**Stuart Dunan (B00371539), Fraser McAulay (B00371477)**

Melvins Quest

Contents

[GAME DESIGN DOCUMENT 2](#_Toc25667578)

[Project Overview 2](#_Toc25667579)

[Story 3](#_Toc25667580)

[Introduction to game world 3](#_Toc25667581)

[Graphics 3](#_Toc25667582)

[Style 3](#_Toc25667583)

[UI Elements 3](#_Toc25667584)

[Level Design 3](#_Toc25667585)

[Mechanics and Gameplay 5](#_Toc25667586)

[Goals 5](#_Toc25667587)

[User Skills 5](#_Toc25667588)

[Game Mechanics 6](#_Toc25667589)

[Losing 7](#_Toc25667590)

[Winning 7](#_Toc25667591)

[Music and Sounds 9](#_Toc25667592)

[TECHNICAL DESIGN DOCUMENT 10](#_Toc25667593)

[User Interface 11](#_Toc25667594)

[Development Methodology 11](#_Toc25667595)

[QA bug testing 11](#_Toc25667596)

[Asset List 13](#_Toc25667597)

[Asset Production 14](#_Toc25667598)

[Game Flow 14](#_Toc25667599)

[Game Creation Process 16](#_Toc25667600)

[Source Control 18](#_Toc25667601)

# GAME DESIGN DOCUMENT

## Project Overview

This game design document or GDD is set out to describe how our ideas are represented within a game concept we have in mind to develop. The game will be a 2.5d platform/puzzle adventure with one main central playable character with a straightforward easy to understand story and end goal.

The game plays a lot like other platform games where the player has no attacks but instead relies on the player to make quick decisions to traverse each area completing the necessary goals to achieve the levels progression with unique puzzles and traps to fit the games genre and overall feel.

Names for the game so far before final decision are:

* Dungeon Diver
* Dungeon Quest
* A Goblin’s Tale
* Imprisoned Imp
* Melvin the Unlucky Goblin
* The Tale of Melvin
* Melvin’s Quest

### Story

### Introduction to game world

The player character is a goblin named Melvin who normally spends his days bothering adventurers and subsequently being beat up by said adventurers. On the day of the game’s events our character wakes up in a dank dungeon with a rather blunt message ringing in his head: “You will need 4 keys to escape.” Looking around his surroundings, he can see entrances to two distinct areas along with a large door, adorned with 4 padlocks. Following this, the story of the game will follow Melvin on his quest to escape the dungeon.

## Graphics

### Style

Our game will take on a semi-realistic 3D style with various realistic aspects such as realistic lighting and shadows. We will do our best to make sure all elements of our game are in keeping with this style so that nothing sticks out or looks awkward. Our player character will take on a similar style as well, looking somewhat realistic but also visually exaggerated to a point.

### UI Elements

For the user interface we’re going to accentuate key elements using vibrant colours and readable text to as to make the game as visually appealing as possible in that department. We’re aiming to emulate the common layout used by countless platformers, that being to display important information in the furthermost corners of the screen and to be in keeping with the game’s overall style.

### Level Design

The challenges seen in each of the distinct areas will make use of both linear and puzzle-based level progression. The linear areas will make use of standard 2D platforming challenges such as precise platforming sections that steadily increase in difficulty. As seen in the left section of *Figure 1*, the player is first presented with a simple challenge wherein they must navigate a set of platforms. In this case, the only punishment for failure is falling down to the beginning of the area and having to climb up again. This is further expanded upon in the following section where the player is required to complete another platforming challenge except this time the punishment for failure is death followed by the loss of a Life and their position being reset. Through this simple difficulty curve in the linear sections of the games levels we aim to let players get accustomed to the rules of the game while also learning what to expect.

Contrary to the linear structure of certain areas of our game, other areas will make use of a non-linear, puzzle-centric gameplay loop. In the right section of *Figure 1*, it can be observed that we focused on switch-based puzzles that unlock doors at different ends of the level.

A close up of a map

Description automatically generated

Figure 1 - Level Area map

## Mechanics and Gameplay

### Goals

The player playing this game in the long term their goal will be to collect Keys in each stage which will ultimately be added to a key count which will guide the player to the end of the game. While collecting these keys the player must attempt to keep at least 1 life in their life pool keeping the player alive to be able to keep playing and achieve their goal. In the short term the player will have to traverse various levels with different themes each having their own unique pitfalls traps and puzzles. These will range from spike traps to burning lava but also have certain puzzles to traverse including pushing a block to a pressure pad to open a door, these will all have to be overcome and all keys collected in each level for the player to proceed to the next level.

### User Skills

The player will have two modes of controlling the game character in the forms of either Joypad or by Keyboard. For this title the skill is in having easy understanding controls for the player with simplified movement and a single jump function where the player will have to use their skills of concentration and strategize their movement to guide the player through each stage using puzzle solving abilities.

