Range (-spawnArea.x, spawnArea.x), Random.Range(-spawnArea.z, spawnArea.z));

        Instantiate (chime, spawnPosition + transform.TransformPoint(0,0,0),gameObject.transform.rotation);

        //how long till spawn new enemy

        yield return new WaitForSeconds (spawnWait);
    }

}
```

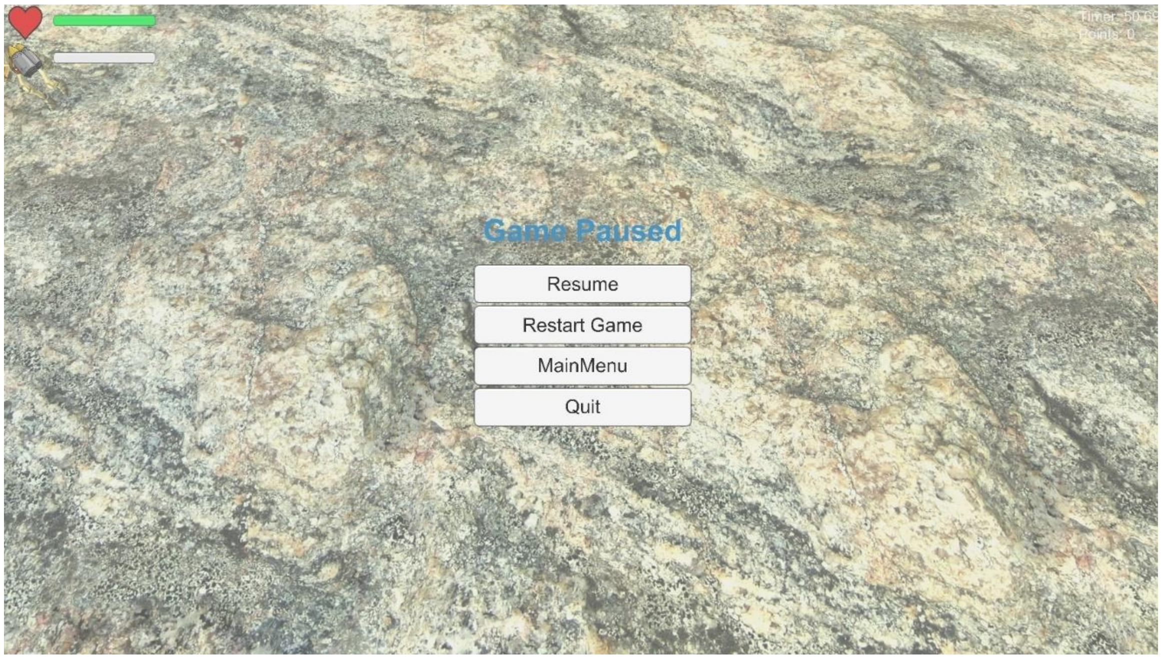
2.7.4 Graphical User Interface (GUI) Layout

Waft Run Main menu



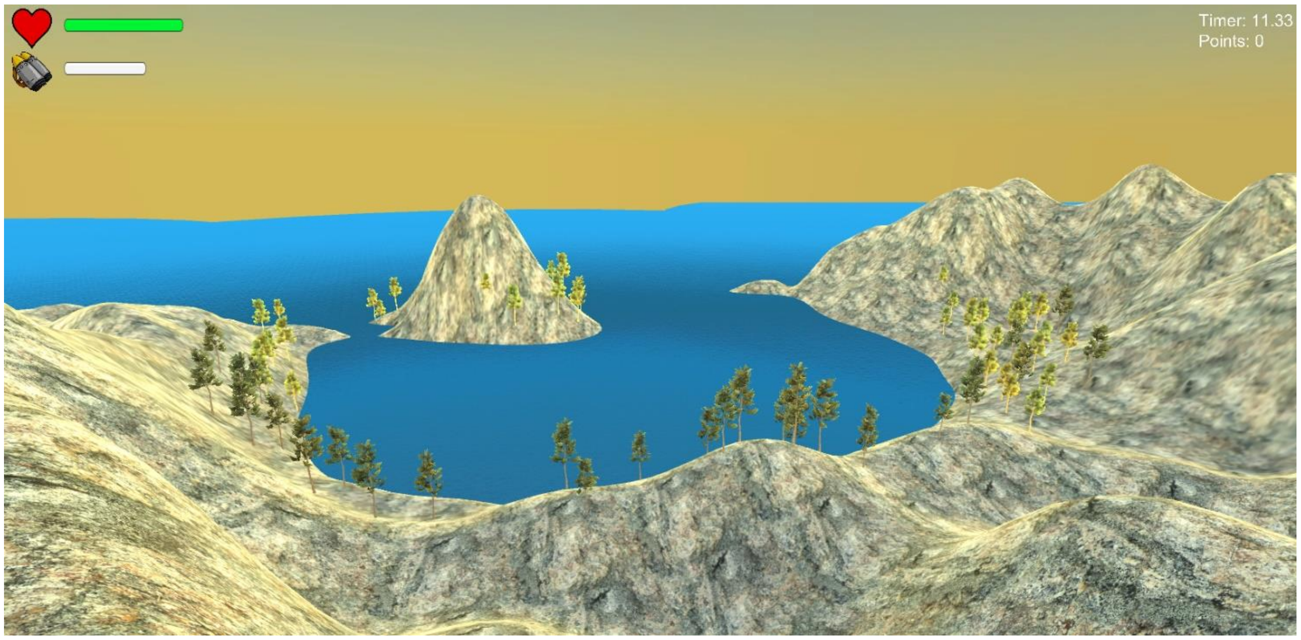
This GUI contains the new game, options, and exit game functions. The new game button will create a new instance of the game for you, the options will show you a few options to change and the exit game button will close the application.

Waft Run Pause Menu



This GUI contains the resume, restart, main menu and quit functions. The resume button will take you back to the game, the restart button will restart the current level, the main menu button will take you back to the main menu where you can see instructions on the game and the quit button will close the application down (Take you back to your desktop).

