National College of Ireland

FINAL REPORT

Dean Byrne

x12337831
x12337831@student.ncirl.ie

B.Sc. (Hons) in Computing: Gaming and Multimedia

DECLARATION COVER SHEET FOR PROJECT SUBMISSION

SECTION 1

Name:
Student ID:
Supervisor:

SECTION 2 Confirmation of Authorship

The acceptance of your work is subject to your signature on the following declaration:

I confirm that I have read the College statement on plagiarism (summarised overleaf and printed in full in the Student Handbook) and that the work I have submitted for assessment is entirely my own work.

Signature: _____ Date: _____

NB. If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the College's Disciplinary Committee. Should the Committee be satisfied that plagiarism has occurred this is likely to lead to your failing the module and possibly to your being suspended or expelled from college.

Complete the sections above and attach it to the front of one of the copies of your assignment,

What constitutes plagiarism or cheating?

The following is extracted from the college's formal statement on plagiarism as quoted in the Student Handbooks. References to "assignments" should be taken to include any piece of work submitted for assessment.

Paraphrasing refers to taking the ideas, words or work of another, putting it into your own words and crediting the source. This is acceptable academic practice provided you ensure that credit is given to the author. Plagiarism refers to copying the ideas and work of another and misrepresenting it as your own. This is completely unacceptable and is prohibited in all academic institutions. It is a serious offence and may result in a fail grade and/or disciplinary action. All sources that you use in your writing must be acknowledged and included in the reference or bibliography section. If a piece of writing proves difficult to paraphrase, or you want to include it in its original form, it must be enclosed in quotation marks and credit given to the author.

When referring to the work of another author within the text of your project you must give the author's surname and the date the work was published. Full details for each source must then be given in the bibliography at the end of the project

Penalties for Plagiarism

If it is suspected that your assignment contains the work of others falsely represented as your own, it will be referred to the college's Disciplinary Committee. Where the Disciplinary Committee makes a finding that there has been plagiarism, the Disciplinary Committee may recommend

- that a student's marks shall be reduced
- that the student be deemed not to have passed the assignment
- that other forms of assessment undertaken in that academic year by the same student be declared void
- that other examinations sat by the same student at the same sitting be declared void

Further penalties are also possible including

- suspending a student college for a specified time,
- expelling a student from college,
- prohibiting a student from sitting any examination or assessment.,
- the imposition of a fine and
- the requirement that a student to attend additional or other lectures or courses or undertake additional academic work.

TABLE OF CONTENTS

EXECUTIVE SUMMARY

TerraCraft is a pixel-2D side-scroller survival sandbox RPG (Role Playing Game) that's been made and developed in Unity. The objective of the game, like all RPGs, is to invest a bit of time and effort into the game to complete multiple tasks, explore the world and gain XP (eXperience Points) to invest in learning skills to become stronger. The player is also not limited to what they spend their XP on, as the game will be fully flexible and will offer the player freedom of choice. The game will feature a dynamic day/night cycle, which, when in use with the parallax effect, will give off an aesthetically pleasing 3D effect. At certain times of the day, i.e. midnight, different enemies will spawn, e.g. zombies, mummies, etc., to give the user a unique experience for each time of day, making it a little bit harder to survive.

The game will be built using C# in Microsoft Visual Studio 2013 and is aimed mainly at Windows PCs. All the graphics used within the game is of my own design, using fonts created online by myself on FontStruct, and using images (sprites) created using Adobe Photoshop CS6. Data for the game, i.e. save data, will be stored locally on the players' PC in their AppData folder, ensuring the data is easily obtainable using a persistent data reader.

1. INTRODUCTION

The project in which I am developing is called TerraCraft. It is a pixel-2D side-scroller survival sandbox RPG. In this section, I will describe the project background, its aims, technologies and the structure of the overall project.

1.1. BACKGROUND

Over the past 30 years, video games have become an integral part of our culture. It's seen remarkable growth and is a multi-billion-dollar industry with all the platforms that it contains. From PCs, consoles, mobile and tablets, gaming is everywhere and it is a great getaway to immerse yourself into the game. Over the past few years, gaming companies have been created and are opening new jobs and new opportunities to prove what capabilities you can achieve. Digit Gaming is an example of a gaming company. Based in Dublin, Digit Gaming is a cross-platform game development company that recently hit headlines for having the "biggest budget video game in Irish history" (Lyons, 2014). The game they released, "*Kings of the Realm*", published with *Kabam*, was picked by Google to be featured on the Play store to millions of people across the globe. With this, they opened 40 new jobs after an investment from a multi-million-dollar company *Scopely* (Whelan, 2015), which is big news because it offers game developers, like myself, an opportunity to apply and possibly acquire a job in the gaming industry.

I reason why I chose to do this project is due to my keen interest in gaming and game design. I frequently engage my free time to playing games and I feel that I could start to develop a game that everybody could enjoy. This project was also of interest to me, as the logic of the game is based off the idea of other games including "*Terraria*" by *Re-Logic*, "*Minecraft*" by *Mojang* and "*Starbound*" by *Chucklefish Games*. I've spent countless hours playing these games and they always keep me interested and focused, as they have endless possibilities to what you can do. This project was also of interest to me from a technical point-of-view since it'll be made using 2D graphics that'll be designed in *Adobe Photoshop CC*. I've always had an interest in 2D graphics as it looks like it's minimalistic but, in fact, are complex but beautifully well represented, if done right.

The reason why I've chosen a 2D role playing game is mainly because the genre greatly intrigues me. It is my favourite genre of gaming and I feel that I would enjoy the opportunity and the challenges that this game can offer. The genre of RPG gaming has a substantial number of players who enjoy games like the "*Grand Theft Auto*" series by *Rockstar*, "*The Elder Scrolls*" series and the "*Fallout*" series by *Bethesda* and the "*Final Fantasy*" series and the "*Life is Strange*" series by *Square Enix*. These games are unique in their own way, offering the players hundreds of hours of gameplay due to storylines, collectables and achievements. The players will also have to think logically and quickly as they've to manage their characters' health along with XP, coins and items in a world filled with weak and strong enemies, dwelling to kill them. Over time, the player can strengthen themselves with perks and upgrades which can have major effects on the game. Engaging in quests from NPCs can reward the player with better equipment or XP and coins. It is a very engaging and highly enjoyable style of gaming and I feel that I could make an incredible contribution to this project.

In terms of development, I've chosen to create this project using the Unity game engine. I felt that Unity can offer me a countless number of features that could benefit the project in terms of finding solutions to in assets or online tutorials. The Unity game engine, recently, have put most of their development into 2D, giving developers more options in creating games. In Unity 2D, developers can create simple 2D characters and move them around the screen easily with the new functions made available. Functions like Rigidbody2D can give a two-dimensional gravity effect, Box Colliders

give an object a 2D collider to detect other objects and Raycasting2D to detect, in two-dimensional space, where the mouse is and where it clicked, whether on a game object or on a UI object. I find that using the Unity game engine will make the progress of the project much easier, as I've also had experience with this in the 6-month internship with *Defiant Games* as part of the Work Placement module in the 2nd semester of 3rd year.

1.2. AIMS

The aim for this project is to create and develop a fully functional and viable game application in the RPG genre. The game will have all, if not, most of the functionalities I have described in the *Project Proposal*. The game itself will allow users to customise their settings, i.e. sound, graphics and keyboard/mouse controls. These settings are vital for players who prefer different playing styles of gameplay, as it'll allow them to customise the game to their needs to give them the experience they deserve. The game will also allow players to create and customise their own character. This will help to achieve giving the players a unique playing experience.

At the beginning of the game, the player will have the opportunity to participate in a tutorial. This is to allow the player to get to know the controls and functionalities of the game, so they can go off, explore and survive. This will take place in the game, so the player can get a sense of survivability. The player can also access the tutorial at any time, if they need it.

Another aim of the game is to allow the player to get XP (eXperience Points) by either crafting items/tools in the inventory system or by killing enemies. XP can be used to invest in certain skills the character possesses. This is a primary aim for any RPG. It's to encourage the player to attempt the game in a range of unique styles each time they play the game.

1.3. TECHNOLOGIES

1.3.1. UNITY GAME ENGINE

Unity is a cross-platform game engine with its own built in IDE called MonoDevelop. Unity is mainly used to create and develop games for a variety of different platforms, including web, consoles and mobile. Unity is the primary tool I'll be using to develop the project. It was chosen because it's the only game engine I feel comfortable with, as I've spent over a year experimenting with distinctive styles of projects, from 3D to 2D desktop and mobile games, and mainly because it's a free-to-use game engine that offers either free or paid assets (available from the Unity Asset Store) that can be added to the project to further enhance the gameplay.

Unity's scripting is based mainly on Mono which is an open-source implementation within the .NET Framework, which any developer working with Unity is given the option of using JavaScript, C# or Boo. Unity makes use of their own IDE, MonoDevelop, but developers can use any scripting software they desire.

Unity supports deployment of projects on a range of different platforms, as mentioned above. When the project is finished, developers can go into the build settings and select what platform to deploy the project to. The list of supported platforms is extensive but includes the likes of PS4, Xbox One, Android, iOS, PC, Linux and many others. It's valuable to have the option to have the choice to deploy the game to other platforms, allowing more players on different platforms to play the game.

The version of the Unity Game Engine being used to create this project is Unity 5.4.1f. It was just recently release, as I like to keep it constantly updated to the latest version.

2. SYSTEM

2.1. REQUIREMENTS

2.1.1. FUNCTIONAL REQUIREMENTS

2.1.1.1. NEW GAME

The player should be able to begin a new game when launching the game application for either the first time or any time henceforth. This requirement has not changed from the original requirements specification document.

2.1.1.2. LOAD GAME

The player should be able to load a game when launching the game application for either the first time or any time henceforth. This requirement has not changed from the original requirements specification document.

2.1.1.3. SAVE GAME

The player should be able to save a game when the game application has been launched and a game has been loaded for either the first time or any time henceforth. This requirement has not changed from the original requirements specification document.

2.1.1.4. QUIT GAME

The player should be able to quit the game when inside the game application for either the first time or any time henceforth. This requirement has not changed from the original requirements specification document.

2.1.1.5. START TUTORIAL

The player should be able to begin a tutorial when the game has launched for either the first time or any time henceforth. This requirement has not changed from the original requirements specification document.

2.1.1.6. COLLECT RESOURCES

The player should be able to collect resources when the game has loaded and the character is inside the generated world. This requirement has not changed from the original requirements specification document.

2.1.1.7. CRAFT ITEMS

The player should be able to craft items when the correct amount of resources is collected and the inventory system is opened. This requirement has not changed from the original requirements specification document.

2.1.1.8. KILL ENEMIES

The player should be able to kill enemies when the game has loaded and the character is inside the generated world for either the first time or any time henceforth. This requirement has not changed from the original requirements specification document.

2.1.1.9. UPGRADE PLAYER SKILLS

The player should be able to upgrade player skills when the character has levelled up and collected XP. This requirement has not changed from the original requirements specification document.

2.1.2. DATA REQUIREMENTS

The users will require the ability to save their progress that they've made within the game, i.e. character customisation and progression. Data will be stored in JSON format in the local disk of the users' PC. The platform therefore will need to be Windows. The user will also require a small amount of free space on their hard drive for the game itself.

2.1.3. USER REQUIREMENTS

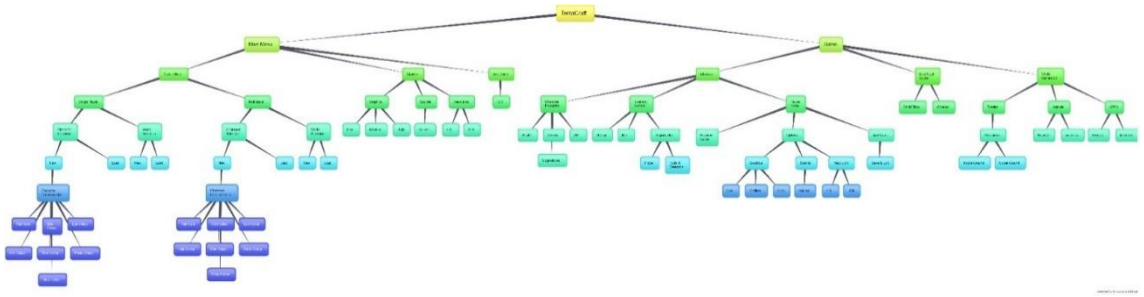
The user must have a desktop PC or a laptop capable of running the latest version of Windows. Internet Access will also be a requirement to download the game application.

2.1.4. ENVIRONMENTAL REQUIREMENTS

The game must be run in a stable environment. The OS that the game is designed for in the Windows 10 platform. But it can work on previous versions like Windows 8, 7 or Vista.

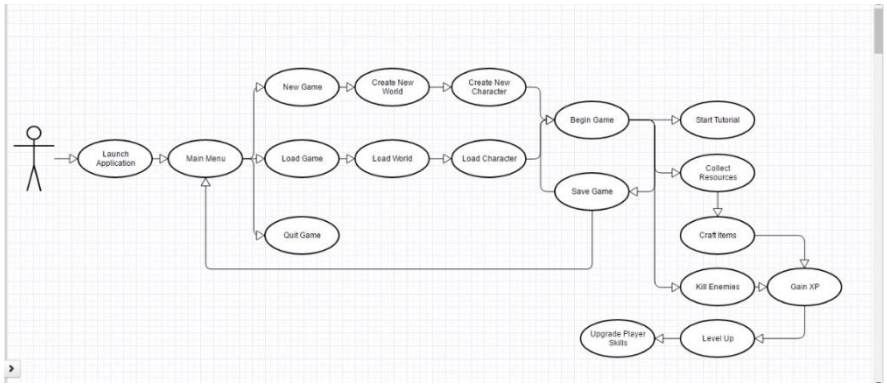
2.2. DESIGN AND ARCHITECTURE

This is the architecture design for my project; TerraCraft. It provides an in-depth view of every functionality of the game.



2.2.1. USE CASE DIAGRAM

This is the Use Case Diagram for my project; TerraCraft. It provides an overview of the functional requirements inside the game.



2.3. IMPLEMENTATION

Each implementation section will include a detailed description and included screenshots from the *Unity Game Engine* and *Microsoft Visual Studio*.

2.3.1. GAME SYSTEMS IMPLEMENTATION

In each section of the implementation, we will cover the overall game systems. These systems are vital to the runtime of the game and are linked to many of the requirements listed above.

2.3.1.1. MAIN MENU IMPLEMENTATION

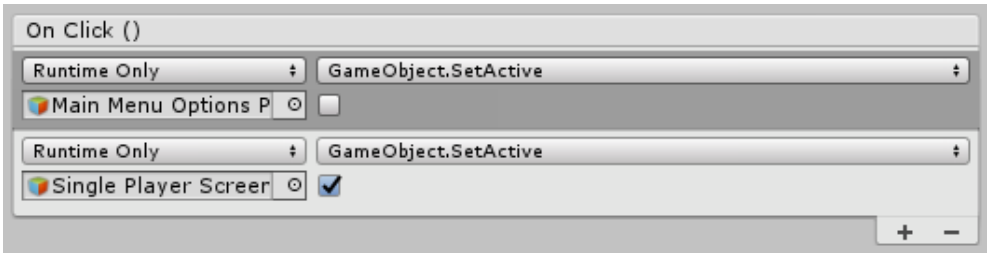
Allows control and maintains what the user selects in the main menu. These systems will be broken down and detailed in sections.

- **Single Player button:** allows the user to access the single player menu. This menu will go through the setup for changing graphics, while the functions of the menu are described in another section – please see *Single Player Implementation*
- **Options button:** allows the user to access the options menu. This menu will go through changing the graphics, while the functions of the menu are described in another section – please see *Options Implementation*
- **Exit button:** allows the user to end the game's runtime, shutting down its systems, returning them to the desktop.

2.3.1.1.1. IMPLEMENTATION & SCREENSHOTS

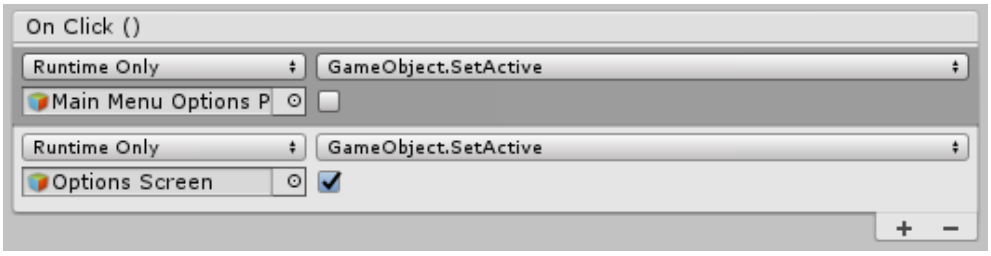
2.3.1.1.1.1. SINGLE PLAYER

This function uses an OnClick event, an automatic script that executes with any button in *Unity*, usually left empty by default so then nothing happens. In this event, the single player button sets the visibility for the main menu to false (meaning its functions will no longer be accessible) and the visibility for the single player menu to true.



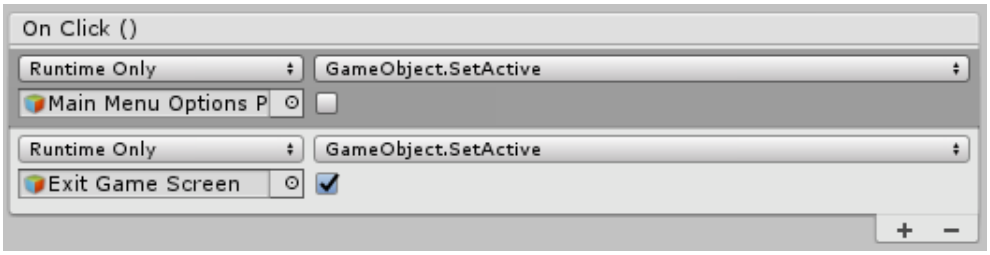
2.3.1.1.1.2. OPTIONS

Like the Single Player button, the Options button uses an OnClick event which, when on the main menu, sets the visibility for the main menu to false, and the visibility for the options to true.



2.3.1.1.1.3. EXIT GAME

The Exit Game button also uses an OnClick event, which turns the visibility for the main menu to false and the Exit Game Confirmation menu to true, to allow the user to select whether they wanted to exit the game – in an event if they clicked the button by accident.



2.3.1.2. SINGLE PLAYER MENU IMPLEMENTATION

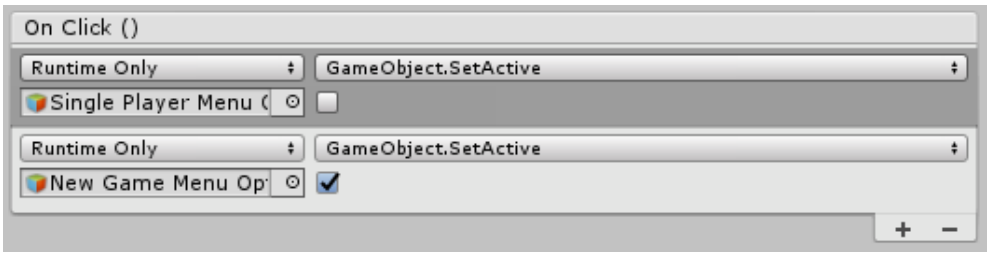
Allows the user to control and maintain:

- **New Game option:** allows the user to create a new game. Game doesn't have a save/load, unfortunately, so the user won't be able to effectively save/load the game).
- **Back button:** allows the user to leave the *Single Player* menu and return to the *Main Menu*.

2.3.1.2.1. IMPLEMENTATION & SCREENSHOTS

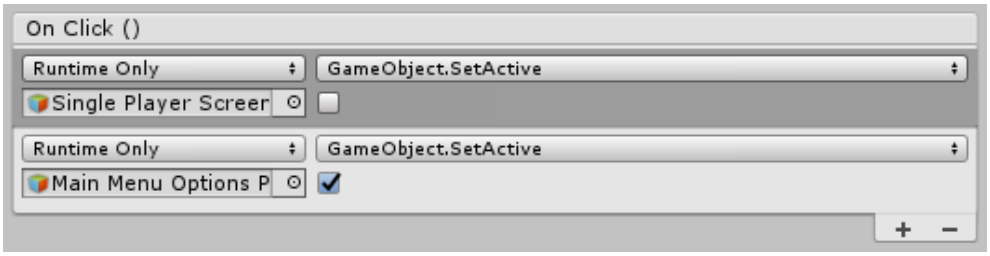
2.3.1.2.1.1. NEW GAME

The *New Game* function is an OnClick event, turning the visibility of the *Single Player* menu to false and the visibility of the *New Game* menu to true, redirecting the user to another menu, to set up their game.



2.3.1.2.1.2. BACK BUTTON

The *Back*-button function is an OnClick event, turning the visibility of the *Single Player* menu to false and the visibility of the *Main Menu* to true.



2.3.1.3. *New Game Menu Implementation*

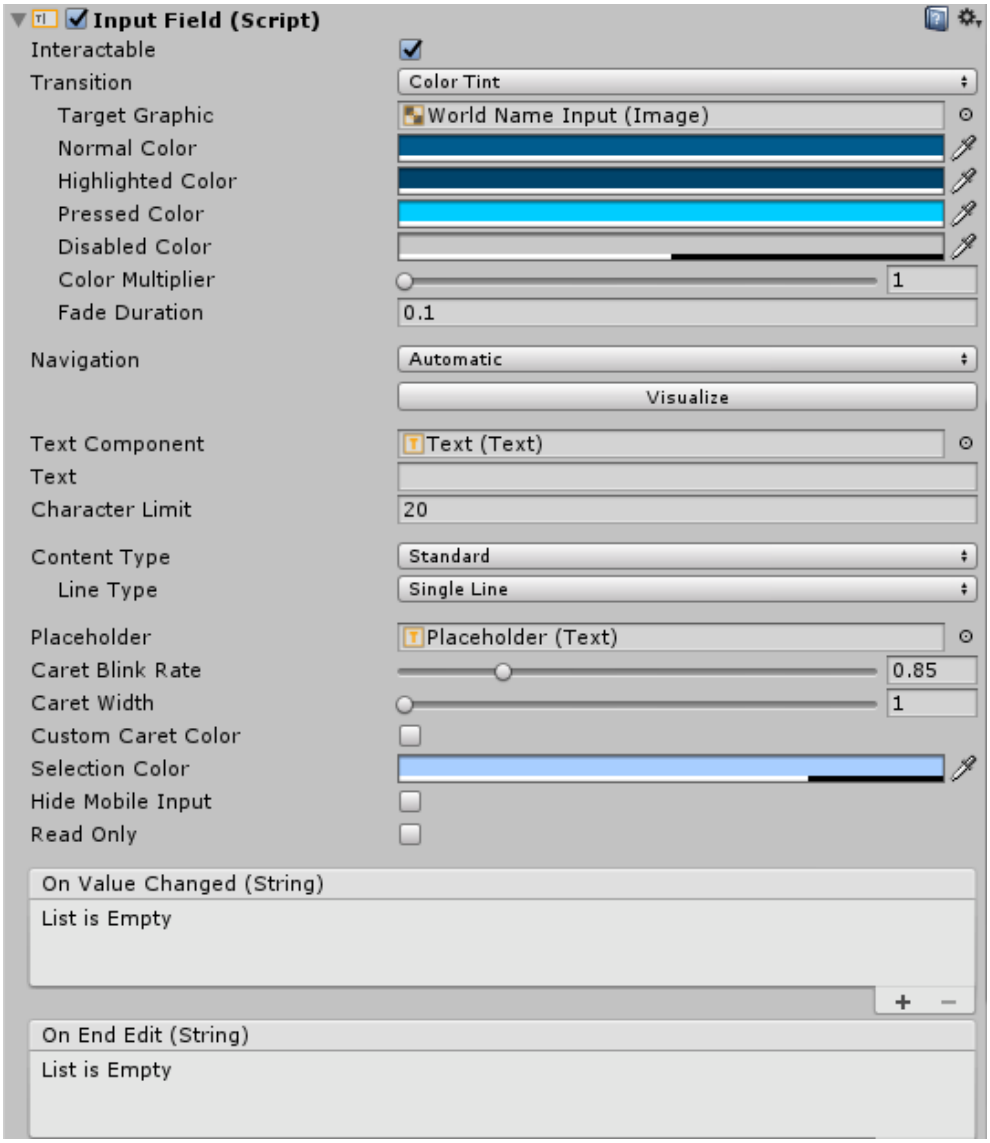
Allows the user to control and maintain:

- **World Name text:** allows the user to name the world that will be generated at the beginning of the game. This would allow the game to be saved using the world's name, making it easier to find and load. Unfortunately, it wasn't implemented into the project.
- **Begin Game button:** allows the user to start the game
- **Back button:** allows the user to leave the *New Game* menu and return to the *Main Menu*.

