National College of Ireland
BSc in Computing
2016/2017

Dylan Walker
x12328836
x12328836@student.ncirl.ie

"Outback"
Platformer game

Technical Report

Table of Contents

Declaration Cover Sheet for Project Submission

SECTION 1 *Student to complete*

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 particular 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.

1. Executive Summary

2. Introduction

Outback is a platform style adventure game set deep in the Australian outback. In the game the player must complete the course of the level while trying to avoid environmental hazards, obstacles and enemies in the process. The player will be equipped with a boomerang with which they can use to fend off and destroy the enemies. They can also jump onto the enemies in order to destroy them and also use their jump to avoid some of the environmental obstacles they encounter during the level. The player has a limited number of lives in order to reach the end before the game is over. However if the player dies while having lives remaining the player will respawn at the nearest checkpoint. The primary objective of the game is to survive the Australian bush while providing an immersive, enjoyable experience for the player with some incentives such as collectibles to gather along the way to motivate the player to progress toward the end.

2.1 Background

The gaming industry has grown exponentially over the past decade and continues to expand at a steady rate. This is largely due to the number of independent developers who can now develop games for the relatively new platforms which have opened up to the market over the past few years. Mobile and Web games are becoming increasingly popular and anyone with access to a smartphone or Internet can now turn their device into a portable gaming console with ease. Previously games would have been developed by large game developing studios in large teams for dedicated gaming platforms such as Playstation, Xbox, and PC. Users would have to own one of the dedicate consoles, purchase a hard copy and be limited to playing their game to where their console was set up. Mobile and Web games have gained a whole new audience for industry. With the vast number of indie developers today, games can be made in smaller teams for little or no cost and deployed almost instantly. This method allows these developers to take more risks in their games and completely implement those risks around their own creative ideas with very little to lose compared to dedicated gaming development companies.

The app stores on IOS, Android and Google Play host hundreds of hundreds of thousands of games. According to IOS statistics the number of games on the Apple store has risen from 7 thousand in July 2008 to over 630,000 in July 2016. This year it has also been reported that as of March 2017 gaming was now the most popular category on the app store with over 25% of apps being games. These numbers are significantly higher with Google Play and Android as their market place is less restrictive to newer developers.

The inspiration for my game was heavily influenced by these indie games and markets. 2D games are very much coming back to the forefront of portable gaming now that we have more and more devices we can use to play them on, while continuing to expand the industries audience.

The project was developed using Unity game engine. Although 2D has only been implemented in Unity over recent years I chose unity over similar game engines due to its vast and active community. This large network of users allowed me to explore a broad range of learning support tools from documentation, forums, tutorials and demos. Unity offers an extensive range of tools to allow any developer to explore their own creativity and potential. It is an extremely powerful game engine, which uses scripts coded in C#.

The game was built with so that it may have the capability to run on a mobile device and be apart of the ever-growing gaming sector within the mobile market places. For the purpose of this project I have made a version which is hosted on Amazon Web Services(AWS) as well a standalone Mac OS version.

2.2 Aims

The aim of this project was to create a fully functional 2D platformer that will be playable on a web browser as well as having a stand alone Mac OS build. The style of the game is to have a more modern style 2D game with a cartoon design rather than pixel art platformers. The game will be developed with the intention of having a broad target audience so that various types of people, of all ages and skill can engage with and enjoy the experience. It will have simple controls and functions to ensure it is easily playable for its intended audience. It will feature incentives in the form of collectibles to persuade the player to progress to the end, only if they can survive the hazards and dangers in the environment and complete the level in order to ultimately survive the Australian outback.

2.3 Technologies

2.3.1 Unity

Unity is a powerful cross platform game engine, which allows for the development of either 2D or 3D games. These games can be created for a variety of different platforms such as Mac OS, Windows, Web, and Mobile devices. In total however unity is compatible with 27 platforms as of 2017 and most recently have added capabilities to include support for VR technologies. Unity is widely considered to be the leading game industry software. More games are developed with unity than any other game engine. Last year (2016) Unity games reported that there was over 5 billion downloads of games made with unity and also stated that over 34% of the top mobile games are developed with their software. Along with independent developers and studios the company also has an extensive client base. Disney, Electronic Arts, Nasa, Ubisoft and Warner Bros are a few of the larger companies who make use of Unity technology. Unity's presence is ever growing in the gaming industry and they expect to be the leading game engine over the next few years.

Unity was the primary application used to build this project. It features a main scene which displays the visual elements of the project. Elements that are used in the game are known as GameObjects. Each of these objects can be customized in the inspector panel. The inspector offers a range of tools where users can modify the scale, position and rotation of that particular object. Additionally users can also add or modify behaviors such as physics or animations. Scripts can then be attached to that object in order to give the object a particular task in the environment or change how it interacts with other game objects.

2.3.2 Visual Studio Code

Unity offers its own integrated development environment to make use of, however I chose to use Microsoft Visual Studio Code as my primary code editor for scripts as it was easy to integrate with Unity. This is just due to my own preference as I felt it provided a more enjoyable experience to work with and greater usability. Visual Studio Code allows users to create C# scripts that can be used within Unity. Since scripts can be attached to game objects these scripts can then access the objects components, co-ordinates as well as other scripts attached to the same object.

2.3.3 Adobe Illustrator and Photoshop
Adobe Illustrator and Photoshop are apart of the adobe creative suite. These applications are designed as graphics editors. Photoshop focuses primarily on raster graphics to edit images in raster formats such as JPEG, PNG, GIF and TIFF. While Illustrator is more of a vector graphics editor. These graphic editors allowed me to customize graphics that would give me the tools to create the cartoon like visual style of the game.

2.3.4 Amazon Web Services (AWS)
Amazon Web Services is a cloud services platform which offers a variety of services from databases, storage, gaming development, analytics to name just a few of its immense catalog. For this project I used AWS to deploy the game online by storing the build on the AWS S3 cloud storage. I could then have users play the game through their own browser.

3. System

3.1 Requirements

3.1.1 Functional Requirements

Start New Game
The user should be able to start a new game when launching the game for the first time. This requirement has not changed from the original requirements specification.

Pause Game
The user should be able to pause the game whenever they want during any time in the game play. The player will press the “Esc” key button which will pause the game and also offer a pause menu screen where they can chose to restart or quit the game.

Quit Game
The user should be able to quit the game whenever they want during game play.

Collect Items
The user should be able to collect objects that are placed at random points throughout the level. The items the player will be able to collect are a health restore, a life restore as well as two different coloured boomerangs that will be worth separate values and act as the incentive to progress further into the level.

Health System
The user should be able to lose and gain health during the level. The player will lose health through the means of obstacles and hazards within the level while they will be able to gain health in the form of a collectible item. When the players health reaches zero they will lose a life. However if they collect a life collectible the life increases by one. Each life has 6 healths. If the player loses one of there 6 healths they can be restored through another collectible. If the players lives run out, the player has failed the level and they can return to the main menu where they can restart or quit the game.

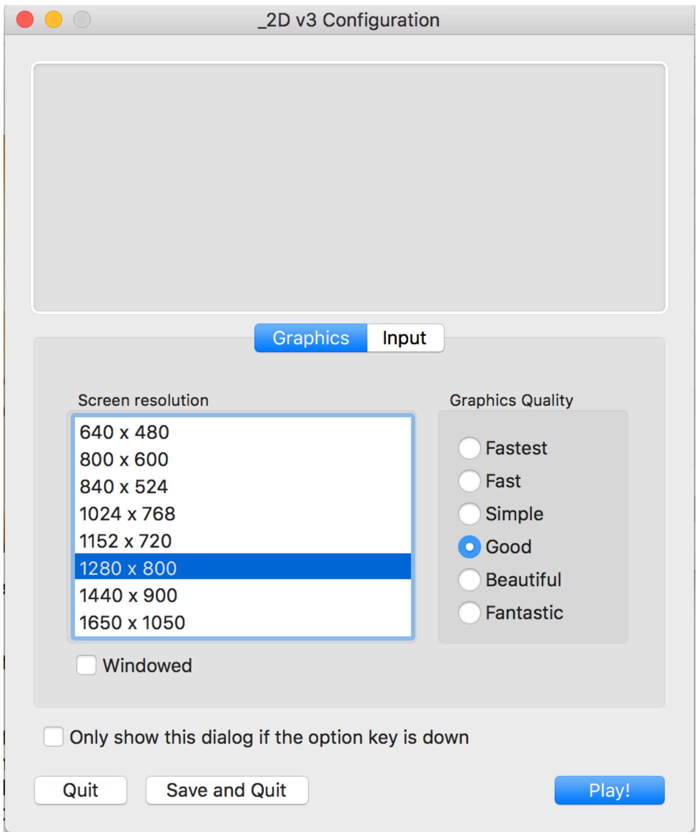
3.1.2 Data Requirements
Unity uses PlayerPrefs which stores and accesses player preferences and information between different sessions. This allows the game to set and get values related to the games session. An example of this is saving values such as health or score and loading it

again when the player returns. These values can also be used if the game had a various levels whereby those values would be carried over into the new scene.

3.1.3 User Requirements

The game has two builds one for web and another for Mac OS. Users who opt for the web version will not be limited to a particular platform or operating system. The web build will run on any platform with a web browser capable of running JavaScript and flash. The users will then only need access to a web browser capable of running the above along with a keyboard, access to the Internet and the game URL.

The Mac OS version will only run on Mac. However upon launch of this version users will be prompted with a settings screen whereby the users can optimize the game to suit their display and graphics card. This will ensure it will run smoothly on any version of Mac OS. Users for this build will need a version of Mac OS and a keyboard in order to play the game.



_2D v3 Configuration

[✓ Graphics] Input

Screen resolution
- 640 x 480
- 800 x 600
- 840 x 524
- 1024 x 768
- 1152 x 720
- [✓ 1280 x 800]
- 1440 x 900
- 1650 x 1050

Graphics Quality
- Fastest
- Fast
- Simple
- [✓ Good]
- Beautiful
- Fantastic

☐ Windowed

☐ Only show this dialog if the option key is down

Quit | Save and Quit | Play!

Figure above shows the dialog box Mac OS users will be given upon launch

3.1.4 Environment Requirements

In order to play the game users must preferably use a stable version of Mac OS or a web browser that runs with JavaScript and Flash player.

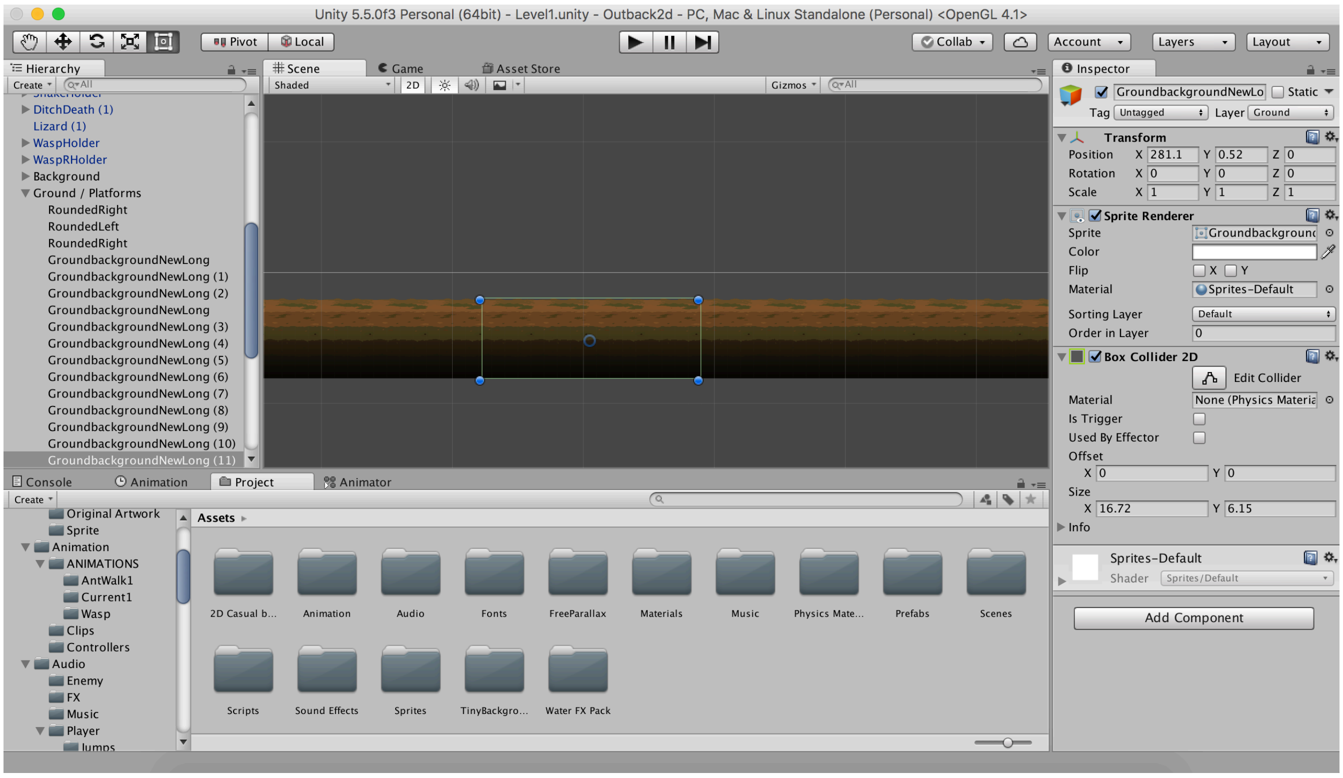
3.1.5 Usability Requirements

The game has a simple, neat and easy to understand graphical user interface (GUI). This was implemented in unity and consists of labels and headings along with colourful buttons with alternating colour to indicate selected or highlighted buttons. This GUI was designed with the goal of being easy to understand, navigate and learn so that it can appeal to a broader audience consisting of various ages, gaming knowledge, and computer skills.