In-Game GUI



This shows the in-game GUI. In the top left corner you can see the player characters health bar and boost meter. The green bar will deplete if the character takes damage. The white bar will deplete when the player is boosting around the map. In the top right corner you can see the chime points and under you can see the time which will count down from 45 seconds.

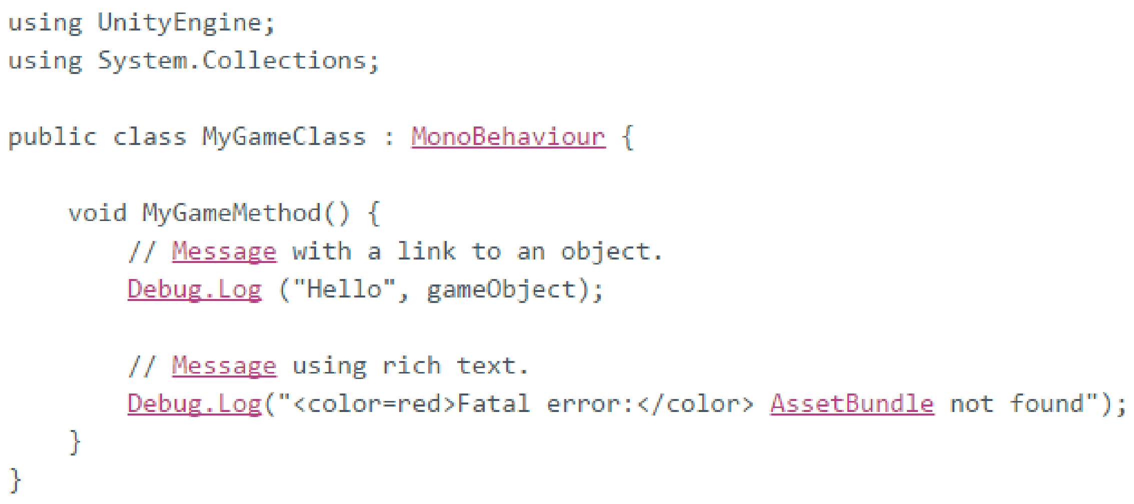
2.8 Testing

Testing is hugely important when it comes to game development progression.

Unit Testing

The Unity debugger was vital in testing certain aspects of the game. Both the Debug.Log() function and the Debug.Print() function can be placed into the scripts to test how the data is being handled. The figure below shows an example. They can be used to see the values of variables and how they are being used. This is a form of Black Box testing as it allows you to input information into the game and monitors the output to see if its returning what is expected.