2.3.1.3.1. Implementation & Screenshots

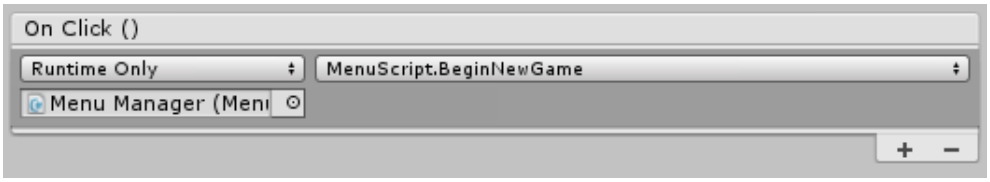
2.3.1.3.1.1. World Name

The *World Name* is a text field that allows input from the user to name the world that will generate when the game begins.



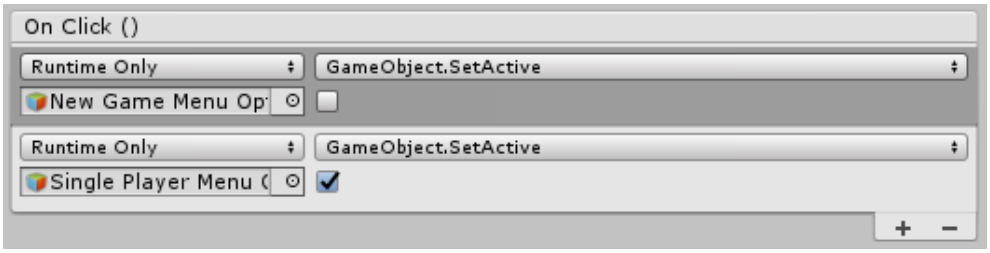
2.3.1.3.1.2. BEGIN GAME

The *Begin Game* button is an OnClick event, that loads the *Main Game* scene, using a custom script method called *BeginNewGame*.



2.3.1.3.1.3. BACK BUTTON

The *Back*-button function is an OnClick event, turning the visibility of the *New Game* menu to false and the visibility of the *Single Player* menu to true.



2.3.1.4. *Options Menu Implementation*

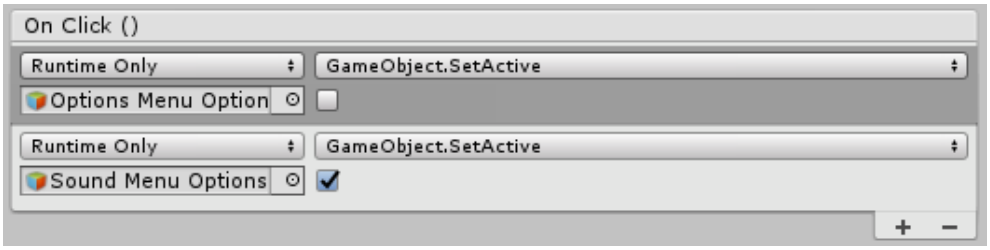
Allow the user to control and maintain:

- **Sound button:** allows the user to view and change the volume of the game sound.
- **Graphics button:** allows the user to view and change the game graphics.
- **Gameplay button:** allows the user to view the gameplay controls.
- **Back button:** allows the user to leave the *Options Menu* and return to the *Main Menu*.

2.3.1.4.1. Implementation & Screenshots

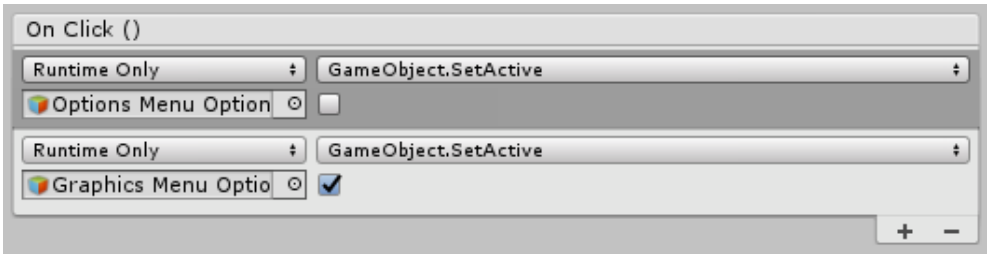
2.3.1.4.1.1. Sound Button

The *Sound* button is an OnClick event, turning the visibility of the *Options* menu to false and the visibility of the *Sound* menu to true.



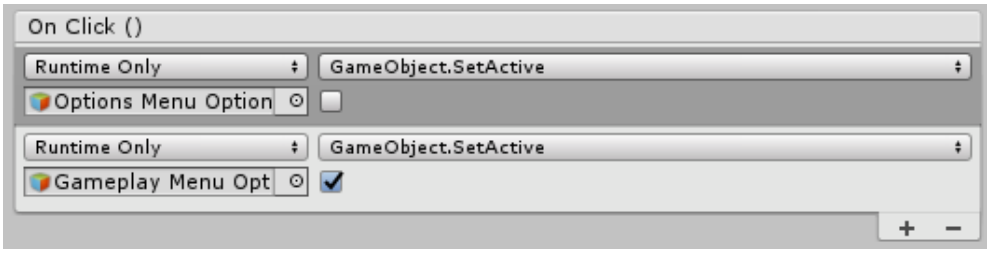
2.3.1.4.1.2. Graphics Button

The *Graphics* button is an OnClick event, turning the visibility of the *Options* menu to false and the visibility of the *Graphics* menu to true.



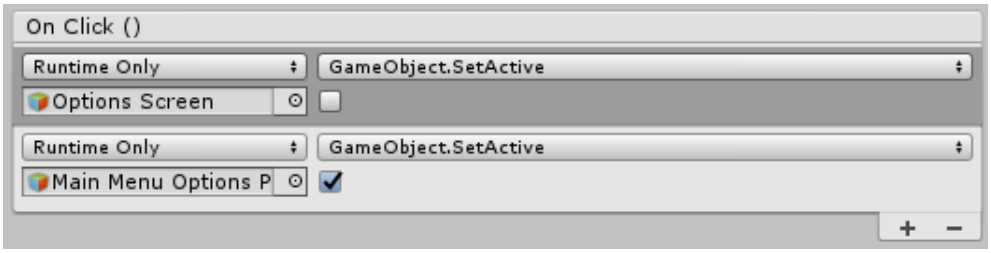
2.3.1.4.1.3. Gameplay Button

The *Gameplay* button is an OnClick event, turning the visibility of the *Options* menu to false and the visibility of the *Gameplay* menu to true.



2.3.1.4.1.4. BACK BUTTON

The *Back*-button is an OnClick event, turning the visibility of the *Options* menu to false and the visibility of the *Main Menu* to true.



2.3.1.5. SOUND MENU IMPLEMENTATION

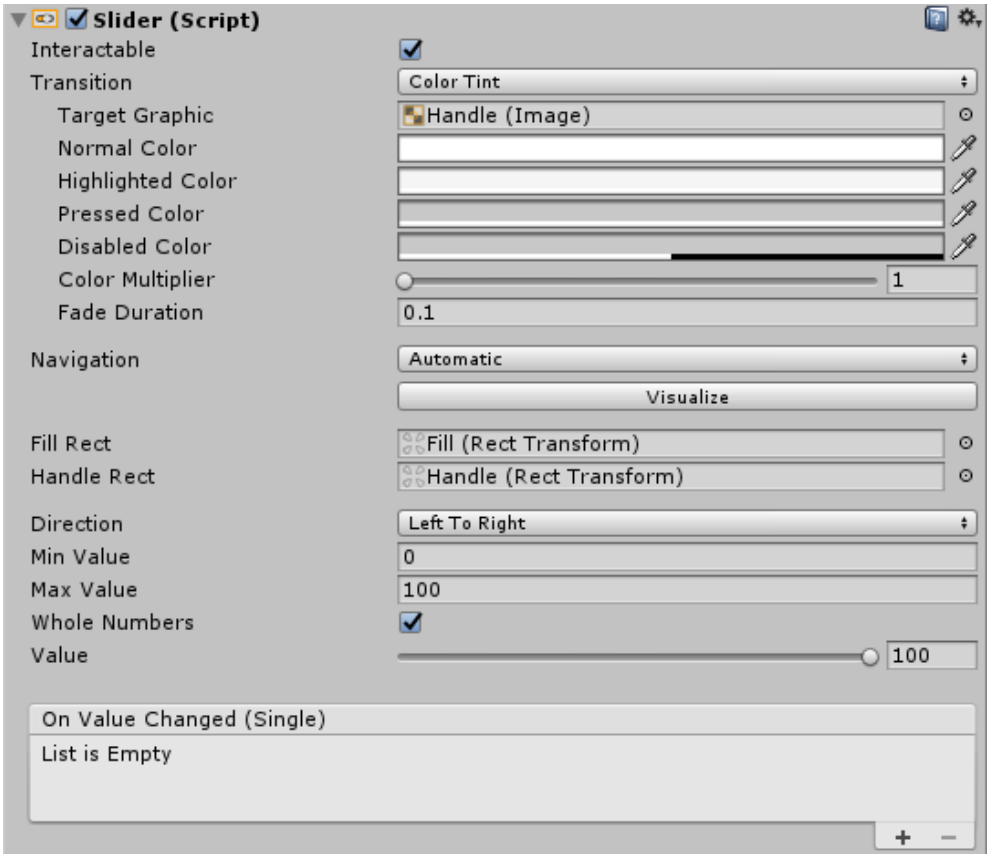
Allow the user to control and maintain:

- **SFX slider:** allows the user to control the volume of the SFX (sound effects) of the game by using a slider, with a minimum value of 0 and a maximum value of 100.
- **Music slider:** allows the user to control the volume of the music of the game by using a slider, with a minimum value of 0 and a maximum value of 100.
- **Back button:** allows the user to leave the *Sound Menu* and return to the *Options Menu*.

2.3.1.5.1. IMPLEMENTATION & SCREENSHOTS

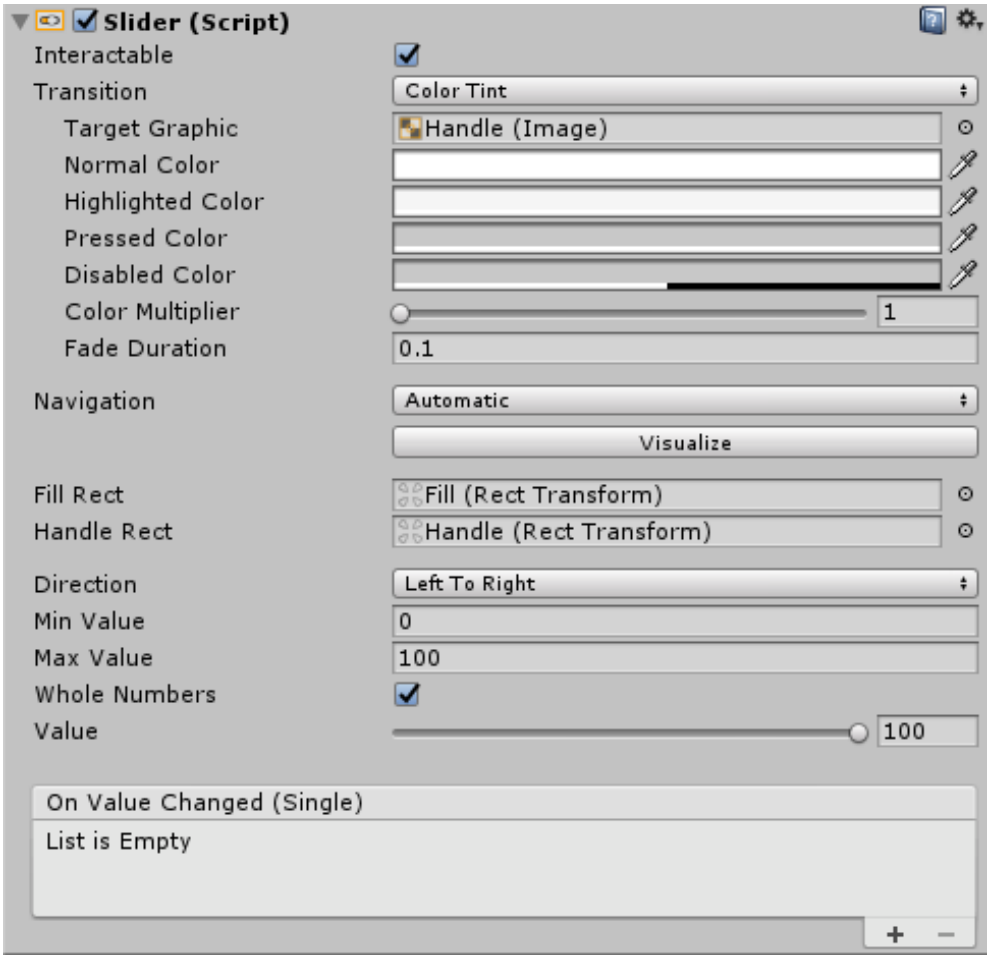
2.3.1.5.1.1. SFX SLIDER

The SFX slider allows the user to click and drag a button to change the value inside the slider. This is then stored and implemented into the game itself.



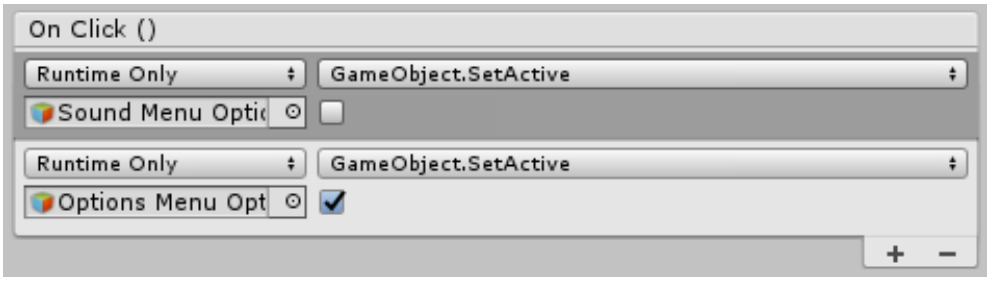
2.3.1.5.1.2. MUSIC SLIDER

The music slider allows the user to click and drag a button to change the value inside the slider. This is then stored and implemented into the game itself.



2.3.1.5.1.3. BACK BUTTON

The *Back*-button is an OnClick event, turning the visibility of the *Sound* menu to false and the visibility of the *Options Menu* to true.



2.3.1.6. *Quality Menu Implementation*

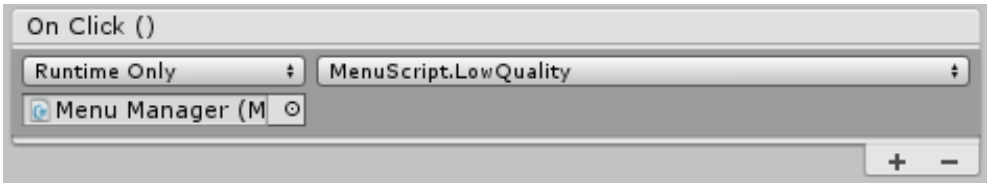
Allow the user to control and maintain:

- **Low quality button:** allows the user to control the quality of the game, setting it to low graphics.
- **Medium quality button:** allows the user to control the quality of the game, setting it to medium graphics.
- **High quality button:** allows the user to control the quality of the game, setting it to high graphics.
- **Back button:** allows the user to leave the *Quality Menu* and return to the *Options Menu*.

2.3.1.6.1. Implementation & Screenshots

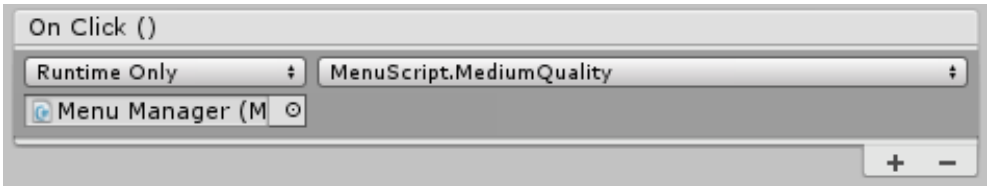
2.3.1.6.1.1. Low Quality Button

The low-quality button allows the user to change the quality of the game's graphics, using an OnClick event, which calls on a script to change the settings.



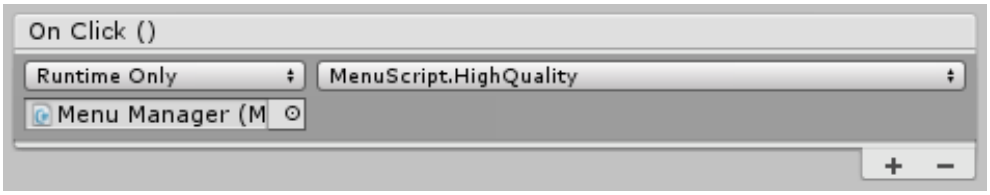
2.3.1.6.1.2. Medium Quality Button

The medium-quality button allows the user to change the quality of the game's graphics, using an OnClick event, which calls on a script to change the settings.



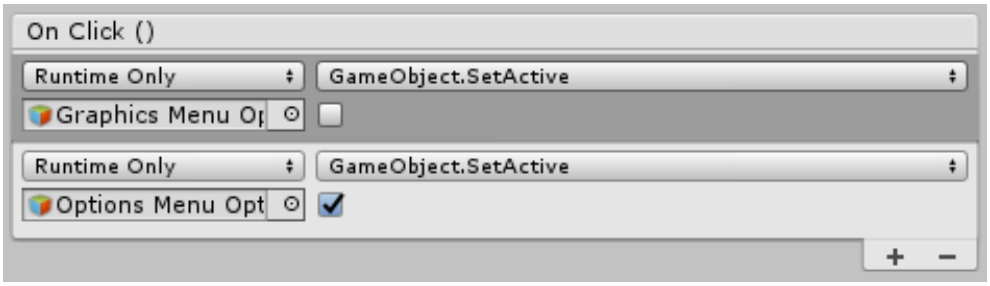
2.3.1.6.1.3. High Quality Button

The high-quality button allows the user to change the quality of the game's graphics, using an OnClick event, which calls on a script to change the settings.



2.3.1.6.1.4. BACK BUTTON

The *Back*-button is an OnClick event, turning the visibility of the *Sound* menu to false and the visibility of the *Options Menu* to true.



2.3.1.7. GAMEPLAY MENU IMPLEMENTATION

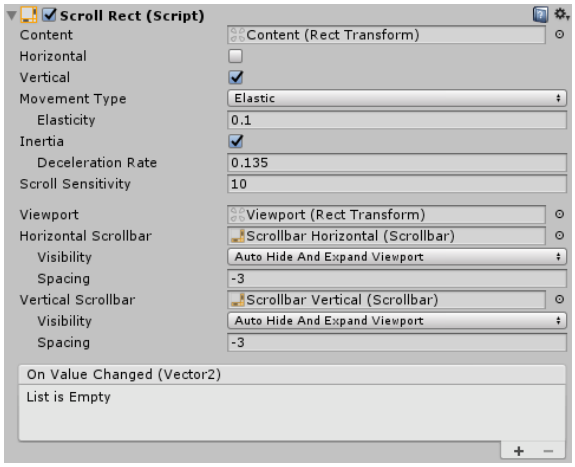
Allow the user to control, maintain and implement:

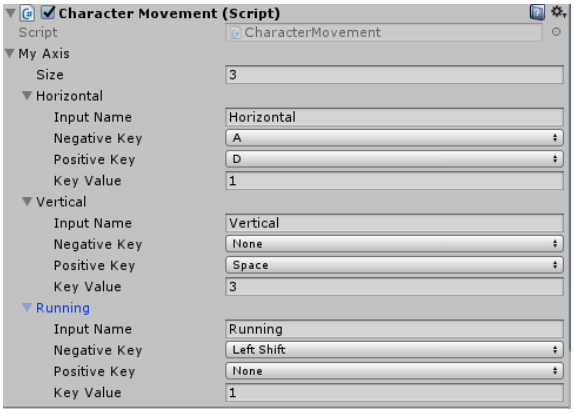
- **Control scroll-view:** allows the user to view the controls of the game using a scrolling viewport.
- **Back button:** allows the user to leave the *Gameplay Menu* and return to the *Options Menu*.

2.3.1.7.1. IMPLEMENTATION & SCREENSHOTS

2.3.1.7.1.1. CONTROL SCROLL-VIEW

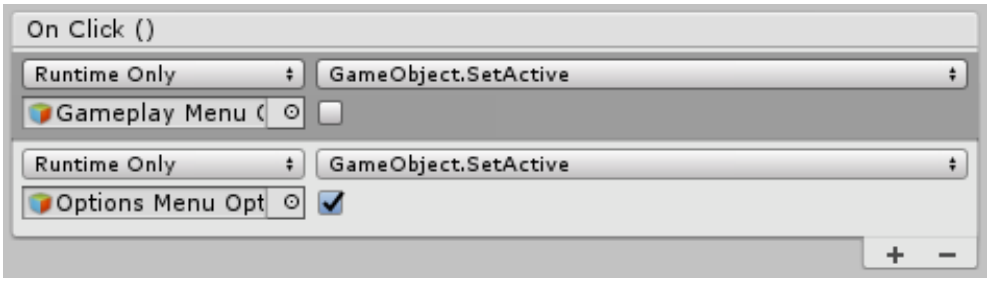
The control scroll-view allows the user to view the controls of the game before they begin. This allows the user to have basic knowledge of the controls. It contains a viewport full of different panels, each of which have different controls inside.





2.3.1.7.1.2. BACK BUTTON

The *Back*-button is an OnClick event, turning the visibility of the *Gameplay* menu to false and the visibility of the *Options Menu* to true.



2.3.2. In-Game Systems Implementation

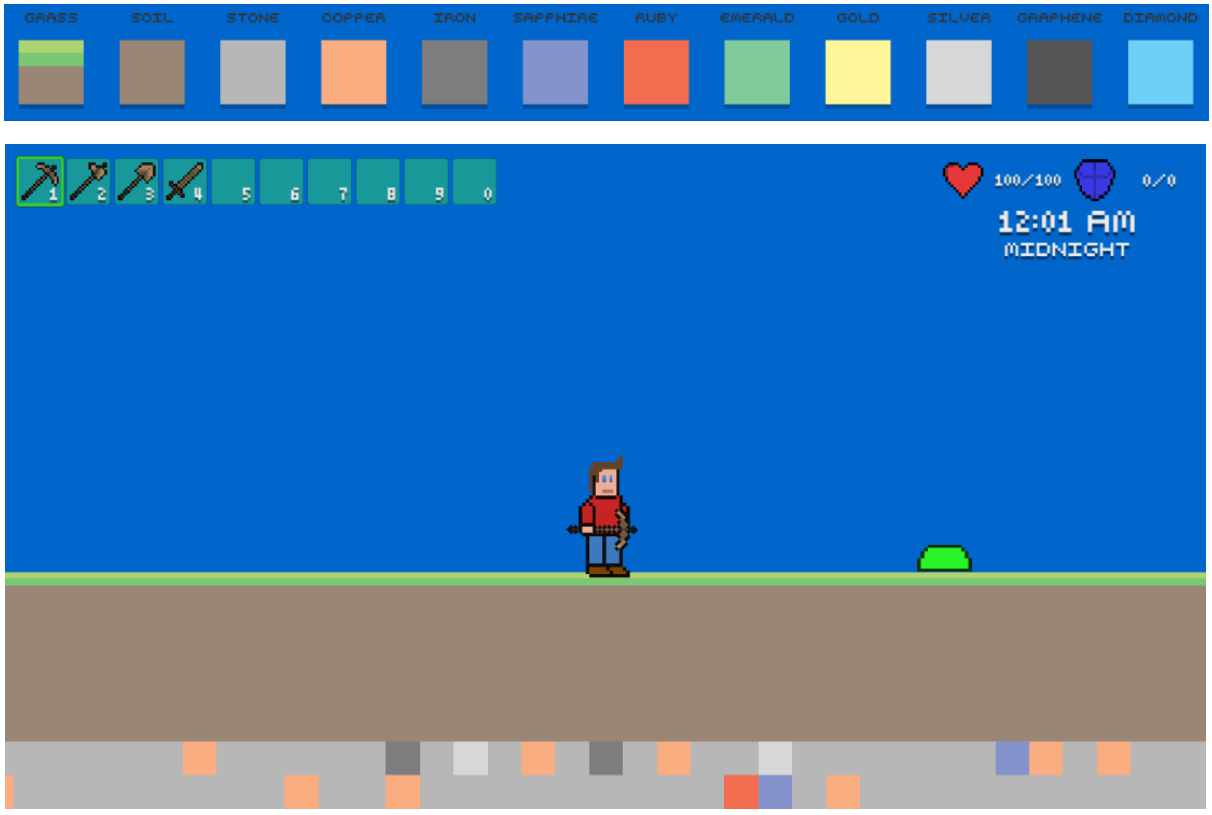
In this section, we will cover the in-game systems – which are vital to the overall gameplay and user experience.

