H446

Rune Raid

H446 Programming Project

Name: Kieran Dhir

Centre Name: South Bromsgrove High School Academy Trust

Centre Number: 24035

Candidate Number: 4699

Contents

Analysis

An outline of the Project

Summary

The name of the game that I am creating is called “Rune Raid” and the main genre will be platform and Puzzle. The game is set in an abandoned temple deep mountains of Bhutan and will consist of multiple levels, each with their own unique puzzle to solve. There will be a timer throughout the game timing how long the player takes to complete the puzzles and finish the game and this time will be recorded along with their username on a leader board for players to compete for 1st place.

Why a computational approach

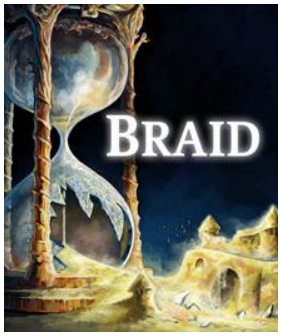
As Rune Raid is a computer game, the only way for this to be developed and played will be on a computer device. Therefore, developing Rune Raid will require a computational approach, without this approach it would be impossible to create Rune Raid as a computer game, it would have to be in another form such as a board game. Rune Raid will require many different algorithms to solve problems such as player movement and collision detection, which I will require a computational approach to create and solve.

The storyline

You, the player, will be playing as Charles an archaeologist, who has been given exclusive access to the long forgotten Himalayan temple in the heart of Bhutan. Apparently, hundreds of years ago when the temple was founded it was built upon an ancient rune ritual site and to this day it is unknown whether the runes still exist and Charles is about to find out if he can solve the mystery surrounding the old temple.

Similar games

Braid



This game is a platform and puzzle video game which are the same genres as my video game. The game was released in August 2008 however, unlike my game where the platform is only PC, Braid’s platforms are Xbox 360, PlayStation and Linux PC. The game’s storyline unfolds as the protagonist, Tim, attempts to rescue a princess from a monster. Throughout the game text passages tell the tale of the story whilst the player is playing the game, which will be similar to my game as I plan to create a storyline to play throughout my game as well.

The player is faced with challenge of finding and assembling jigsaw puzzle pieces to complete each

level which I may use as inspiration and also create a level with a jigsaw puzzle involved.[1]

Identification of all the stakeholders

Rune Raid will be targeted towards years 10 and above because my game involves puzzles which younger players may find too difficult to understand and therefore be unable to complete the game. However, as my game will contain no violence, bad language etc. and so Rune Raid will be rated at PEGI 3 even though my game is targeted towards age 10 and above. As I myself and my friends enjoy playing computer games I will be using myself and my friends to review Rune Raid and give feedback throughout my development process. I want to make Rune Raid and enjoyable puzzle platform game for me and others to play and have fun with.

The PEGI regulations

PEGI 3

The content of games given this rating is considered suitable for all age groups. Some violence in a comical context (typically Bugs Bunny or Tom & Jerry cartoon-like forms of violence) is acceptable. The child should not be able to associate the character on the screen with real life characters, they should be totally fantasy. The game should not contain any sounds or pictures that are likely to scare or frighten young children. No bad language should be heard.

PEGI 7

Any game that would normally be rated at 3 but contains some possibly frightening scenes or sounds may be considered suitable in this category.[2]

As Rune Raid is PEGI 3 and targeted towards ages 10+ my game design will be cartoon as this is the most attractive towards a young audience as they can relate to cartoon designs due the majority of children watching cartoon shows on the television such as, SpongeBob, Regular show and Adventure time.[3]

However, cartoon designs can also be targeted towards older audiences as well such as 16+ because they grew up with cartoons and cartoon games such as Pokémon and so can also relate to the design. There are also cartoon programs targeted towards adults such as Rick and Morty which is very popular for older audiences.[4]

As cartoon designs are relatable towards young and old audiences it is the perfect design for Rune Raid, as my target audience is young players but I also don't want to deter older players from enjoy the game too.

[1] https://en.wikipedia.org/wiki/Braid_(video_game) – Date accessed 03/09/17
[2] http://www.pegi.info/en/index/id/33/ - Date accessed 28/09/17
[3] http://www.cartoonnetwork.co.uk/show - Date accessed 28/09/17
[4] http://www.imdb.com/title/tt2861424/?ref_=nv_sr_1 - Date accessed 28/09/17

Why puzzle and platform genre?