Here are the simplified character controls mapped out showing how the player will move left and right and the corresponding input to make the player jump on the keyboard the P key is mapped for pause which will also be used to pause for players using controllers.

A close up of a keyboard

Description automatically generated

### Game Mechanics

The game will utilise simple platforming mechanics which many players may be used to having played other platformers. Additionally, the game will make creative use of button puzzles which work in conjunction with the aforementioned platforming mechanics. Additionally, there will be dangerous hazards to avoid such as swinging blades that move between and out of the foreground, emphasising the 2.5D nature of the game world.

#### Player lives and keys collected

The player as mentioned will need lives to keep playing and will initially be set at 3 they will be shown on the UI in the top left of the screen this allows the player to keep track of the remaining life pool. With having lives comes the challenge of losing them and the game finishing, in the game their will be extra lives placed in areas of each level where the player will have the ability to either add a life to their pool or replace a previously lost life during play, when a new life is picked up it will be shown in the correct area of the UI. In addition to the players lives being displayed the UI will also represent how many keys the player has collected in the style of 0/8 to 8/8 style so the player can see how many are needed to complete the game this will be located in the top right of the screen. As well as the amount needed for the games completion the number of keys needed to be collected in each individual level will be shown on the UI at the top middle also allowing the player to keep track and give a sense of objective during each level.

#### Progression and Challenge

The aim of the game is to collect keys level by level adding to an ultimate tally and the games completion, for our demonstration of this prototype we will showcase 2 unique levels both offering their own challenges to the player. We are aiming to use an old-style platform system where simply an arrow signpost will aim in the direction the player needs to go being a platform game a simple visual guide like this is usually enough to help the player progress. Both levels shown will have an easy first part having the player make a few jumps in order to collect the first key then ramp up as the player goes farther through till the end, each new obstacle adding another challenge to the player the skill level set to the player will be gradual with certain obstacles repeating where the player then knows what they need to do without over taxing the player helping the game feel fun in its playthrough but still having an element of challenge added in. The puzzle element will be set out in a fun manner without making the progression hindered but have the feel of excitement that the player has solved it and can move forward to the next key or stage completion which is important that the player feels goals being achieved as they play.

#### Losing

We have set out that the 3-life system is a good choice and staple design of platformers which allows players to lose a life and learn what they need to overcome to beat the obstacle that lost the life for them. When the player interacts with hazards in the game be it a spike trap, Lava pit or swinging cleaver these will result in the player instantly losing a life from their stored pool. Upon losing a life the player will be transported back to the start of the level where they will have to traverse back to where they lost the life no need to collect keys again but will have to beat the traps and pitfalls to the previous loss of life. Even thought the player will have initially 3 lives and the ability to find more within the stages upon losing the last life in the pool will result in the player’s run ending and the game will end upon where a game over screen will be shown with the message on screen “GAME OVER” and prompted to press the jump input to take the player back to the main menu for another fresh start game.

#### Winning

The player wins the game by keeping at least 1 life in their pool allowing the game to proceed, collect each key within each level add them to the total key tally ending for a successful playthrough. Upon returning to the door at the beginning with all keys collected the player will unlock the door completing the game where upon a message to the UI will inform the player of their triumph “CONGRATULATIONS YOU HAVE FOUND THE ALL THE KEYS!!” as like game over the player will also be prompted here to press the jump input and return to the start screen for another fresh playthrough.

#### Player Challenges

In lieu of any actual enemies, the challenge of Melvin’s Quest will instead come from platforming challenges, hazards and various button-based puzzles. This type of challenge is the main source of difficulty within the game as we seek to create a gameplay experience free of pressure so that the player may enjoy the game world and take their time to figure out and complete the puzzles laid before them. Over the course of the game, these challenges will increase in difficulty as a way of testing the player’s understanding of the game’s mechanics and systems.

## Music and Sounds

For Melvin’s Quest we will source a majority of the music and SFX from websites like freesound.org. For the game’s soundscape we will use some fantasy-esque music mixed with occasional use of environmental sounds based on which are the player is currently in. With the music we hope to emulate the tone of other fantasy games like *The Elder Scrolls V: Skyrim* or *Fable 2*. For the music we will look for royalty-free tracks from websites such as incompetech.com.

We will record sounds for Melvin ourselves, such as jumping or taking damage. Sounds like opening a door or collecting a key will be short, upbeat tunes that provide positive reinforcement for the player.

# TECHNICAL DESIGN DOCUMENT

* PLATFORM:

The platform the game is being aimed at is the PC for starters to see how the game is accepted and how successful. If the game gathers traction, then porting to Console will be a high priority as soon as available to do so while interest in the title is still gathering there is the possibility of a mobile version but will be discussed at length later as the game may not suit mobile devises style of gameplay.