2.3.2.1. World Generation Implementation

When you begin a game in TerraCraft, each time you begin, a layout of blocks will fill up the world, and every time, each one will be different.

2.3.2.1.1. Implementation & Screenshots

The script used for the world generation generates approximately 100,000 blocks, each one layered on top of each other, starting off with grass, soil, stone and a combination of different minerals, i.e. copper, iron, sapphire, ruby, emerald, gold, silver, graphene and diamond. And inside each of these blocks, they contain their own properties, i.e. name, location, collider and sprite.



The world generation uses a probability algorithm which decides what blocks can be generated. A random number is called by a variable and whatever number is picked, chooses the block being placed. In this circumstance, the rarer blocks have a smaller probability of being placed than common blocks. This is what gives each world its own unique property.

```
void Awake()
{
    BlockLocation = StartingBlockLocation;
    for (int i = 0; i < (BlockRowNumber / 10); i++)
    {
        for (int j = 0; j < BlockRowNumber; j++)
        {
            if (i < 1)
                LayGrass(i, j);
            else if (i < 5)
                LaySoil(i, j);
            else
                LayOtherBlocks(i, j);
            BlockLocation.x += BlockSpacing;
        }
        BlockLocation.y -= BlockSpacing;
        BlockLocation.x = StartingBlockLocation.x;
    }
}

void LayGrass(int row, int column)
{
    CreateBlock("Grass", row, column);
}

void LaySoil(int row, int column)
{
    CreateBlock("Soil", row, column);
}

void LayOtherBlocks(int row, int column)
{
    int randomNum = Random.Range(1, 1000);

    //  Stone
    if (randomNum < 800)
    {
        CreateBlock("Stone", row, column);
    }
    //  Copper
    else if (randomNum >= 800 && randomNum < 900)
    {
        CreateBlock("Copper", row, column);
    }

    //  Iron
    else if (randomNum >= 900 && randomNum < 950)
    {
        CreateBlock("Iron", row, column);
    }
```

```
void CreateBlock(string blockType, int row, int col)
{
    //R: 0 | C: 0 - Grass Block
    GameObject go = new GameObject("R: " + row + " | C: " + col + " - " + blockType + " Block");
    go.gameObject.layer = LayerMask.NameToLayer("Block");
    go.transform.parent = gameObject.transform;
    go.transform.localPosition = BlockLocation;
    SpriteRenderer renderer = go.AddComponent<SpriteRenderer>();
    BoxCollider2D collider = go.AddComponent<BoxCollider2D>();
    collider.size = new Vector2((go.transform.localScale.x) / 10, (go.transform.localScale.y) / 10);
    for (int i = 0; i < BlockObjects.Count; i++)
    {
        if (BlockObjects[i].name.Contains(blockType))
        {
            renderer.sprite = BlockObjects[i];
        }
    }

    TotalBlocks++;
}
```

2.3.2.2. TIME IMPLEMENTATION

For this game, being a survival RPG, time was implemented to allow for a day/night cycle, which, unfortunately, wasn't developed into the game. But the time was still implemented into the game, alongside with the periods of the day. It would've allowed for easier sunrise/sunset animations and NPC spawning.

2.3.2.2.1. IMPLEMENTATION & SCREENSHOTS

The development of the time involves it taking 2 seconds real-time for it to increment 1 minute in-game. This means, for every 2 minutes in real-time, an hour passes in the game. This is used using a timer variable, which increments using Time.deltaTime, a real-time clock (as the game updates 30 times every second), and when it hits the 2 second mark, it increments the game time and resets the timer, effectively going forward.

```
void IncreaseTime()
{
    Minutes++;
    if (Minutes > MaxMinutes)
    {
        Minutes = 0;
        Hours++;
        checkDayPeriod();
        if (Hours > MaxHours)
        {
            Hours = 0;
        }
    }

    WorldTime.updateTime(getTime(WorldTime.getClockType()));
}
```

But when the game-clock hits a certain hour, the period of the day changes, i.e. midnight into dawn, dawn into morning, etc. These periods of the day help spawn different enemies into the game, varying difficulty.

```
//  Periods of the Day; in a 24-hour clock format
const int Midnight = 0;      //  00:00 or 12:00 AM
const int Dawn = 4;          //  04:00 or 04:00 AM
const int Sunrise = 6;       //  06:00 or 06:00 AM
const int Morning = 7;       //  07:00 or 07:00 AM
const int Afternoon = 12;    //  12:00 or 12:00 PM
const int Evening = 17;      //  17:00 or 05:00 PM
const int Sunset = 18;       //  18:00 or 06:00 PM
const int Dusk = 20;         //  20:00 or 08:00 PM
const int Night = 21;        //  21:00 or 09:00 PM
```

2.3.3. CHARACTER IMPLEMENTATION

This section will outline the all the features of the character of TerraCraft – the only character the user will play in the game. This is the most vital system, as the game focuses primarily on the character and without it, the game would be unplayable.

2.3.3.1. CHARACTER PROPERTIES SETUP IMPLEMENTATION

This section will cover the setup for the character's properties, which includes its health and armour

The properties setup implementation controls and maintains:

- **Character health:** sets and measures the health of the character.
- **Character armour:** sets and measures the armour of the character.

2.3.3.1.1. IMPLEMENTATION & SCREENSHOTS

2.3.3.1.1.1. CHARACTER HEALTH

The character's health is the most important property to the character. Without it, the character would either be dead or immortal.

When the game starts, the health is given the maximum property, which at the beginning is 100. This can be taken away by attacking NPCs. If the health reaches 0, the character will die and respawn.

```
//  Variables

int MaximumHealth, MaximumArmour;
int CurrentHealth, CurrentArmour;

//  Methods
void Awake()
{
    PropertiesUI = FindObjectOfType<CharacterPropertiesUI>();

    MaximumHealth = 100;
    MaximumArmour = 0;
    CurrentHealth = MaximumHealth;
    CurrentArmour = MaximumArmour;
}

public void TakeDamage(int damage)
{
    if (CurrentHealth > 0)
    {
        int tempHealth = CurrentHealth;
        CurrentHealth = tempHealth - damage;
    }
    PropertiesUI.UpdateHealth(CurrentHealth, MaximumHealth);
}
```

2.3.3.1.1.2. CHARACTER ARMOUR

The character's armour is like the character's health, but is less important. It helps with the survival of the character, allowing the NPCs to attack the armour before the health,

Armour was not implemented into the game, so I cannot show any screenshots or show how it was implemented.

2.3.3.2. CHARACTER MOVEMENT IMPLEMENTATION

All games have some sort of input to allow for interactivity, to allow the user to immerse themselves into the game. These inputs can be anything, i.e. mouse, keyboard, controllers, and they require functions for each key-press. For this project, a custom keyboard control was implemented, to allow users with different tastes of controls, i.e. WASD controls, Arrow controls, etc., and could change the keys with ease.

The movement implementation controls and maintains:

- **Custom key binding:** sets the keys used to trigger the character's movement.

2.3.3.2.1. IMPLEMENTATION & SCREENSHOTS

For this, a script was developed to create a custom axis, i.e. for horizontal and vertical movement. The custom axis is a *C#* class, which contains an input name, a positive and negative key, and a value.

The name would allow for easy access to the class, as it would be stored in a List<> object – please see *Control Scroll-View Implementation* for an example of this. The positive and negative keys allow for easy changes for the key presses, e.g. the positive key could be 'A'; used to move left and the negative key could be 'D'; used to move right. The value is used for the movement of the character, as it interacts with another script, allowing the movement speeds to change when walking or running.

```
54      void Movement()
55      {
56          HorizontalMovement = GetCustomAxis("Horizontal").getMovement();
57          VerticalMovement = GetCustomAxis("Vertical").getMovement();
58
59          //  Jumping
60          if (OnBlock)
61          {
62              if (Input.GetKeyDown(GetCustomAxis("Vertical").getKeyType("Positive")))
63              {
64                  HasJumped = true;
65                  OnBlock = false;
66
67                  CurrentMovementSpeed = JumpHeight;
68
69                  //transform.position += Vector3.up * (VerticalMovement * (CurrentMovementSpeed * Time.deltaTime));
70                  gameObject.GetComponent<Rigidbody2D>().velocity = new Vector2(0.0F, (VerticalMovement * CurrentMovementSpeed));
71              }
72          }
73
74          //  Moving
75          if (Input.GetKey(GetCustomAxis("Horizontal").getKeyType("Negative")))
76          {
77              if (Input.GetKey(GetCustomAxis("Running").getKeyType("Negative")) ||
78                  Input.GetKey(GetCustomAxis("Running").getKeyType("Positive")))
79              {
80                  CurrentMovementSpeed = RunningPace;
81              }
82              else
83              {
84                  CurrentMovementSpeed = WalkingPace;
85              }
86              transform.position += Vector3.right * (HorizontalMovement * (CurrentMovementSpeed * Time.deltaTime));
87          }
88          else if (Input.GetKey(GetCustomAxis("Horizontal").getKeyType("Positive")))
89          {
90              if (Input.GetKey(GetCustomAxis("Running").getKeyType("Negative")) ||
91                  Input.GetKey(GetCustomAxis("Running").getKeyType("Positive")))
92              {
93                  CurrentMovementSpeed = RunningPace;
94              }
95              else
96              {
97                  CurrentMovementSpeed = WalkingPace;
98              }
99              transform.position += Vector3.right * (HorizontalMovement * (CurrentMovementSpeed * Time.deltaTime));
100         }
101         else
102             return;
103     }
```

```
using System;
using UnityEngine;

 [Serializable]
public class CustomAxis
{
    [SerializeField]
    string InputName;
    [SerializeField]
    KeyCode NegativeKey, PositiveKey;
    [SerializeField]
    float KeyValue;

    public float getMovement()
    {
        if (Input.GetKey(NegativeKey))
            return -KeyValue;
        else if (Input.GetKey(PositiveKey))
            return KeyValue;
        else
            return 0.0F;
    }

    public String getInputName()
    {
        return InputName;
    }

    public KeyCode getKeyType(String type)
    {
        if (type.Equals("Negative"))
            return NegativeKey;
        else
            return PositiveKey;
    }
}
```

```
CustomAxis GetCustomAxis(string AxisName)
{
    CustomAxis returnedAxis;

    for (int i = 0; i < MyAxis.Count; i++)
    {
        if (MyAxis[i].getInputName().Equals(AxisName))
        {
            returnedAxis = MyAxis[i];
            return returnedAxis;
        }
    }
    return null;
}
}
```

2.3.3.3. CHARACTER MELEE IMPLEMENTATION

To interact with NPCs, the character can only attack them, once they are within a radius of the character. Once inside the radius, the character can deal damage to the NPC.

The character melee implementation controls and maintains:

- **NPC detection:** allows the character to detect an NPC and add it to a List<> to attack.
- **Attacking:** allows the character to effectively attack the NPC.

2.3.3.3.1. IMPLEMENTATION & SCREENSHOTS

2.3.3.3.1.1. NPC DETECTION

To allow for easy attacking, the NPCs need to be close to the character, to stop any long-range hits. This is done by adding them to a List<> when they (NPCs) enter the character's box collider. This trigger allows them to be added and when exited, they are removed. To stop any clone objects in the List<>, the NPC has a unique ID, assigned by Unity, and is compared to objects within the List<> to the objects to be added, and, if they exist within the List<>, they won't be added.

```
void OnTriggerEnter2D(Collider2D other)
{
    if (other.gameObject.layer.Equals(LayerMask.NameToLayer("NPC")))
    {
        GameObject tempObject = other.gameObject;
        if (EnemyObjects.Count >= 1)
        {
            for (int i = 0; i < EnemyObjects.Count; i++)
            {
                if (EnemyObjects[i].GetInstanceID().Equals(tempObject.GetInstanceID()))
                {
                    return;
                }
                else
                {
                    EnemyObjects.Add(tempObject);
                }
            }
        }
        else
        {
            EnemyObjects.Add(tempObject);
        }
    }
    else
    {
        return;
    }
}

void OnTriggerExit2D(Collider2D other)
{
    if (other.gameObject.layer.Equals(LayerMask.NameToLayer("NPC")))
    {
        for (int i = 0; i < EnemyObjects.Count; i++)
        {
            if (other.gameObject.name == EnemyObjects[i].gameObject.name)
            {
                EnemyObjects.RemoveAt(EnemyObjects.IndexOf(EnemyObjects[i]));
            }
        }
    }
    else
    {
        return;
    }
}
```

2.3.3.3.1.2. ATTACKING

When a NPC is detected and is within the box collider and the List<>, the character can attack anything that is within the List<>, with the left mouse button. The damage inflicted is the amount of damage a weapon can do, i.e. fists do small damage while swords do large damage. When attacked, the NPC will move towards the character – please see *NPC Movement Implementation*, and attack the player too – please see *NPC Melee Implementation*.

```
void Attack(int damage)
{
    if (CanAttack == true)
    {
        for (int i = 0; i < EnemyObjects.Count; i++)
        {
            EnemyObjects[i].GetComponent<NPCProperties>().TakeDamage(damage);
            EnemyObjects[i].GetComponent<Rigidbody2D>().AddForce(((EnemyObjects[i].transform.position - transform.position).normalized) * 100);
        }
    }
    else
    {
        return;
    }
}
```

2.3.4. NON-PLAYABLE CHARACTER (NPC) IMPLEMENTATION

This section will outline the all the features of the NPCs – which can be interacted with by the character. The only way the character can interact with the NPCs is to attack them, which will deplete their health until they die.

2.3.4.1. NPC PROPERTIES SETUP IMPLEMENTATION

This section will cover the setup for the NPC's properties, which includes its health and armour

The properties setup implementation controls and maintains:

- **NPC health:** sets and measures the health of the NPC.
- **NPC armour:** sets and measures the armour of the NPC.

2.3.4.1.1. IMPLEMENTATION & SCREENSHOTS

2.3.4.1.1.1. NPC HEALTH

The character's health is the most important property to the NPC. Without it, the character would either be dead or immortal.

When the game starts, the health is given the maximum property, which at the beginning is 100. This can be taken away by attacking NPCs. If the health reaches 0, the character will die and respawn.

```
public int CurrentHealth, CurrentArmour;

bool Friendly, Attacking, Dead;

//  Methods

void Awake()
{
    Name = gameObject.name;

    MaximumHealth = Random.Range(10, 25);

    if (Name.Equals("Slime"))
    {
        Friendly = false;
    }

    Dead = false;
}

void Start()
{
    CurrentHealth = MaximumHealth;
}

void LateUpdate()
{
    if(CurrentHealth <= 0)
        DestroyImmediate(gameObject, true);
}

public void TakeDamage(int damage)
{
    if (Attacking != true)
        Attacking = true;

    if (CurrentHealth >= 0)
    {
        int tempHealth = CurrentHealth;
        CurrentHealth = tempHealth - damage;
    }
}
```

2.3.4.2. NPC MOVEMENT IMPLEMENTATION

This section will cover the setup for the NPC's movement, which demonstrates how it moves throughout the game.

The movement implementation controls and maintains:

- **NPC movement:** sets and implements the movement of the NPC to the character.

2.3.4.2.1. IMPLEMENTATION & SCREENSHOTS

The implementation of the movement of the NPC is very simplistic, meaning that it only moves to the character, but only when it is interacted with, i.e. attacked – please see *Character Melee Implementation*.

```
//  Variables

readonly int WalkingPace = 1;

float MovementSpeed;

Vector2 CurrentPosition;

//  Methods

void Awake()
{
    MovementSpeed = WalkingPace / 1.25F;
}

void LateUpdate()
{
    MoveNPC();
}

void MoveNPC()
{
    if (gameObject.GetComponent<NPCProperties>().IsAttacking() != true)
    {
        return;
    }
    else
    {
        MoveNPCToPlayer();
    }
}

void MoveNPCToPlayer()
{
    Debug.Log("Moving to Player");
    if (GetComponent<NPCProperties>().GetCurrentHealth() > 0)
    {
        CurrentPosition = GetPosition(gameObject);
        Vector2 characterPosition = GetPosition(FindObjectOfType<CharacterProperties>().gameObject);
        if (characterPosition != CurrentPosition)
        {
            float step = MovementSpeed * Time.deltaTime;
            Vector3 distance = characterPosition - CurrentPosition;
            if (distance.magnitude < step)
            {
                transform.position = characterPosition;
                return;
            }
            distance.Normalize();
            distance *= step;
            transform.position += distance;
        }
    }
}
```

When the NPC is attacked, it gets the character's current position continuously, so it can keep moving when the character moves. When the NPC reaches the character, it will attack – please see *NPC Melee Implementation*.

2.3.4.3. NPC MELEE IMPLEMENTATION

Like the character, the NPC can also deal its damage. This is done differently, as it doesn't need to detect the character, as the NPC only moves when the character interacts – causing the NPC to move towards the character.

The melee implementation controls and maintains:

- **Attacking:** allows the NPC to deal damage to the character

2.3.4.3.1. IMPLEMENTATION & SCREENSHOTS

The implementation was done very simply – if the box collider of the NPC hits off the box collider of the character, the NPC would deal its damage to the player, as well as knocking the player back.

```
int Damage = 1;

void OnCollisionEnter2D(Collision2D other)
{
    Debug.Log(other.gameObject.name + " is detected!");
    Debug.Log("Attacking is " + GetComponent<NPCProperties>().IsAttacking());

    if (other.gameObject.name.Equals("Character"))
    {
        if (GetComponent<NPCProperties>().IsAttacking() != false)
        {
            Attack(other.gameObject, Damage);
        }
    }
}

void Attack(GameObject go, int damage)
{
    Debug.Log("Dealing Damage!");
    //  Knockback
    go.GetComponent<Rigidbody2D>().AddForce(((go.transform.position - transform.position).normalized) * 100);
    go.GetComponent<CharacterProperties>().TakeDamage(damage);

}
```

2.4. TESTING

2.4.1. OVERVIEW

Testing is necessary to gaming, as it gives an insight to the functionalities of the game, ensuring that they are working, that there are no errors caused and that no crashes occur.

When testing TerraCraft, these tools were used:

- **Unity Runtime Simulation:** Unity has a built-in simulator that allows the developer to test the game and its functionalities.
- **Unity Test Scenes:** Built solely to test out the layout and features of systems before being implemented into the main game.
- **TerraCraft beta:** a latest build version of the TerraCraft project.

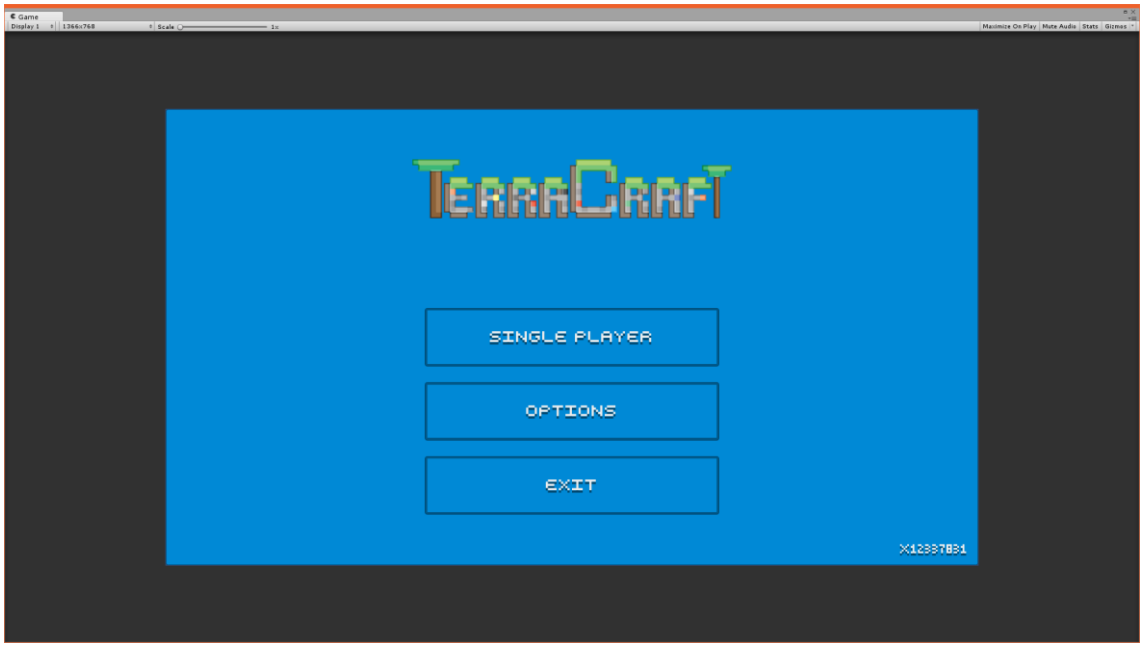
For the main testing – both usability and public testing, I didn't get a chance to release it and/or create surveys for anyone to participate in.

2.5. GRAPHICAL USER INTERFACE (G.U.I) LAYOUT

2.5.1. MAIN MENU

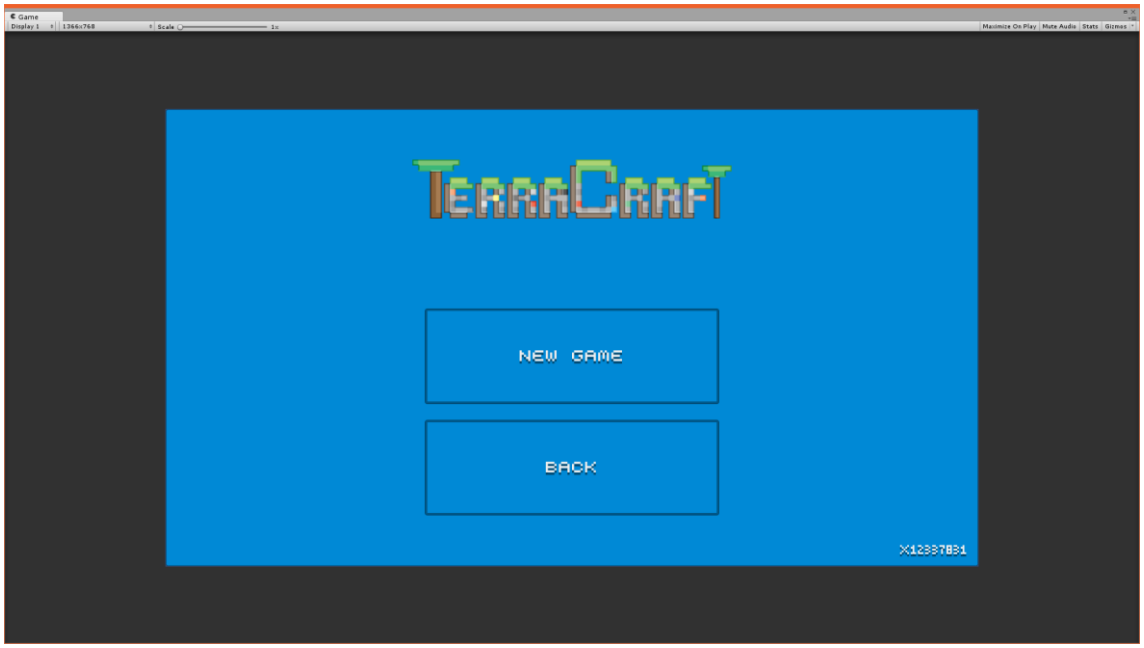
This is the first interface the user will be presented with. It has three buttons to interact with:

- Single Player – allows the user to go to the Single Player menu
- Options – allows the user to go to the Options menu
- Exit – allows the user to exit the game