There are a few reasons why I chose Rune Raid to be of the puzzle and platform genre which are;

- I very much enjoy the platform genre myself as I grew up with games such as Sonic the Hedgehog, New Super Mario Bros and Super Mario Galaxy. And so I very much wanted to create a platform game for myself.
- I like a challenging game and so games with puzzles incorporated within them I believe gives the game more depth and creates a brilliant gaming atmosphere because the game forces you to use your brain and once a puzzle is completed it gives me a buzz like no other.
- Other than my personal opinions the Puzzle and Platform genres are very popular with gamers and are often ranked in the top 10 or 20 most popular gaming genres.[5][6]
- With me very much liking the two genres and also them being popular in the gaming industry it is the perfect combination for Rune Raid.

Reason for male player character

The reason why I chose my player character to be a male character and not female was because I did some research and surveys convey that there is a larger percentage of male gamers compared to female gamers. [7]

Male gamers: 59%

Female gamers: 41%

Therefore, with a larger percentage of male gamers it is more viable to have a male character rather than a female because a larger percentage of my target audience can relate to the main characters gender.

Colour Scheme

As Rune Raids design is cartoon the colour scheme I will be using will be bright and colourful because I want to create a positive atmosphere whilst playing the game. Also having bright and eye-catching colours will make my game more attractive towards a younger audience because they tend to side with colourful images rather than dull and dark colours.[8]

The graphics

The graphics that Rune Raid will have will be simple and old school graphics because they are much easier to implement into a game and also gamers like games that take the look of old retro style graphics. [9]

There are many current popular puzzle games on the market that have cartoon designs and old school graphics such as;

- Braid - **(9/10 Steam, 8.8 IGN)**
- Tetris - **(3.9/5 iTunes, 4/5 Common Sense Media)**
- World of Goo - **(9/10 Steam, 4.7/5 Google Play)**
- Candy Crush Saga - **(4.4 Google Play, 4.7/5 iTunes)**
- Fez - **(9/10 Steam, 81% Metacritic)**

[5] https://www.thetoptens.com/video-game-genres/ - Date accessed 29/09/17
[6] https://www.gamefaqs.com/top10/2422-the-top-10-video-game-genres - Date accessed 29/09/17

[7] http://www.bigfishgames.com/blog/2017-video-game-trends-and-statistics-whos-playing-what-and-why/ - Date accessed 29/09/17
[8] http://www.resene.co.nz/homeown/use_colr/colours-for-living.htm - Date accessed 30/09/17
[9] http://www.dorkly.com/post/50754/8-things-gamers-want - Date accessed 30/09/17

All these games have very high ratings and so the cartoon Puzzle and Platform genre with bright colours is clearly a very popular game design. Therefore, Rune Raid will be using the same format as it is popular in the video game market.

Identification of the target platform

Rune Raid will be exclusive to the PC platform, there are a few reasons why;

- I'm using my desktop PC to code Rune Raid as I'm using Unity and so to have Rune Raid on the PC platform will make testing the game much easier as I can code and test Rune Raid on the same platform.
- The majority of games that are released all are compatible on PC platform because it's the most popular platform for gaming and so it makes sense to have Rune Raid on the PC platform. [10]
- I only want to make Rune Raid on one platform because it's much easier than creating it on multiple platform because each platform has to be coded differently such as PC has a keyboard whereas, PS4 has a controller and no keyboard.
- For me to create Rune Raid on a console platform it would be much more difficult than to just export my game from Unity engine which is ready to be played on PC platform.

Therefore, because of all these reasons I decided Rune Raid would only be for PC platform. The operating system Rune Raid will be compatible for will be Windows as it's the most popular operating system.[11]

Hardware for PC

- Keyboard
- Mouse
- Monitor
- Headphones (Accessory)
- Speakers
- Gaming Controller (Optional)

Controls

For Rune Raid the controls will only for keyboard and mouse, it will not be compatible for gaming controllers. This is because I believe it is better to get the full PC platform experience of a keyboard and mouse rather than a gaming controller.

Headphones/earphones/speakers

In Rune Raid there will be a soundtrack which will be played through the game whilst the player is playing to create a nice atmosphere. However, sound is not needed to play the game because all the narrative storyline will be in the form of text. Therefore, the player can decide whether they want sound or not.

Monitor

Rune Raids only user interface will be on a pc monitor/TV screen connected to a desktop and so no other devices will be needed.

Mouse

The mouse will be used for a cursor in Rune Raid for selecting objects and moving on the narrative text for the storyline or just skipping the text popups.