* GAME ENGINE:

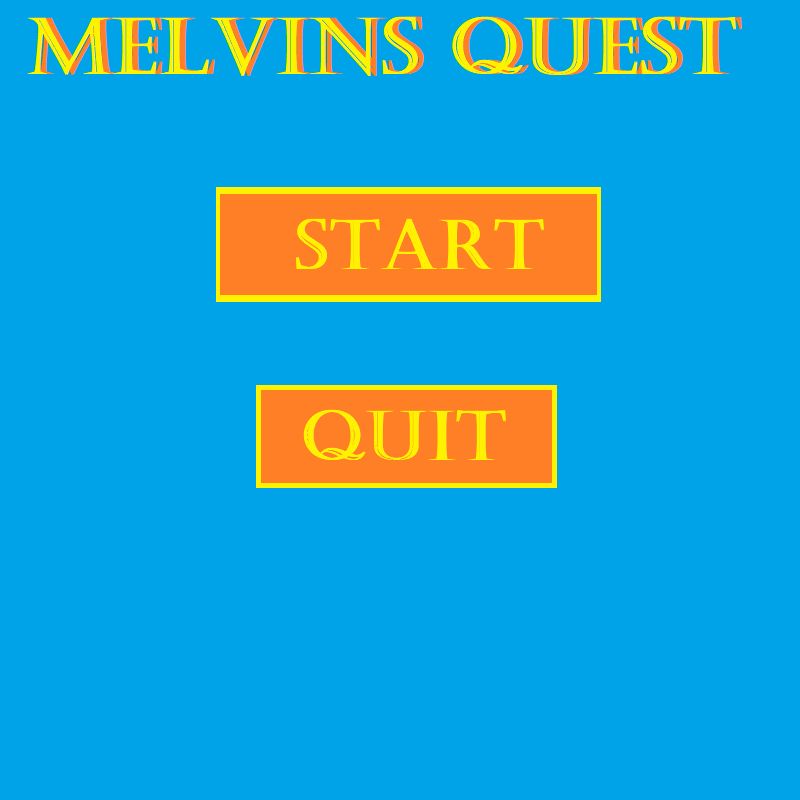
The game itself being at the prototype stage choosing the correct Engine to suit the needs of what we want to build was discussed through either Unity or the Unreal Engine. Both are very powerful, and both suit the needs of what we want to build, with Unity using scripting by the means of C# and Unreal using the Blueprint system with added C++ if desired. When deciding what we wanted to do we chose to use the Unreal Engine as we are familiar with the editor and adding elements and gameplay through the use of the Blueprint method which will be more than we need to build a prototype quickly giving us a great foundation to show of our concept and have the ability to progress if we take the project farther with the ability to port to all the platforms we would like to use.