Instructions on game play will be printed on the main menu screen, which will appear on the initial launch of the game.

3.2 Design and Architecture

Unity uses a variety of various interfaces in the development screen in order to develop projects. Within the development area there consists a scene view, hierarchy and inspector. Game Objects in unity refers to any object which can potentially be used in the scene for the game. These can vary from characters, shapes, particle effects etc. A project within unity is created by using game objects and scripts which will provide the objects do their desired function in the scene view. The objects are then arranged in the hierarchy panel. This panel organizes where objects in the world are in relation to each other. Game Objects can also have parents or children. Child objects which have parents are positioned relative to the parent and not by the world co-ordinates. Below, figure 1, is a screenshot of the unity workspace showing the hierarchy on the top left, the inspector along the right, the scene view in the middle along with project folder panel along the bottom.



GroundbackgroundNewLo Static
Tag Untagged Layer Ground

Transform
Position X 281.1 Y 0.52 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1

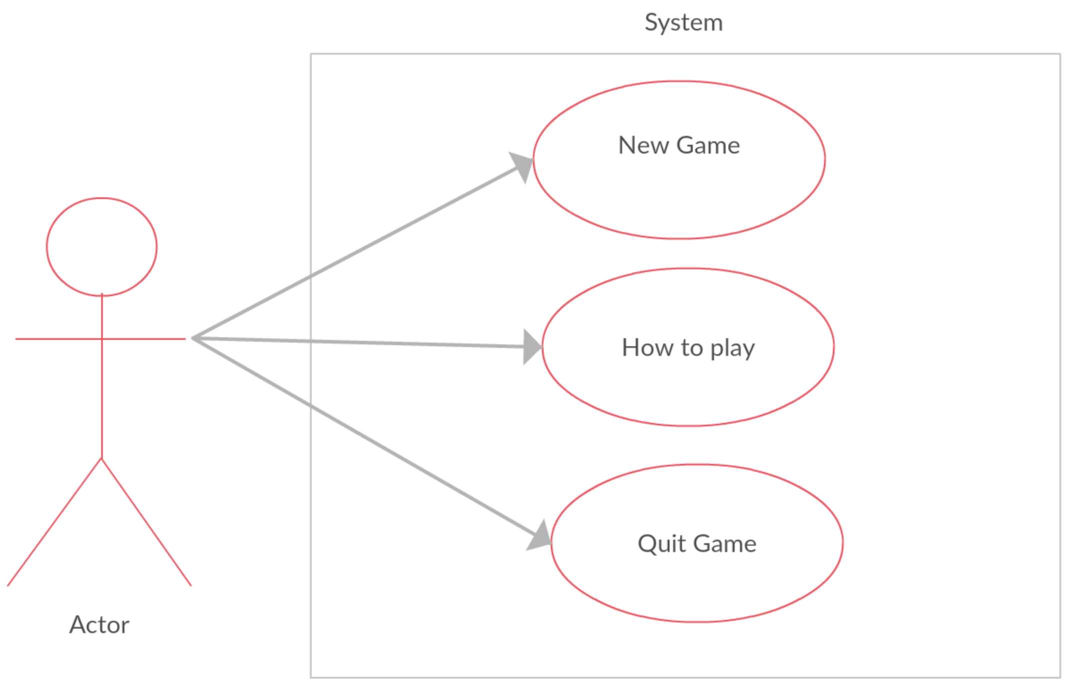
Sprite Renderer
Sprite: Groundbackground
Color
Flip: X Y
Material: Sprites-Default
Sorting Layer: Default
Order in Layer: 0

Box Collider 2D
Material: None (Physics Materia
Is Trigger
Used By Effector
Offset X 0 Y 0
Size X 16.72 Y 6.15
Info

Sprites-Default
Shader Sprites/Default

(Figure 1: Unity Workspace, showing hierarchy, scene view, and inspector)

3.2.1 Class Diagram

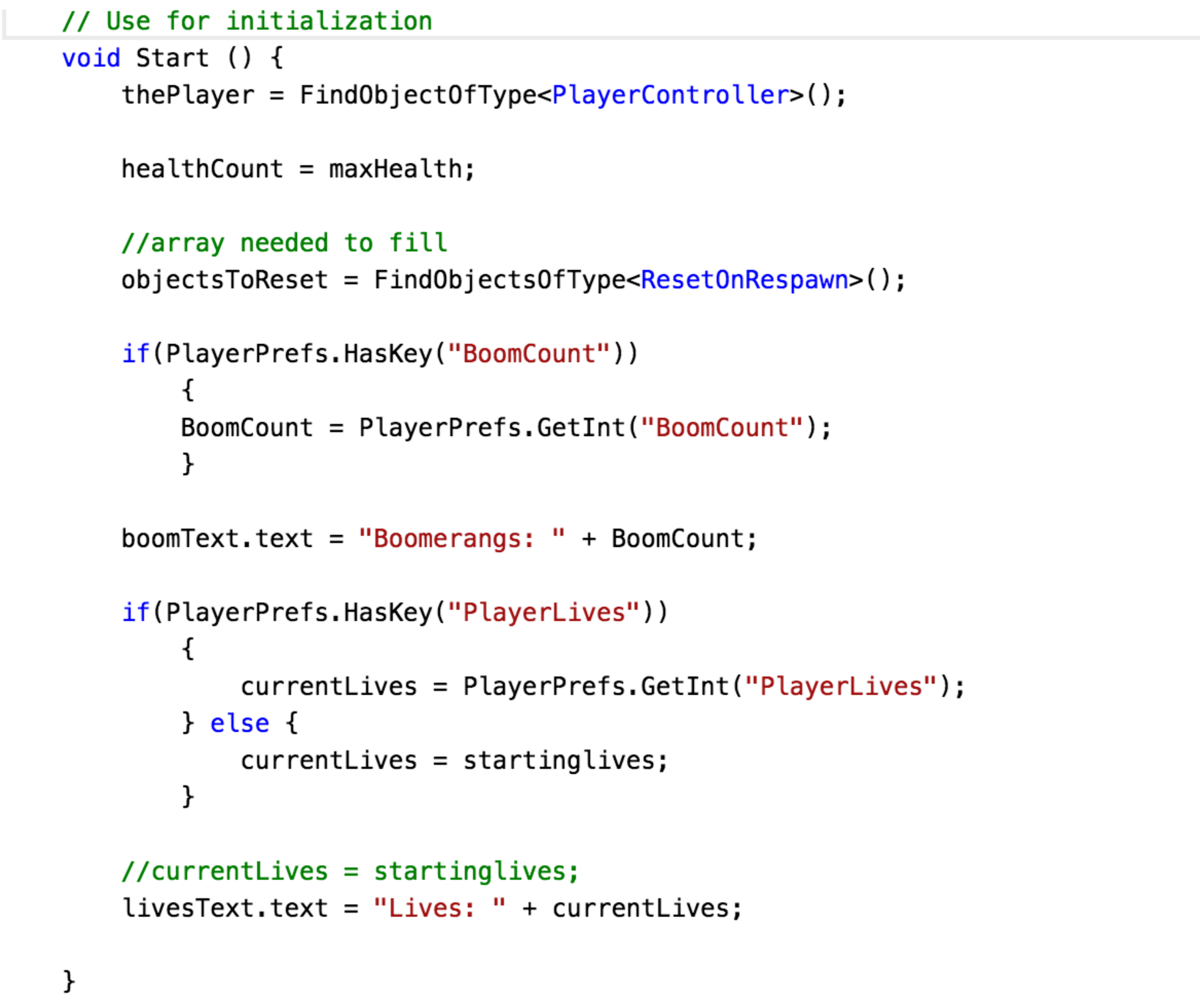


When the player launches the game the main menu will appear and be the first interaction the user has with the system. The main menu will display three options where the user can select only one. The first option is “new game”. This option will load the scene for the player to begin playing the game. The “how to play” option will bring the player to a page displaying some brief information about the game objectives and the player controls. Finally the “quit game” option, which will simply close the system.

3.3 Implementation

3.3.1 LevelManager.cs

For the purpose of the project I created a level manager script which controlled all the main functions to handle, such as Player Health, Respawn, UI, collectibles, and reset. The level manager handles these functions to minimise the amount of scripts needed to attach to game objects. This also allows easier implementation of the same functions that will be used in later levels. Below is snippets along with brief explanations from the script.



```
    // Use for initialization
    void Start () {
        thePlayer = FindObjectOfType<PlayerController>();

        healthCount = maxHealth;

        //array needed to fill
        objectsToReset = FindObjectsOfType<ResetOnRespawn>();

        if(PlayerPrefs.HasKey("BoomCount"))
            {
            BoomCount = PlayerPrefs.GetInt("BoomCount");
            }

        boomText.text = "Boomerangs: " + BoomCount;

        if(PlayerPrefs.HasKey("PlayerLives"))
            {
                currentLives = PlayerPrefs.GetInt("PlayerLives");
            } else {
                currentLives = startinglives;
            }

        //currentLives = startinglives;
        livesText.text = "Lives: " + currentLives;

    }
```

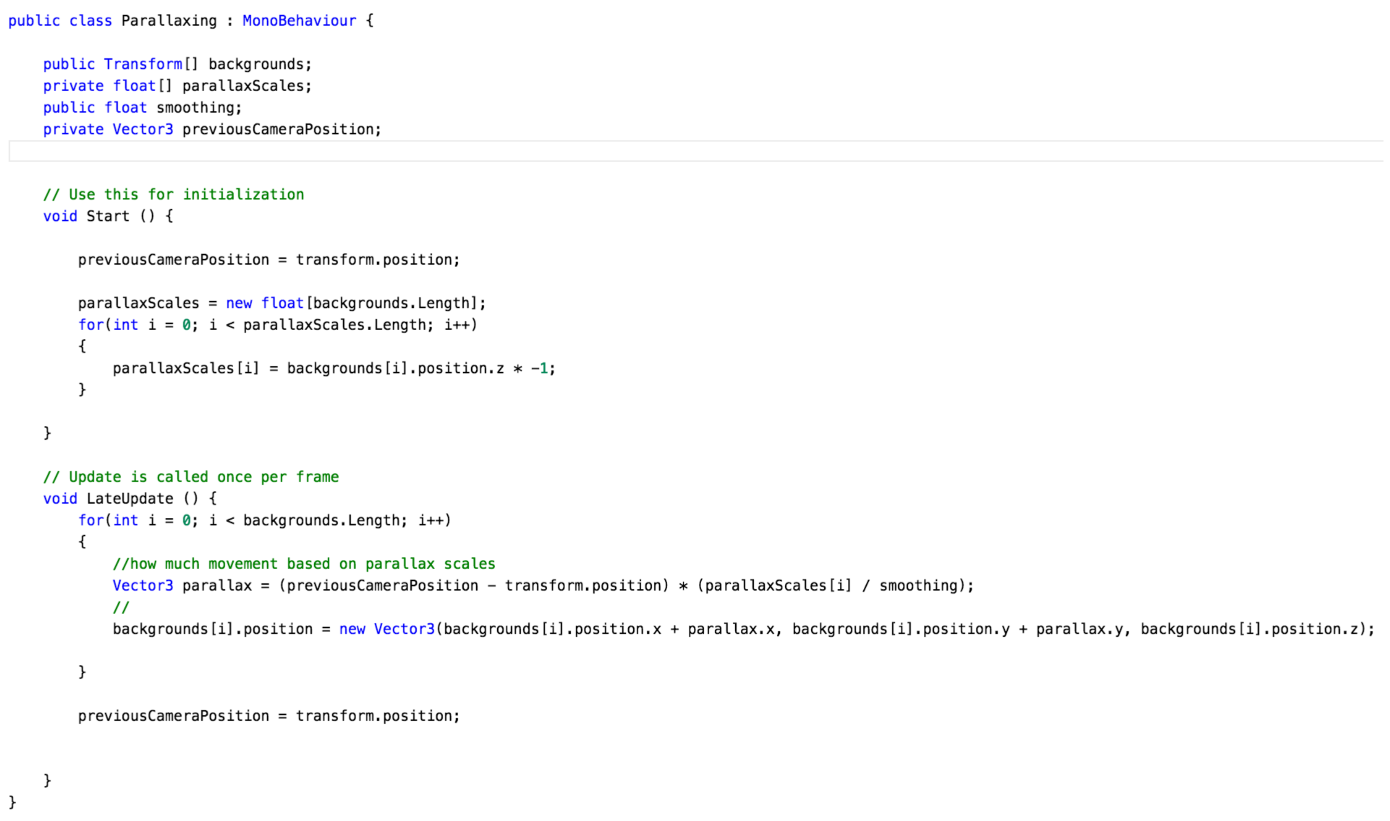
Above: The level manager finds the object it interacts with. In this case the object is the player controller script. The healthCount is then set to maxHealth for the beginning of the level. The next script the level manager interacts with here is the ResetOnRespawn. This is simple an array of game objects I have set that will reset when the player respawns into the level. The PlayerPrefs are the valuables which are saved from the game session. These values are just to make sure the player does not have any values in which the game will carry over. Lives and Boomerangs are the only values it will check.

```
    public void HurtPlayer(int damageToTake)
    {
        if(!invincible)
        {
            //take away damage from health
            healthCount -= damageToTake;
            UpdateHeartMeter();

            thePlayer.Knockback();

            thePlayer.hurtSound.Play();
        }
    }

    public void GiveHealth(int healthToGive)
    {
        healthCount += healthToGive;

        if(healthCount > maxHealth)
        {
            healthCount = maxHealth;
        }

        heartSound.Play();

UpdateHeartMeter();

    }
```

Above: This next snippet from the level manager script demonstrates apart of the player health system of the game. This function works alongside a HurtPlayer script. Objects with this script attached and set as a trigger will cause the players health to decrease when the two game objects collide. The objects get there own value set in the inspector to take from the player allowing different objects to use the same script but to have different effects on the players health. If one of the objects collides with the player a knock back effect and animation is activated along with a sound to signal the user that the player is hurt.
The method below interacts with HeartPickup script, which is the opposite to the hurt player script. This method gives health to the player when the player collides with objects with the script attached. A sound effect is played to signal the player of the interaction.