Files

[10] http://www.bigfishgames.com/blog/2017-video-game-trends-and-statistics-whos-playing-what-and-why/ - Date accessed 30/09/17

[11] https://www.computerhope.com/issues/ch001777.htm - Date accessed 03/09/17

As I'm coding and programming Rune Raid in the Unity engine the majority of the files will be stored in Unity whilst I'm programming the game and any code that I make outside of Unity will be a C# script in Notepad ++ which is linked to the Unity Engine.

When I have completed my game and it is ready to be all put together there is an option in Unity to build my game and once it is built it creates a .exe file and a data folder which can then be both used to run Rune Raid[12]. The file exported will be only compatible for Windows and not Mac or Linux because I cannot test the game in Linux or Mac because I only have a Windows operating system.

Explanation of the user needs

The PC platform has a different game style compared to that of smart phones or console games. This is because gamers expect different aspects of a game for when they play on the PC platform to when they play on their smart phone.

Smart Phone Games

When gamers play on their smart phones they expect the game to be either short, easy to learn, fast paced and small levels, such as Candy Crush Saga, so that they can play on the go e.g. on the train for 10 minutes or a Massively Multiplayer Online (**MMO**) strategy game, such as Clash of Clans, which they can pick up and play for 5 minutes and then put down to play again later.

Console Games

Console games tend to be similar to PC games with only a slight difference which is that there are a lot less small developer games on the console platform because it is very expensive to release and support console compared to PC due to console games mainly being sold as disks rather than digital download. Therefore, it is much more expensive to create and distribute these disks. Also to publish a game on all three platform the developers will need a publisher to publish their game which will significantly increase the price of releasing the game.

As a result there are very few games on console which are developed by individuals or small teams unless the game was very popular on PC first and then when the company got enough funds they got a publisher to release their game on the other platforms.[13]

Hence, games with smaller developer teams that rely on digital distribution, which are called "Indie games", are less likely to be found on platforms other than PC or smart phone. And so console games are more draw towards "triple A" games. Triple A games (**A lot of money, A lot of time and A lot of resources**) is a classification term for games that have the largest development budgets and amount of promotion. Thus, triple A games are very much expected to be the highest quality and to be in the year's top bestsellers.[14]

PC games

The PC platform has the biggest range of games for any platform, which vary from very small indie games to massive triple A games. Each type of game is accepted by the PC gaming community and so Rune Raid fits best on the PC platform.

[12] http://answers.unity3d.com/questions/325384/how-to-export-a-finished-game-help-please.html - Date accessed 30/09/17
[13] https://www.quora.com/Why-are-console-games-more-expensive-than-PC-versions - Date accessed 03/10/17
[14] https://en.wikipedia.org/wiki/AAA_(video_game_industry) – Date accessed 03/09/17

PC gamers don't expect amazing graphics or complex game mechanics from a game, as long as the game is fun and bug proof the game has a chance of becoming a big hit on the market. This is why Rune Raid's key aspects will not be the graphics or the amazing mechanicsHowever, the PC games cannot be in the form of a smart phones game which are very short, to be able to be picked up played and then put down. This is because a smartphone is mainly used when people are on the move or are just relaxing and want a quick game. Whereas, PC games are mainly played when people have more than 30 minutes of time to play the game and so PC games need to be more in-depth and much longer than smart phone games.

In conclusion, as Rune Raid is on the PC platform it doesn't need stunning graphics nor incredible mechanics but must have depth, be longer than smart phone games and have another aspect such as a gripping storyline or great gameplay to become successful. The key aspects that I will be using for Rune Raid will be the overall gameplay of the game through the complex puzzles, a little storyline and enough play time to satisfy a PC gamer.

There will be no levelling up or achievements throughout Rune Raid but there will be a timer which records the time it takes for the player to complete each level which is then recorded in a leader board that displays all the best times for each level. This is to create a competitive environment as to who can complete Rune Raid the fastest and get their name on the top of the leader board.

Detailed research into existing similar products

There are many puzzle and platform games on the PC platform market but I'm only interested in the games which that were very popular. This is because I can research into each of the games to find out what made them so popular and to see if I could take any inspiration from these games and implement those ideas into my game to make my game more enjoyable and of a higher quality.

These games below are a few very popular games that stood out to me as similar to my vision of Rune Raid.

- Braid - **(9/10 Steam, 8.8 IGN)**
- World of Goo - **(9/10 Steam, 4.7/5 Google Play)**
- Fez - **(9/10 Steam, 81% Metacritic)**

After researching into each of these games I have come up with a list of things that I liked and what other gamers liked about each of these games for me to decide what I should implement into Rune Raid to make it an enjoyable game.