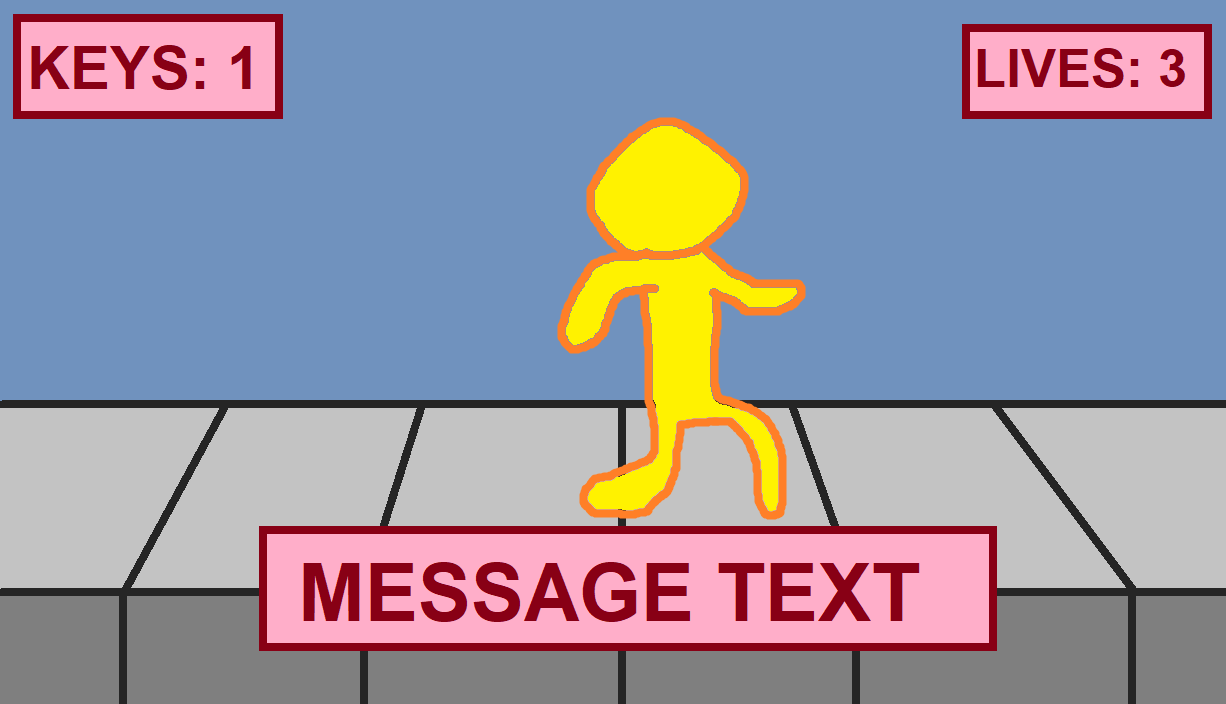
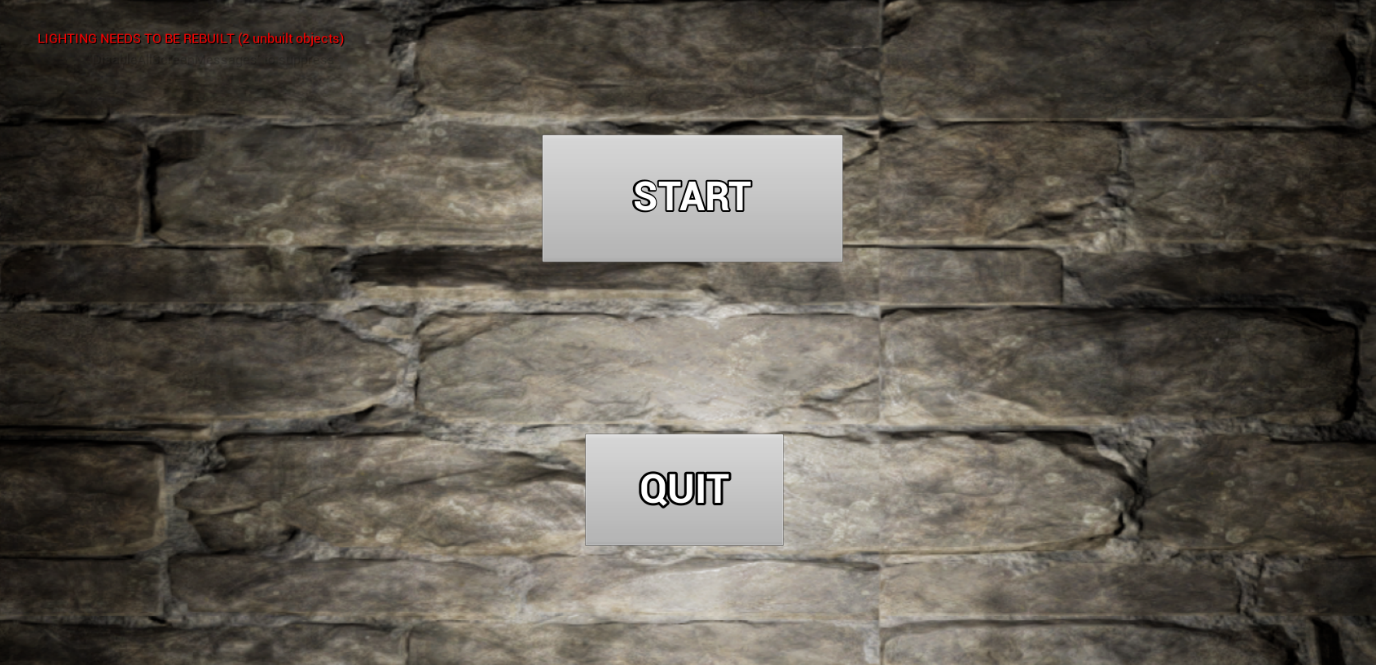
* GRAPHICS ANIMATION AND SOFTWARE:

For the level design we will be using the Unreal Engines editor to create platforms ledges lifts etc which are deployed easily enough. Animations being created through Unreal’s Matinee feature for the movements of platforms doors opening etc. The character will be imported from Mixamo with animations for the character coming from this site as well the character movement animations. Assets we create will come from using Maya as our 3D creation software assets port to Unreal easily enough.

## User Interface

In our initial plans for the game, we chose to use vibrant colours and readable text along with emulating the commonly seen UI layouts in other platformers. With this as a baseline we iterated upon it until we ended up with our own distinct visual style. Below is a progression of the Main Menu as well as the onscreen UI during gameplay, each in 3 stages of development (Mock-up, Stage 1, Final).





## Development Methodology

When it came time to decide on a development methodology for our project, we quickly settled on using Agile. There was little debate between us over which one we would use as we had both used it in previous projects in our time at college. We found that Agile being an iterative approach was extremely beneficial as it allowed us to quickly respond to new developments within our projects as well as work around what we had at that time. Additionally, all other methodologies that we evaluated simply didn’t fit our needs.