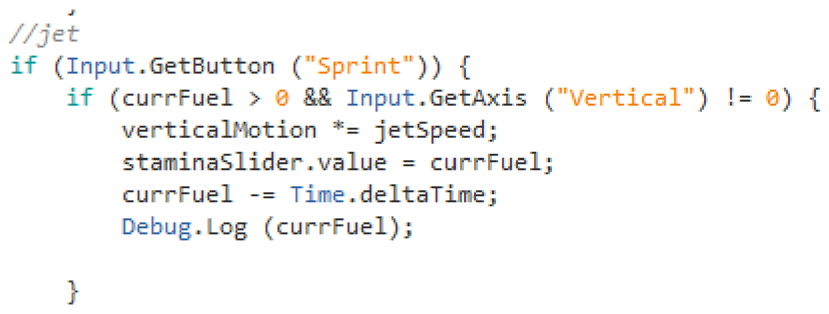
Debug.log



```
using UnityEngine;
using System.Collections;

public class MyGameClass : MonoBehaviour {

    void MyGameMethod() {
        // Message with a link to an object.
        Debug.Log ("Hello", gameObject);

        // Message using rich text.
        Debug.Log("<color=red>Fatal error:</color> AssetBundle not found");
    }
}
```

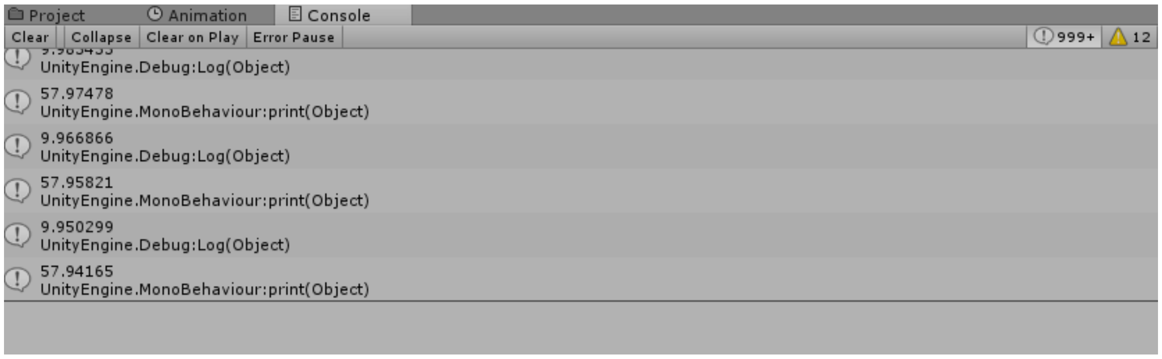
This debugger was used when adding functionality to the game and examining where the problems in the code lay. Below shows testing the jet fuel stamina. It displays the amount of fuel currently.

Code:



```
//jet
if (Input.GetButton ("Sprint")) {
    if (currFuel > 0 && Input.GetAxis ("Vertical") != 0) {
        verticalMotion *= jetSpeed;
        staminaSlider.value = currFuel;
        currFuel -= Time.deltaTime;
        Debug.Log (currFuel);

    }
```

Console:



```
9.983433
UnityEngine.Debug:Log(Object)
57.97478
UnityEngine.MonoBehaviour:print(Object)
9.966866
UnityEngine.Debug:Log(Object)
57.95821
UnityEngine.MonoBehaviour:print(Object)
9.950299
UnityEngine.Debug:Log(Object)
57.94165
UnityEngine.MonoBehaviour:print(Object)
```

Customer Testing

User testing was also very important. I had both students from my course and friends with little gaming knowledge test the game functions throughout. This allowed me to catch small bugs that I wouldn't have noticed before. It also gave me an idea of what the user enjoyed about the game and what he/she did not enjoy.

When first developing the game the timer was set to count upwards (i.e. the game was never ending unless player died) but after review from customer testing I concluded that it was more enjoyable to give the game a competitive feel. To achieve this I set the timer to count down from 45 seconds, after 45 seconds the users score is displayed on the screen. Players can now compete with each other.

3 Conclusions

Overall I have really enjoyed created this project. Beginning the project, I felt that I had a good understand of Unity but previously I had only worked on a 2D game. Getting the opportunity to fully develop a 3D I feel I have achieving a much higher understanding of aspects in Unity and the C# language.

Several personal milestones were created and achieved throughout the development.

- Creating multiple scenes.
- Creating a character within a scene that can move throughout the map.
- Achieve a higher understanding of the C# Language.
- Create an artificial intelligence agent in the scene that attacks the character.
- Implement the animations foe the AI character.
- Create an appealing player GUI that displays all the relative information about the player.