Before I talk about each game, there is a source which I have used for most of the games to look at customer reviews and that resource is a PC application called Steam, this is very easy to access via either the application or the website.

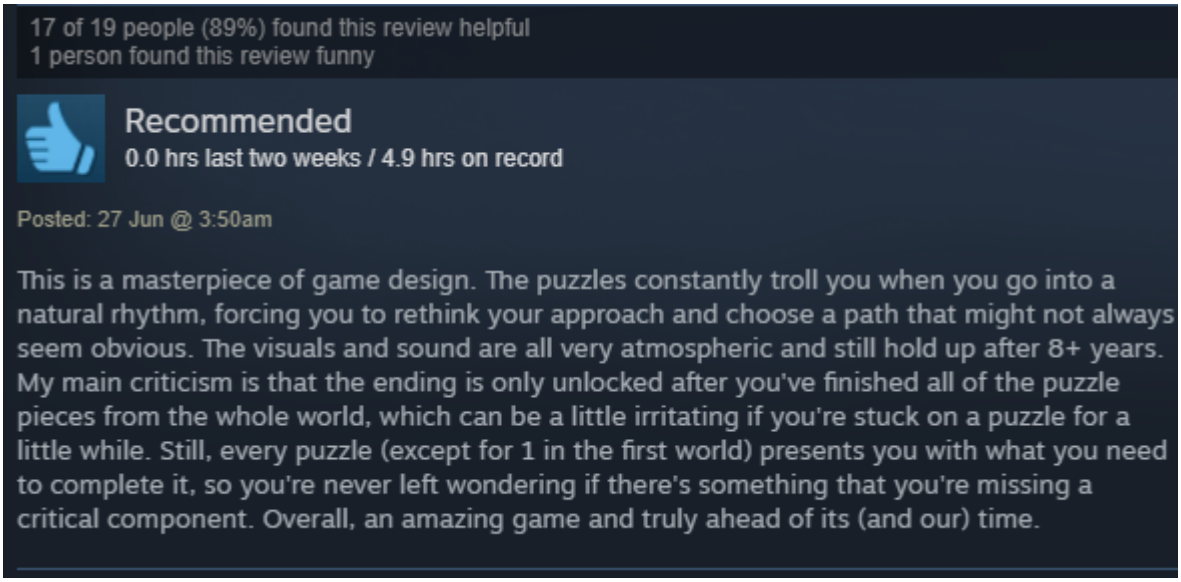
Steam is a platform application that provides a place for;

- PC games to be bought.
- PC games to be played.
- A network connection for online gamers to talk on and invite players to play multiplayer games with them.
- Fellow gamers in the steam community to discuss about games they have played and leave reviews for other gamers to read and for developers to take feedback for future improvements or idea.

Braid

I've already previously discussed about Braid with a quick summary of the game and so to read it again it's on page 3 of this document. However, this time I've delved deeper into the game this time to find out why this game was so popular.

Gamer reviews



17 of 19 people (89%) found this review helpful
1 person found this review funny

Recommended
0.0 hrs last two weeks / 4.9 hrs on record

Posted: 27 Jun @ 3:50am

This is a masterpiece of game design. The puzzles constantly troll you when you go into a natural rhythm, forcing you to rethink your approach and choose a path that might not always seem obvious. The visuals and sound are all very atmospheric and still hold up after 8+ years. My main criticism is that the ending is only unlocked after you've finished all of the puzzle pieces from the whole world, which can be a little irritating if you're stuck on a puzzle for a little while. Still, every puzzle (except for 1 in the first world) presents you with what you need to complete it, so you're never left wondering if there's something that you're missing a critical component. Overall, an amazing game and truly ahead of its (and our) time.

This reviewer's key points are that Braid is:

Pros

- The puzzles are each unique and are not just re-skinned previous puzzles – this makes the game more exciting and prevents the player from easily learning the pattern of the puzzles.
- The visuals and sound atmosphere are great.
- The puzzles are not impossible and everything you need to complete it is presented before you – this prevents players from getting frustrated by not being able to complete a puzzle.