## QA bug testing

For our playtesting we adopted the White Box approach to testing, meaning that we were able to observe the code before and during testing a number of test cases. This allowed us to have the maximum amount of insight into our game and how it was handling the various mechanics we had implemented. Below is a list of various aspects of the game that were tested along with the results of each test case and how any problems were solved. These tests were conducted as and when required, as is the nature of White Box Testing.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test No.** | **Item/Action being tested** | **Test data/How to test** | **Expected Results** | **Actual Results/Test Results** | **Comments** |
| **1** | ‘START GAME’ button loads Hub level | Click the ‘START GAME’ button | Hub level is loaded | Hub level loaded | Was able to avert an issue with loading before running as the level name was case-sensitive |
| **2** | ‘QUIT GAME’ button exits game | Click the ‘QUIT GAME’ button | Game is closed | Game was closed |  |
| **3** | Level loading from Hub | Enter a door in the Hub | Level is loaded | Both levels loaded with no problems |  |
| **4** | Hub loading from Level 1 | Press ‘E’ in front of door to Hub | Hub is loaded | Hub loaded with no issue |  |
| **5** | Hub loading from Level 2 | Walk into door at end of level | Hub is loaded | Hub loaded with no issue |  |
| **6** | Player movement - Walking | Press Left and Right arrow keys | Player character moves left and right while playing walking animation | Player character moved left and right with proper animation |  |
| **7** | Player movement – Running | Press ‘Shift’ key | Player character moves faster and plays running animation | Player character ran by default (unintended) | This issue was caused by the default speed being set to the one required to activate the Run functionality. This was fixed by just changing the default speed below this level. |
| **8** | Player movement - Jumping | Press ‘Space’ key | Player character moves up and plays Jump animation | Player character moved upward and played Jump animation |  |
| **9** | Player interaction – Keys | Collide with Key pickup | 1 key is added to total, ‘Jingle’ sound plays, Key is removed |  |  |
| **10** | Player interaction – Losing a life | Collide with a trap (Spikes / Spinning Spikes /  Lava) | Player loses a life | Player did not lose a life | Had to implement extra logic for Hit events as well as Overlap events with objects tagged as ‘hurt’. This fixed the problem. |
| **11** | Player respawning | Collide with a trap (Spikes / Spinning Spikes /  Lava) | Player loses a life and respawns | Player lost a life and respawned | When implemented, this functionality affected the elements tested in cases 2,3&4 as the life subtraction was tied to the respawning logic. The game would subtract a life after a level load as the respawn code was used in the process. This was fixed by relocating the execution of the life subtraction function to the collision event itself. |
| **12** | Key count persisting across levels | Collect a Key then enter a level transition | Key count stays the same | Key count stayed the same |  |
| **13** | Life count persisting across levels | Lose a life then enter a level transition | Life count stays the same | Life count stayed the same |  |
| **14** | Button opens door in Level 2 | Step on button | Button moves down and door is removed from level | Button moved down and door was removed from level |  |
| **15** | Game Over functionality | Collide with a trap and lose last life | Game loads Main Menu and resets Lives + Keys | Main Menu loaded, Lives + Keys reset |  |
| **16** | Win functionality | Enter Hub with a Key count of 4 | Win message displays then game loads Main Menu after 5sec delay | Win message displayed then Main Menu loaded after 5sec delay |  |

## Asset List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Asset name** | **Asset type** | **Created by** | **Sourced** | **Notes** |
| buttontone | .wav | Fraser McAulay | / |  |
| doormove | .wav | Fraser McAulay | / |  |
| KeyJingle | .wav | Fraser McAulay | / |  |
| 8bit blix | .wav | Questiion (Freesound) | sourced by Stuart Dunan |  |
| 8bit\_chase\_music-loop | .wav | ShiftKun (Freesound) | sourced by Stuart Dunan |  |
| NormalKey | .fbx | Fraser McAulay | / |  |
| KeyDoor | .fbx | Fraser McAulay | / |  |
| SpikeTrap | .fbx | Fraser McAulay | / |  |
| Wood\_Spike | .fbx | Stuart Dunan | / |  |
| Mixamo\_Goblin | .fbx | Mixamo | sourced by Stuart Dunan | animated by Stuart Dunan |
| Medieval Dungeon Pack | Unreal Engine content pack | Infuse Studio | sourced by Stuart Dunan | implemented by Stuart Dunan |

For the sake of maximising time spent working on the game itself we decided to source the majority of our assets such as the player character and various level assets (the dungeon pack in particular). We found that this greatly sped up the development time of the game though there were some issues early on with implementing the model for the player character and subsequently adding its animations.

## Asset Production

For creation of our 3d models we used Autodesk Maya 2019. Any graphics were created in GIMP and all sounds were recorded and edited in Audacity. We chose these particular programs as we were both well accustomed to using them from previous work in college. Importing these assets into Unreal Engine was relatively simple, though there were a few glitches with some of our 3d models when imported into our levels. These issues were fixed with a bit of research and tinkering which in itself was a learning experience in dealing with unexpected and sudden problems.