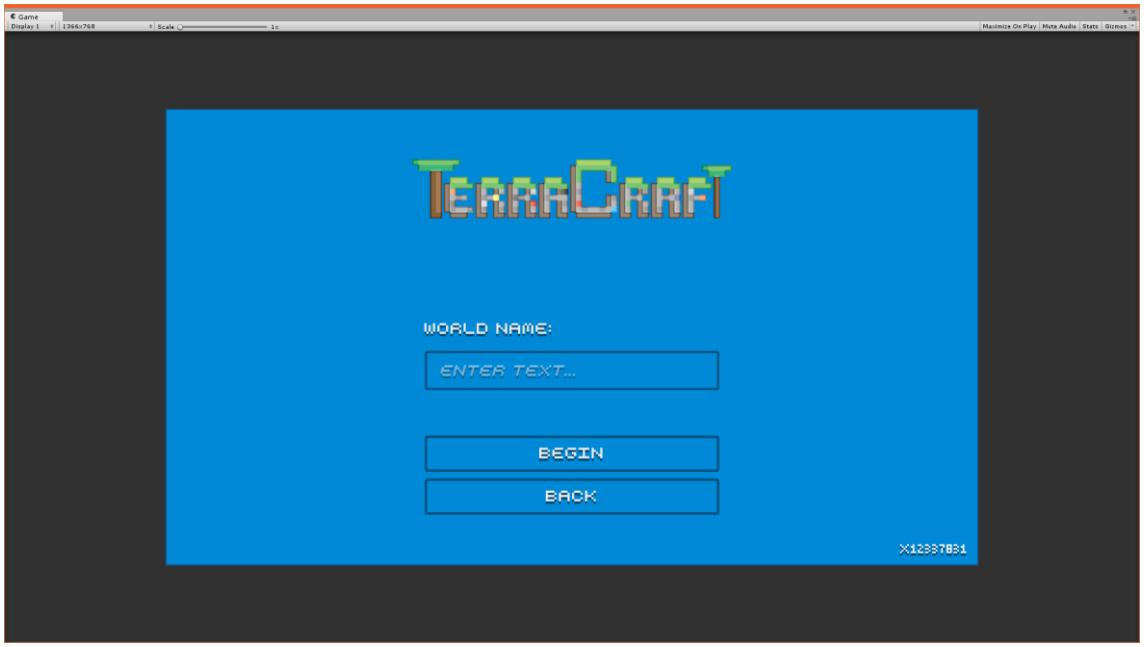
2.5.2. SINGLE PLAYER MENU

This interface allows the user to either start a new game or to go back to the main menu:



2.5.2.1. NEW GAME MENU

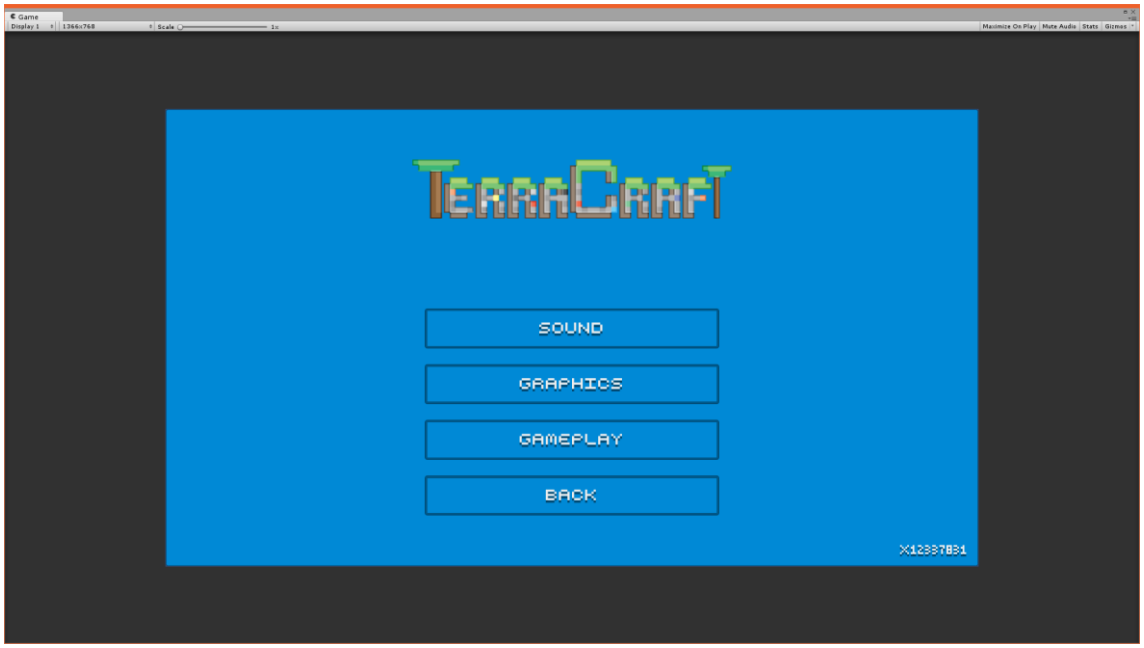
This interface allows the user to name the world and to effectively begin the game, or to go back to the single player menu.



2.5.3. OPTIONS MENU

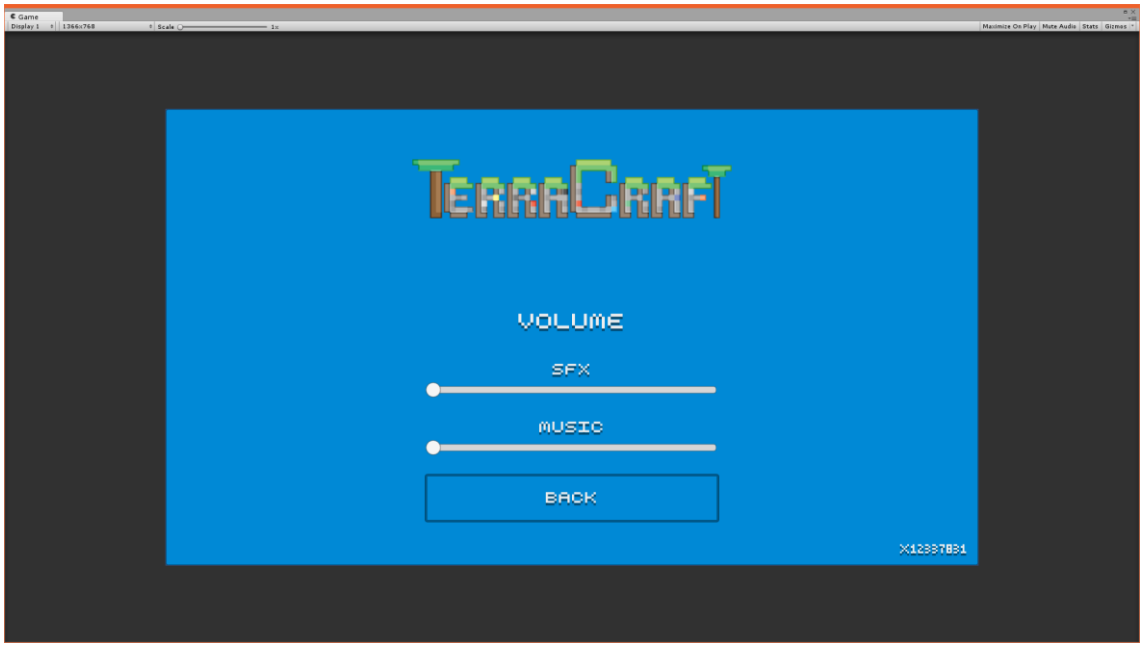
This interface has four buttons to interact with, allowing the user to change the settings of the game. These buttons are:

- Sound – allows the user to go to the sound menu
- Graphics – allows the user to go to the graphics menu
- Gameplay – allows the user to go to the gameplay menu
- Back – allows the user to go back to the main menu



2.5.3.1. SOUND MENU

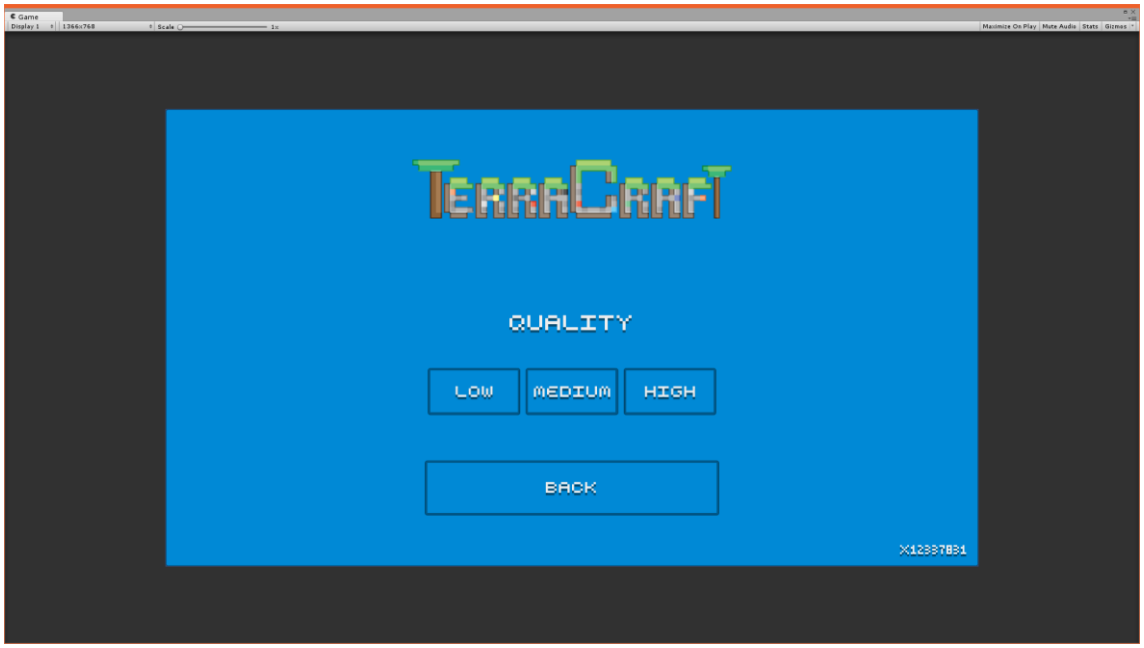
This interface has two sliders and one button, allowing the user to change the volume of the game or go back to the options menu



2.5.3.2. GRAPHICS MENU

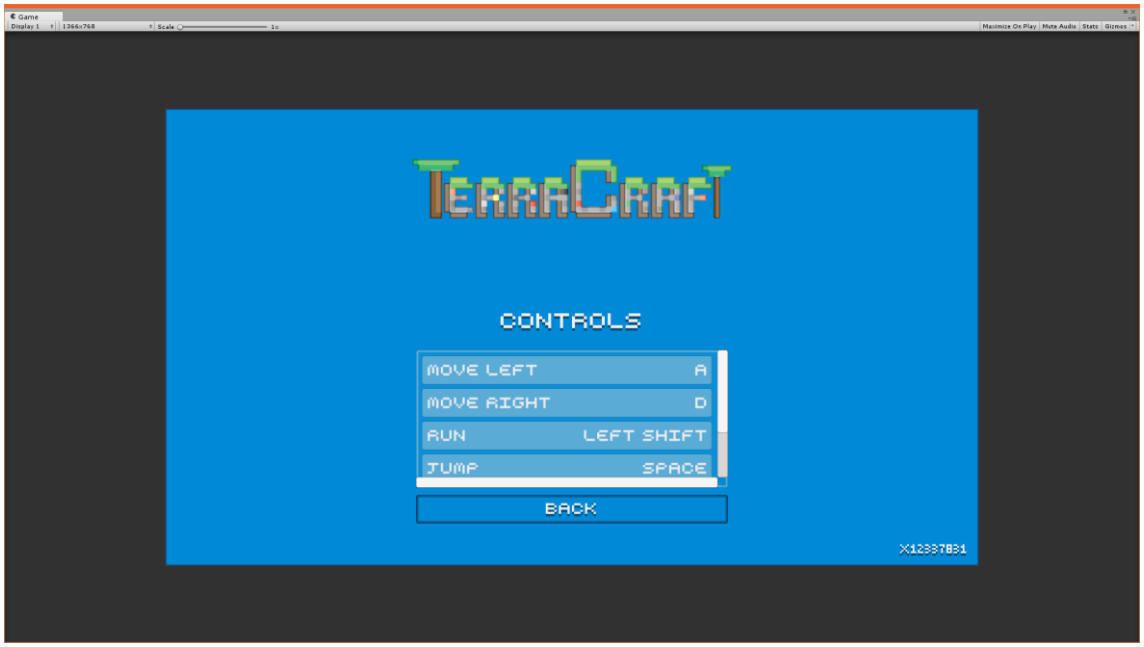
This interface has four buttons, allowing the user to go back to the options menu or to change the graphics settings from either:

- Low Quality
- Medium Quality
- High Quality



2.5.3.3. GAMEPLAY MENU

This interface has a scroll-view and a button, allowing the user to view gameplay controls or to go back to the options menu



2.5.4. EXIT GAME CONFIRMATION MENU

This interface has two buttons, allowing the user to confirm whether they would like to exit the game.

3. CONCLUSIONS

Over the course of this project, I've learned that major projects like these require more time to develop and design to a professional standard. Due to my circumstances, I couldn't dedicate myself as such and it resulted with a project that I'm not satisfied with – as it doesn't show my true potential as a game developer.

I came into many obstacles during the development:

- **Laptop breaking down:** during the middle of the development, my laptop (now old laptop) began breaking down, i.e. slow response, battery errors, cooling fan failure, so I had to back-up every file and transfer it to the new laptop. This cost me over a month in development time.
- **Unity memory crash:** when I was developing the mining feature for TerraCraft, I came across an error that wasn't easily fixed. The mining technique was like the melee feature – it used a List<> to see what was nearby; in this case, it was blocks. So, nearby blocks were added to the List<>, compared to see if they weren't duplicates, removed if they weren't in the nearby boundaries and being able to be interacted with. This caused a major Unity crash – which lead developers contacting me, letting me know about the huge memory error; meaning that it wasn't able to process the functions properly.

4. FURTHER DEVELOPMENT OR RESEARCH

The way the development of TerraCraft went so far, I would like to further develop this game, to include all the functionalities it was promised in the project proposal. I would also like to further develop it in different areas, i.e.:

- **Missions:** add missions to the game, to allow the user to immerse themselves to obtain items and interact with NPCs.
- **Achievements:** add achievements to the game; fitting in well with missions and that can be obtained when finishing a mission and/or doing certain tasks, i.e. crafting, building, etc.
- **Multiplayer:** allow users to play in a co-op multiplayer LAN by using the Photon Unity Network (P.U.N) to allow users to play a local game together.
- **Controller support:** allow different forms of input for the game, so that it's not just limited to a keyboard/mouse duo, i.e. PS4 controller, Xbox One controller, etc.
- **Cloud support:** develop the game so it could be supported in the cloud to make it a multiplatform game and store saved data online, rather than locally.
- **Character customisation:** allow the user to customise their character before beginning a new game. This will allow for a unique playthrough for the user.
- **Bosses:** implement boss battles to make the game more difficult, yet, more exciting, as they could drop important items.
- **World generation optimisation:** build a faster and more reliable world generation algorithm that can place blocks in an instant, rather than waiting for it to load.

For the research of this project, I would like to release various surveys, i.e.:

- **Questionnaires:** to allow the public to voice their opinions on what they'd like to see in this game
- **Surveys:** based on what has been developed, what has been the best/worst features and what can be done to better them.

Developing games is a huge hobby of mine. Since my internship at Defiant Games, I've developed many games that gave me a lot of knowledge in the gaming area, as well as graphic design.

BIBLIOGRAPHY

Lyons, T., 2014. *Digit launches biggest budget video game in Irish history.* [Online] Available at: http://www.irishtimes.com/business/digit-launches-biggest-budget-video-game-in-irish-history-1.1918371
[Accessed 17 October 2016].

Whelan, F., 2015. *DIGIT Game Studios to create 40 new jobs in Ireland.* [Online] Available at: http://www.independent.ie/entertainment/games/digit-game-studios-to-create-40-new-jobs-in-ireland-31261550.html
[Accessed 17 October 2016].

APPENDIX

1. PROJECT PROPOSAL

1.1. OBJECTIVES

The main objective of this project is to create, design and develop a fully-functional 2D-gaming application using *Adobe Photoshop* to design the look of the game and the *Unity* game engine to develop and program the game. This game will be aimed mostly towards the PC gaming market, this includes Windows, Linux and Mac operating machines. I'm aiming to ensure that I have the most up-to-date version of this software and create a large-scale game, playable for many gamers. I plan to undertake a learning opportunity of creating a large project in gaming as it's an area of expertise that I'd like to build on and pursue a career in. This project, I feel, will allow me to show what I've learned in the course, as well as the internship in 3rd year, and show a great deal of diversity to my degree. Over the years in the BSHC course, many the projects I've been involved in have been related to websites, Java applications or databases, and I feel that this software project will give me an opportunity to make a gaming application that will show my capabilities as a fellow gamer and as a game designer/developer.

The type of gaming application I will be making is a paper 2D side-scroller survival sandbox role-playing game called *TerraCraft*. A role-playing game or *RPG* is a style of game where the player starts off, creating a character that the game designs for them, and with nothing to begin with but a few basic tools in a world that is generated for them, they'll start to develop skills that will affect their gaming experience. The objective of the game is to survive. By any means necessary. This could be achieved by crafting tools, armour and weapons to protect yourself, building houses to stop creatures and enemies from reaching you, and by upgrading your own perks or weapons, eventually becoming stronger and resilient. Non-playable characters, or *NPCs*, will be available nearby to help aid you by selling certain items that could play a helpful role in the way the game is played.

I'm also planning to make this game with the use of 2D graphics, or *sprites* as it's known in the *Unity* game engine, in a *'pixel'* form. It will give of a minimalistic look, but will look aesthetically pleasing for players. It will also enable me to create and design a more unique world with certain aspects and terrains. I'm currently planning to use my own designs for this game, using programs like *Adobe Photoshop CC 2015.5* or *Paint.NET*. Yet, I have small projects, created in spare free time, to help me in certain ways to think of the logic inside this game, like character movements, block-building and using weapons. The 'plan' for this project is, once I've created the basic game logic, that I deem myself comfortable with, I'll work on developing a better experience for the player, by creating small quests, building a better UI system and allowing character customisation and upgradables. I'm also aiming to make use of a certain technique called *parallax scrolling*. I will learn this by researching online forums and videos/tutorials. The *parallax scrolling* gives off a 3D-like effect in a 2D game, where the background appears to move at a slower rate than the foreground. This gives off an illusion, to the player, that there is a distance, i.e. clouds, sunrise/sunset, mountains in distance, etc.

The development of the game *TerraCraft* will be made in the *Unity* game engine using *C#* coding, given that I worked and developed with it for 6 months in *Defiant Games* as part of the Work Placement module in the 2nd semester of 3rd year. I've been building on that for the past year, developing smaller projects, teaching myself better ways to code logic and the experience has

given me vital concepts of planning the code and its organisation and, I hope, that it will show during the development of this project.

1.2. BACKGROUND

Over the past 30 years, video games have become an integral part of our culture. It's seen remarkable growth and is a multi-billion-dollar industry with all the platforms that it contains. From PCs, consoles, mobile and tablets, gaming is everywhere and it is a great getaway to immerse yourself into the game. Over the past few years, gaming companies have been created and are opening new jobs and new opportunities to prove what capabilities you can achieve. Digit Gaming is an example of a gaming company. Based in Dublin, Digit Gaming is a cross-platform game development company that recently hit headlines for having the "biggest budget video game in Irish history" (Lyons, 2014). The game they released, "*Kings of the Realm*", published with *Kabam*, was picked by Google to be featured on the Play store to millions of people across the globe. With this, they opened 40 new jobs after an investment from a multi-million-dollar company *Scopely* (Whelan, 2015), which is big news because it offers game developers, like myself, an opportunity to apply and possibly acquire a job in the gaming industry.

I reason why I chose to do this project is due to my keen interest in gaming and game design. I frequently engage my free time to playing games and I feel that I could start to develop a game that everybody could enjoy. This project was also of interest to me, as the logic of the game is based off the idea of other games including "*Terraria*" by *Re-Logic*, "*Minecraft*" by *Mojang* and "*Starbound*" by *Chucklefish Games*. I've spent countless hours playing these games and they always keep me interested and focused, as they have endless possibilities to what you can do. This project was also of interest to me from a technical point-of-view since it'll be made using 2D graphics that'll be designed in *Adobe Photoshop CC*. I've always had an interest in 2D graphics as it looks like it's minimalistic but, in fact, are complex but beautifully well represented, if done right.

The reason why I've chosen a 2D role playing game is mainly because the genre greatly intrigues me. It is my favourite genre of gaming and I feel that I would enjoy the opportunity and the challenges that this game can offer. The genre of RPG gaming has a vast number of players who enjoy games like the "*Grand Theft Auto*" series by *Rockstar*, "*The Elder Scrolls*" series and the "*Fallout*" series by *Bethesda* and the "*Final Fantasy*" series and the "*Life is Strange*" series by *Square Enix*. These games are unique in their own way, offering the players hundreds of hours of gameplay due to storylines, collectables and achievements. The players will also have to think logically and quickly as they've to manage their characters' health along with XP, coins and items in a world filled with weak and strong enemies, dwelling to kill them. Over time, the player can strengthen themselves with perks and upgrades which can have major effects on the game. Engaging in quests from NPCs can reward the player with better equipment or XP and coins. It is a very engaging and highly enjoyable style of gaming and I feel that I could make an incredible contribution to this project.

In terms of development, I've chosen to create this project using the Unity game engine. I felt that Unity can offer me a countless number of features that could benefit the project in terms of finding solutions to in assets or online tutorials. The Unity game engine, recently, have put most of their development into 2D, giving developers more options in creating games. In Unity 2D, developers can create simple 2D characters and move them around the screen easily with the new functions made available. Functions like Rigidbody2D can give a two-dimensional gravity effect, Box Colliders give an object a 2D collider to detect other objects and Raycasting2D to detect, in two-dimensional space, where the mouse is and where it clicked, whether on a game object or on a UI object. I find that using the Unity game engine will make the progress of the project much easier, as I've also had experience with this in the 6-month internship with *Defiant Games* as part of the Work Placement module in the 2nd semester of 3rd year.

1.3. TECHNICAL APPROACH

1.3.1. RESEARCH

Before beginning this project, I did a great deal of research beforehand, thinking about what concepts to include in this project and whether I had enough time to do so. My main priority is to fully create and develop a project that I would love, so, from learning about the *Unity* game engine in the internship at Defiant Games, I decided that it'd be best if I could continue using *Unity* to develop the project. Having decided that I'm making a game, I spent a lot of time researching on Google, YouTube and through various gaming forums like the *Unity* forums or *StackOverflow* about the implementations of game logic and ideas that I could add to the development of the game. The purpose was to give me inspiration on the game and what to expect during the development.

I also did some research regarding the style of game I'm making – a paper 2D side-scroller survival sandbox RPG. This will take a considerable amount of time to complete, but, I plan on making sure that it doesn't reach beyond my capabilities. Before development, I laid out all the features and concepts that I felt would be necessary for this project, gaining inspiration from other games, YouTube videos and tutorials.

1.3.2. INTERFACE REQUIREMENTS

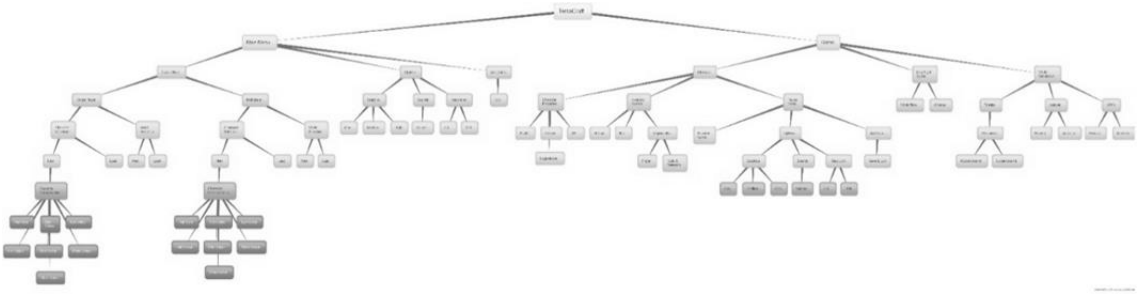


Figure 1. Mind-Map of TerraCraft Requirements

1.3.3. IMPLEMENTATION

The implementation of this project will be carried out using the *C#* programming language within the *Unity* game engine. I aim to have three development stages over the duration of this project, building the project without losing focus of the tasks ahead. I'll start with making a basic concept of the game, having the player move around the screen and from then on, I'll implement most the features that I'd want in the game. I expect to have the basics done by the end of the 1st semester. I'll then move onto implementing more advanced functions later, possibly during the second semester, maybe a little bit earlier. These functions include the Day/Night Cycle, Parallax Scrolling, Multiplayer, etc. During the development of the project, I'll be designing the graphics of the whole project with the *Adobe Photoshop CC* graphics software tool, including sprites and fonts. During the final stages of the project, I'll be testing the game by myself, asking friends to participate and by releasing it on *Facebook*, asking users what they think of the game, which will aid in the development by adding optimisations and fixes.

1.4. SPECIAL RESOURCES REQUIRED

As this project is a gaming application, I don't expect the need for any special resources to undertake in the development of the project. The hardware on which I'll create it will need to minimum/recommended requirements of having an up-to-date set of components and assets. This is applicable to the Graphics Processing Unit (*GPU*). If this is not powerful enough to run the project, then I'd need to reconsider the hardware on which to create/run the game. Now however, I don't expect any issues to arise about exceeding the hardware resources on my computer.