3.3.2 Parallaxing.cs

One of the features I was keen to integrate into the project was a paralleling effect. This effect creates an illusion of having much more depth to 2D games as layers in the background will move at different speeds in relation to the players movement. This script is attached to the main camera in the scene so it runs throughout the whole level giving an over all better feel to the game. The effect is achieved by creating an array of background objects at different positions along the Z axis in the scene. Since each object is at a different position in relation to the camera it appears to move at a different speed.



```
public class Parallaxing : MonoBehaviour {

    public Transform[] backgrounds;
    private float[] parallaxScales;
    public float smoothing;
    private Vector3 previousCameraPosition;


    // Use this for initialization
    void Start () {

        previousCameraPosition = transform.position;

        parallaxScales = new float[backgrounds.Length];
        for(int i = 0; i < parallaxScales.Length; i++)
        {
            parallaxScales[i] = backgrounds[i].position.z * -1;
        }

    }

    // Update is called once per frame
    void LateUpdate () {
        for(int i = 0; i < backgrounds.Length; i++)
        {
            //how much movement based on parallax scales
            Vector3 parallax = (previousCameraPosition - transform.position) * (parallaxScales[i] / smoothing);
            //
            backgrounds[i].position = new Vector3(backgrounds[i].position.x + parallax.x, backgrounds[i].position.y + parallax.y, backgrounds[i].position.z);

        }

        previousCameraPosition = transform.position;


    }
}
```

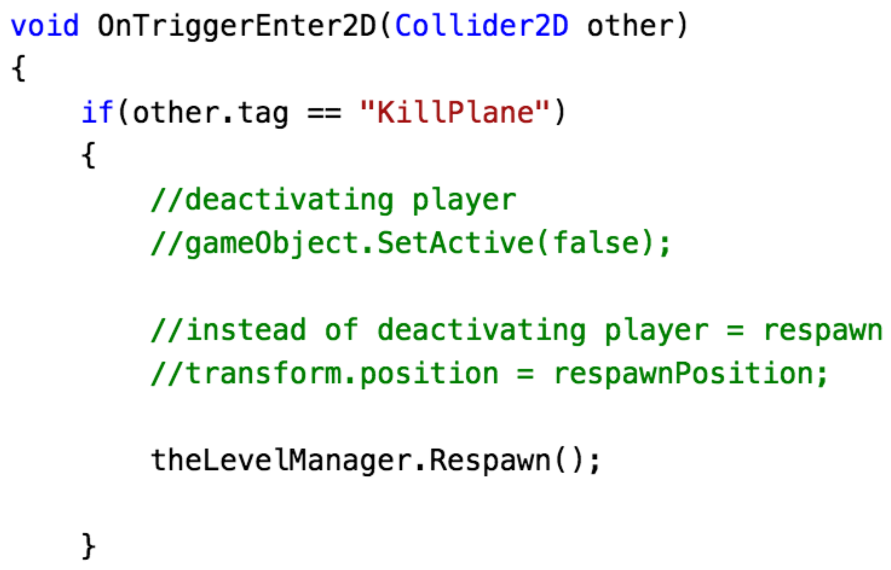
3.3.3 EnemyPatrol.cs

The Enemy Patrol script is a simple script, which will allow an enemy, or any game object with the script attached to move between two points, which are defined in the inspector within unity. The way this script was implemented was by creating two points a left and right and having the object move back and forth between those co-ordinates. It’s a useful way to give a simple game object some characteristics or function.

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

    if(movingRight && transform.position.x > rightPoint.position.x)
    {
        movingRight = false;
    }

    if(!movingRight && transform.position.x < leftPoint.position.x)
    {
        movingRight = true;
    }

    if(movingRight)
    {
        myRigidbody.velocity = new Vector3(moveSpeed, myRigidbody.velocity.y, 0f);
        transform.localScale = new Vector3(-1f,1f,0f);


    } else {
        myRigidbody.velocity = new Vector3(-moveSpeed, myRigidbody.velocity.y, 0f);
        transform.localScale = new Vector3(1f,1f,0f);



    }
}
```

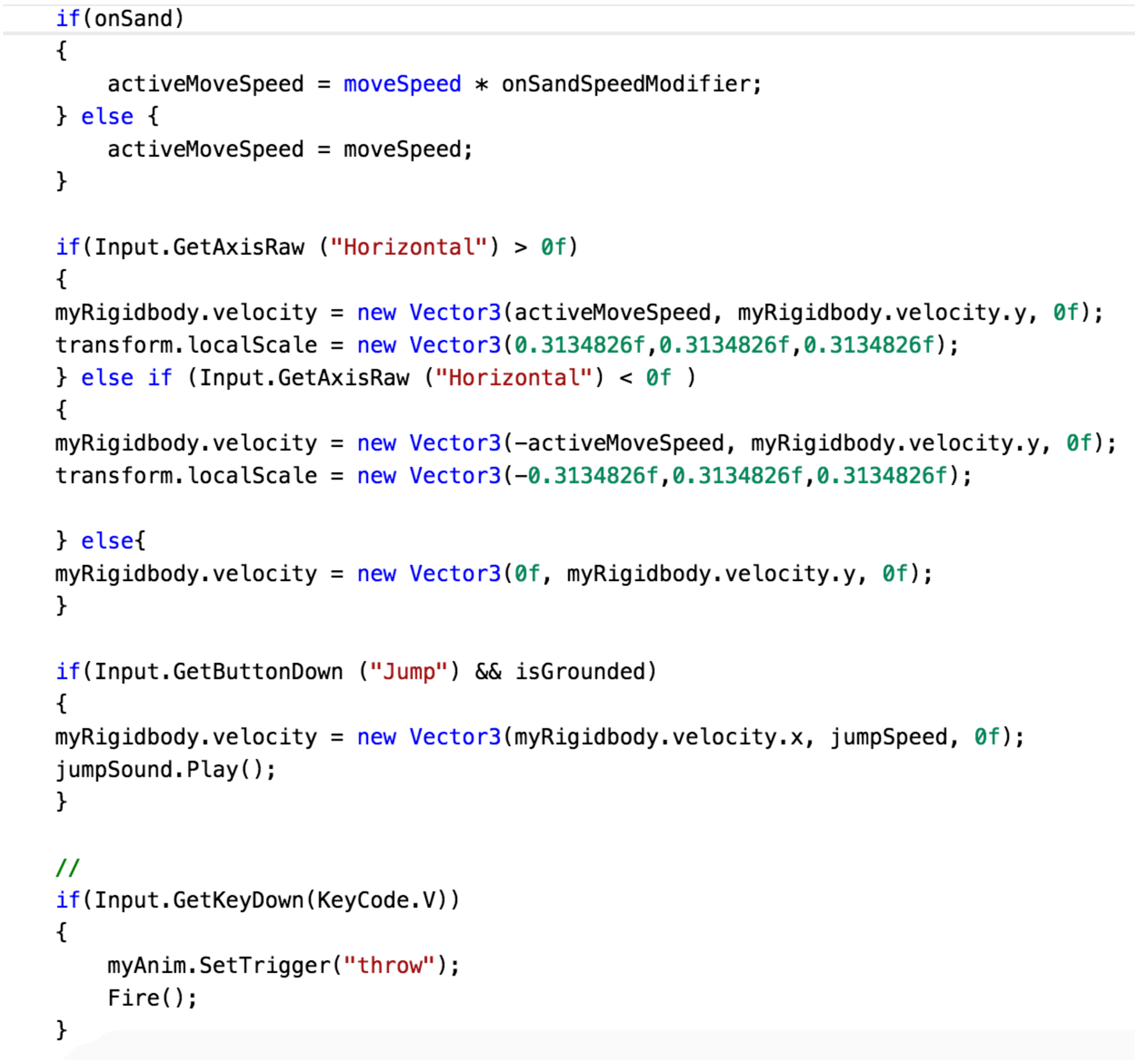
3.3.4 PlayerController.cs

The Player Controller is the script attached to the main character that controls all functions related to the player. The script also interacts with the level manager script as mentioned above. This script controls the characters movement along the X and Y axis along with telling the character which keys do a particular action. The script also controls whether the player can throw a boomerang. A common way of having the player interact with other game objects in a method known as OnCollision2D. This method works by carrying out a specific action or task when two colliders meet. A great way to demonstrate this is having a kill box along the bottom. The kill box is nothing other than a collider position along the bottom of the screen out of the players view. If the player falls down and the collider meets the kill box collider the player respawns. As mentioned above the level manager script controls the resets however the image below we see how that function is carried out when the collider of the player meets the game object tagged with “KillPlane”.



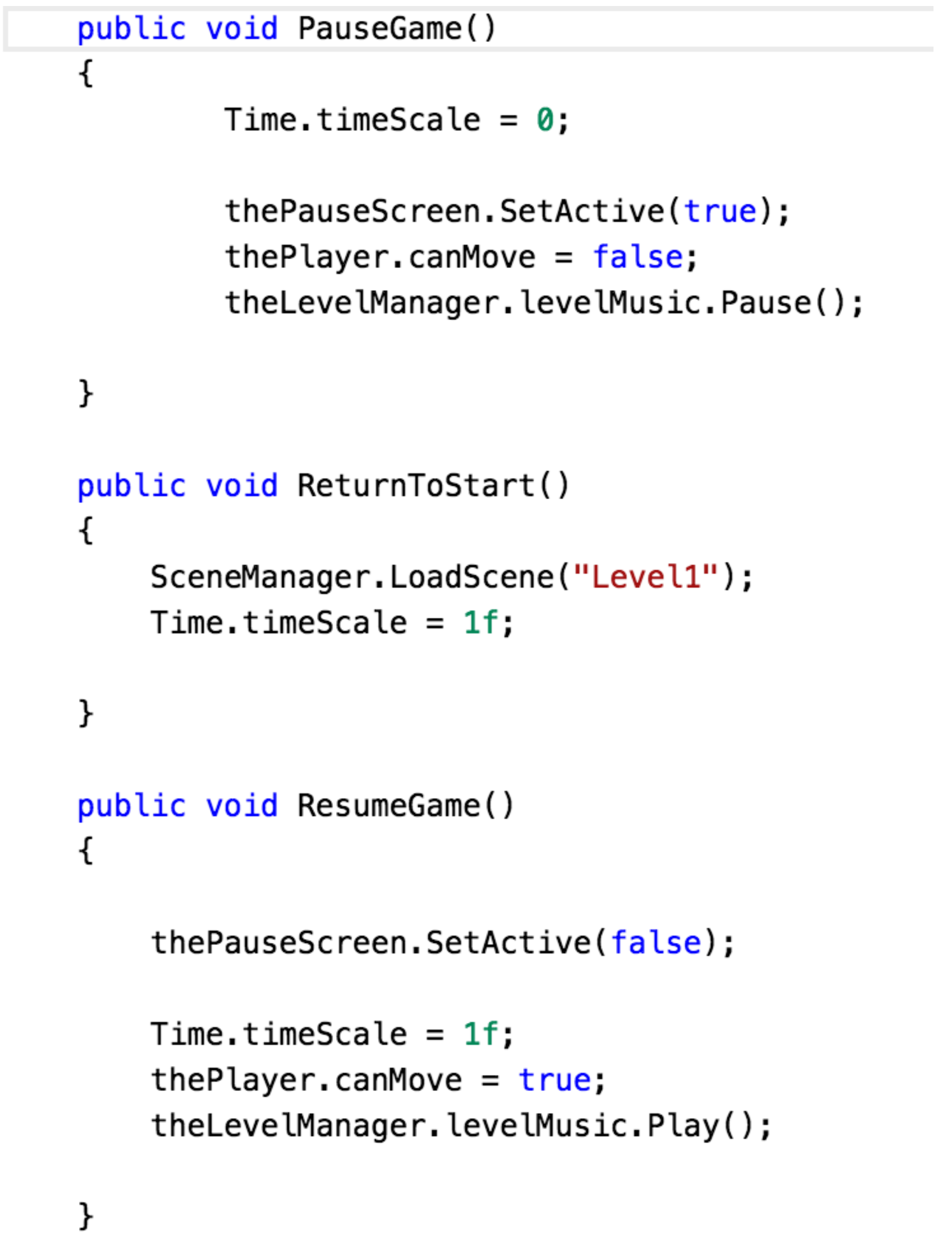
```
void OnTriggerEnter2D(Collider2D other)
{
    if(other.tag == "KillPlane")
    {
        //deactivating player
        //gameObject.SetActive(false);

        //instead of deactivating player = respawn
        //transform.position = respawnPosition;

        theLevelManager.Respawn();

    }
```

The next image below is another snippet from the player controller. This snippet largely represents the code used to move the player along the Y axis. It also demonstrates how the player can jump if the player is on the ground. A lot of platformers use double jumping where a player can jump once again while they are in the air. However for the purpose of this project and given that the character is human I chose to only let the player jump when he is on the ground. The snippet also shoes how Key Codes are used to implement a particular action with a specific key. Horizontal movement along the Y axis is controller with the arrow keys by “GetAxisRaw”, spacebar is represented by “GetButtonDown”, and the V key used to fire is represented as “GetKeyDown(KeyCode.V)”. These keys are changeable to whichever keys the developer prefers. The first “if” statement controls the players speed when on sand. This gives the effect of harder terrain in the game, as the player will drag along the sand at the pace set in the inspector. The last “if” statement is the triggers the method fire along with a short animation for the player.