- Create a spawner that spawns game objects throughout the map. These game objects should be intractable with the player.

There were also many hurdles that I faced over the months creating the project:

- **Unity3D** – As stated above, I was completely new to the 3D side of unity. Becoming familiar with the software took some time. I now feel very comfortable with the software.
- **Animation Setup** – Implementing the enemy animation was quite difficult as I had not used animations in unity before.
- **Leap Motion** – Although I did not use the leap motion in the end, I spent many hours in the early months trying to understand how to integrate the software with Unity.
- **Researching Tutorials** – I spent a huge amount of time researching tutorials online to achieve the smoothness feel I wanted in my game.

3.1 Further development or research

The game has multiple ways in which it can evolve with an increased budget, larger workforce or without a short time requirement:

- **Leap Motion/Oculus Rift:** As stated above original project was to be developed with a Leap Motion device. If the Leap Motion device becomes more stable and if a larger time frame it would be possible to implement the Leap Motion device to allow users to control player movement. The Oculus Rift would also be a good addition.
- **Improved Game Features:** Graphics and sound effects can be updating.

- **New Levels:** Multiple levels can be implemented to ensure the experience doesn’t become stale.
- **Story:** A story telling aspect could be implemented to the game.
- **Game Objects:** The terrain could get more advanced.
- **Character Improvements:** More functionality could be implemented to improve gameplay such as multiple attack options or character items.
- **Multiplayer:** Online multiplayer options could be added where you could interact with other players online.
- **Artificial Intelligence:** More variety of Artificial intelligence characters could be implemented as at the moment there is only one AI character.
- **Release Methods:** Release the game on multiple online platforms.
- **Evolving the Chime Script:** The chime script could be expanded so that the transforms rotate and move around the map.

4 References & Unity Assets

Bibliography:2016, S. (2016) *U.S. Most popular video game genres 2015*. Available at: https://www.statista.com/statistics/189592/breakdown-of-us-video-game-sales-2009-by-genre/ (Accessed: 11 December 2016).**In-line Citation:**(2016, 2016)

Bibliography: Technologies, U. (2016) *Unity - Scripting API: Debug.Log*. Available at: https://docs.unity3d.com/ScriptReference/Debug.Log.html (Accessed: 11 December 2016).**In-line Citation:**(Technologies, 2016)

Bibliography: Technologies, U. (2016) *Unity - developer advice*. Available at: https://unity3d.com/learn/tutorials/topics/developer-advice (Accessed: 11 December 2016).**In-line Citation:**(Technologies, 2016)

Bibliography: YouTube. (2017). Basic Artificial Intelligence for a Non-Player Character with Unity 5. [online] Available at:

https://www.youtube.com/watch?v=gXpi1czz5NA&t=1092s [Accessed 9 May 2017].

Bibliography: Unity. (2017). *Unity - Environment Details*. [online] Available at: https://unity3d.com/learn/tutorials/topics/graphics/environment-details [Accessed 9 May 2017].

Bibliography: YouTube. (2017). *Unity Tutorial | Spawning Random Enemies at Random Times and Positions*. [online] Available at:

https://www.youtube.com/watch?v=WGn1zvLSndk [Accessed 9 May 2017].

5 Appendix

5.1 Project Proposal

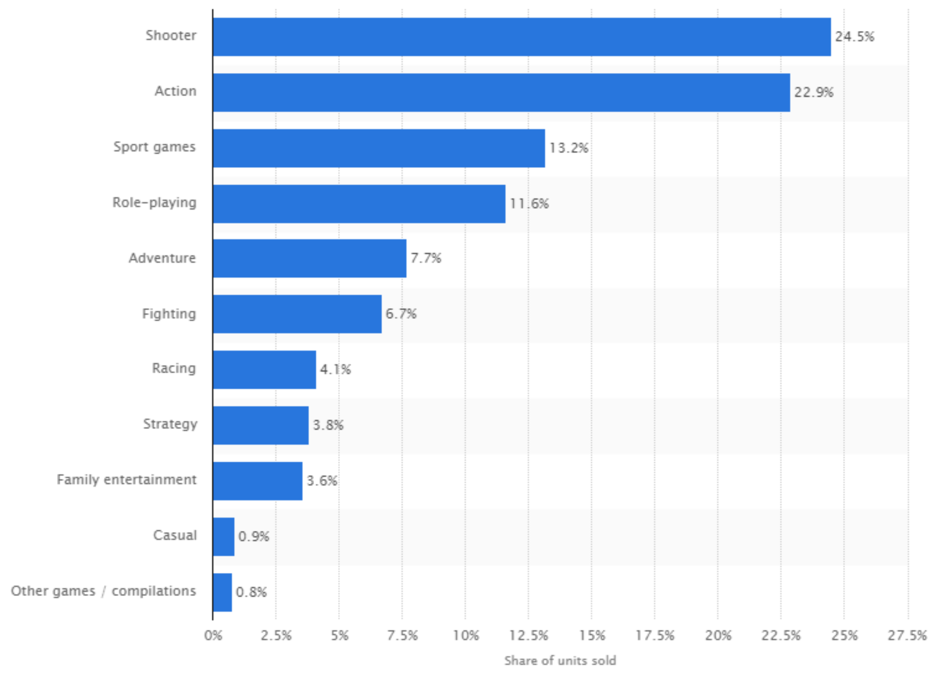
5.1.1 Objectives

- The main objective of this project is to develop a multi-model Unity3D game that gives the user a unique experience. This unique experience will be achieved through the user having a feeling of complete control like they haven't had before.