I'll make use of several tutorials on *Udemy*, *Pluralsight* and *YouTube* or research topics on forums like *StackOverflow* to learn and implement certain aspects of concepts into the game. These will be referenced in the project when and if they're used.

1.5. PROJECT PLAN

I have attached a Gantt Chart, made using *Microsoft Project* with details of the implementation of steps and timelines of this projects. See the attached .mpp file for more details:

Project Plan.mpp

1.6. TECHNICAL DETAILS

The programming language I'm planning to use for this project is C#. This is one of the languages the Unity game engine offers to use, others being JavaScript and Boo, for the creation and development of games. The IDE which I'll do all my coding in is *Microsoft Visual Studio 2013*. Unity has its own IDE built-in called *MonoDevelop* but it doesn't appeal to me, so I've overridden the settings to use *Microsoft Visual Studio 2013* as it's easier to see errors, format and write code in.

The game will make use of the full spectrum of the asset library, free or paid on offer, within the Unity Asset Store. These will provide a range of methods to creating and aiding game concepts, such as the use of debugging, which appear inside the game log.

1.7. EVALUATION

I'll evaluate the project through the process of releasing the game to many Facebook users to test and give feedback, thus giving me knowledge of the development on the game and what errors, bugs and concepts that need to be fixed.

Using Facebook will give a wide range of statistics, showing what countries are opted in the testing process, the number of hours spent playing, feedback and rating of the game. I will also respond to users, asking what types of concepts they like/dislike and what they would like more to see, and, if I have the time, I could add a few of the concepts, if there are any.

2. REQUIREMENTS SPECIFICATION

DOCUMENT CONTROL

Date	Version	Scope of Activity	Prepared	Reviewed	Approved
14/10/2005	1	Create	AB	X	X
21/10/2005	2	Update	CD		

DISTRIBUTION LIST

Name	Title	Version
Eamon Nolan	Lecturer	
Anu Sahni	Supervisor	

RELATED DOCUMENTS

Title	Comments
Project Proposal	
Reflection Journals	

2.1. INTRODUCTION

2.1.1. PURPOSE

The purpose of this document is to set out the requirement for the development of TerraCraft – a game made with the Unity Game Engine. The game itself will be a 2D side-scroller sandbox survival RPG style of game – this means that the player will create their own character, using tools that I will make and then, will be placed into a generated world where they can destroy and place blocks, craft items, and survive from spawning enemies, which will spawn randomly, each having their own strengths and weaknesses. When you defeat an enemy, they will drop objects. The objects that are dropped can be either: items, coins or XP. XP is essential to the character. It will determine what level he/she is, allowing upgrades for the player.

The game will include many features that exist in many games today, including a Day-Night Cycle, a Mini-Map, Inventory System, Multiplayer, Crafting System and a Save/Load System. In the Day-Night Cycle, the sun and moon will rise and set in the sky at specific times of the day. In the daytime, the sun will rise and the sky will go from a dark black to a light blue – signalling the morning-afternoon. As the day progresses, the sky will go from a light blue, to a mild red-orange to a dark black – signalling the evening-night. The Inventory System and Crafting System will be connected – as you will be able to carry around items in the inventory and craft with those items.

The intended customers are gamers who love the style of 2D pixel gaming and those who love survival RPGs. This game will appeal to fans who love games like: **Terraria**, **Minecraft**, **Starbound** and **Edge of Space**. This game will have its own world generated, so each time it's played, the look of the environment will always be different – therefore the game will appeal to any players who enjoy a large setting they can explore. It will also be specifically aimed towards the PC gaming market, with the potential of transferring to consoles or even mobile. Potential distribution platforms could include Steam – with their Greenlight system, GreenManGaming, Indie gaming sites or possibly a physical copy of the game, selling in retailers like HMV or GameStop. If the game was to make it to a commercial space and needed funding – it could be Kickstarted by online contributions and could be investigated further to guarantee the progress of the development.

2.1.2. PROJECT SCOPE

The scope of the project is to develop a fully functional game within the Unity Game Engine using C# in Microsoft Visual Studio 2013 and creating images using the photo-editing tool Adobe Photoshop CC. The game will have a primary focus on the use of 2D graphics.

I have been involved in research with users on forums on sites like Unity, StackOverflow and others, to enquire about in-game requirements and specifications. The project will use the following requirements:

Requirement 1.	New Game
Requirement 2.	Save Game
Requirement 3.	Load Game
Requirement 4.	Quit Game
Requirement 5.	Start Tutorial
Requirement 6.	Collect Resources
Requirement 7.	Craft Items
Requirement 8.	Kill Enemies
Requirement 9.	Upgrade Player Skills

Scheduling: This game will be made within an eight-month period and the project will be broken down into three phases, as described in the *Gantt Chart* in the *Project Proposal*. These are the ***Planning***, ***Development*** and ***Testing*** phases.

Cost: This game will be generally free to make, with some exceptions of using paid game assets from the Unity Asset Store.

Software to develop Requirements: The requirements diagrams will be made using a free-to-use online tool known as Gliffy.

2.1.3. DEFINITIONS, ACRONYMS AND ABBREVIATIONS

2D – Two-dimensional: a perspective-based camera angle which only shows objects as 'flat'. It can move on two axes: the x-axis and the y-axis.

FPS – **F**rames **p**er **S**econd: a unit that measures display device performance. It consists of the number of complete scans of the display screen that occur each second. The faster it is, the better.

Microsoft Visual Studio 2013: allows easy interaction with scripts in Unity, either C# or JavaScript and easy-to-use coding techniques, e.g. predictive text, easy code formatting and highlighting coding errors.

RPG – **R**ole **P**laying **G**ame: a style, or genre, of gaming where the player takes on a role of a character and creates it in their own style, which is then used in the game and its world. The player will then become stronger over the course of game through defeating enemies and upgrading perks.

Sandbox – another style, or genre, of gaming, which basically means 'open world' – a game where the player can roam freely or can interact with the world, e.g. destroying blocks, placing blocks, etc.

XP – E**x**perience **P**oints: in-game bonus that defines what level, or how strong, you are. The more experience you have, the better, as it contributes towards upgrading the players' perks.

2.2. USER REQUIREMENTS DEFINITION

The user requirements are as followed:

- The users' machine must meet the minimal requirements of the Unity Game Engine. Those requirements are:
    - Windows 7, 8 or 10 (developed in Windows 10),
    - Minimum of 2GB of RAM installed,
    - Dual-Core Intel/AMD processor, 2.0 GHz,
    - Intel HD Graphics 4000 or higher,
    - Desktop PC, Mac or Laptop.
- For best experience, the users' machine should meet the recommended requirements for the Unity Game Engine. Those requirements are:
    - Windows 7, 8 or 10, 64-bit,
    - 8GB of RAM installed,
    - Quad-Core Intel or AMD processor, 2.5 GHz,
    - NVIDEA GeForce GTX 970 graphics card or higher.
- The user must also have a functional keyboard and mouse.
- The user must have the latest, up-to-date build version of the game.

2.3. REQUIREMENTS SPECIFICATION

The requirements specification will outline several variables that the user will achieve while playing the game. That are as followed:

- This game will be designed to be enjoyable to play and will require some number of tutorials when the player faces a new feature of the game.
- The player should be able to begin a new game and start a tutorial.
- Having finished the tutorial, the player should make less errors per gaming session. The player can then decide whether to continue with the tutorial or whether to quit the tutorial.
- The player should be able to save the game whenever they decide to quit the game.
- The player should be able to load the game when selecting the world that they want to play in.
- The player should be able to quit the game anytime they desire at any stage.
- The player should be able to upgrade perks whenever they have the right amount of XP.

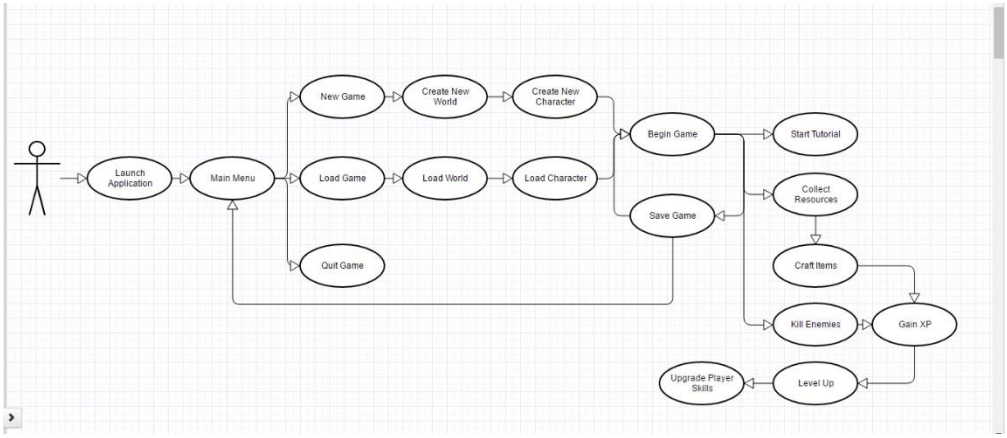
2.3.1. FUNCTIONAL REQUIREMENTS

The functional requirements define function(s) of the system – being a set of inputs and outputs, along with defining the behaviour. This will be supported with a section which details the non-functional requirements, as seen below in the list of functional requirements in a ranked order:

Requirement 1. New Game
Requirement 2. Save Game
Requirement 3. Load Game
Requirement 4. Quit Game
Requirement 5. Start Tutorial
Requirement 6. Collect Resources
Requirement 7. Craft Items
Requirement 8. Kill Enemies
Requirement 9. Upgrade Player Skills

2.3.1.1. USE CASE DIAGRAM

This is the Use Case Diagram for my project; TerraCraft. It provides an overview of the functional requirements inside the game.



2.3.1.2. REQUIREMENT 1: NEW GAME

2.3.1.2.1. DESCRIPTION & PRIORITY

The user must be able to play a new game, and by play, I mean to achieve movement using the controls, thus being able to explore the world, hence the priority on this requirement is high.

2.3.1.2.2. USE CASE

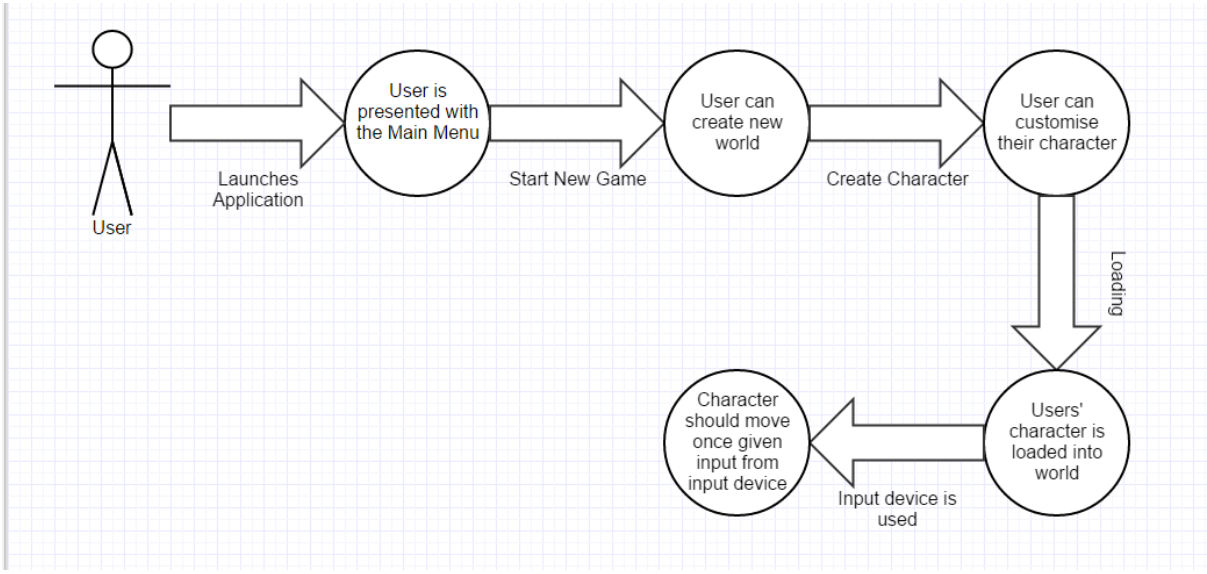
2.3.1.2.2.1. SCOPE

The scope of this use case is to allow the user to play the game, hence, moving around the world of the game.

2.3.1.2.2.2. DESCRIPTION

The use case describes the process of which the player launches the application.

2.3.1.2.2.3. USE CASE DIAGRAM



2.3.1.2.2.4. FLOW DESCRIPTION

2.3.1.2.2.4.1 PRECONDITION

The game is installed and the shortcut for the application is displayed on the desktop of the computer. The computer should meet the minimum requirements.

2.3.1.2.2.4.2. ACTIVATION

The Use Case Diagram starts when the user launches the game. The user should then create a new world and character and use their input device to make their character move.

2.3.1.2.2.4.3. MAIN FLOW

1. The game is launched
2. The user selects new game
3. The user creates a new world
4. The user creates a new character
5. The game loads the world and character
6. The user sends input using their input device
7. The game reads the input and moves the character.

2.3.1.2.2.4.4. ALTERNATE FLOW

Does not apply.

2.3.1.2.2.4.5. EXCEPTIONAL FLOW

1. Optimal Way of Starting New Game
   1.1. The game is launched
   1.2. The user selects new game
   1.3. The user creates a new world
   1.4. The user creates a new character
   1.5. The game loads the world and character
   1.6. The user sends input using their input device
   1.7. The game reads the input and moves the character.

2.3.1.2.2.4.6. TERMINATION

The process only stops once the user has successfully started a new game and has moved the character. Otherwise, the process does not terminate.

2.3.1.2.2.4.7. POST CONDITION

The game has no post condition for this as the game will always be active from this point on, and will get continuous input for movement.

2.3.1.3. REQUIREMENT 2: SAVE GAME

2.3.1.3.1. DESCRIPTION & PRIORITY

The user must be able to start the game, go into a selected world with their selected character and from there, they must be able to save the game at any point.

2.3.1.3.2. USE CASE

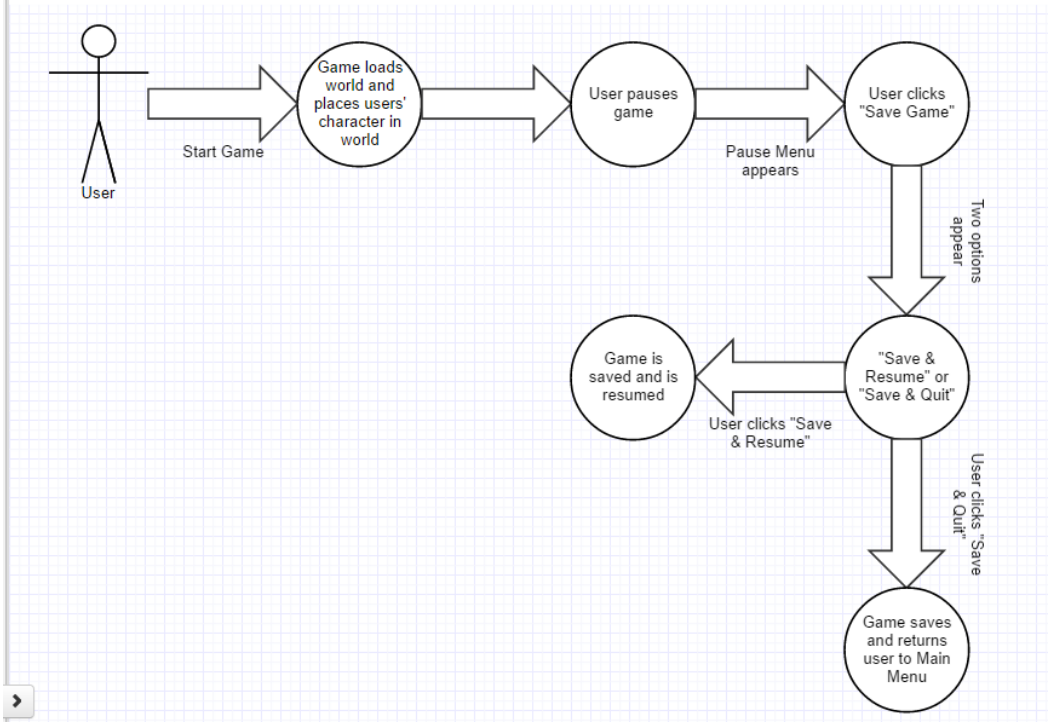
2.3.1.3.2.1. SCOPE

The scope of this use case is to allow the user to save the game, hence, the location, time of day, current player properties, i.e. health, and the storage in the inventory system.

2.3.1.3.2.2. DESCRIPTION

The use case describes the function of saving an instance of the game. The save file will store the users' status, i.e. world location, world time, character properties, etc.

2.3.1.3.2.3. USE CASE DIAGRAM



2.3.1.3.2.4. FLOW DESCRIPTION

2.3.1.3.2.4.1 PRECONDITION

The game is installed and the shortcut for the application is displayed on the desktop of the computer. The computer should meet the minimum requirements. The user should also have a new game begun or a loaded game save file.

2.3.1.3.2.4.2. ACTIVATION

The Use Case Diagram starts when the user pauses the game, showing a pause menu, just after starting a new game and loading into the world.

2.3.1.3.2.4.3. MAIN FLOW

1. The game has loaded the world
2. The user can move their character and progress
3. The user pauses the game
4. The game brings up the pause menu
5. The user selects the “Save Game” option
6. The game presents with two options: “Save & Resume” or “Save & Quit”
7. The user selects “Save & Resume”
8. The game creates a save file
9. The game writes to this file on the computer and stores information about the game

2.3.1.3.2.4.4. ALTERNATE FLOW

1. Save & Quit option
   1.1. The game has loaded the world
   1.2. The user can move their character and progress
   1.3. The user pauses the game
   1.4. The game brings up the pause menu
   1.5. The user selects the “Save Game” option
   1.6. The game presents with two options: “Save & Resume” or “Save & Quit”
   1.7. The user selects “Save & Quit”
   1.8. The game creates a save file
   1.9. The game writes to this file on the computer and stores information about the game
   1.10. The game then quits the game and brings user back to Main Menu screen

2.3.1.3.2.4.5. EXCEPTIONAL FLOW

1. Optimal Way of Saving the Game
   1.1. The game has loaded the world
   1.2. The user can move their character and progress
   1.3. The user pauses the game
   1.4. The game brings up the pause menu
   1.5. The user selects the “Save Game” option
   1.6. The game presents with two options: “Save & Resume” or “Save & Quit”
   1.7. The user selects “Save & Resume”
   1.8. The game creates a save file
   1.9. The game writes to this file on the computer and stores information about the game

2.3.1.3.2.4.6. TERMINATION

The process only stops once the user has successfully saved the game and the save launces successfully to a target location and either returns the user to the game or to the main menu screen.

2.3.1.3.2.4.7. POST CONDITION

The game goes into a waiting state, as the game saves, either allowing the user to continue to play, and save as many times as they like, or bring them back to the main menu screen.

2.3.1.4. REQUIREMENT 3: LOAD GAME

2.3.1.4.1. DESCRIPTION & PRIORITY

The user must be able to start the game and navigate through the main menu. From there, the user must be able to click the “Load Game” option and from there, they can load a saved game version to continue in.

2.3.1.4.2. USE CASE

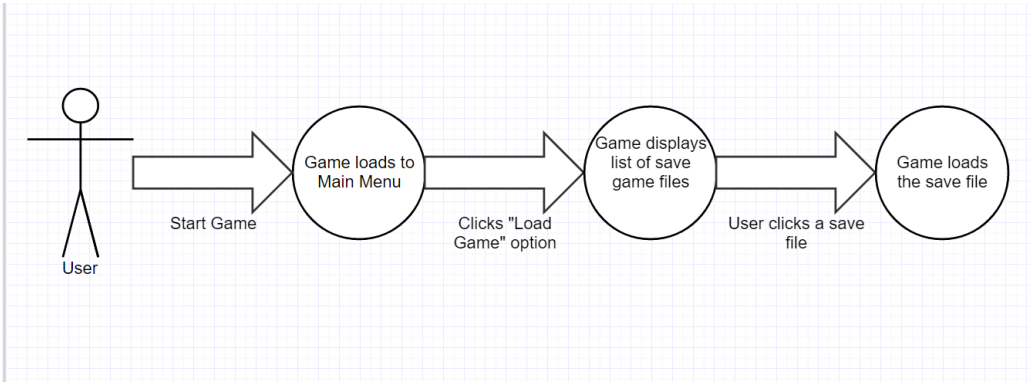
2.3.1.4.2.1. SCOPE

The scope of this use case is to allow the user to load a saved version of the game, hence, continuing from where they left off.

2.3.1.4.2.2. DESCRIPTION

The use case describes the function of loading an instance of the game. The loaded save file will read the information of the world and character, load it into the game and allow the user to pick up where they left off, insuring they saved the game previously.

2.3.1.4.2.3. USE CASE DIAGRAM



2.3.1.4.2.4. FLOW DESCRIPTION

2.3.1.4.2.4.1 PRECONDITION

The game is installed and the shortcut for the application is displayed on the desktop of the computer. The computer should meet the minimum requirements. The user should also have a save game file (see *Requirement 2: Save Game*).

2.3.1.4.2.4.2. ACTIVATION

The Use Case Diagram starts when the user starts the game, and is in the main menu.

2.3.1.4.2.4.3. MAIN FLOW

1. The user starts the game
2. The game loads the main menu screen
3. The user selects the “Load Game” option
4. The game displays the save files
5. The user selects a save file to load
6. The game loads the save file

2.3.1.4.2.4.4. ALTERNATE FLOW

Does not apply.

2.3.1.4.2.4.5. EXCEPTIONAL FLOW

1. Optimal Way to Load the Game
   1.1. The user starts the game
   1.2. The game loads the main menu screen
   1.3. The user selects the “Load Game” option
   1.4. The game displays the save files
   1.5. The user selects a save file to load
   1.6. The game loads the save file

2.3.1.4.2.4.6. TERMINATION

The process only stops once the user has successfully loaded an instance of the game and starts the game from the saved location.

2.3.1.4.2.4.7. POST CONDITION

The “Load Game” system is inactive when the game is currently being played.

2.3.1.5. REQUIREMENT 4: QUIT GAME

2.3.1.5.1. DESCRIPTION & PRIORITY

The user must be able to start the game and then either quit the game from the main menu or from within the game, in the pause menu, at any instance.

2.3.1.5.2. USE CASE

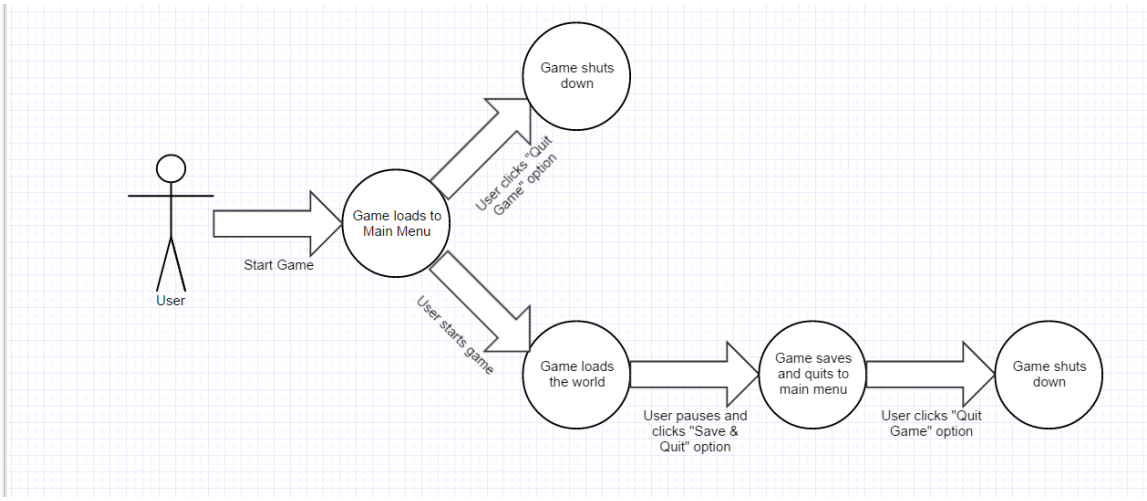
2.3.1.5.2.1. SCOPE

The scope of this use case is to allow the user to quit the game from either the main menu or from within the game, in the pause menu.

2.3.1.5.2.2. DESCRIPTION

The use case describes the function of quitting the game from either the main menu or from within the game, from the pause menu, at any given time they want. The game will either shut down or bring back to main menu where the game will then be effectively shut down, all functions will be stopped and the users' desktop will be displayed.

2.3.1.5.2.3. USE CASE DIAGRAM



2.3.1.5.2.4. FLOW DESCRIPTION

2.3.1.5.2.4.1. PRECONDITION

The game is installed and the shortcut for the application is displayed on the desktop of the computer. The computer should meet the minimum requirements.