```
if(onSand)
{
    activeMoveSpeed = moveSpeed * onSandSpeedModifier;
} else {
    activeMoveSpeed = moveSpeed;
}

if(Input.GetAxisRaw ("Horizontal") > 0f)
{
myRigidbody.velocity = new Vector3(activeMoveSpeed, myRigidbody.velocity.y, 0f);
transform.localScale = new Vector3(0.3134826f,0.3134826f,0.3134826f);
} else if (Input.GetAxisRaw ("Horizontal") < 0f )
{
myRigidbody.velocity = new Vector3(-activeMoveSpeed, myRigidbody.velocity.y, 0f);
transform.localScale = new Vector3(-0.3134826f,0.3134826f,0.3134826f);

} else{
myRigidbody.velocity = new Vector3(0f, myRigidbody.velocity.y, 0f);
}

if(Input.GetButtonDown ("Jump") && isGrounded)
{
myRigidbody.velocity = new Vector3(myRigidbody.velocity.x, jumpSpeed, 0f);
jumpSound.Play();
}

//
if(Input.GetKeyDown(KeyCode.V))
{
    myAnim.SetTrigger("throw");
    Fire();
}
```

3.3.5 PauseScreen.cs
The pause screen is activated by pressing the escape key on the keyboard. The pause screen was developed using the unity UI tools to display a menu when the key is pressed. The pause screen is inactive for the duration of the game until the player pressed the key. Then the time scale of the game is set to zero so that all the game objects in the game freeze in position. The pause menu offers three buttons. New Game, Resume and Quit. New game will reset the player and game objects back to the beginning. Resume will fix the time scale back to 1, hide the pauses screen so the player can continue while quit will close the application completely. Below is a snippet from the script.



```
public void PauseGame()
{
        Time.timeScale = 0;

        thePauseScreen.SetActive(true);
        thePlayer.canMove = false;
        theLevelManager.levelMusic.Pause();

}

public void ReturnToStart()
{
    SceneManager.LoadScene("Level1");
    Time.timeScale = 1f;

}

public void ResumeGame()
{

    thePauseScreen.SetActive(false);

    Time.timeScale = 1f;
    thePlayer.canMove = true;
    theLevelManager.levelMusic.Play();

}
```

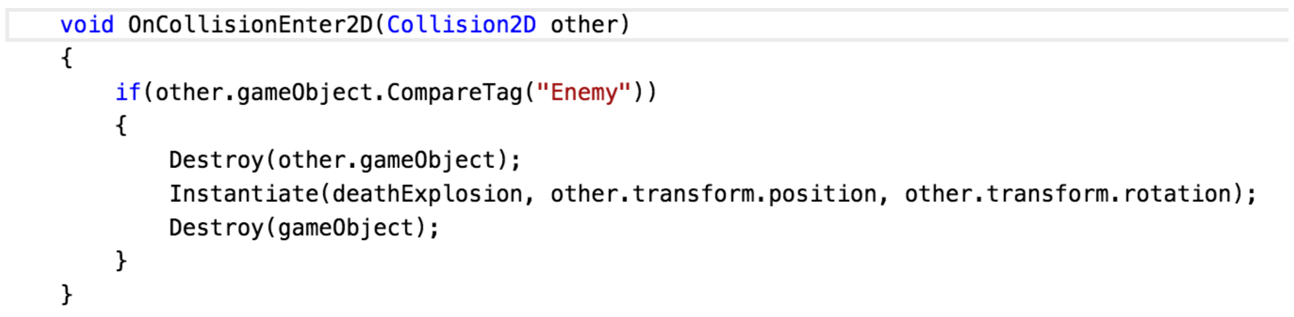
3.3.6 CameraController.cs

The camera controller was set up so I would be able to customize the feel of the game like I had done with the parallax effect. A simple way of getting the camera to follow the player is to make the player a child of the main camera. However this would make the camera follow the player when the player jumped as well as center the player in the screen. For the purpose of this game I wanted to have the camera move in a linear straight path along the Y axis and have a smooth transitioning from when the player changed direction while also having the camera look further along the axis so the player has enough time to react to obstacles or hazards. The way which this script works is by setting a target for the camera to follow i.e. player, while setting a follow ahead which alters the cameras positioning and smoothing value which sets the speed of the camera returning to its look ahead state.

```
public class CameraController : MonoBehaviour {

    public GameObject target;
    public float followAhead;
    private Vector3 targetPosition;
    //camera flip smoothness
    public float smoothing;

    public bool followTarget;



    // Use this for initialization
    void Start () {
        followTarget = true;
    }

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

        if(followTarget)
        {

            targetPosition = new Vector3(target.transform.position.x, transform.position.y, transform.position.z);
            //moves target of the cam ahead of player in both directions
            if(target.transform.localScale.x > 0f)
            {
            targetPosition = new Vector3(targetPosition.x + followAhead, targetPosition.y, targetPosition.z);
            } else {
            targetPosition = new Vector3(targetPosition.x - followAhead, targetPosition.y, targetPosition.z);
            }
            //transform.position = targetPosition;

            transform.position = Vector3.Lerp(transform.position, targetPosition, smoothing * Time.deltaTime);
        }



    }
}
```

3.3.7 boomCtrl.cs

This script controls the boomerangs that the player throws. The boomerangs are fired from the players position from the Player Controller script. The below snippet is the code showing the interaction between the boomerang and game objects with the enemy tag. Once the objects collide the objects are destroyed while also calling a particle effect to that position to indicate that the enemy was killed.