## Game Flow

The intended flow of the game was to follow the traditional structure of a platformer. In our game, the player starts in a ‘hub’ area of sorts so that they may choose which level to enter first. Following level selection, the player is to explore the level in the regular platforming fashion. Throughout the level they will have to avoid deadly traps as well as pick up 4 keys that will allow them to win the game. Should the player return to the hub area with all 4 keys in their possession, a brief congratulatory message will be displayed before sending them back to the Main Menu. This process is detailed in the diagram below (*Figure 3*).

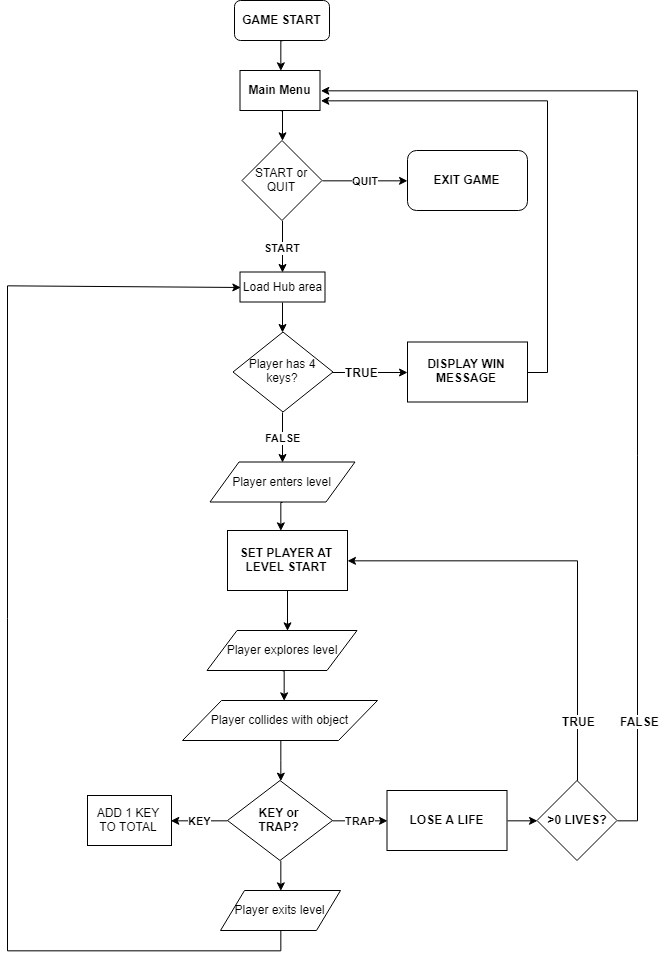


Figure 3 - Intended Game Flow

## Game Creation Process

As our game was created in Unreal Engine, we were able to make great use of one of its key features: Blueprints. Blueprints avoided the need to use any C++ in our project which was a blessing as neither of us are very adept with that particular programming language. Additionally, the way Blueprints are displayed allowed us to easily visualize the flow of every specific gameplay event that we had created. Any errors would prevent us from immediately running the game which in the end did help us keep at it until we’d fixed the problem at hand. The error messages, while sometimes filled with confusing jargon, were useful as they helped guide us toward the solution to any errors flagged up by the compiler. A source of frustration was occasionally having to figure out which highly-specific command was the one that would do what we needed for a particular function, though in turn it was highly rewarding to see features we’d racked our brains over work after an hours-long struggle. The function we are most proud of is the player respawning (*Figure 4*) as it required extensive research and numerous deep dives into the Unreal Engine forums, learning from others mistakes and picking up helpful tips from some of the people who actually worked on the engine.

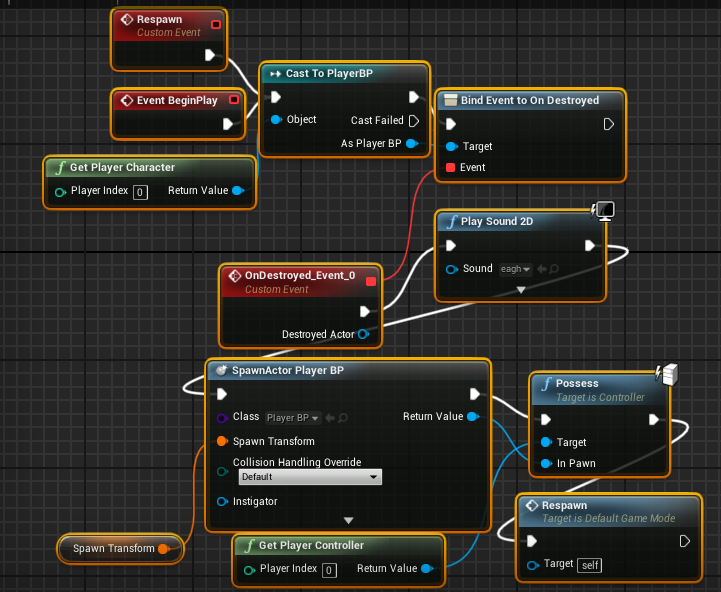


Figure 4 – Respawning

Below are some more examples of our use of Blueprints in Unreal Engine. The following five display our logic for when the player collides with or overlaps with objects with certain tags. This logic was integral to our game functioning properly and thus was rigorously tested and meticulously examined whenever an error happened to appear.