2.3.1.5.2.4.2. ACTIVATION

The Use Case Diagram starts when the user starts the game, and is in the main menu.

2.3.1.5.2.4.3. MAIN FLOW

1. The user starts the game
2. The game loads the main menu screen
3. The user selects the "Quit Game" option
4. The game shuts down all functions

2.3.1.5.2.4.4. ALTERNATE FLOW

1. Exit from in-game pause menu
   1.1. The user launches the application
   1.2. The game loads to the Main Menu screen
   1.3. The user starts an instance of the game
   1.4. The user pauses the game
   1.5. The user clicks the “Save & Quit” option (see *Requirement 2: Save Game Alternate Flow*)
   1.6. The game saves the game and quits back to the Main Menu screen
   1.7. The user clicks the “Quit Game” option
   1.8. The game shuts down all functions

2.3.1.5.2.4.5. EXCEPTIONAL FLOW

1. Optimal Way to Quit the Game
   1.1. The user starts the game
   1.2. The game loads the main menu screen
   1.3. The user selects the “Quit Game” option
   1.4. The game shuts down all functions

2.3.1.5.2.4.6. TERMINATION

The process only stops once the user has successfully exits the game and stops all functions.

2.3.1.5.2.4.7. POST CONDITION

The game has been stopped successfully.

2.3.1.6. REQUIREMENT 5: START TUTORIAL

2.3.1.6.1. DESCRIPTION & PRIORITY

The user must be able to start the game and then participate in the tutorial to get to know the controls and functionalities.

2.3.1.6.2. USE CASE

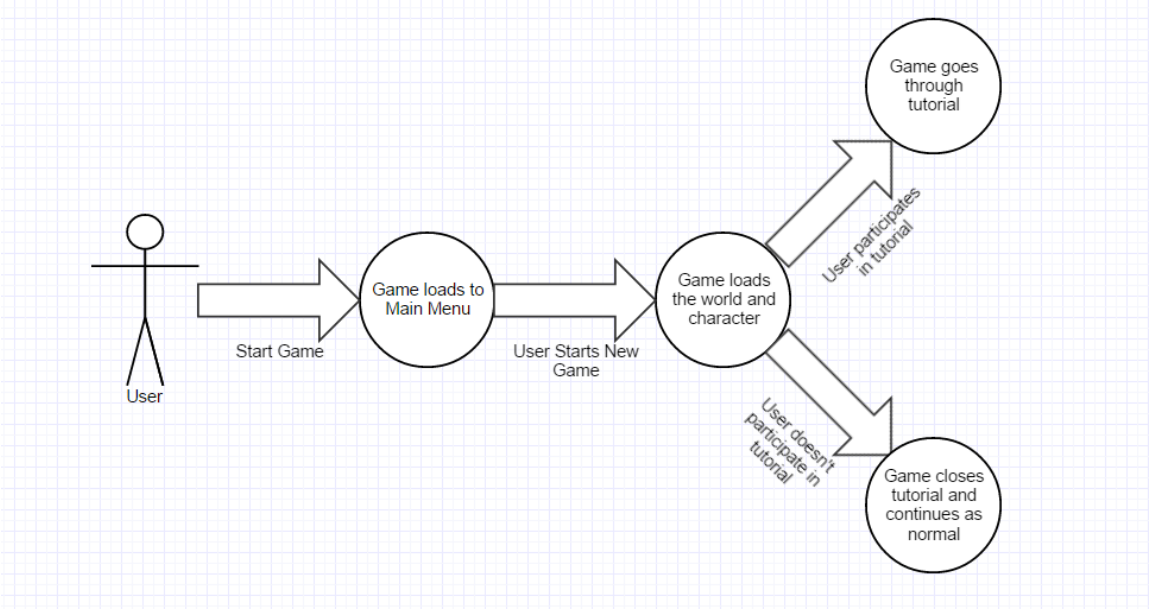
2.3.1.6.2.1. SCOPE

The scope of this use case is to allow the user to engage with the tutorial to get knowledge about the character movements, game functionalities and challenges.

2.3.1.6.2.2. DESCRIPTION

The use case describes the function of starting a new game and when in the new game, a tutorial is displayed to take place in to allow the user to gain knowledge about the games' functionalities and controls. The user will have a choice to either take part or to dismiss it and continue without the tutorial.

2.3.1.6.2.3. USE CASE DIAGRAM



2.3.1.6.2.4. FLOW DESCRIPTION

2.3.1.6.2.4.1 PRECONDITION

The game is installed and the shortcut for the application is displayed on the desktop of the computer. The computer should meet the minimum requirements. The user should also have a new game

2.3.1.6.2.4.2. ACTIVATION

The Use Case Diagram starts when the user starts a new game and has loaded inside the world.

2.3.1.6.2.4.3. MAIN FLOW

1. The user starts the game
2. The game loads the main menu screen
3. The user selects the "New Game" option
4. The game starts a new game
5. The user participates in the tutorial

2.3.1.6.2.4.4. ALTERNATE FLOW

1. Doesn't Start Tutorial
   1.1. The user starts the game
   1.2. The game loads the main menu screen
   1.3. The user selects the "New Game" option
   1.4. The game starts a new game
   1.5. The user doesn't participate in the tutorial
   1.6. The game continues as normal, without tutorial

2.3.1.6.2.4.5. EXCEPTIONAL FLOW

1. Optimal Way to Start the Tutorial
   1.1. The user starts the game
   1.2. The game loads the main menu screen
   1.3. The user selects the "New Game" option
   1.4. The game starts a new game
   1.5. The user participates in the tutorial

2.3.1.6.2.4.6. TERMINATION

The process only stops once the user has either successfully participated in the tutorial or has exited the tutorial.

2.3.1.6.2.4.7. POST CONDITION

The game has no post condition for this as the game will always be active from this point on, and will get continuous input for movement.

2.3.1.7. REQUIREMENT 6: COLLECT RESOURCES

2.3.1.7.1. DESCRIPTION & PRIORITY

The user must be able to start the game, know basic movement and functionalities and then go off and collect resources.

2.3.1.7.2. USE CASE

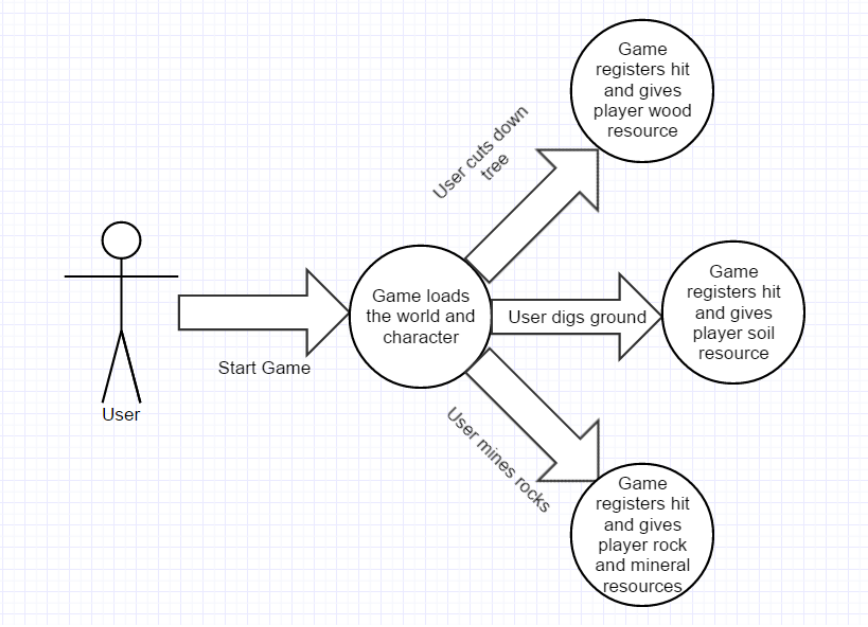
2.3.1.7.2.1. SCOPE

The scope of this use case is to allow the user to collect resources by using tools provided in the game.

2.3.1.7.2.2. DESCRIPTION

The use case describes the function of collecting resources within the game, by using the tools provided in the game, i.e. axe, pickaxe and shovel, to either chop, dig or mine their resources from their generated world and then collect them into their inventory.

2.3.1.7.2.3. USE CASE DIAGRAM



2.3.1.7.2.4. FLOW DESCRIPTION

2.3.1.7.2.4.1 PRECONDITION

The game is installed and the shortcut for the application is displayed on the desktop of the computer. The computer should meet the minimum requirements. The user should also have a new game or loaded save to play in, and must also have basic knowledge of the game from the tutorials.

2.3.1.7.2.4.2. ACTIVATION

The Use Case Diagram starts when the user starts the game, and is in the main menu.

2.3.1.7.2.4.3. MAIN FLOW

1. The user starts the game
2. The game loads the main menu screen
3. The user either selects the “New Game” option or the “Load Game” option (Please see either *Requirement 1: New Game* or *Requirement 3: Load Game*)
4. The game loads the world and character
5. The user selects a tool to use, i.e. axe for wood, pickaxe for rocks and shovel for soil
6. The user chops down a tree
7. The game registers the hit and destroys the tree and spawns wood for the user
8. The user collects the wood
9. The game adds the wood to the users’ inventory

2.3.1.7.2.4.4. ALTERNATE FLOW

1. Collect Soil Resources
   1.1. The user starts the game
   1.2. The game loads the main menu screen
   1.3. The user either selects the “New Game” option or the “Load Game” option (Please see either *Requirement 1: New Game* or *Requirement 3: Load Game*)
   1.4. The game loads the world and character
   1.5. The user selects a tool to use, i.e. axe for wood, pickaxe for rocks and shovel for soil
   1.6. The user digs the soil
   1.7. The game registers the hit and destroys the soil block and spawns it for the user
   1.8. The user collects the soil
   1.9. The game adds the soil to the users’ inventory
2. Collect Rock/Mineral Resources
   2.1. The user starts the game
   2.2. The game loads the main menu screen
   2.3. The user either selects the “New Game” option or the “Load Game” option (Please see either *Requirement 1: New Game* or *Requirement 3: Load Game*)
   2.4. The game loads the world and character
   2.5. The user selects a tool to use, i.e. axe for wood, pickaxe for rocks and shovel for soil
   2.6. The user digs the soil
   2.7. The game registers the hit and destroys the soil block and spawns it for the user
   2.8. The user collects the soil
   2.9. The game adds the soil to the users’ inventory

2.3.1.7.2.4.5. EXCEPTIONAL FLOW

1. Optimal Way to Collect Resources
   1.1. The user starts the game
   1.2. The game loads the main menu screen
   1.3. The user either selects the “New Game” option or the “Load Game” option (Please see either *Requirement 1: New Game* or *Requirement 3: Load Game*)
   1.4. The game loads the world and character
   1.5. The user selects a tool to use, i.e. axe for wood, pickaxe for rocks and shovel for soil
   1.6. The user chops down a tree
   1.7. The game registers the hit and destroys the tree and spawns wood for the user
   1.8. The user collects the wood
   1.9. The game adds the wood to the users’ inventory

2.3.1.7.2.4.6. TERMINATION

The process only stops once the user has successfully destroyed a block and collects the resource(s) to their inventory.

2.3.1.7.2.4.7. POST CONDITION

The game has no post condition for this as the game will always be active from this point on, and will get continuous input for movement.

2.3.1.8. REQUIREMENT 7: CRAFT ITEMS

2.3.1.8.1. DESCRIPTION & PRIORITY

The user must be able to start the game, know basic movement and functionalities and have collected resources in their inventory.

2.3.1.8.2. USE CASE

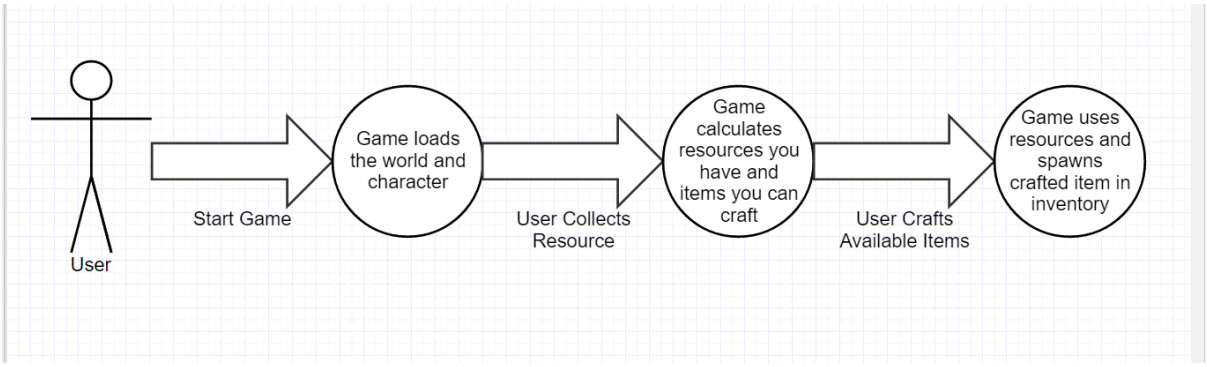
2.3.1.8.2.1. SCOPE

The scope of this use case is to allow the user to craft items within their inventory system, using collected resources.

2.3.1.8.2.2. DESCRIPTION

The use case describes the function of crafting items from the inventory system, at any given time they want, if they have the resources required. The game will calculate the resources you have and see if it can craft certain items.

2.3.1.8.2.3. USE CASE DIAGRAM



2.3.1.8.2.4. FLOW DESCRIPTION

2.3.1.8.2.4.1 PRECONDITION

The game is installed and the shortcut for the application is displayed on the desktop of the computer. The computer should meet the minimum requirements. The user should also have a new game or loaded save to play in, and must also have basic knowledge of the game from the tutorials.

2.3.1.8.2.4.2. ACTIVATION

The Use Case Diagram starts when the user starts the game and has sufficient resources collected.

2.3.1.8.2.4.3. MAIN FLOW

1. The user starts the game
2. The game loads the main menu screen
3. The user either selects the “New Game” option or the “Load Game” option (Please see either *Requirement 1: New Game* or *Requirement 3: Load Game*)
4. The game loads the world and character
5. The user collects resources (Please see *Requirement 6: Collect Resources*)
6. The user opens the inventory system
7. The game calculates the resources obtained to the resources required for the items to be crafted
8. The user selects what item(s) to craft, depending on resources
9. The game spawns the item(s) in the users’ inventory

2.3.1.8.2.4.4. ALTERNATE FLOW

Does not apply

2.3.1.8.2.4.5. EXCEPTIONAL FLOW

1. Optimal Way to Craft an Item
    1.1. The user starts the game
    1.2. The game loads the main menu screen
    1.3. The user either selects the “New Game” option or the “Load Game” option (Please see either *Requirement 1: New Game* or *Requirement 3: Load Game*)
    1.4. The game loads the world and character
    1.5. The user collects resources (Please see *Requirement 6: Collect Resources*)
    1.6. The user opens the inventory system
    1.7. The game calculates the resources obtained to the resources required for the items to be crafted
    1.8. The user selects what item(s) to craft, depending on resources
    1.9. The game spawns the item(s) in the users’ inventory

2.3.1.8.2.4.6. TERMINATION

The process only stops once the user has successfully crafts the item(s) and exits the inventory system.

2.3.1.8.2.4.7. POST CONDITION

The game has no post condition for this as the game will always be active from this point on, and will get continuous input for movement.

2.3.1.9. REQUIREMENT 8: KILL ENEMIES

2.3.1.9.1. DESCRIPTION & PRIORITY

The user must be able to start the game, know basic movement and functionalities and have weapons in their inventory.

2.3.1.9.2. USE CASE

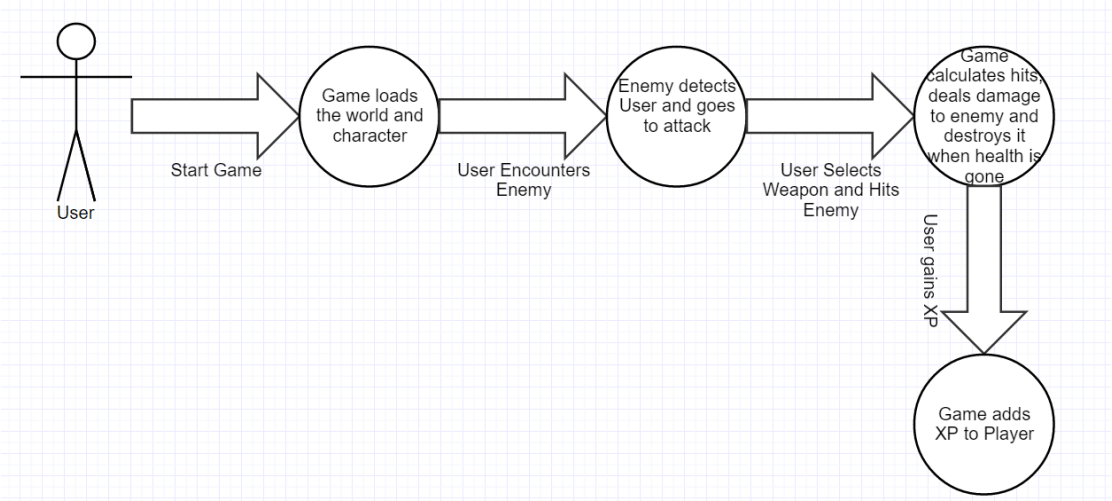
2.3.1.9.2.1. SCOPE

The scope of this use case is to allow the user to kill enemies

2.3.1.9.2.2. DESCRIPTION

The use case describes the function of killing enemies, using weapons from their inventory system, whenever it's required to do so. The game will calculate each hit, dealing damage to the enemies from the damage given to the weapon.

2.3.1.9.2.3. USE CASE DIAGRAM



2.3.1.9.2.4. FLOW DESCRIPTION

2.3.1.9.2.4.1 PRECONDITION

The game is installed and the shortcut for the application is displayed on the desktop of the computer. The computer should meet the minimum requirements. The user should also have a new game or loaded save to play in, and must also have basic knowledge of the game from the tutorials.

2.3.1.9.2.4.2. ACTIVATION

The Use Case Diagram starts when the user starts the game and encounters an enemy.

2.3.1.9.2.4.3. MAIN FLOW

1. The user starts the game
2. The game loads the main menu screen
3. The user either selects the “New Game” option or the “Load Game” option (Please see either *Requirement 1: New Game* or *Requirement 3: Load Game*)
4. The game loads the world and character
5. The user encounters an enemy
6. The user selects a crafted weapon (Please see *Requirement 7: Craft Items*)
7. The user hits enemy
8. The game calculates the hit and deals damage to enemy
9. The game destroys enemy when health is depleted
10. The game adds XP to player when enemy is killed

2.3.1.9.2.4.4. ALTERNATE FLOW

Does not apply

2.3.1.9.2.4.5. EXCEPTIONAL FLOW

1. Optimal Way to Kill an Enemy
   1.1. The user starts the game
   1.2. The game loads the main menu screen
   1.3. The user either selects the “New Game” option or the “Load Game” option (Please see either *Requirement 1: New Game* or *Requirement 3: Load Game*)
   1.4. The game loads the world and character
   1.5. The user encounters an enemy
   1.6. The user selects a crafted weapon (Please see *Requirement 7: Craft Items*)
   1.7. The user hits enemy
   1.8. The game calculates the hit and deals damage to enemy
   1.9. The game destroys enemy when health is depleted
   1.10. The game adds XP to player when enemy is killed

2.3.1.9.2.4.6. TERMINATION

The process only stops once the user has successfully killed an enemy.

2.3.1.9.2.4.7. POST CONDITION

The game has no post condition for this as the game will always be active from this point on, and will get continuous input for movement.

2.3.1.10. REQUIREMENT 9: UPGRADE PLAYER SKILLS

2.3.1.10.1. DESCRIPTION & PRIORITY

The user must be able to start the game, know basic movement and functionalities and have acquired XP to level up.

2.3.1.10.2. USE CASE

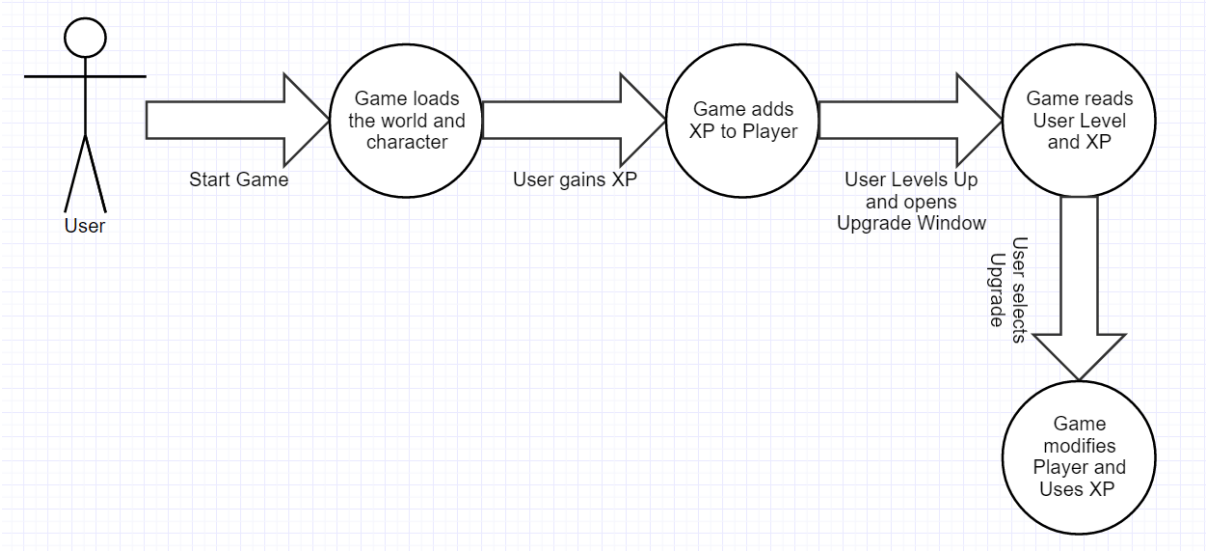
2.3.1.10.2.1. SCOPE

The scope of this use case is to allow the user to upgrade their characters' skills.

2.3.1.10.2.2. DESCRIPTION

The use case describes the function of upgrading their characters' skills, depending on their level and the amount of XP acquired. The game will calculate the players level and XP, and allow the user to upgrade their skills, being health, armour, resistance, speed, etc.

2.3.1.10.2.3. USE CASE DIAGRAM



2.3.1.10.2.4. FLOW DESCRIPTION

2.3.1.10.2.4.1. PRECONDITION

The game is installed and the shortcut for the application is displayed on the desktop of the computer. The computer should meet the minimum requirements. The user should also have a new game or loaded save to play in, and must also have basic knowledge of the game from the tutorials.

2.3.1.10.2.4.2. ACTIVATION

The Use Case Diagram starts when the user starts the game and gains enough XP to level up.

2.3.1.10.2.4.3. MAIN FLOW

1. The user starts the game
2. The game loads the main menu screen
3. The user either selects the "New Game" option or the "Load Game" option (Please see either *Requirement 1: New Game* or *Requirement 3: Load Game*)
4. The game loads the world and character
5. The user levels up, gaining enough XP (Please see *Requirement 8: Kill Enemies***Error! Reference source not found.**)
6. The user opens Upgrade window
7. The game calculates the users' level and XP and shows what can be upgraded
8. The user selects an upgrade
9. The game takes XP from user and upgrades the character

2.3.1.10.2.4.4. ALTERNATE FLOW

Does not apply

2.3.1.10.2.4.5. EXCEPTIONAL FLOW

1. Optimal Way to Upgrade the Players' Skills
   1.1. The user starts the game
   1.2. The game loads the main menu screen
   1.3. The user either selects the "New Game" option or the "Load Game" option (Please see either *Requirement 1: New Game* or *Requirement 3: Load Game*)
   1.4. The game loads the world and character
   1.5. The user levels up, gaining enough XP (Please see *Requirement 8: Kill Enemies***Error! Reference source not found.**)
   1.6. The user opens Upgrade window
   1.7. The game calculates the users' level and XP and shows what can be upgraded
   1.8. The user selects an upgrade
   1.9. The game takes XP from user and upgrades the character

2.3.1.10.2.4.6. TERMINATION

The process only stops once the user has successfully upgraded their character.

2.3.1.10.2.4.7. POST CONDITION

The game has no post condition for this as the game will always be active from this point on, and will get continuous input for movement.

2.3.2. NON-FUNCTIONAL REQUIREMENTS

2.3.2.1. PERFORMANCE/RESPONSE TIME REQUIREMENT

This game should run at an average of 60 frames per second (FPS) on a user's PC. The user will be able to directly change the quality of the graphics, to make sure the game can run optimally, change the resolution of the screen so it can fit their monitor and higher/lower the sound so it won't be as loud. The response time of the game should be immediate, so that when the player begins an action, the game should react instantly without delay in some meaningful way to what the user intended to achieve.