- The game will demonstrate the full capabilities of games/application that are developed using Leap Motion.
- Create movements within the game that are appealing and addictive.
- Create a well-designed main menu for the user to navigate through. They will be able to adjust settings to suit their preference. Settings such as sensitivity, field of view and sound will be adjustable.
- Allow functionality for both Leap Motion and mouse + keyboard. This will allow users that do not have access to a leap motion controller to still experience the game, although not to its full potential.
- The game will have a stylish GUI that will display the players score, previous best score, time, health and minimap. The minimap will show the players position on the map and surrounding terrain.
- Sound effects and backtrack will be pleasing to the ear and in sync with gameplay.
- Create custom characters using Autodesk Maya, going into good detail.
- Create multiple levels that drastically differ from one another.

5.1.2 Background

My project uses Unity3D, voted #2 best game engine in 2015, covers multiple industry areas and is free to download. I was debating between this and Unreal Engine 4 but decided this was my best option. The scope of my project involves developing a single player Unity3D game which takes the input from a Leap motion controller to control the game. The aim of the game is to use the Leap Motion device to glide your character across multiple sets of terrain in the fastest time possible, destroying any AI obstacles. You will compete with other users to try to achieve the highest score on the ranking leader board. The inspiration for this project mainly comes from a popular game named Tribes Ascend. The unique difference will be that the objectives of the game is completely different and my game will be completely controlled using the Leap Controller.

The genres I choose for this game include shooter, adventure, racing and action. While researching I found that very few games had been released for PC gaming in the racing genre (Only about 5% across all platforms) and thought it could be an opening in the market.



https://www.statista.com/statistics/189592/breakdown-of-us-video-game-sales-2009-by-genre/

Also you can see from this chart above that shooter and action genres are by far the most popular.

5.1.3 Technical Approach

Since I have a small bit of previous experience with using Unity2D developing a small 2D sidescroller game I began to research deeper into unity3D. It is a well-known piece of software with thousands of tutorials on many aspects including physics and code environments. It is also free to download, with no yearly subscription.

I plan to use Autodesk Maya to develop my main character in the game. The main character will need to be very detailed to appeal to the user. I feel Autodesk Maya will help me achieve this. There are many tutorials available both online and pre-installed in the software.

5.1.4 Special resources required

- ➢ Leap Motion controller - For this project I plan to use an IoT device called Leap Motion. Leap Motion is a sensor device that supports hand and finger motions as input into unity. It requires no contact once plugged into the computer via USB.

  I purchased the leap motion from www.leapmotion.com for 70 euro, this included a stand for the device.
- ➢ Unity3D – This is the game engine I plan to create my project in.
- ➢ Desktop/Laptop - Either of these will be needed to run the game.
- ➢ Autodesk Maya – This is a computer animation and modelling software that I plan to use for many aspects of the game.