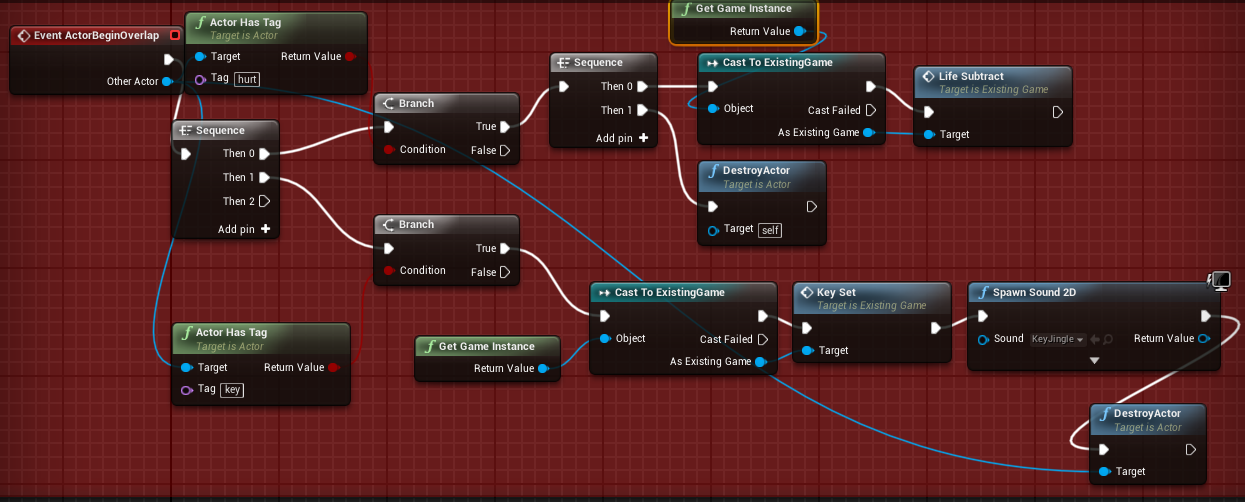


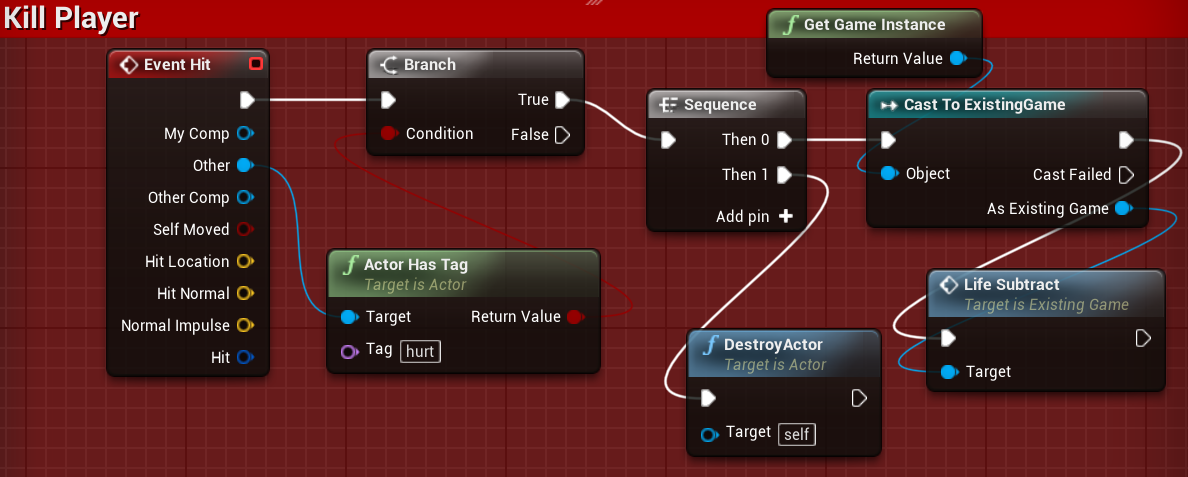
Figure 5 - Player Overlap Check

Figure 6 - Player Hit Check

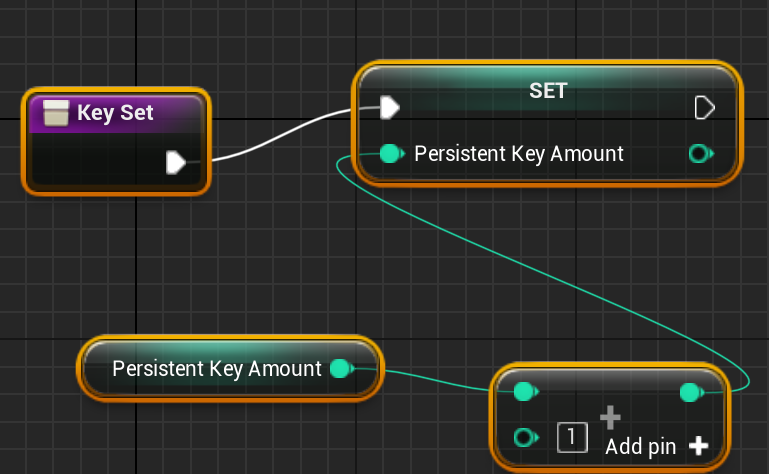


Figure 7 - Adding Key when Key is collected



Figure 8 - Subtracting Life and by extension checking for a Game Over



Figure 9 - Function for resetting lives and score, called upon game start

## Source Control

We later found out that Unreal projects and GitHub do not go well together as projects often get quite bulky very