2.3.2.2. AVAILABILITY REQUIREMENT

The game itself should be constantly available to the user always when they wish to play. The game will eventually, and hopefully, feature Multiplayer. This concept of the game may sometimes be unavailable to the users, to perform maintenance on the system.

2.3.2.3. RECOVER REQUIREMENT

The game will save the users progress at constant intervals during the duration of the gameplay. This is to ensure there is no data loss, so that, in an event of a crash, the user can just pick up where they left off, easily. The user will also have the option to manually save the game in the pause menu.

2.3.2.4. ROBUSTNESS REQUIREMENT

The game will have a constant safeguard against errors – a small program will detect any errors that may arise in the game, allowing the game to continue, without crashing. Alpha and Beta tests will take place prior to the release along with bug fixing so the users can't encounter any bugs.

2.3.2.5. SECURITY REQUIREMENT

The save files that the game creates will be encrypted using JSON and/or Binary. This prevent users from 'tapping in' to the files and change the data up to cheat.

2.3.2.6. RELIABILITY REQUIREMENT

This game is expected to be accessible always, once the user has downloaded and installed all the files onto their PC.

2.3.2.7. MAINTAINABILITY REQUIREMENT

The game will be supported for a considerable amount of time post-release of the game, keeping track of users' progress, the game's progress and what needs improvements in terms of errors or bugs.

2.3.2.8. PORTABILITY REQUIREMENT

Once the user has successfully downloaded the game, they can take it anywhere with them, to any PC they desire using a USB flash drive or any other means of storage.

2.3.2.9. EXTENDIBILITY REQUIREMENT

The game may be updated with additional features over time – these will be free updates to help bring the game to bigger and better means, aiming for better gameplay and experience for all users.

2.3.2.10. REUSABILITY REQUIREMENT

The game will offer the user a way to go back to the beginning – to start over, to perceive a unique experience of the game.

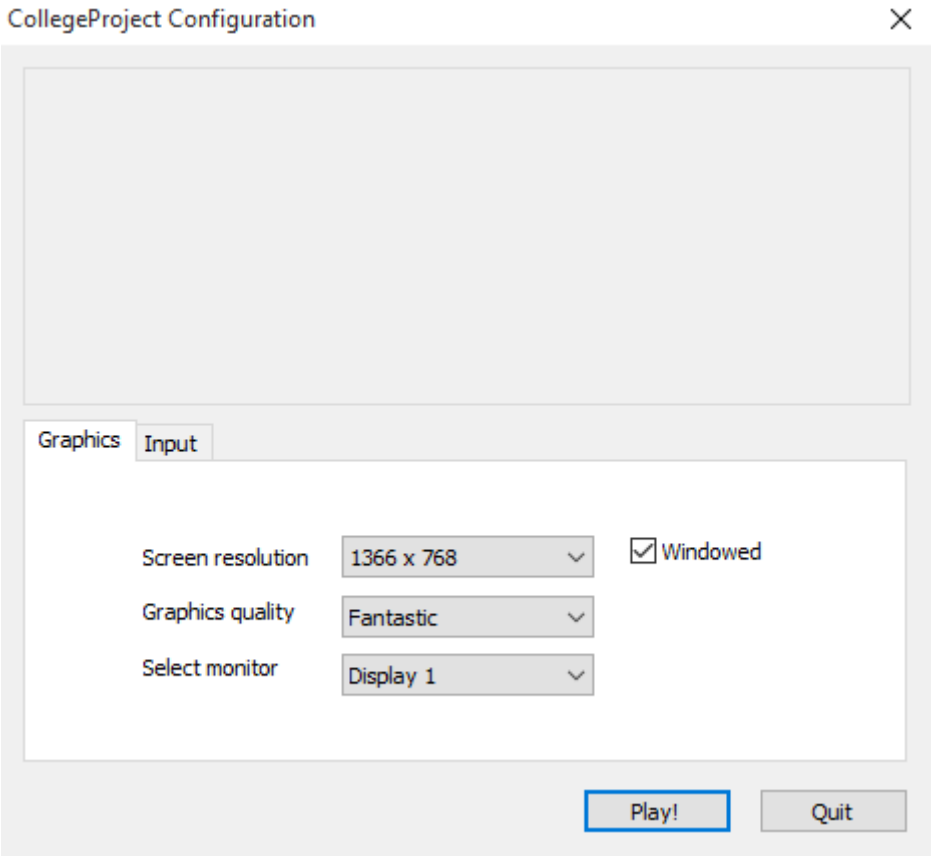
2.3.2.11. RESOURCE UTILISATION REQUIREMENT

This game should use as much available resources as it requires, with reason, from the users' system to reach a standard of stability and playability.

2.4. INTERFACE REQUIREMENTS

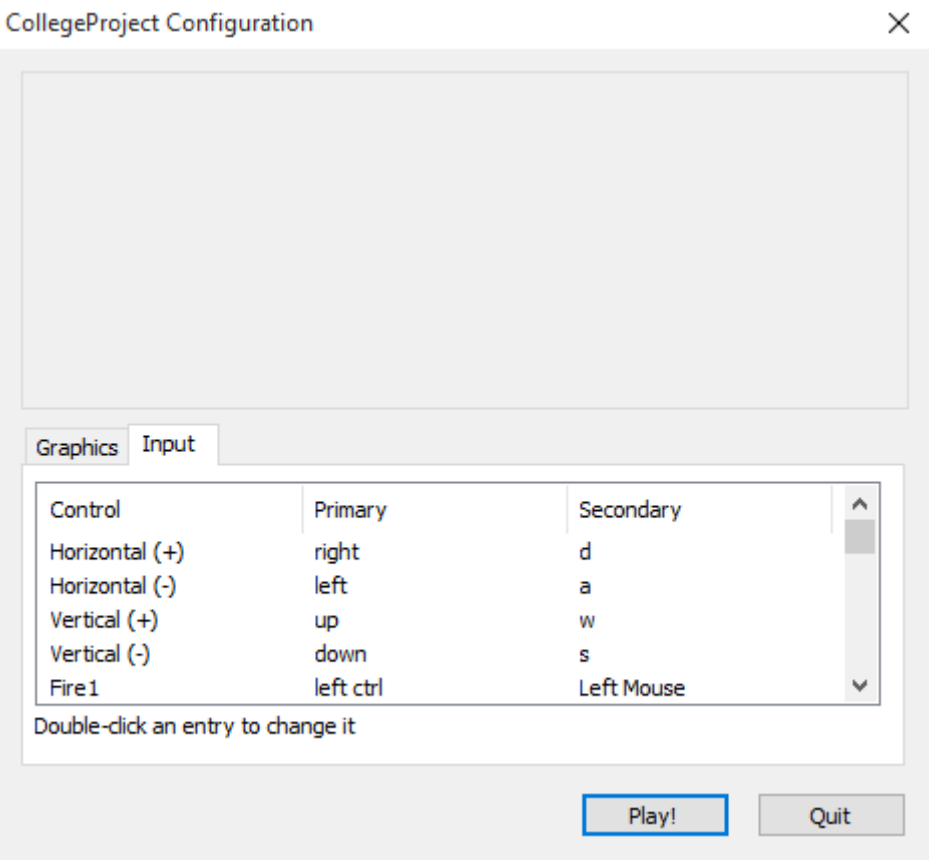
2.4.1. GRAPHICAL USER INTERFACE (G.U.I)

2.4.1.1. GRAPHICS SETTINGS



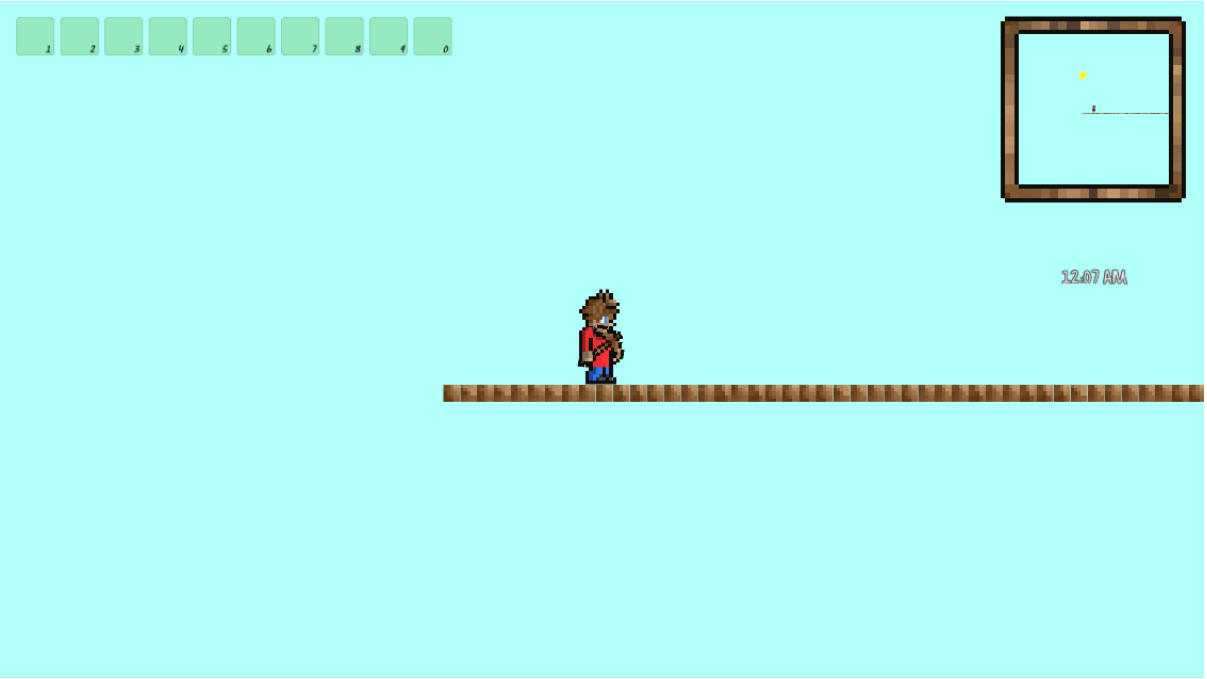
The above screenshot shows how the new game requirement is implemented. This is the default launch window from Unity and it allows the user to change settings regarding the game. The settings include the screen resolution, graphics quality and the monitor you wish to show it on. It also allows the user to set the game to run in either full screen or windowed mode.

2.4.1.2. INPUT SETTINGS



This screenshot shows the ability of the user to change their input settings. The user can tweak the controls of the game, if they're not happy with the default settings, they can double-click on either the primary or secondary key-bindings and enter their own key-binding. Clicking **play** will commit these changes and apply to the game.

2.4.1.3. IN-GAME SCREEN



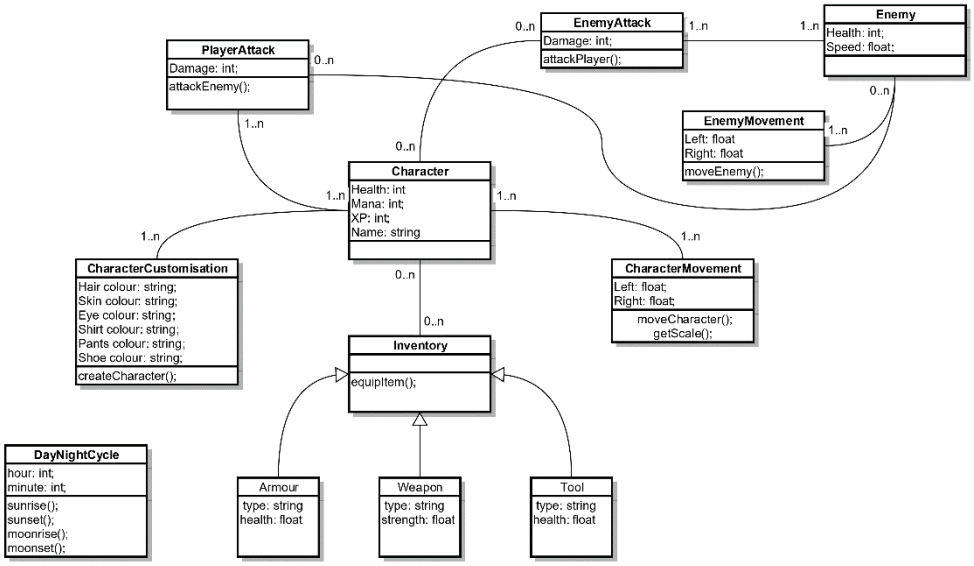
So far, this is how the game looks. I haven't done as much as I'd like to but it is the basics that will kick off the final look. You can see the inventory system in the top left corner, the mini-map in the top right corner and, just under that, you can see the time. It can be previewed in 12-hour or 24 hours. This will help with the Day-Night Cycle.

The blocks that the character is standing on is randomly generated. It is only the beginning, so it won't have any landmarks like hills or cliffs, so it's just a continuous straight line of blocks. These will be able to be destroyed and placed later in the development.

2.4.2. APPLICATION PROGRAMMING INTERFACE (A.P.I)

The game will be made using the Unity Game Engine – which is freely available.

2.5. SYSTEM ARCHITECTURE



I feel that the architecture, as shown above, is suitable to the type of game I am intending to make. Character customisation is the first class the player will interact with – creating their own character in a way they'd like to see. The player will then enter the game and the next class that will be interacted with is the movement class – allowing the player to move along the x-axis and y-axis. The player may or may not encounter an enemy that will attack them and the class will be called to handle this. The enemy will need to move and attack the player.

When defeating the enemy, the player may find an item they'd like to equip – this can either be a tool, weapon or armour. These will be held inside the inventory system.

A Day/Night Cycle will be present in the game, allowing the sun and the moon to rise and set at specific times of the day, changing the colour of the sky also.

For now, this is how I envision the basics of the starting architecture of the classes to be. I'll need to add more classes in to handle distinct functions, like crafting.

2.6. SYSTEM EVOLUTION

The game, even though it's in its initial stages, will have additional content. This additional content will be optional to download and free to use and will customise the way the game works and looks. My aim is for this game to have its own multiplayer component – connecting players through the game and explore the game together, strengthening friendships.

I would like it so that users could also add to the game themselves – using their own scripts to add their additional content e.g. creating maps, worlds and characters.

3. REFLECTION JOURNALS

INTRODUCTION

My name is Dean Byrne. I'm 21 years old at the time of starting this reflective journal. I have one daughter – Robyn, aged 9 months in another week. She lives with her mother in Killina, Co. Kildare, while I live in Ballyfermot, Co. Dublin. I'll be returning to finish the final year of a Bachelor of Science (Honours) in Computing at the National College of Ireland in September 2016.

I'm in part-time employment in Marks & Spencer as a Twilight Sales Advisor – meaning I put stock onto the shelves when the store is open and after it closes.

I've decided to go with this style of journal for these reasons:

1. I wanted to go into the course with something that can support the final year project – recording my thoughts, ideas and the information I write down, on paper, throughout the year.
2. It will help keep myself well-organised for the final year project, showing that I worked hard to design and develop the project.

I know, for a fact, that if I don't have a well-organised and disciplined approach to the final year project, I'll find it very challenging to complete the project. I love designing and programming. Ever since I started the internship with Defiant Games, I've become immersed in developing games. But, it's funny; come up to me, 2 years ago, and I could not explain how or even where to start on developing games. I didn't know what Unity was. I never knew the C# programming language, but, ever since I began to learn about it, it's become like a hobby to me.

So, now, I'm on my own. To plan and develop my final year project, which I started last year before I deferred the course, due to my mother's passing of cancer. I promised that I'd finish the course for her and finish the project that I started.

I know it's tough to write down everything you did every day, but the objective of this journal is to have a complete description of the year-long journey that I'm going through in the final year.

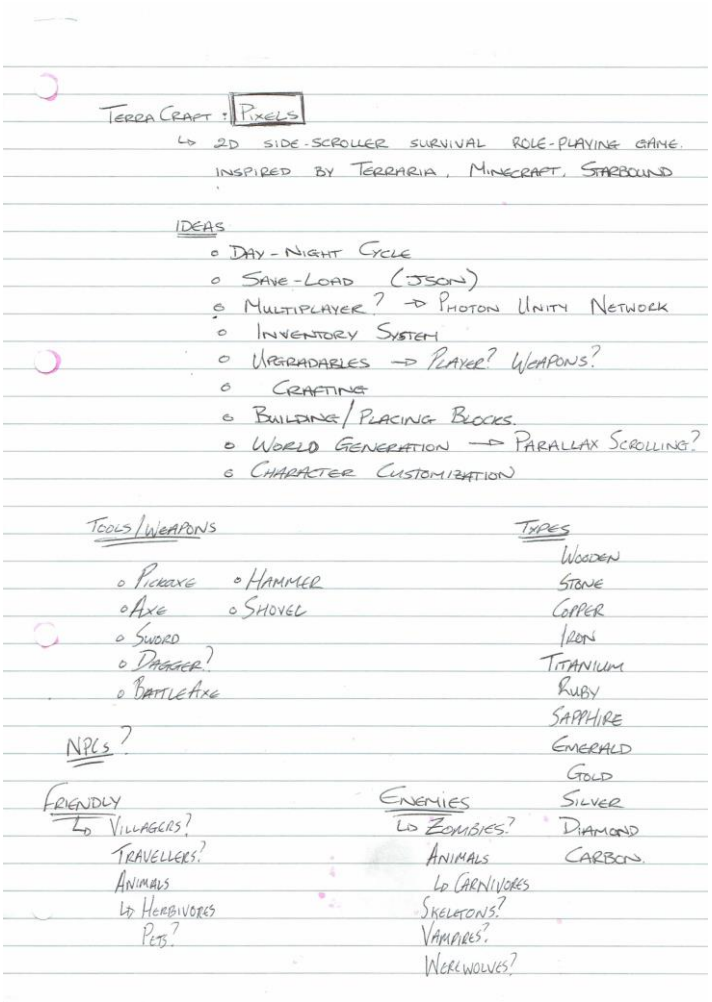
3.1. SEPTEMBER 2016

3.1.1. SEPTEMBER 19TH, 2016

Today is the first day back at college. It's so nerve-wrecking coming back in after a year, knowing absolutely no one.

I was in the Software Project module at 9am with Eamon. He ran over the project deliverables, telling us about what ideas to think of and not to worry if we didn't have any ideas yet, as it's only week 1. He ran over past projects, showing us the documents of past students who got high grades with their projects.

I've known what I was going to do for my project. I was going to plan, design and develop a game called *TerraCraft*. It started out as a small idea back in 2015, with inspiration from other similar games in that field.



3.1.2. SEPTEMBER 20TH, 2016

Today I thought I'd consider some of the features the game will have – the Day/Night Cycle. It's a little complex, but I think it'll add a nice effect to the game – to give it a sense of time. The main functions of the Day/Night Cycle are:

- In the morning, around 5 – 6am, the Sun rises. The sky goes from a dark, black colour to a light orange-to-blue colour, representing a sunrise.
- As the morning continues into the afternoon, the sun rises to the top, the sky becomes a bright blue colour.
- As the evening sets in, around 6 – 7pm, the sky goes from the bright blue colour into a light blue-to-orange colour to represent a sunset.
- As the sun sets down, the sky turns back into the dark, black colour. This happens around the same time that the Moon begins to rise, in the same way as the Sun does, while the stars begin to appear.

I found some answers online, in the Unity forums [1], about the Day/Night Cycle – about controlling the sky colour, by using a gradient variable, and the Sun/Moon rising and setting, by using animation curves.

When I get into the design/development stage of the project, I'll test out this method to see if I can get a Day/Night Cycle working. In the project, the Day/Night Cycle will synchronise with an in-game clock, which the player can use in a 12-hour (12:00am) setting or a 24-hour (00:00) setting.

[1] Day/Night Cycle – Unity Forum: https://forum.unity3d.com/threads/2d-day-night-cycle-and-weather-system-a-la-altos-adventure.377959/

3.1.3. SEPTEMBER 27TH, 2016

I'm beginning to worry about the integrity of the project now, for the simple being: *am I plagiarising myself?*

I started the project last year. I handed in two reflective journals and showed off a prototype to a supervisor. I'm worried that, if I attempt to develop *TerraCraft*, I'll be plagiarising my own work.

I spoke to Eamon today. I was in the atrium, as I was on a two-hour break and the SU was closed due to a mess from the pre-sesh Fresher's Ball. I spotted him walking past and spoke very quickly to him about my situation, about my worries about the project. He spoke very briefly, stating that, as I'm a returning student, I shouldn't worry about it and that I can continue developing the project.

This took a lot off my mind, as I have no other ideas and considering as it's week 2, it's a bit of a late start to plan and develop a new project.

Now, I'm starting to do the Project Pitch presentation, which will run into the October reflective journal. I've to go up in front of three lecturers and talk about my project without the use of any presentation or props. It's a bit nerve-wrecking but I hope that they'll allow my project to be developed.

3.1.4. SEPTEMBER 30TH, 2016

I just finished up with the Project Pitch presentation. It's completed and uploaded to Moodle.

Now, I'm just looking up most of my features on forums. One feature that stands out is the Parallax Scrolling effect. It gives off a 3D-like effect in a 2D game. It gives the sense of distance and can be useful to implement and would be a great skill to learn.

I found a website [2] that take you through a tutorial that can help give off the Parallax Scrolling effect. I haven't tried testing it out yet, but I will soon.

This is the last entry for the September reflective journal. I'll continue October's reflective journal, where I'll describe the project pitch, the presentation and my studies for next month.

[2] PixelNest Parallax Scrolling Effect Tutorial - http://pixelnest.io/tutorials/2d-game-unity/parallax-scrolling/

3.2. OCTOBER 2016

3.2.1. OCTOBER 2ND, 2016

Today I started working on the Project Pitch, which is due in 2 days, where I'll have to describe my project in front of 3 judges, i.e. what my project is, what functionalities it will include, if it's original and/or good enough to work on for the year. I'm extremely nervous as I tried making this project last year before I deferred the course, but, I'm hopeful that it will be accepted.

The presentation pitch can only be 3 slides. It seems so little for such a huge year-long project! And you can't use the presentation slides during your presentation, as no props are allowed.

I have it completed and uploaded to Moodle, a day early, and am now waiting for my presentation time, which is on Wednesday 5th October 2016 at 14:30. The three judges I had was Damien McNamara, Anu Sahni and Padraig De Burca.

I've attached the Project Pitch presentation, so you can view it here.

Software Project Pitch

3.2.2. October 5th, 2016

Today is the day of the Project Pitch presentation. I'm anxious to get it finished with and I'm starting to doubt whether the project will be accepted. I'm afraid that, when I go in and present my project, it will be dismissed and I'll have to start from scratch on a new project – one I haven't planned for the past year.

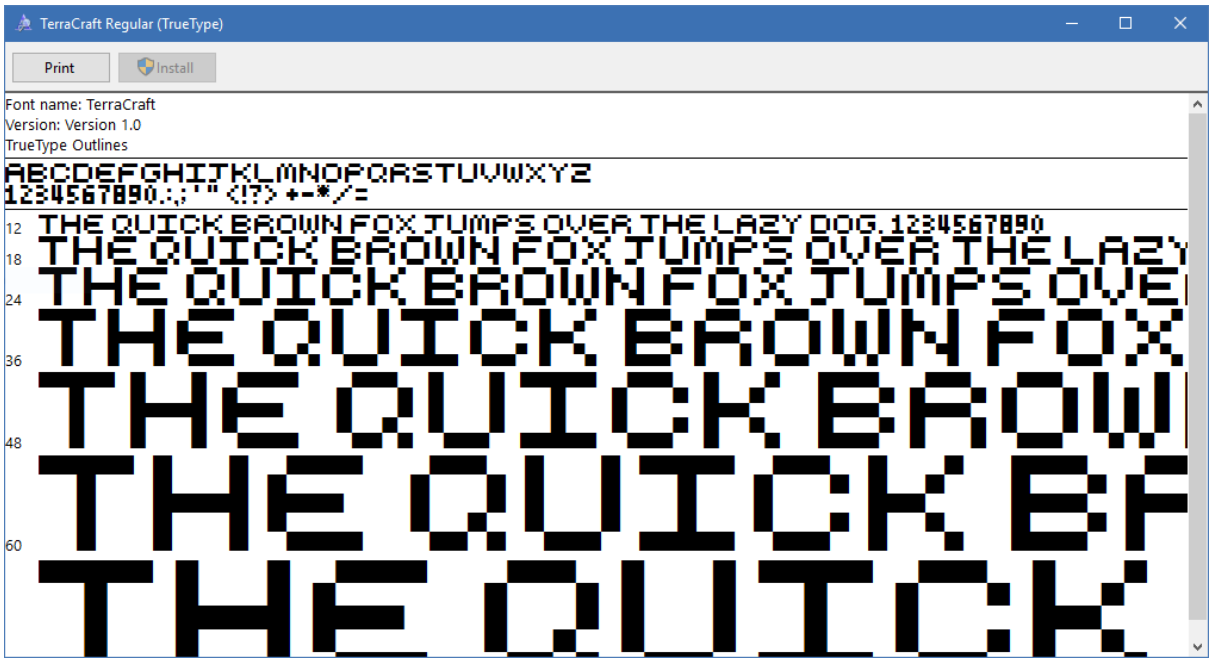
I've just had the presentation around 2 hours ago and I must say it went okay. It didn't go as bad as I thought it out to be in my head. It took about a good 5 minutes to present my project – going through each aspect of the functionalities and another 5 minutes of questions about my plans for the project, i.e. will I be diving in head first into development or am I going to plan carefully about the development of my project?