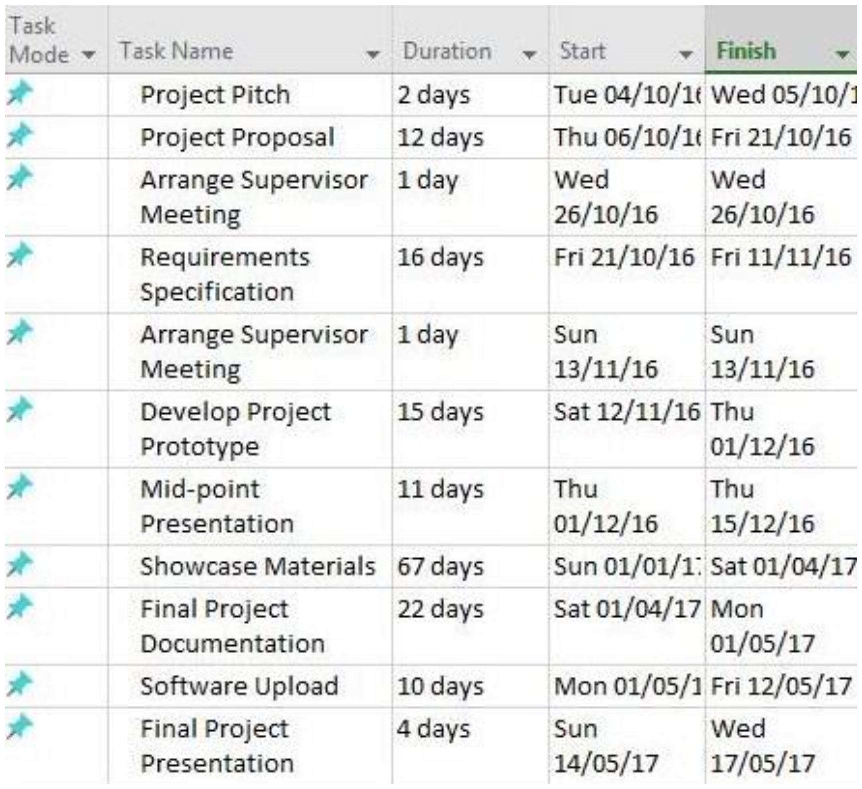
5.1.5 Technical Details

- The game will be coded in Unity using mainly C#.
- The game engine will be Unity3D.
- The model design environment will be developed in Autodesk Maya.
- I will use the unity inbuilt simulator to test the game.

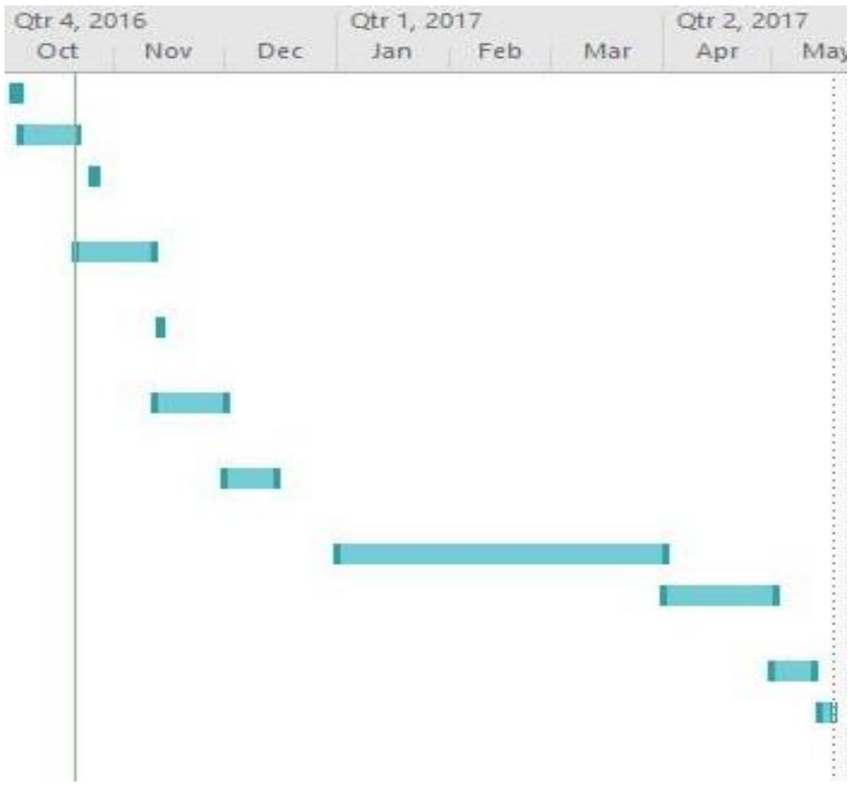
5.1.6 Evaluation

With the research I completed I am confident that my game will appeal to a large majority of both hard-core and casual gamers. Hard-core gamers will battle to fight for the best score while casual gamers will enjoy the flow and relaxing movements within the game.