quickly. Storing the whole project would have required Git LFS which, unfortunately, we were unable to figure out how to implement into our project. As a result, we had to resort to just using GitHub to keep track of versions of our Game Design Document as well as a few other miscellaneous files. Upon reflection we wish that we’d been able to better understand Git and version control in general.

Below is an overview of our GitHub page:

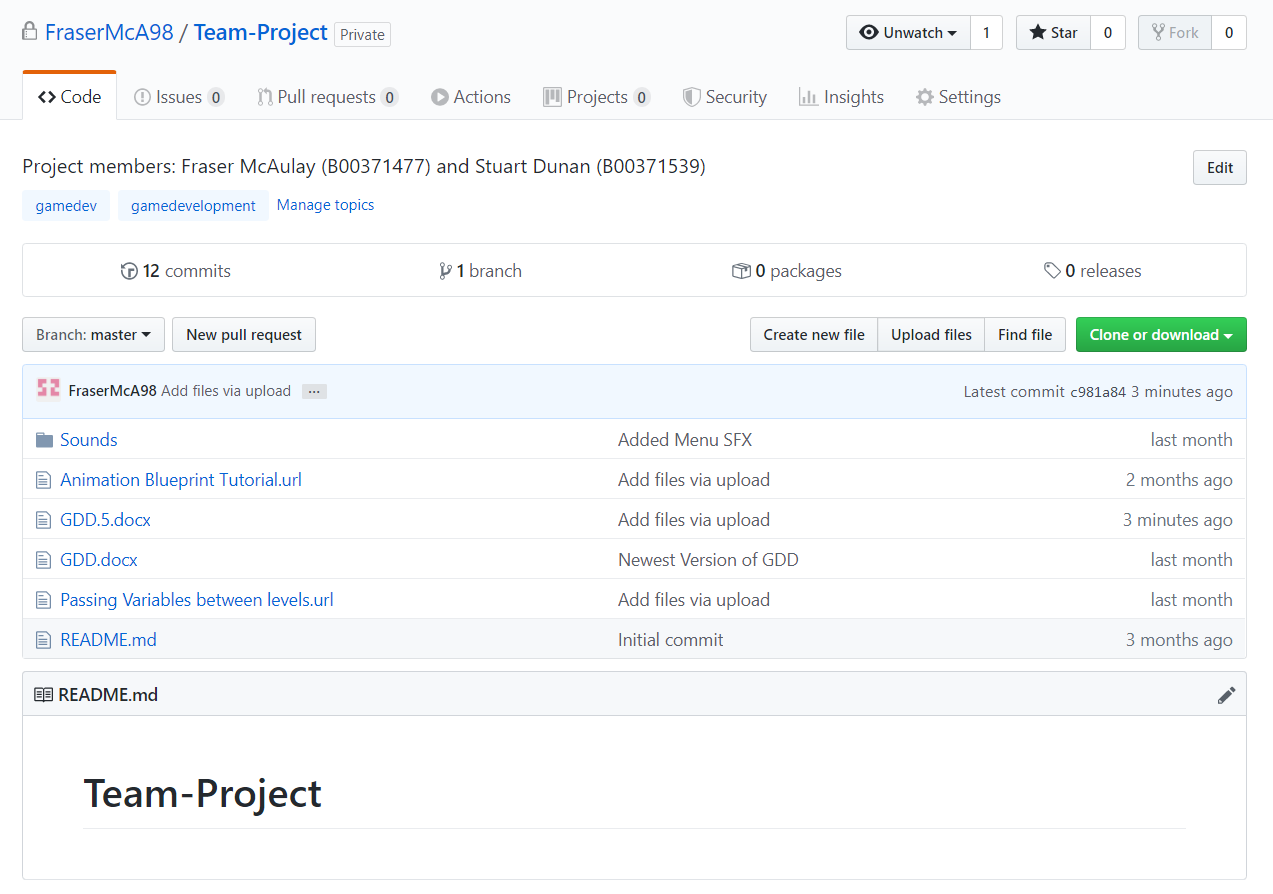


Figure 2 - GitHub project page