```
void OnCollisionEnter2D(Collision2D other)
{
    if(other.gameObject.CompareTag("Enemy"))
    {
        Destroy(other.gameObject);
        Instantiate(deathExplosion, other.transform.position, other.transform.rotation);
        Destroy(gameObject);
    }
}
```

3.4 Testing

3.4.1 Unit Testing

Unity has its own built in debugger and console which can be used to monitor functions and information. The debug mode can be activated on the inspector panel where the developer can view the specific information related to the game object or script. This is extremely useful in order to check when variables have been modified. For example, when Booleans have changed or float values have been modified. This ultimately is to ensure the scene runs as expected. This type of testing was carried out when new game objects or scripts were added to into the game. If a compiling error occurred after scripting the error would be noted immediately in the console where it also pointed to the location of the error so that it may be inspected and resolved. This type of testing was extremely useful during the development of the game as errors got resolved very quickly, subsequently preventing a build up of bugs in the system that would later become a more serious issue in the build.

3.4.2 Assertion Testing

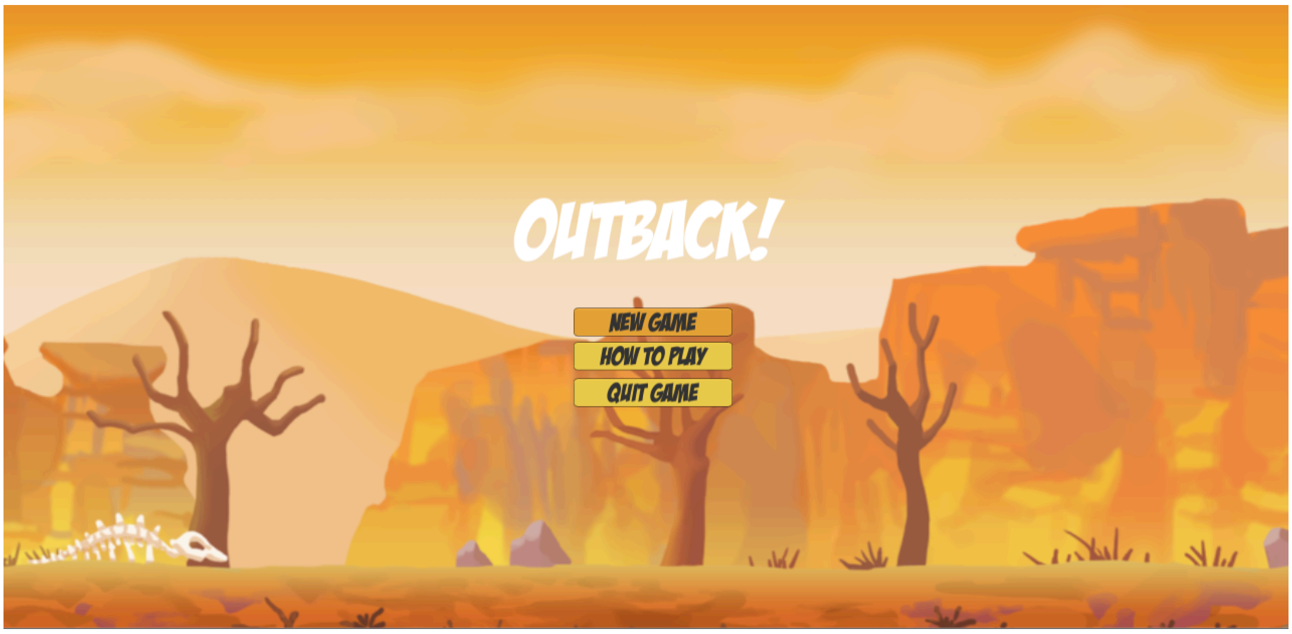
Unity provides the user with an asset store where users can browse a vast catalog of assets, tools, and graphics for their game. One useful package is Unity Test Tools. This package provides a more specific type of testing that can be carried out without having to write your own scripts. This type of testing was conducted later on in the development stages of this project. Assertion testing is used to indicate to the developer of an error. An example of how assertion testing was used in the project monitoring the player's health during a collision with another object. Once the assertion script is attached to the game object we can chose what type of comparison we want to perform. In my case it was a int for the health value. I then set up the parameters of the comparison by stating that if the player's health goes is greater than a value of six to perform the check. I then indicated that I want to compare the value whenever an object enters the trigger. If nothing happened it would indicate or assert that this is true until the assertion failed where it would then highlight the failure in the console. In conjunction with this assertion test you can also chose to pause the scene whenever an error occurs. Assertion tests are a very useful way in identifying subtle or minor errors in the game that may go unnoticed to the developer and were useful in comparing values and checking Boolean conditions during the course of my development.

3.4.3 Integration Testing

Integration testing tests how objects interact with each other, how scenes are integrated together and how game objects are behaving in the world. Integrating testing was carried out on specific game objects ensure that they operated correctly with the player. A simple integration test was carried out with one of the layers of ground the player runs along. This layer was unique to other ground levels as the purpose was to slow the player down and created a drag effect. This simple test was carried out by setting up parameters on when to start and end the test. The script was also provided by the unity test tools package. The time values are entered into the inspector panel where the test is created and the scene can run. If the character exceeds the value of time set in the inspector panel to move along the specific layer of ground the test fails. Integrating testing mainly used to ensure the game objects in the scene carried out their specific function when they entered triggers or collisions.

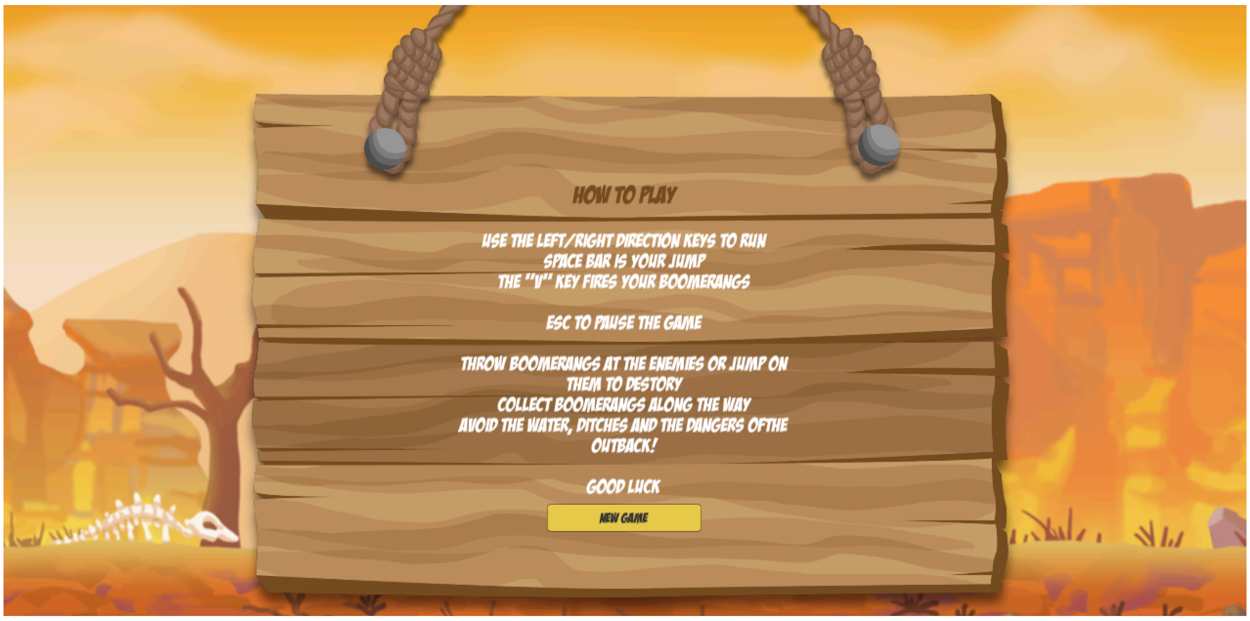
3.5 Graphical User Interface (GUI) Layout

3.5.1 Main Menu



The main menu gives the player three options to begin interacting with the game. The “New Game” button will load the level where the player can begin to play the game. The “How to play” button was a later addition to the menu. This loads a separate scene where instructions on how to play the game and the objectives are briefly outlined. The “quit game’ button simply exits the system. The design is minimalistic to appeal to the target audience I had in mind when beginning this project. Buttons are clearly labeled and displayed in the center of the screen. I have also included a highlighted selection colour on buttons that are being pointed at by either the mouse or selected with the vertical directional keys on the keyboard. This is just to give the user a better indication of which button is being selected.

3.5.2 How To Play (Main Menu)


HOW TO PLAY

USE THE LEFT/RIGHT DIRECTION KEYS TO RUN
SPACE BAR IS YOUR JUMP
THE "V" KEY FIRES YOUR BOOMERANGS

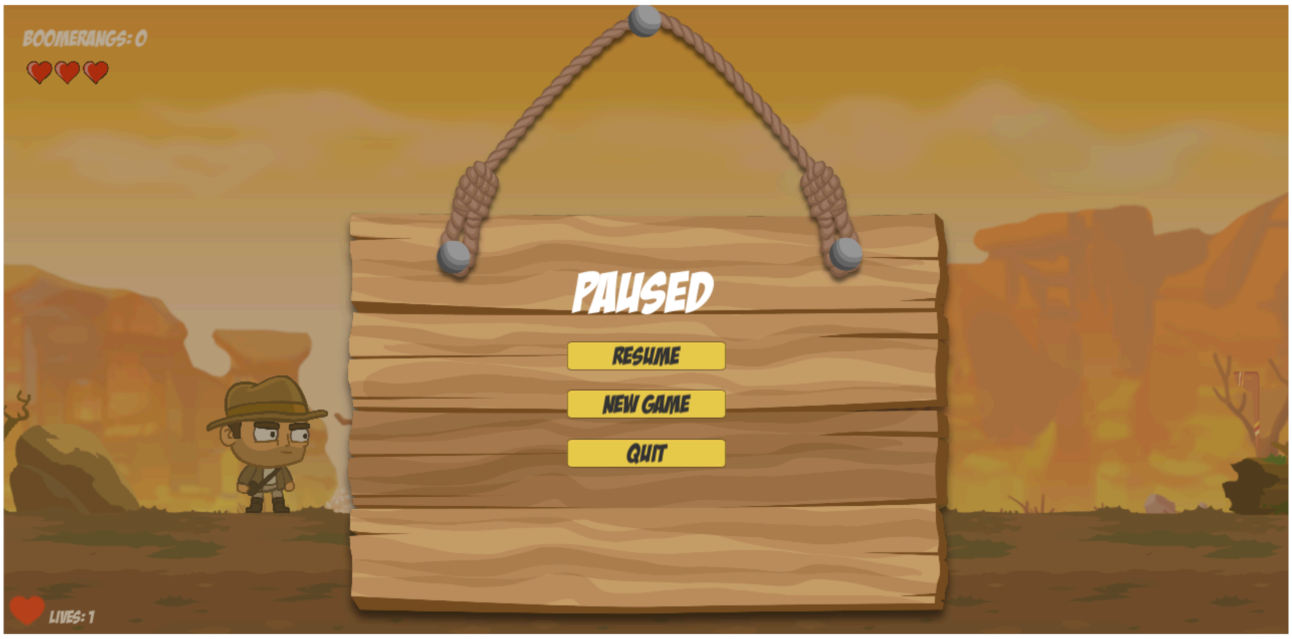
ESC TO PAUSE THE GAME

THROW BOOMERANGS AT THE ENEMIES OR JUMP ON THEM TO DESTORY
COLLECT BOOMERANGS ALONG THE WAY
AVOID THE WATER, DITCHES AND THE DANGERS OFTHE OUTBACK!

GOOD LUCK

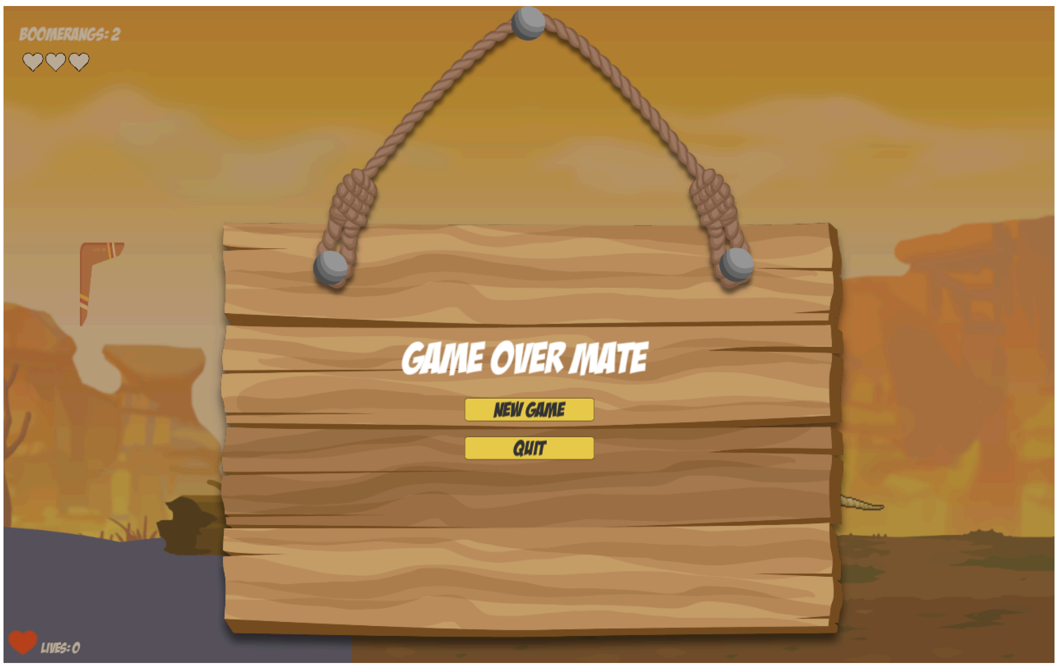
The How to play scene is displayed as a simple page with a brief set of instructions and information about the game. Once the user feels comfortable with the objectives and controls they then have the option of selecting new game to load the level instead of having to return to the main menu and selecting the same option.

3.5.3 Pause Screen



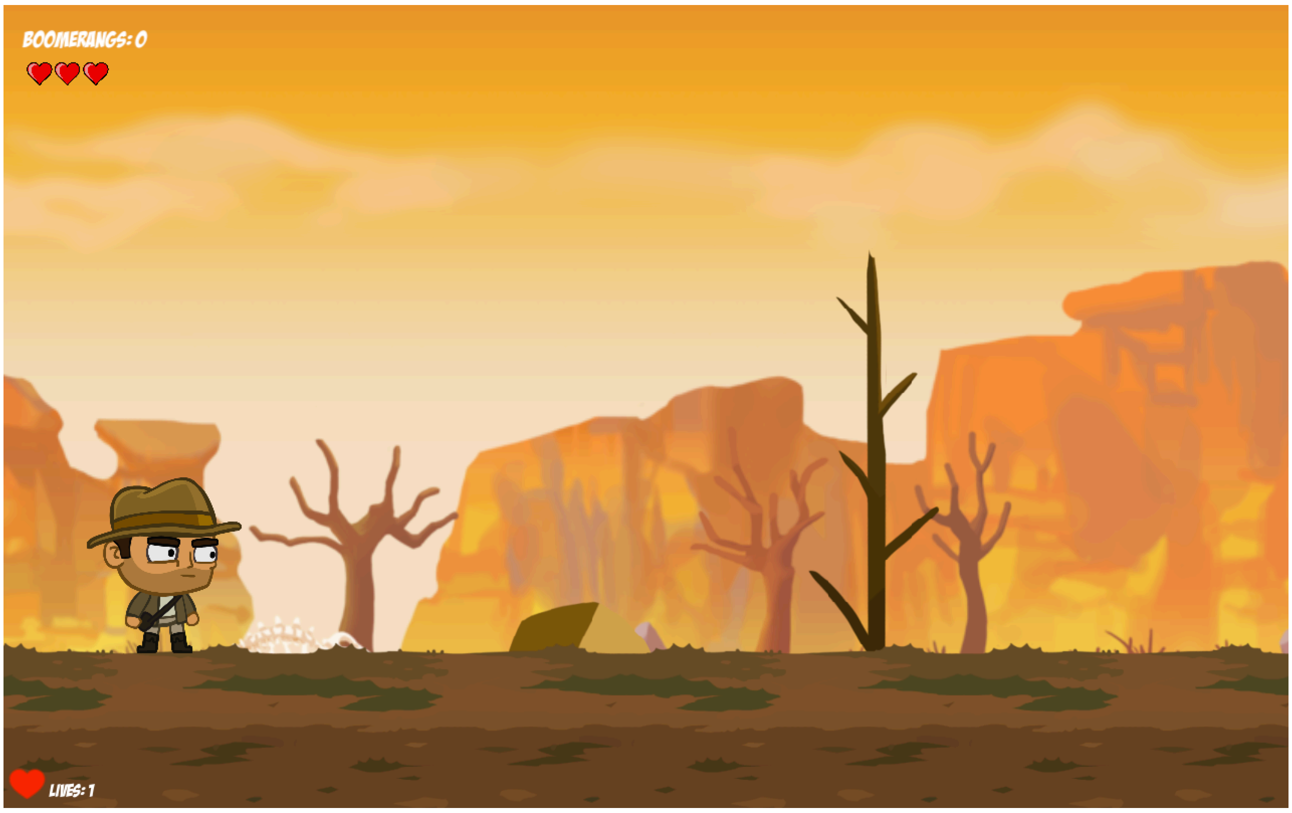
The pause screen is activated once the user presses the escape key (Esc) on their keyboard. The screen then freezes the game time while dimming the background to indicate that the background is no longer active. The board that is displayed in the center gives the user three options. The “Resume” button to go return the game to continue their progress. The “New Game” button is selected to restart the level and the “Quit” option to exit the game.

3.5.4 Game Over Screen



The game over screen is activated when the player has run out of lives in the level and therefore cannot continue any further. This screen is designed to have the same theme as the pause screen to keep with the minimalistic and neat feel to the GUI. This screen indicates that the game as ended for the player and gives the user two options to choose from. By selecting “New Game” the user can restart the level or selecting “Quit” to exit the system.

3.5.5 Game Play Screen



This screen displays how the screen will be displayed to the user once they begin the game. Prior knowledge of the instructions and objectives are needed however they are provided in the main menu as mentioned above. The screen displays the number of collectible items the player has in the top left corner. Below are three hearts. Each heart has two lives. When a player loses a life the heart breaks to indicate some health has been lost. Below in the bottom left corner the number of lives the player has is displayed. When the number of lives reaches zero. The Game Over screen will be activated. This display also shows the character and the ground. The ground that the character runs along is displayed in a much darker colour than the background items, this is to highlight that it is the ground in which the player can move along during the level.

3.6 Customer Testing

During the development of the game extensive user testing took place. This was implemented to receive feedback on numerous aspects of the game, which subsequently led to a number of changes in the project. The testing was done with the build created and hosted on AWS. Since I had created this build for the project to run online I decided to make use of customer testing by sending the URL to my testers. The testers of the game were made up of people who had experience in playing games, moderate experience and little to no experience with games. As well as people who has programming knowledge or no programming experience at all. This was to ensure that the game was playable to a broad audience and not just a specific group that would act as my target audience. I found this experience to be extremely useful during the course of my development and had implemented this testing throughout all beta builds to ensure that I meeting the customer demands. Some of the ideas from the feedback I have listed below.

Version 1

This build was extremely basic. This build was used for testing functionality for the later release. The build consisted of pixel art, moving the character along platforms and jumping to reach different platforms. Feedback consisted of customers noting that it was simplistic, no music, no objectives and limited functionality. All this however was expected with the first build.

Version 2

This build adding moving platforms for the character to jump along and move around. There was also a scoring system that added points to the screen for every moment the player was able to survive. Feedback consisted of customers noting it was more enjoyable, include more player capabilities and better design.

Version 3

This build was closer to the finished product. The design was heavily modified to give a more modern feel. The player could now run and jump on enemies to destroy them. The build also had a health system in place. Feedback consisted of customers requesting a “gun” or firing option, more enemies, obstacles, longer levels. I had also noted to include an instructions screen as I realised I had to explain the objectives and controls to the users.

3.7 Evaluation

Throughout the development of this game I had consistently evaluated the system by testing, checking the console for errors every time I had implemented a new script or game object into the world. This was done to ensure that all the functions worked alongside each other and carried out their specific tasks in the world. Unity’s powerful testing tools along with the console and inspector were all used to ensure that the game ran how I wanted it to. Customer testing was also pivotal to the progression and development of this game. In doing so I was able to also witness how the various groups used the system and what they expected or did not want within a game. Their feedback provided me with clearer objectives in mind for the game and I am very pleased with their feedback as all testers said the game was very enjoyable to play and would recommend it to others.

4. Conclusions

There are many advantages when creating this type of project. The primary reason I decided to use Unity was because of the immense learning support and community that exists around the software. I had no experience with using Unity or playing games before this project. Unity really allowed me to develop and create an idea which I had visualized at the beginning. For me choosing a game as my final year project was down to creativity. I myself am not a gamer and had little knowledge about games and their development process. However I wanted to create something fun and enjoyable for my project and have complete creative control over it. Using the knowledge the college has provided me with over the past few years allowed me to do just that. This project allowed me to explore the world of games and expand my knowledge on how they are created. The experience has left me overwhelmed with the amount capabilities that are available to game developers out there. I now feel that I was naive in my expectations of game development before beginning this. A lot of my game has changed over the course of the past few months because I had not acknowledged the difficulties, challenges and time that it takes when creating a game from nothing. It was important to myself that the style and feel of the game were how I wanted it to be from the beginning. As I wanted a game that was visually appealing and enjoyable to play so anyone of any age or skill can start the game and enjoy it.