5.2 Project Plan



Task Mode	Task Name	Duration	Start	Finish
📌	Project Pitch	2 days	Tue 04/10/1	Wed 05/10/1
📌	Project Proposal	12 days	Thu 06/10/1	Fri 21/10/16
📌	Arrange Supervisor Meeting	1 day	Wed 26/10/16	Wed 26/10/16
📌	Requirements Specification	16 days	Fri 21/10/16	Fri 11/11/16
📌	Arrange Supervisor Meeting	1 day	Sun 13/11/16	Sun 13/11/16
📌	Develop Project Prototype	15 days	Sat 12/11/16	Thu 01/12/16
📌	Mid-point Presentation	11 days	Thu 01/12/16	Thu 15/12/16
📌	Showcase Materials	67 days	Sun 01/01/1	Sat 01/04/17
📌	Final Project Documentation	22 days	Sat 01/04/17	Mon 01/05/17
📌	Software Upload	10 days	Mon 01/05/1	Fri 12/05/17
📌	Final Project Presentation	4 days	Sun 14/05/17	Wed 17/05/17


Qtr 4, 2016
Oct
Nov
Dec
Qtr 1, 2017
Jan
Feb
Mar
Qtr 2, 2017
Apr
May

5.3 Monthly Journals

5.3.1 Month: September

My Achievements

This month, I was able to:

- Define what my project will accomplish: *'My project will consist of using motion sensors to display information about a chair in an application. Details such as if the chair is occupied, the chairs movement and how frequently the chair is used will be stored.' (Later will be changed)*
- Research Internet of Things section while looking for information that may help me when developing my project.
- Discuss with multiple friends and family my idea and get their take on what is good/bad.
- Plan for my project pitch coming up at the beginning of next month.
- I pitched my project to 3 college lecturers. It went well.

My Reflection

The lecturers I presented to seemed to like the project idea but also had a few recommendations. They recommended that I pick a specific target market that the project will be used for. One lecturer said Dublin Bus would be a good idea. I will keep researching my target market.

The next few days I plan to concentrate on developing my mock-ups and defining exactly what software I will use for the project.

Intended Changes

As Said before, I intend to change the project to be more focused on a specific target market as per the feedback given from the lecturers.

5.3.2 Month: October

My Achievements

Note: At the beginning of this month I decided to switch project ideas.

This month, I was able to:

- Complete and upload my project proposal.
- Continue my research into Leap Motion development with Unity.
- Begin my requirement specification.
- I contacted my supervisor (Adriana Chis) and she organised a schedule for meetings throughout the year.
- I began messing around with the game scene in unity and looking at potential assets I can use for my project (Mainly Prefab ideas).

My Reflection

I felt, it worked well to work on my project frequently. I made good progress when I did this. In the middle of the month a had a few poor working days where I did not research or work on my project at all. I completed the project proposal and I feel it gave myself a very good idea and groove in my project direction.

However, I was not successful in developing any functionality within the game, I feel I am not at the stage where I am willing to start. I want to concentrate on the document and research side and make sure it is completed properly.

Intended Changes

Next month, I will try to complete my requirement spec by the 11th, I will need to meet with Adriana multiple times as I have questions about how I should go about some parts of the project and I will dive into working on some of the functionality in the game.

I realize I need to research more into implementation of the Leap Motion device with unity. This is my first time working with the device. From what I understand it can be difficult. I have found multiple tutorials I plan to watch throughout the month.

Supervisor Meetings

There were no supervisor meeting this month as we were only assigned in the last week.

5.3.3 Month: November

My Achievements

This month, I was able to:

- Meet with my supervisor Adriana Chis and discuss my project. She was very helpful and she discussed my requirement specification with me.
- I developed the first version of my terrain in Unity. It is very basic only containing small hills and water.
- I began the script for my player movement (Not with the Leap Motion device).
- I completed and uploaded my requirement specification on the 11th of the month.

My Reflection

Working on the requirement specification this month allowed me to get a deeper understanding of the scope of my project. I developed a much better understanding of the flow and functions in my game. I was unable to get started on some tutorials I had prepared the previous month. This was mainly due to the deadline of some projects in different modules.

Intended Changes

I again plan to complete watching the tutorials I have prepared to aid in the development of my prototype.