I was told that my game is like some games out there already, including Minecraft: Story Mode and was told to go and have a look at a few games and see what makes my game different to each one. So, I did just that. I researched Minecraft: Story Mode, seen the concepts in the game, evaluated them against the plans for my own game and can say that my game will be different compared to it.

3.2.3. October 13th, 2016

Today I done a bit for the interface of the main game – the font. It takes around 30-60 minutes to complete, depending on the complexity of the font. I used this website[3] to design my font and allows me to design every character, from letters to symbols. This font is made by myself, not copying anybody else's designs or ideas.

I've attached an image below to show my font design.



[3] Font website – http://fontstruct.com/

3.2.4. October 21st, 2016

Over the past week, I've been working on the Project Proposal document. This is a document that reflects over the main objectives, background, technical approaches, requirements, plans, details and the evaluation of the main project. Each heading goes into in-depth detail about the project, i.e. objectives shows what the project is about, what is to be expected and how you're doing it.

This document is essential to have as it allows you to keep track of the development progress of the project. Inside the project proposal, you've to develop a project plan, using Microsoft Office Project. This is a professional planning tool which you can decide what to develop, when to do it and estimate how long it may take you to do so. Having this is basically like having a check-list. Every time you finish a development task, you can mark it off as complete until the project is 100% complete.

I have it completed the document, put it into a .zip file with the Project Plan file and uploaded to Moodle with over 6 hours to spare.

I have it attached below, so it can be viewed at any time:

TerraCraft Project Proposal

3.2.5. OCTOBER 27TH, 2016

Today I had a meeting with my assigned project supervisor; Dr. Anu Sahni. She's my lecturer for a module I take called 'Computer Graphics Design and Animation". The meeting took place right after a class CA, where we presented a modern technology we learned over the past 2-3 weeks prior.

In the meeting, I told Anu what I've done so far in my project, what challenges I will be facing, i.e. Multiplayer later during development, and what I hope to do. Her advice to me was to keep my ideas as original as possible, to make my game stand out and to try to make Multiplayer a priority in my project, as it's a complex task and will be heavily rewarded in the final grading of the project.

The meeting only lasted around 10-15 minutes, a short but brief meeting, and I was happy with the results. It got me thinking of the tasks I need to be focusing on and will help me gain a better grade.

3.2.6. October 30th, 2016

This week, I've been documenting all the records of this month. While it wasn't one of my most productive months, I'm happy to say I'm ready to progress into the development stage of the project. By the end of the next month (November), I'm hoping to have a good bit done for the "prototype" to show off to my supervisor.

The functionalities I'd like to have done is as follows:

- Basic character movement
- World generation
- Basic day/night cycle
- Inventory system

If I manage to get the above list integrated and developed into the game, I'd be delighted, as it would give me a boost in the long run of development.

This will be the last entry for the October monthly journal. I'll begin writing the November reflective journal after Halloween, where I plan to do the above tasks and get a start on the project.

3.3. NOVEMBER 2016

3.3.1. NOVEMBER 1ST, 2016

Today I began to design the games' user interface (UI) for the main menu. This will be a basic design to allow the user to start the game, change the games' options and/or exit the game, if they are finished playing the game.

The way the main menu will be laid out is as followed:

- Single Player
  - New Game
    - Create Character
  - Load Game
    - Select World
- Multiplayer (greyed out)
- Options
  - Sound
  - Graphics
- Exit

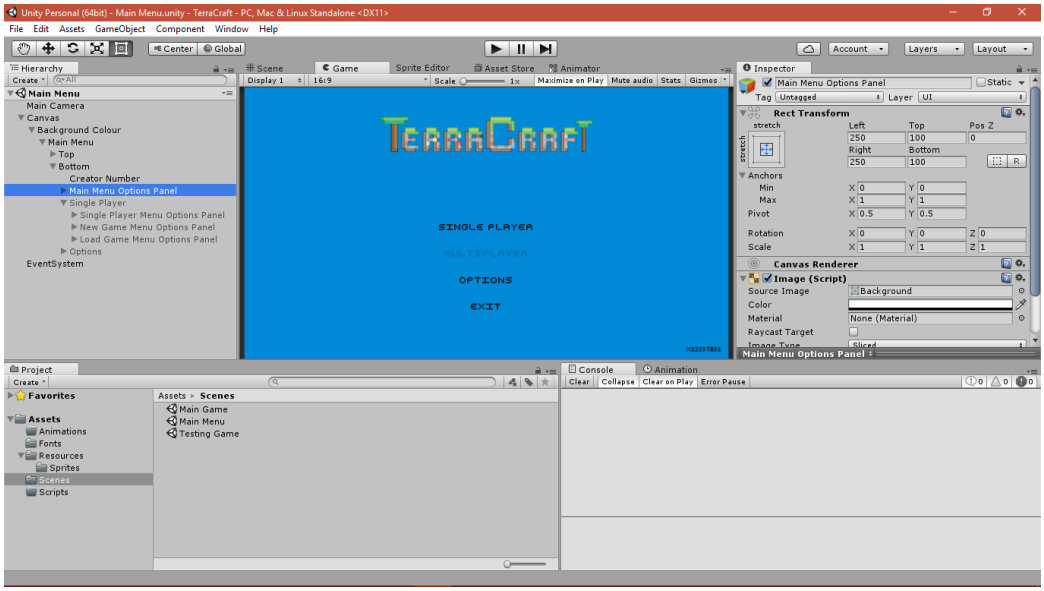


Figure 2. Main Menu layout

3.3.2. NOVEMBER 5TH, 2016

Over the past two days, I've been creating the sprites (graphics) for TerraCraft. The graphics consist of basic tools, weapons and character body parts, that, when put together in the game, come together to form the character of your creation. By separating each body part of the character will allow me to easily animate and control the character when playing the game.

When added to Unity, I cut the one image into multiple, smaller and individual images, i.e. swords, pickaxe, character parts. The image below shows each piece in the image and how they're divided:

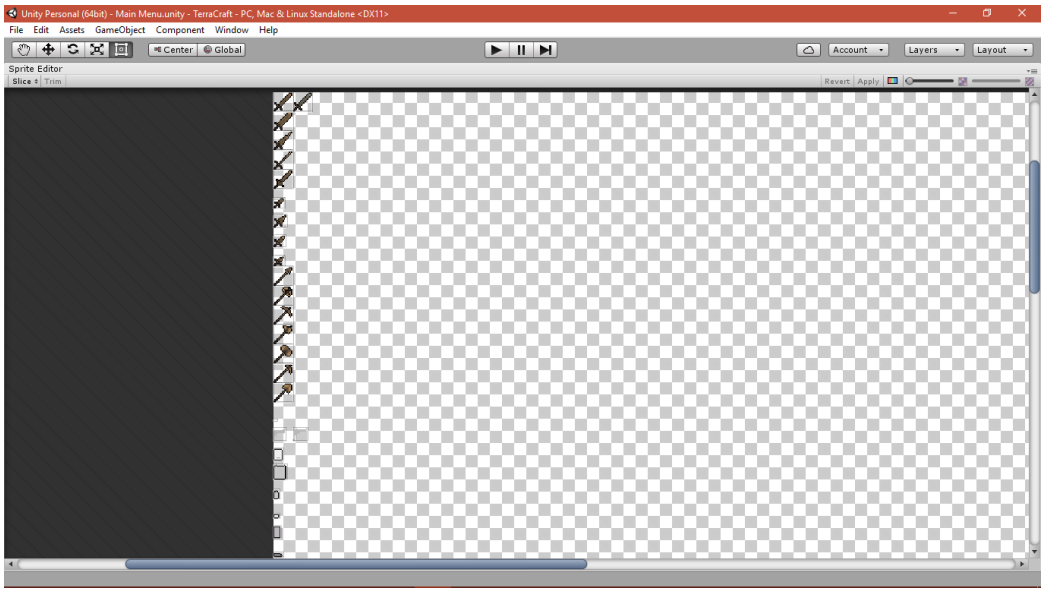


Figure 3. TerraCraft Sprites

Shown below is the character which is made up of the small images from the above image. Each body part is nested correctly, so, when it animates, the images stay in a position lock relative to their parent objects.

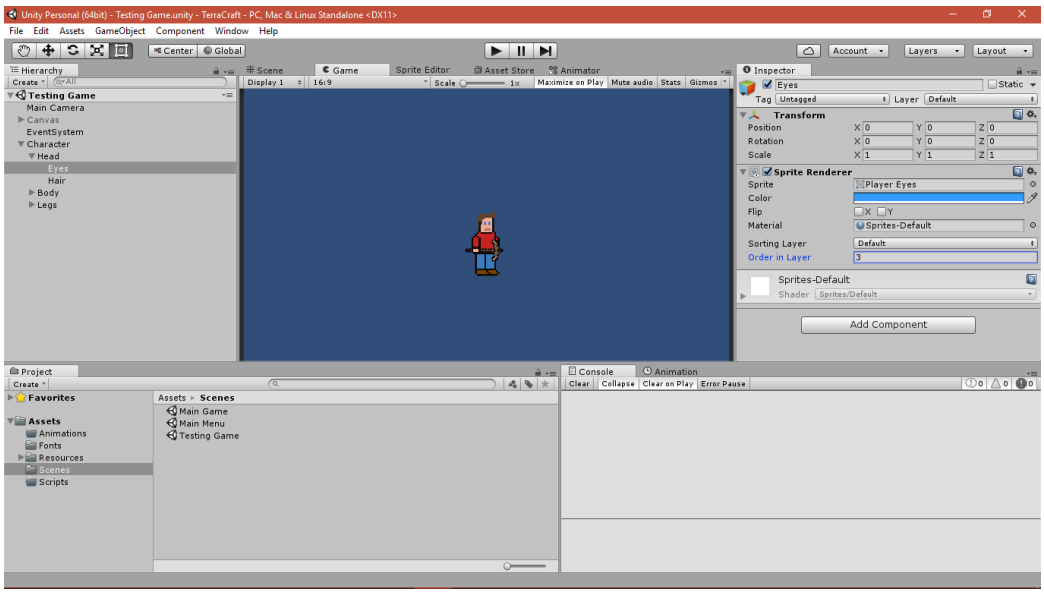


Figure 4. TerraCraft Character

3.3.3. NOVEMBER 30TH, 2016

This is the last entry for the November reflective journal. As far as the development goes, I'm not happy with the progress of this month, due to the intensity of the assignments and poor time management. I should've given more time into the development of the project.

During the next month, I'll be prioritising my assignments and working with this project more. I need to get the development of TerraCraft started and to get started on developing more complex structures of the game.

3.4. DECEMBER 2016

3.4.1. DECEMBER 1ST, 2016

Today I began working on the UI end of the game – to make it a little bit easier on the eyes and to give it a proper game look. I started with the in-game UI, adding the look of the inventory system, the player properties window and an in-game clock (which will aid in developing the Day/Night Cycle).

In the inventory system, I've only done the basics. This includes the layout of the items – split between a hot bar (for quick easy use of items) and the remaining inventory. The inventory will be fully customisable and can stack items.

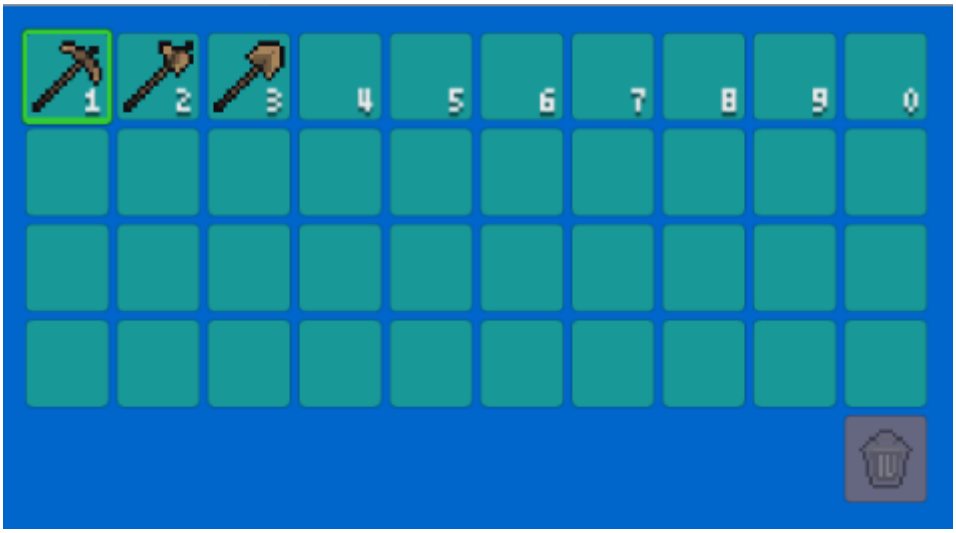


Figure 5. Basic Inventory System w/ items

With the player properties window, it includes a players' health and the players' armour. They will give the user an overview of their characters' properties.

Figure 6. Player health and armour

3.4.2. DECEMBER 3RD, 2016

Today I implemented Time with the game. Time will integrate with the Day/Night Cycle. The Day/Night Cycle is complex in its own way. To judge what time the sky changes colour at and then to implement the sun/moon rising and setting. To do this, I added an in-game clock. It increments using Unity's *Time.deltaTime* function, which is basically real-time, where every second is 1 second – considering that Unity updates the game at 60 frames per second (meaning that a second real-time is 60 seconds in Unity-time, without Time.deltaTime). The in-game time will increment +1 every 2 real-time seconds.

```
0 references
void LateUpdate()
{
    Timer += Time.deltaTime;
    if (Timer >= TimeDelay)
    {
        ResetTimer(0.0F);
        IncreaseTime();
    }
}
```

Figure 7. Time incrementing after a TimeDelay (2 seconds)

```
1 reference
void IncreaseTime()
{
    Minutes++;
    if (Minutes > MaxMinutes)
    {
        Minutes = 0;
        Hours++;
        checkDayPeriod();
        if (Hours > MaxHours)
        {
            Hours = 0;
        }
    }

    WorldTime.updateTime(getTime(WorldTime.getClockType()));
}
```

Figure 8. How time increments

The clock is modifiable between two settings – the 12-hour clock format (12:00 AM) and the 24-hour clock format (00:00). I then separated certain times of the day and gave it a name. The times of the day are as followed: **00:00 – 05:59** is known as "*Early Hours*". **06:00 – 11:59** is "*Morning*". **12:00 – 17:59** is "*Afternoon*". **18:00 – 20:59** is "*Evening*" and **21:00 – 23:59** is "*Night*". With these barriers in place, it will help give a boost with the Day/Night Cycle development, giving each time of day a different sky colour.

```
2 references
void checkDayPeriod()
{
    if (Hours >= Midnight && Hours < Morning)
    {
        this.DayPeriod = "Early Hours";
    }
    else if (Hours >= Morning && Hours < Noon)
    {
        this.DayPeriod = "Morning";
    }
    else if (Hours >= Noon && Hours < Evening)
    {
        this.DayPeriod = "Afternoon";
    }
    else if (Hours >= Evening && Hours < Night)
    {
        this.DayPeriod = "Evening";
    }
    else if (Hours >= Night && Hours < 24)
    {
        this.DayPeriod = "Night";
    }
    else
    {
        this.DayPeriod = "Early Hours";
    }
    WorldTime.updateDayPeriod(DayPeriod);
}
```

Figure 9. CheckDayPeriod code that gives Time of Day

Figure 10. The clock in-game UI

3.4.3. DECEMBER 9TH, 2016

I'm starting to work on the Mid-Point Technical Report today. It seems a bit a bit unnecessary to have so much documentation for the project as it causes a lot of pressure with both the upload and the development of the project. It will take some time doing it, as I'm looking over the past reports and the document template. I will add the finished document later when it's uploaded.

3.4.4. DECEMBER 11TH, 2016

Okay, I managed to upload it, but it was late, due to me being in work. Only in the door at 12:00 AM and took ages to get the document processed and ready to upload. I've attached it below to allow easy access.

Technical Report.docx

3.4.5. DECEMBER 20TH, 2016

Today I had to present my game to two judges in the college, Anu Sahni and Catherine Mulwa. It was to give a brief overview of the entire project – giving specs and talking them through each functionality that the game will have, hopefully, by the end of the development process and the plans on what the testing phase of the project will be like. They were interested in my idea to release the game on Facebook, Steam and my website and it'll give a boost in the feedback. The feedback will be directed back to me and I will implement any changes that are suggested.

3.4.6. DECEMBER 31ST, 2016

I didn't get any more development done for this month, because of the Christmas rush. Being in work and looking after family really halted the development. I'm looking to get the Day/Night Cycle done next month (hopefully) and start developing more like the world generation and/or the parallax effect.

3.5. JANUARY 2017

3.5.1. JANUARY 19TH, 2017

Today I started to develop the Day/Night Cycle further. I researched online about diverse ways it could be done, but all of them seem to focus on the 3D development side of things – which don't seem to benefit me at all. I've tried different tactics to try get colours to change at a specific moment and to change back at another using a Unity function called *'Color.Lerp'*. This allows a change in colour after a certain amount of time, i.e. from red to green in 3 seconds.

Using the day periods, I added more, to allow for better accuracy of the sky colour. The day periods are as followed: *midnight, dawn, sunrise, morning, afternoon, evening, sunset, dusk* and *night*. Using these, I tried to use the *'Color.Lerp'* function, but it didn't work well, as it changed too quickly and/or kept changing back to the original colour, which was not intended.

3.5.2. JANUARY 20TH, 2017

World Generation is always a tough point. It involves making hundreds or thousands of little blocks, each of which will have its own properties, and set them at a specific point in the game, creating a world-like setting. Today I started to create a C# script to implement the basics of this.

Starting off, I added a List that would hold all the block types. The blocks are:

- Grass,
- Soil,
- Stone,
- Copper,
- Iron,
- Sapphire,
- Ruby,
- Emerald,
- Gold,
- Silver,
- Graphene, and
- Diamond.

With help from the List, I implemented a function that has two *for* loops with some properties inside, i.e. how many blocks to implement into the game, width and height. In the *for* loops, they would repeat numerous times, as many times as I tell it to, using variables, and other methods, until every block is placed.

When placing a block, I measure what depth each block is placed at, so I know what types can be placed there, i.e. the first 6 layers of the world will be grass and soil. Below this, it will contain a mix of stone and every other block, minus the grass and soil blocks. This will allow for block randomisation, giving the game common blocks and rare blocks.

The only problem I'm facing with this, is how to implement trees into the game, so that wood can be obtained. Maybe I'll focus on that sometime this week.

3.5.3. JANUARY 25TH, 2017

Today I'm starting to implement the NPCs of the game. I'll start off with something simple: a slime. They're small and move very easily so they will provide a boost to production, as it won't get too complicated.

To get started, I worked on the properties of the NPC, i.e. the health, damage it deals, etc., and worked with how it interacts with the player. When in a certain range, using colliders, the player will have the opportunity to attack the slime, slowly degrading its health until it depletes and leaves the slime with no health. But as Newton's Third Law states: *"Every action has an equal and opposite reaction"*. As the player can attack the slime, the slime has the chance to attack the player. When the player encounters the slime, damage will be dealt to the players' health and knock the player back a certain distance. This will show the intensity of the attack and maybe warn the player off attacking.

3.6. FEBRUARY 2017

3.6.1. FEBRUARY 28TH, 2017

I got nothing done this month because my laptop is fully broken. The cooling fan inside the laptop is causing problems with the overall performance and memory. Whenever I tried to use Unity, it would freeze and crash.

I'll have to go out and buy a new laptop next month.

3.7. MARCH 2017

3.7.1. MARCH 19TH, 2017

Unity is installed and updated, as well as Microsoft Visual Studio and Adobe Photoshop.

I got started today with the melee from NPCs, which allows them to move towards the player, when attacked, and start attacking, which slowly decreases the (player's) health.

I used a 2D box collider to detect nearby NPCs, adding them to a List<>, that could then be attacked, if within the bounds, and when they're attacked, if they were not friendly, they would attack the player until either they or the player died.

I found difficulty with this as the NPCs would keep adding to the List<> and/or wouldn't be detected. I fixed the problem by making the box collider a little bigger and then comparing each of their predefined IDs that Unity assigns, and if they were the same, they wouldn't add to the List<>.

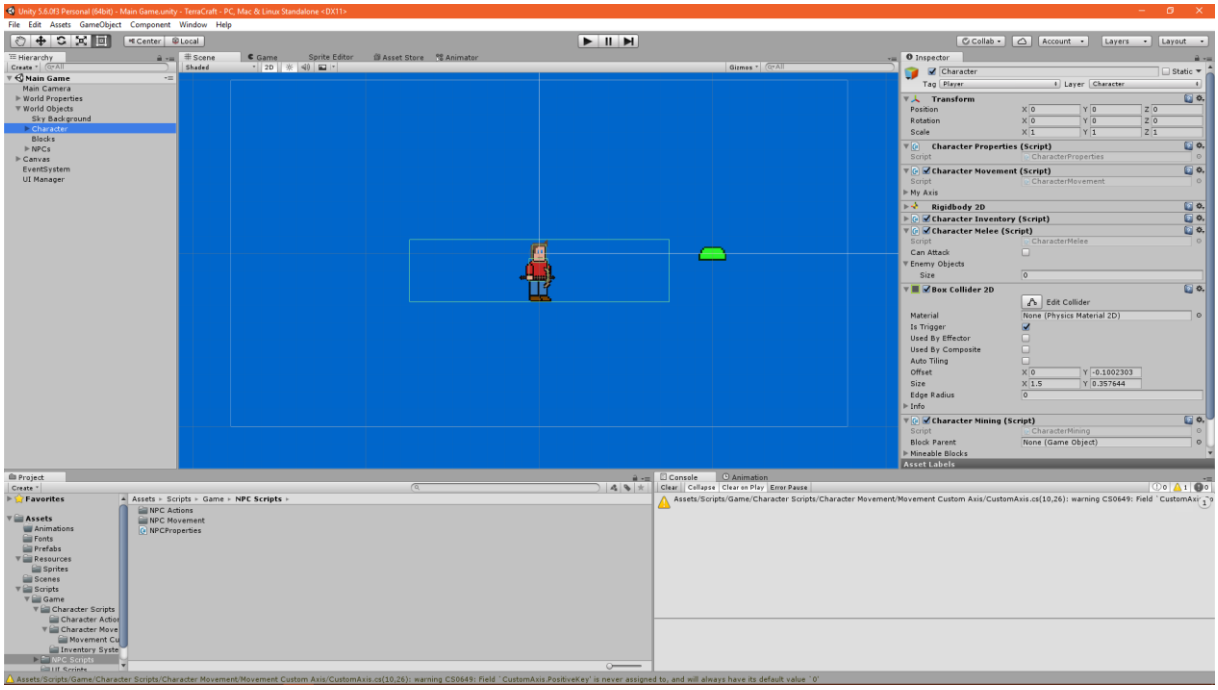


Figure 11. 2D Box Collider detecting NPCs

3.7.2. MARCH 20TH, 2017

Today I began working on the mining of the blocks. I approached it in the same way I done the melee – by using the 2D box collider to detect nearby blocks that could be mined, making sure that the same blocks can't get added when within the bounds and that they get removed when they exit the bounds. This caused massive errors with the Unity engine, as it had an out-of-memory crash every time that it started the game, thus, making the mining an impossible task to do.

I logged an error with Unity and got nowhere in return.

3.7.3. MARCH 21ST, 2017

Today I began working on the mining of the blocks. I approached it in the same way I done the melee – by using the 2D box collider to detect nearby blocks that could be mined, making sure that the same blocks can't get added when within the bounds and that they get removed when they exit the bounds. This caused massive errors with the Unity engine, as it had an out-of-memory crash every time that it started the game, thus, making the mining an impossible task to do.

I logged an error with Unity and got nowhere in return.

4. OTHER MATERIAL USED

During the development of the project, I haven't used any assets from Unity and/or any other website/forum.