The challenges that presented itself with this project were difficult to over come. Becoming familiar with the Unity interface took some practice and getting used to. At first glance the software looks intimidating to start with. This took some time to feel comfortable with for the first half of the development stage. Another issue was with C#. I had little knowledge of the language as I had only touched on the language in various modules within the college. However ultimately I found it to be a great opportunity to expand on my knowledge and found it to be extremely useful and powerful programming language that worked well with Unity. One of the biggest challenges during the development stages was to do with art work. Finding the art that would blend well in the world I had created. This was partly because in using art in animations I needed to have various frames to display the animation correctly. Graphic design is a passion of mine and I was able to edit some of the art work myself however it was a lot more time consuming and frustrating than I had initially expected.

Ultimately I have thoroughly enjoyed developing this project. I now feel I have a greater appreciation for game developers, animators and game programmers than I had before. There have been a few ups and downs but overall I have achieved what I had set out in the beginning. I'm also extremely proud of the feedback that came from the users who have played the game during the development stages. Through the course of this project I have had the great opportunity to expand my knowledge and skill of various aspects of the computing industry and ultimately achieve the goals which I had set in the beginning.

5. Further development and research

There are a variety of ways in which this project can be expanded. The first that I would like is to provide a mobile format that would be present in the app stores for IOS, Google Play and Android. I feel that this game would be well suited to smart devices, as it is enjoyable and simple to play especially on a portal device.

For gameplay one potential idea I would like to achieve is to have the players health slowly deteriorate as the player progressed in the level. This would create a sense of urgency in the game for the player to survive. This could be implemented by creating the effect that the character is dehydrated in the outback as he progresses and as the health begins to deteriorate the player must collect water along the way in order to boost their time in the level. Another feature I would like to implement is more levels. More levels creates he opportunity to expose the player to new challenges and experiences. The primary level is set in the Outback however if the player completed the level they would more to a second level featuring a different environment like a jungle for example. This level would have different obstacles and hazards for the player to avoid ultimately survive. This method could be used to any other harsh environments in the world i.e. Sahara, Antarctica etc.

I would also like to feature a boomerang and destroy enemy system. In level 1 the player collect boomerangs along the course of the level however in future builds the player might have a limited number of boomerangs to throw and have to collect more to throw along the way. A scoring system could also be used to count how many enemies the player has destroyed and offer rewards to level completion or enemies destroyed. Rewards could be in the form as different weapons, level access, or even player skills. For example, having the option to create a higher jump but would ultimately cost the player some speed or life.

Overall I feel there is vast range of capabilities for this game, which could subsequently create new opportunities and expand the target audiences. This could lead to having the game being purchased through the app stores with in store add ons as well.

6. References

Webpages:

(cumulative), N. (2017). *Number of apps from the iTunes App Store 2008-2016 | Statistic*. [online] Statista. Available at: https://www.statista.com/statistics/268251/number-of-apps-in-the-itunes-app-store-since-2008/ [Accessed 1 May 2017].

Unity. (2017). *Unity - Fast Facts*. [online] Available at: https://unity3d.com/public-relations [Accessed 1 May 2017].

Unity. (2017). *Unity - Learn - Modules*. [online] Available at: https://unity3d.com/learn/tutorials [Accessed 1 May 2017].

Youtube tutorials:

https://www.youtube.com/watch?v=UbPiCgCkHTE

https://www.youtube.com/watch?v=86Bgt--Ww7w

7. Appendix

7.1 Project Proposal

BSc (Hones) in Computing
Specialisation: Gaming and Multimedia

Name:	Dylan Walker
Student Number:	x12328836
Email:	x12328836@student.ncirl.ie

7.1.1 Objectives:
My objective is to develop a 2D game. The game will be an action based survival game. I feel that a good game story is pivotal to a well-constructed game and I want to be able to incorporate that into my project. The story I wish to implement is player that is actively trying to avoid both environmental obstructions and health hazard in order to complete the level.

To develop this sufficiently I will be using Unity & Visual Studio code and building with C#. My personal aims in undertaking this project are to explore new technologies, research game structures and stories. To be able to efficiently implement characters and objectives that I will create into a game story and allow users to interact with that environment.

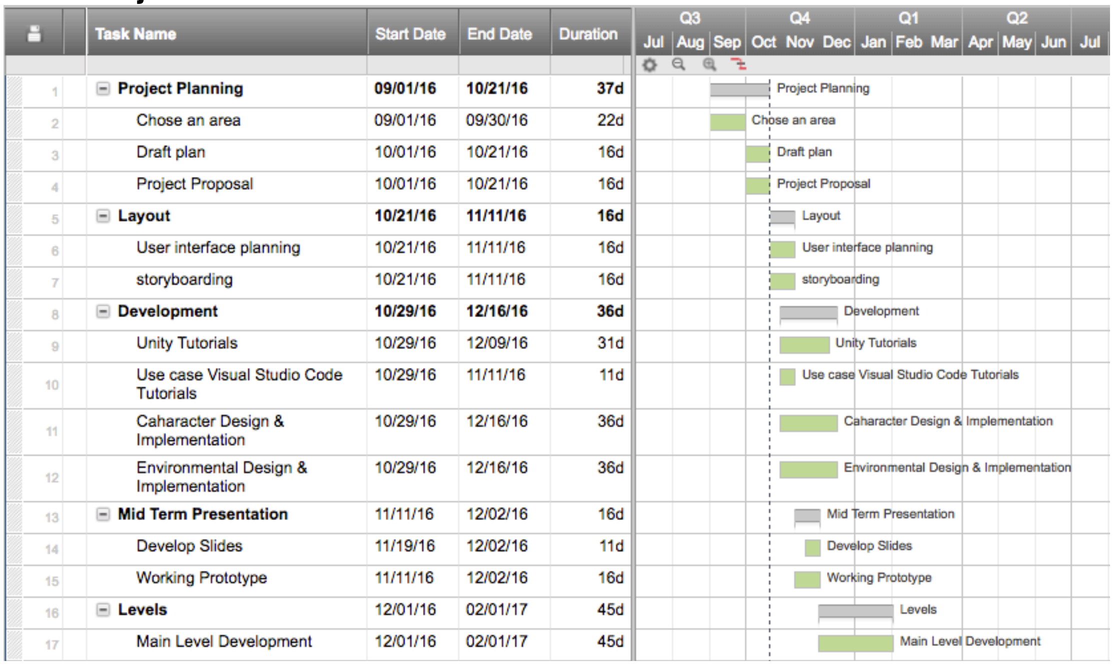
7.1.2 Background:
The gaming industry is one of the largest industries in the world. There are a wide range of game categories as well as many different platforms and consoles to engage with. Throughout the decades gaming has developed at an exponential rate. The golden ages of the 1980s which brought us arcade style games, and Nintendo as well as the probably the most famous game of that decade Pac-Man. The 90's brought us huge advancements in technology and subsequently the gaming industry thrived. 3D graphics technology was introduced and consoles such as the Sony PlayStation were the stand out consoles of the decade. Since then after the introduction of the smart phone many third party developers began emerging and of the indie games that surfaced became extremely popular. The introduction of the smartphone market for games has dominated the revenue for the gaming industry the past couple of years. This is due to the fact that consoles are dedicated gaming systems whereas your smart phone is almost like a portable gaming console that is likely on your person at all times. Games are now widely available in any major mobile marketplace such as the Android Store, IOS App Store and Google PlayStore.
I chose to develop a game because my interest in the industry. Since starting in NCI I had hoped to pursue the Gaming and Multimedia specialisation as I feel that developing a project based in an industry I have an interest in will allow me to perform to the best of my abilities. The project itself will be a challenging one, as I've no prior experience in developing a game nor have I had any experience in the software that I will be using.

7.1.3 Technical Approach
In order to complete this project I will be using Unity and Visual Studio Code. Unity is a game engine designed to be able to develop games for consoles, mobile devices and websites and is popular among develops due to its vast capabilities. Also alongside Unity I

will be using Visual Studio Code as the main code editor for C#. For the game design and images I will mostly be working with Photoshop. A lot of research will be required for using unity as its capabilities seem quite overwhelming, therefore I will dedicate a lot of time for practice in order to get used to the software. Currently this week I will be using unity in my Computer graphics design and animation module as well which will be a great introduction to the software.

7.1.4 Special Resources Required
Unity have quite a good catalogue of tutorials available on their website as well as many tutorial videos available for both unity and visual studio code available on YouTube.

7.1.5 Project Plan



	Task Name	Start Date	End Date	Duration
1	**Project Planning**	**09/01/16**	**10/21/16**	**37d**
2	Chose an area	09/01/16	09/30/16	22d
3	Draft plan	10/01/16	10/21/16	16d
4	Project Proposal	10/01/16	10/21/16	16d
5	**Layout**	**10/21/16**	**11/11/16**	**16d**
6	User interface planning	10/21/16	11/11/16	16d
7	storyboarding	10/21/16	11/11/16	16d
8	**Development**	**10/29/16**	**12/16/16**	**36d**
9	Unity Tutorials	10/29/16	12/09/16	31d
10	Use case Visual Studio Code Tutorials	10/29/16	11/11/16	11d
11	Caharacter Design & Implementation	10/29/16	12/16/16	36d
12	Environmental Design & Implementation	10/29/16	12/16/16	36d
13	**Mid Term Presentation**	11/11/16	12/02/16	16d
14	Develop Slides	11/19/16	12/02/16	11d
15	Working Prototype	11/11/16	12/02/16	16d
16	**Levels**	12/01/16	02/01/17	45d
17	Main Level Development	12/01/16	02/01/17	45d

7.1.6 Technical Details
The project will be specifically developed within unity using Microsoft visual studio code and the implementation language being C#

7.1.7 System
My system will be evaluated on the functionality of the main objectives of the game. The functions of the main player, the environmental obstructions and health hazards being the primary focus as they are the most important tasks in order for the game to function as a game. Testing I believe will be relatively easy as I visual representation to go by and I will be able to actively test during my progression in development. I intend to get opinions of those with a strong interest in gaming in order to build my project to a standard where a user can enjoy their interaction with the game.

7.2 Requirements Specification

7.2.1 Introduction

7.2.2 Purpose
The purpose of this document is to set out the requirements for the development of a game made with Unity. The game will be a 2D platformer styled game. This style of game is intended to be developed with the intent of making the game simple but fun echoing the approach of the popular 2D mobile games that have emerged over the past few years. The objective of the game is that a player will successfully get their character to the end of the level, whilst encountering many obstacles and hazards along the way. Furthermore, I want to involved a timer, that will limit the characters health as the player progresses through the scene. This is to keep with the story of the game where the character is trying to survive the harsh environment and the longer they are in the level the more their health is at risk. I feel this will make the game very interactive and fun, as the player will experience pressure in trying to reach the end level as quickly as possible. I also wish to incorporate a day/night cycle that will determine the obstacles and hazards the character will face throughout the scene.
The intended customers are fans of arcade style games, or mobile games. A game that isn't solely dedicated to those who want to spend hours playing but appeal to anyone who wishes to do play something fun and interactive and challenging.

7.2.3 Project Scope
The scope of the project is to develop a game with Unity and built with C# while using applications such as Photoshop and illustrator to make the game scenes and player attributes.
The requirements of this concept as follows:

- New Game Requirement
- Save Game Requirement
- Continue Game Requirement