Supervisor Meetings

We met Adrianna for the first time as a group this week. She discussed what is expected for the mid-point presentation and briefly went through our projects with her. I now have a much better understand of what is to be expected in the presentation.

5.3.4 Month: December

My Achievements

- This month we completed the mid-point presentation. I felt mine went well. The feedback given was helpful in the future development of my project.
- We briefly discussed how the project could use an oculus rift in the future, I think maybe this is an option after the semester has finished.
- I continued development of the terrain, implementing the water and the UnderWater script. This script gives the player a blueish tint of fog while under water.

- I also added a small house to the terrain just to improve the look.

My Reflection

I feel relieved that the mid-point presentation is now out of the way. I can now focus more of my time on the development on the game.

The Terrain is starting to look close to finished.

Intended Changes

Next month I plan to implement the Leap Motion prefab supplied by unity.

5.3.5 Month: January

My Achievements

- This month, after uploading the requirement specification I had more time to spend on the development of the game. I focused on implementing the Leap Motion assets to my existing project. At first this was difficult because I had developed on a previous version of Unity and it needed to be updated to allow for Leap assets. I was able to copy over the majority of my prototype to the newer version and spent a bit of time implementing what I could not copy over.
- I also continued development on the game world. The world has been expanded as the previous world was too small for the objectives of the game. If needed, the world will be expanded further but for the moment it is okay.

My Reflection

I felt it worked well to contain the development of the project to a few days at a time. This meant that I would get a lot done in a few days then I could concentrate on my other 3 modules.

Intended Changes

Next month, I will try to implement movement with the alternative controller (keyboard & mouse). This was always the plan as not all users will have the availability of the Leap Motion device.

The main menu will be created next month. It will be a simple feature that provides instructions on how to play the game.

I will also develop the AI character to interact with my player. This might run into the next month but I plan to begin the development this month.

I realised that I need to research more into AI development. I have little experience with developing AI.

5.3.6 Month: February

At the beginning of this month I decided to scrap the idea of the Leap Motion completely. The reason for this was because I strongly felt that in the time frame I could not complete the game I wanted to complete if I was to implement the Leap Motion. The Leap Motion device itself had very good hand recognition but the models that were provided for unity were lacklustre. Since the technology is fairly new there are constant updates and fixes. Maybe in the future I could implement this feature in my own time.

My Achievements

- This month I successfully implemented the playerController script into my game. My character can move freely around the map. The character can use the jetpack to increase their speed.
- The AI character was implemented into the scene this month. The 3 animations used in the project are idle (used when the game first starts), walk (used when the player is in view of the character) and attack (used when the Ai is within a certain distance from the player).
- I implemented the Main menu scene this month. The user has the other to view the instructions, play the game and quit the game. There is no save function in the game as with the level only lasting 45 seconds I did not see a need.
- The pause menu was also implemented. The user can restart the level, resume the game, quit the game and go back to the main menu. When the pause menu is enabled, the game is paused.

My Reflection

Intended Changes

Next month I will implement the movement script for the Ai.

I will develop the GUI for the player. I want to have the player health, jetpack fuel, time and score all displayed on the GUI. I will need to look up various tutorials in order to ensure I complete this task.

The chime spawner will also need to be implemented. The player needs to collect chimes to add to their score.

5.3.7 Month: March

My Achievements

- This month was by far the most productive. I started by implemented the chasePlayer script on the AI character. The AI now follows the character if the player is within a certain distance and if the player is in the view of the AI.
- The GUi for the character was implemented. The health and jetpack slides were implemented. The health bar depletes if the character takes damage from the AI and the jetpack slider depletes while the character is sprinting.
- The GUI also displays the time and score. The time counts down from 45 seconds, when the timer hits 0 the game ends. The aim is for the user to get the highest score possible in the time given.
- The Spawner GameObject was added to the scene, along with the chimeSpawner script. The script allows us to spawn chimes randomly around the map. If these chimes are collected it will add to the players score. Not all chimes that are spawned will be reachable for the player. They will need to judge this themselves.

My Reflection

A huge amount was completed this month. The remaining aspects that need to be completed next month are fairly small.

Intended Changes

AudioSources will be added to some GameObjects next month. The chime, the jetpack and the Ai needs audio.

Apart from adding small aspects to the game the report from December will need updating as I have made many changes to the game throughout.

I will also need to focus on preparing for the project presentation and showcase.