- Character Progression Requirement
- Quit Game requirement

Schedule: The following game will be made of two terms of a college year, consisting of roughly 9 months in total. The phases will be broken down and to complete tasks and milestones along the way up until the completion of the finished product.
Costs: The game will be free to make and play
Software engineering environments used to develop requirements: The requirements diagrams will be made using Creatly.com, an online tool allowing users to create diagrams.

7.2.4 Definitions, Acronyms, and Abbreviations
Platformer: A game type which allows characters to move freely amongst platforms which act as the gaming environment.

NPC: non-player character, which refers to any character that is in the game that is not directly controlled by the user or player but interacts with them player within the environment.

7.2.5 User Requirements Definition

The objective of the game is for the player to make it to the end of the level alive. During the level health will slowly decrease as they progress into the scene. There will also be hazards and obstacles that are part of the level, which will either attack or obstruct the player from trying to reach the end. The player will obtain and equip certain items that may help their survival in the environment, which will benefit them to reach the end of the scene.

7.2.6 Requirements Specification

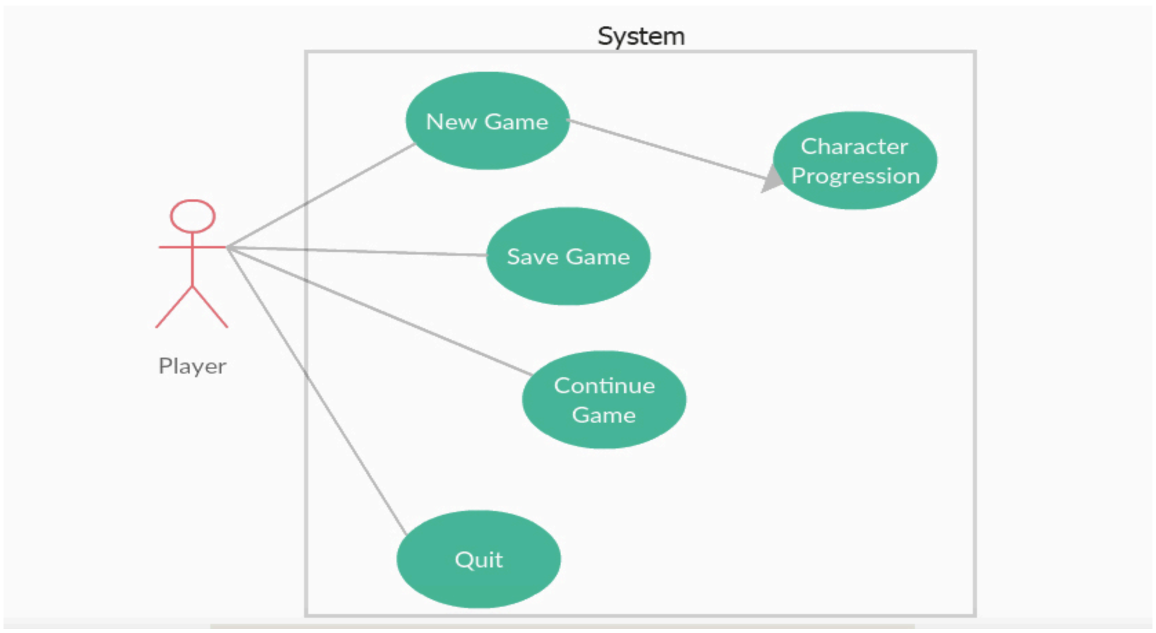
The game will require no practice or tutorials. The basic controls will be prompted to the user before the level begins. The user will also be provided with a background of the level and what is to be expected in the forthcoming scene before proceeding. The user should be able to start a new came with little delay. The user should be able to save their progression once a milestone has been reached or they manually select if from the menu screen. The user should be able to load a previous saved file and continue from a previous saved point. The user should be able to quit the game once they select it from the menu screen. The user should be able to collect and use items collect in order to aid them in their survival of the level.

7.2.7 Functional requirements

The functional requirements are pivotal in defining the system. Any input to or output from the system that ultimately changes the systems behavior. The following points below are a list of the functional requirements ranked in order.

1. New Game
2. Character Progression
3. Save Game
4. Continue Game
5. Quit Game

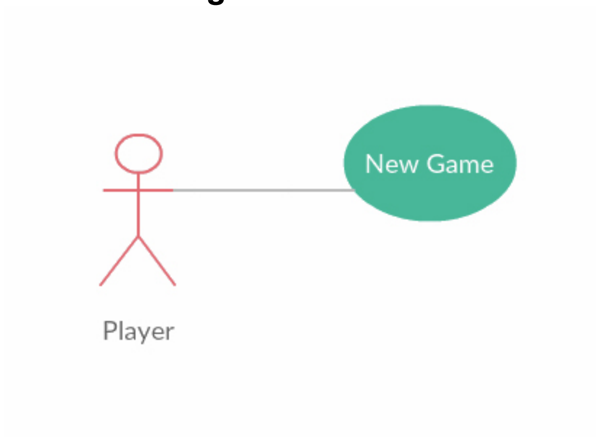
Use Case Diagram



Requirement 1 <New Game>
This requirement starts the initial process of allowing the player to participate in playing the game. They can do so by selecting New Game from the menu screen displayed in the user interface. Players will select this option in order to start the game for the first time or start all over again from the beginning if they select the option from the menu screen during a previous game.

Scope
The scope of this use case is to allow the user to engage in a new game.
Description
This use case describes the process that allows the player to start a new game.

Use Case Diagram



Flow Description
Precondition
The system is in initialization mode.

Activation
The use case starts when the user selects new game from the menu.
Main flow

1. The system identifies the player.
2. The player selects New Game
3. The system creates a save file for the game.
4. The system loads up the initial level.

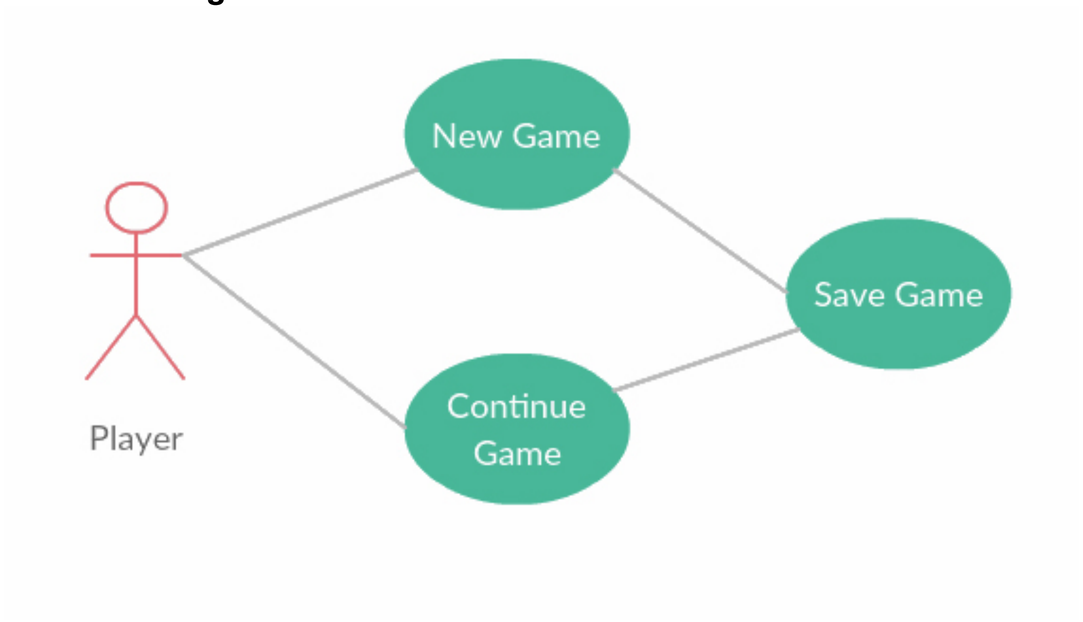
Alternate flow
A1: <Unexpected error>

1. The system identifies the player
2. The player selects New Game
3. The system creates a save file for the game.
4. The system prompts player for an Unexpected error,

Termination
The system presents the level for the player.
Post condition
The system goes into a wait state.

Requirement 2 <Save Game>
Description & Priority
This requirement allows the player to save their progress in the game. This is an important feature as it allows the player to continue on from their previous time invested in the game and encourages them to complete the level from a previous save point.

Scope
The scope of the use case is to allow the player to save the game and/or progress.
Description
This use case describes the process by which allows the user to save the game.
Use Case Diagram



Flow Description
Precondition
The system is in initialization mode.
Activation
The use case starts when a player brings up the menu and selects save game from the options.
Main flow

1. The player pauses the game.
2. The system shows the menu screen.
3. The player selects Save Game from the options displayed.
4. The system stores the game data up to that point.

Alternate flow
A1: <Completion of level>

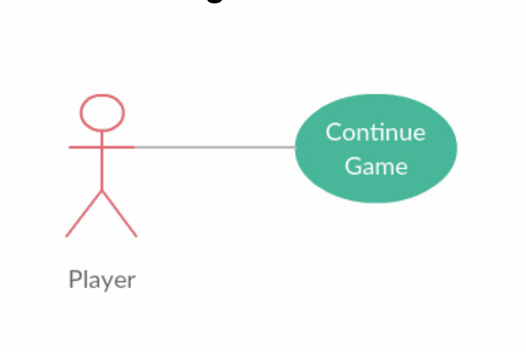
5. The player completes level.
6. The system identifies the player at a save point.
7. The system stores the game data up to that point.

Termination
The system presents the previous scene to the user.
Post condition

The system goes into a wait state.

Requirement 3 <Continue Game>
The player should be able to continue their previous saved progress by selecting to do so on the initial menu screen. This requirement is pivotal for the player as it allows the to return to the game with their progress from a previous session.

Scope
The scope of this use cases it to allow the player to continue their game.
Description
This use case describes the process by which a player continues a game from their previous saved progress, which is stored by the system.
Use Case Diagram



Flow Description
Precondition
The player has played the game before and has made a save file of their progress.
Activation
The use case starts when the system has stored the saved game data.
Main flow

1. The system shows the menu screen at the start.
2. The player selects Continue Game from the options.
3. The system locates the save file.
4. The system loads the file and spawns the character to their previous location within the level.

Alternate flow
A1: <Return to start>

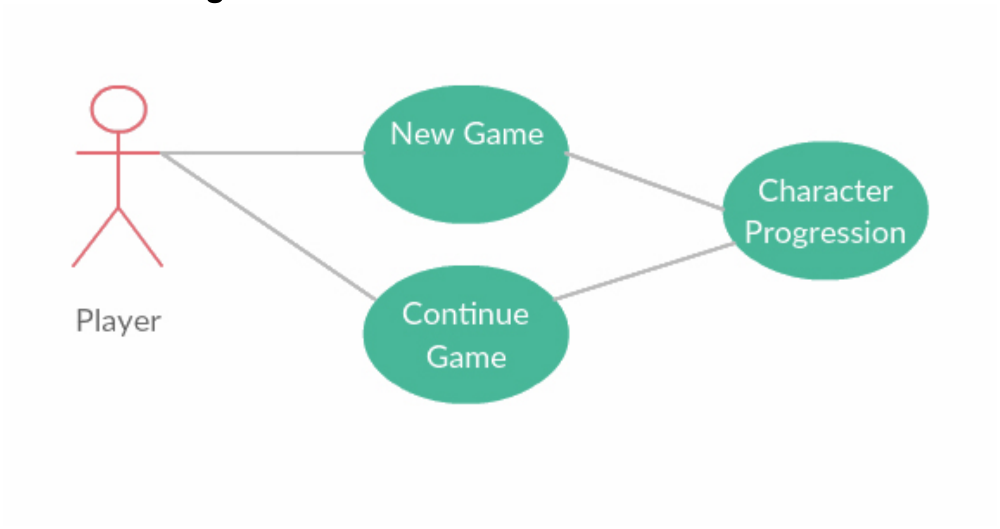
1. The system shows the menu screen at the start.
2. The player selects Continue Game from the options.
3. The system cannot locate any previous saved file.
4. The system prompts user of unexpected error.
5. The system loads a new game for the player.

Termination
The system presents the requested file to the user.
Post condition
The system goes into a wait state.

Requirement 4 <Character Progression>
During the progression of the level the player will be able to collect experience points which will have an immediate effect in the characters health and abilities within the game. The is important as it encourages the player to make use of these features in order to successfully complete the level.

Scope
The scope of this use case is to allow the player to increase the potential of their character.
Description
This use case describes the process by which a player can gain points.

Use Case Diagram



Flow Description
Precondition
The system is in initialization mode.
Activation
This use case starts when the player starts a new game or continues a previous game.
Main flow

1. The player collects items whilst progressing through the level.
2. The system rewards the player by increasing certain character abilities

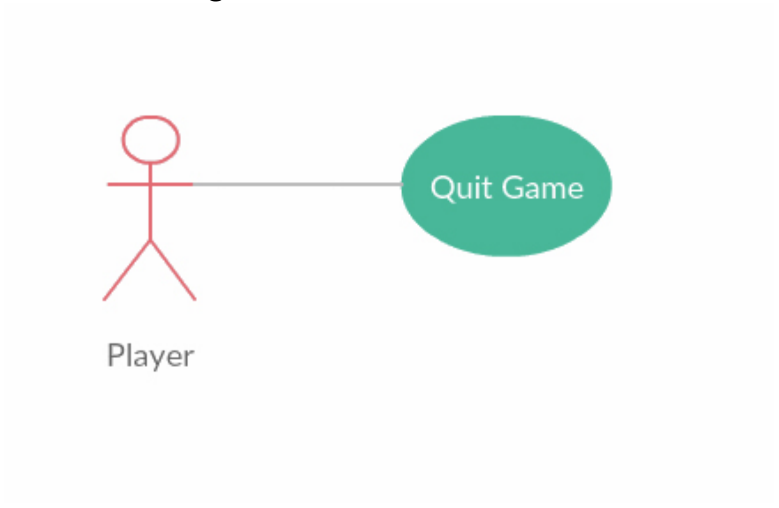
Termination
The system presents the requested file to the user.
Post condition
The system goes into a wait state.

Requirement 5 <Quit Game>
The player may quit the game at any point. However the system must prompt the user before the action is complete to warn that any unsaved progress will be lost. This requirement is important for the player to end a session.

Scope
The scope of this use case is to quit the game.
Description

The use case describes the process that is involved when the player opts to quit the game.

Use Case Diagram



Flow Description

Precondition

The system is in initialization mode.

Activation

This use case starts when the player opts to end the session.

Main flow

1. The player pauses the game.
2. The system displays the menu screen
3. The player selects Quit Game from the options.
4. The system prompts user about losing unsaved progress.
5. The player acknowledges the warning by selecting OK.
6. The system exits the game.

Termination

The system exits the game.

Post condition

The system goes into a wait state.

Non-Functional Requirements

Performance/Response time requirement

The user will be able to select graphics that are appropriate to their machine to ensure the game is running optimally for their experience. However, The game will run at an average at 60 frames per second on most machines. The response time for the character within the game should be immediate. This ensures that when a player makes an action the character within the level responds appropriately and immediately to the players request.

Availability requirement

The game should be available to the user whenever they request.

Recover requirement

The system will save the players progress at certain points during the game. This ensures that if a player reaches a milestone within the level their progress is saved up until that point. The requirement is essential as in the event of a performance failure or crash the user can continue from their previous sessions progress. As well as this the user will have the option of saving the game manually during the level.

Robustness requirement

Developing in Unity allows the developer to view changes in the play-mode view, this will hopefully eliminate any bugs or major errors that might have occurred otherwise and acts as a continuous beta test during the development. However error detection will be programmed in to ensure that any errors will be caught and handled rather than crash the game.

Reliability requirement

The game will be available at all times once the user has downloaded it to their device.

Maintainability requirement

During the development stage on going tests will be carried out to ensure the game continually runs as it should. Maintenance will be carried when the game is finally finished to ensure any major issues will be addressed immediately.

Portability requirement

The user will be able to play the game at any time they desire once they have it installed on their device.

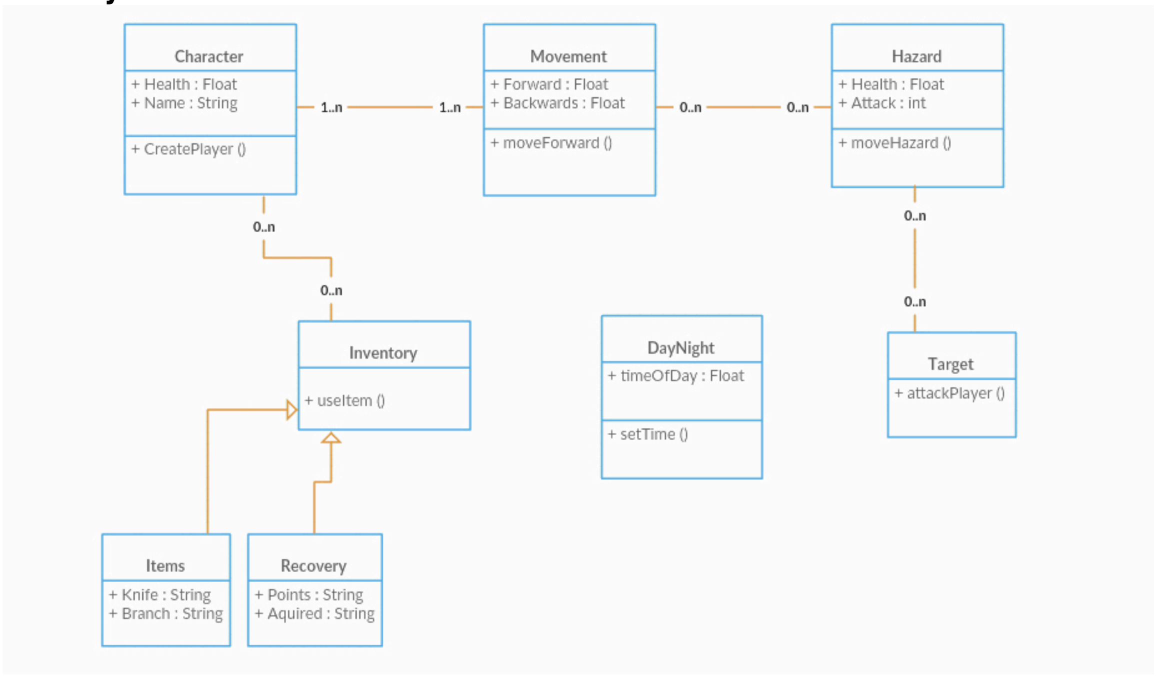
Extendibility requirement

The game may be updated with additional levels or character abilities over time to constantly improve on the game play and story.

Resource utilization requirement

The game should use a certain level of system resources to ensure that the game will run efficiently on the users system to ensure optimal playability.

7.2.8 System Architecture



Character class with the first that the player will meet upon reviewing the character and initialization of the game the next class that will be in operation will be the movement class. The movement class allows the player to move forward and backwards to progress through the level. As the player progresses through the environment they will encounter hazards, which will be made up of environmental aspects of in the form of animals, which may attack the player once they are targeted.
The player will have a brief inventory of weapons that will allow them to fight against the hazards and also have items that will benefit the characters health and time progression. Other classes will be added as the game development progresses further.

7.2.9 System Evolution
I feel the game has great potential to evolve over time. For example further levels can be added, which could bring new environments, hazards or obstacles to the scene. The players inventory could also be greatly expanded to compensate for more challenging environments or perhaps the character could develop new skills on the success competition of a level, which would subsequently be put in to practice for the next level. E.g. the character could level up to swing from vines, which could be appropriate to a level set an Amazonian style forest. As well as further character ability development and additional environments to play, I feel the game would be very appropriate on mobile devices given its format and style. I would also like to make it possible for players to be able to create their own character. Allowing them to customise their characters appearance and select character attributes and abilities that have an impact on their performance throughout the game for example, a character could potentially have a lot of strength but as a consequence not much speed and via versa.
Over all I feel there is a lot of room for expansion within this concept and I’m very pleased about the opportunities and ideas that I can add to help the game evolve over time.

7.3 Reflective Journals

7.3.1 <u>Reflective Journal September</u>

Student name:	Dylan Walker
Programme:	BSc in Computing
Month:	September

My Achievements
Throughout this month I have been extremely worried and stressed about the project pitch proposals, as my intention was to dive straight in to my research and development as soon as I could, as I know the amount of practice it's going to take in order to complete this successfully. My project however did get approved this week, which is more than a relief. I'm now free to plan this out and get started which I'm really looking forward to.

My Reflection
The project pitches were constantly in my thoughts throughout the month, as I did not know what to expect. However I am glad that they had remained in my thoughts actively as I believe I thought more about the construction of the game itself. I research game types and architectures and developed a project pitch that was accepted by the judges, which I'm very pleased with. Their feedback was brilliant and something that was pointed out to me was that I would need some unique element that will make the game stand out. Especially in the mid term presentations, which is were I will have to address that question.

Intended Changes
Next month I want to experience a lot of practice and development. I have never built a game before not used the software required. I really want to be able to know my capabilities in this project and know my weaknesses as soon as I can in order to develop the game efficiently. I also wasn't to be able to find a unique selling point in the game. Also I wish to come up with my unique selling point that the judges suggested. So I will have to do a bit of research on what elements I can add to this project in order to make it unique.

7.3.2 <u>Reflective Journal October</u>

Student name:	Dylan Walker
Programme:	BSc in Computing
Month:	October
Supervisor:	Francis Sheridan

My Achievements
This month I was able to focus on understanding and developing in Unity. We were able to do get the basics in college in my Graphics and multimedia labs and furthermore I watched a number of YouTube tutorials to practice with. Overall I'm very happy with my developments in Unity so far however much more work needs to be done in order to have a functional prototype for next month!

My Reflection
I've been slightly concerned about the timescale of the project. Initially I was optimistic about staying ahead of myself for this project, however due to a number of assignments and other coursework alongside the project I've had a growing concern over completing everything to what I have in mind. Reading week I hoped to have a great deal of work done however a lot of my focus was spent studying for exams. I understand now that I must be more prepared in dividing up my time.

Intended Changes
Next month I wish to have a proper prototype developed. So I can get feedback from my supervisor before the presentations. I intended to have at least one level made and ready to demonstrate. I also intended to make use of a calendar to stick to so I wont let myself sink in the sea of work that's to be accomplished in our final year. I'm looking forward to developing more especially after watching the capabilities of various games designed within unity.

7.3.3 Reflective Journal November

Student name:	Dylan Walker
Programme:	BSc in Computing
Month:	November
Supervisor:	Francis Sheridan

My Achievements
This month I was able to do the labs in my graphics module, which were all based in Unity. I felt this gave me a great foundation of practice for my project. After having a bit of practice I was able to develop a bit of the game for the demonstration coming up shortly. This was my man goal for the month. As I want to try and have a good presentation and get some feedback of my position.

My Reflection
I'm disappointed with my progress this month as I intended to have enough done to be able to go through more with the supervisor. The past few weeks were dominated by the projects in our other modules that I wasn't able to dedicate as much time to it as I would have liked. However, I'm still going to continue to get a little bit further before the presentation, so I can have a better demonstration. I still feel I'm making progress and if I'm able to get feedback of my position at the mi-term presentation I'll be very happy.

Intended Changes
I'm looking forward to the feedback from the mid point presentations. I want to ensure that I'm on the right track and also to see if there is anything more I can implement or what the final examiners will look for in the final project. Overall I think this will be a great milestone to reach in order to evaluate the goals I have set. Over the Christmas break I hope to try and be able to have a level complete with the timer function. This ensures I've a good baseline for the rest of the work to be done and also takes pressure off myself from the upcoming exams if I succeed!

7.3.4 Reflective Journal December

Student name:	Dylan Walker
Programme:	BSc in Computing
Month:	December
Supervisor:	Francis Sheridan

My Achievements
With the mid point presentations this month I was finally able to put together pieces of the project I have been practicing and combine them to have a demo working and ready. The demo itself was quite basic however it did show off some features which I will be using and altering for the finish product. I got a lot of practice with scripting in unity and I'm more confident in my ability to reach the end with a project I'm going to be happy with.

My Reflection
I feel this has been the best month so far, possibly due to the mid point presentation approaching I was motivated to demonstrate something I would be proud to walk in with. When all the individual scripts I had been working on were piece together for one demo I could really see how far I had come along since the start and it really put the end goal into perspective as I had something to show that will be a part of my final project.

Intended Changes
I've a lot more work to get done. A few of the game objectives I have decided to change, such as a timer which I will probably decided to drop from the final project. Also I've decided to bring together the platforms in the level to have a consistent ground rather than hoping from one to the other. I really want to have one full level complete in the next month so I can focus on design and completely another 2 or 3 levels for the end. All in all I've been happy with this month and I'm looking forward to seeing my progress in the next few weeks.

7.3.5 Reflective Journal January

Student name:	Dylan Walker
Programme:	BSc in Computing
Month:	January
Supervisor:	Francis Sheridan

My Achievements
This month I was researching how to put the game on a mobile platform. Initially I thought the process would be a lot more simple than I had imagined but I may have either rebuild the application using unity mobile development tools unless I figure out how to continue my work as is and ensure it can be displayed on a mobile device. I'm hoping that it will not take long as I still need to complete the level one to have a full level completed soon.

My Reflection
This month was a lot more difficult than I had initially expected but that is partly down to exams and the start of a new semester. Subsequently I did not get as much done as I

would have liked. There's still a good bit more to go, but I'm confident that it's piecing together nicely at the moment.

Intended Changes
By next month I will like to resolve the mobile issue. Whether that be a rebuild in unity or further development on the current project. Other than that I will have to push having a full level completed by the next month due to the difficulty of the past few weeks. However I still feel as though I'm on track and I'm happy with the progress so far.

7.3.6 <u>Reflective Journal February</u>

Student name:	Dylan Walker
Programme:	BSc in Computing
Month:	February
Supervisor:	Francis Sheridan

My Achievements
This month I was able to get back on track with my progress. I completed a base level for the game which I'm happy about so far. I do think that I may change some of the initial aspects of the game as implementation of certain features is providing me with some difficulty so far and I don't what to linger on a problem for too long and have issues later on. The only features I'm thinking about changing is the to have the character move by the use of the actual player rather than have it automated. Also to provide a character battle at the end of the level instead of just running to completion.

My Reflection
Over all I'm happy so far with the progress and more so happy with that changes. Game development is a lot more difficult than I expected as there are so many features to implement. I'm happy that I took time to reflect on where the game was going in order to make the amended changes. I feel this will give the game a more enjoyable experience to players and a great sense of achievement than having the character automated for the journey through the level.

Intended Changes
To implement the new features and begin work on the second level. I would also like to to see if a rebuild is required to get my game on android if possible as I did not investigate it as much as I would have liked this past month.

7.3.7 Reflective Journal March

Student name:	Dylan Walker
Programme:	BSc in Computing
Month:	March
Supervisor:	Francis Sheridan

My Achievements
This month I focused on new aspects that I had integrated into the game. As mentioned in my previous report some features were providing difficulty to implement and I was hesitant about implementing other features. So I took a step back and started trying to identify what really makes a game enjoyable. I believe that my new level will provide users with a much more satisfying experience and I'm very happy about the past 2 months progress.

My Reflection
Being hesitant with the format of my game was something I struggled with as I don't have much experience in playing games let alone designing games. But I'm extremely proud of the past couple of months progress as I feel it gives my game a much more rounded format.

Intended Changes
The only thing I really have left to do is create new levels for the players to engage with but due the success of my first I'm positive this will fall into place effortlessly. This is because the 2nd and 3rd levels will feature many of the same aspects as the first but with just a change in scenery and objects but the coding has already but done for the objects and I've created prefabs for quick re-use.