Cons

- The game makes the player complete all the puzzles in the game before the ending occurs which is slightly annoying if there is a puzzle you can't seem to complete – the majority of games allow the player to get the game ending without completing the game 100% but then allows the player to achieve 100% after if they desire to do so. This then allows both types of gamers to enjoy the game to the fullest, the 100% completion gamers have the opportunity to achieve 100% and the gamers who just enjoy playing the game and don't care for 100% completion can get the ending too without being forced to do 100%.
- This reviewer's key points are that Braid is:

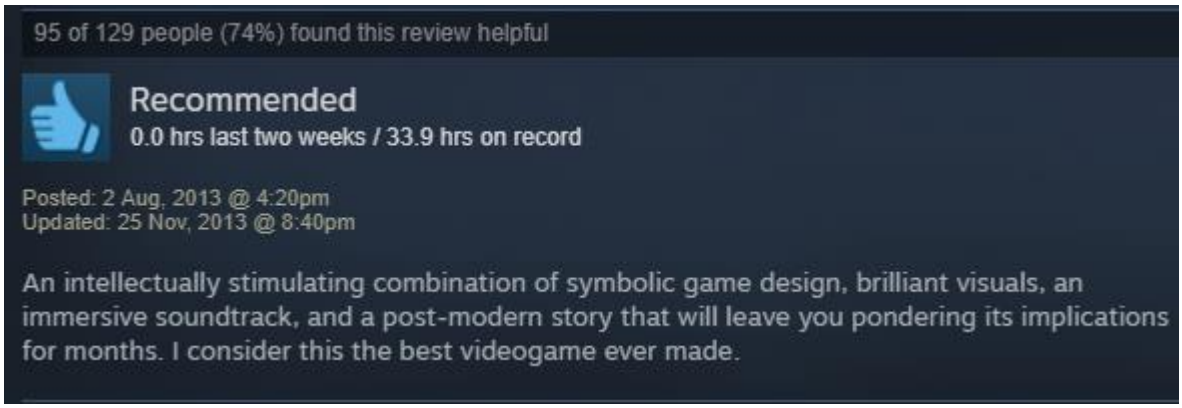
This reviewer's key points about Braid are:

Pros

- Puzzle designs are flawless as they explain what each new aspect of the puzzle mechanics but also make the puzzles a challenge.
- Most of the puzzles you can skip and go back to later as there is no linear campaign that forces the player to complete a puzzle to get to the next. - This allows the player to pick and choose which the order of puzzles they want to play. This prevent them from getting frustrated and angry because they can't progress because of one puzzle.
- Beautiful graphics.
- The story has multiple layers of meaning. – Creates a story that is not just one-sided and simple.

Cons

- Would have liked a hint system



This review's key points were:

- Brilliant visuals
- Immersive soundtrack
- Post-modern story that will leave you pondering implications for months.
- Intellectually stimulating game design.

This reviewer however doesn't recommend Braid to gamers, their reasons are:

- For someone who doesn't enjoy constant puzzles after puzzles it is a boring and gets old very fast.
- They got bored extremely fast due to the game only consisting of puzzles.

My own view on Braid

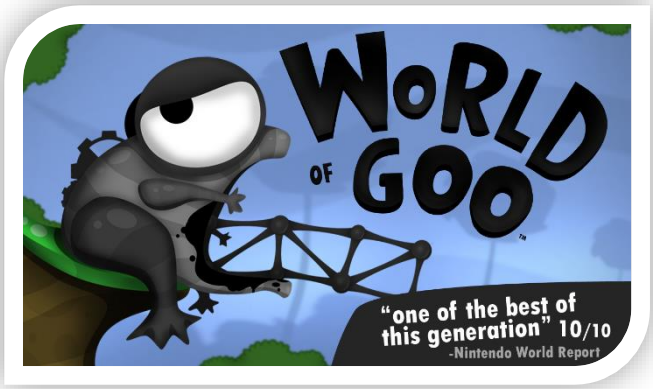
As I don't own a copy of Braid myself, I decided to watch some videos of other gamers playing Braid to get an idea of how to game played out. It is true that the graphics are great for a platform game even though it was released in 2009. The design was clearly inspired form Mario Bros which I am a fan of and so I rate it highly. The controls seem very simple: spacebar, arrow keys/WASD for movement and a cursor for interacting with jigsaw puzzle pieces to complete the jigsaw. There are monsters in the game which have very simple AI which is to just walk in the facing direction until there is a pitfall or a jump. When it collides with another object e.g. a wall it then turns the other way and continues to walk in the new direction.

The puzzles are simplistically made but are not simple, they get progressively harder after each level which I like because it eases you into the game to first get an understanding of how the game works before delving into much harder puzzles. The storyline also is interesting which adds atmosphere to the game.

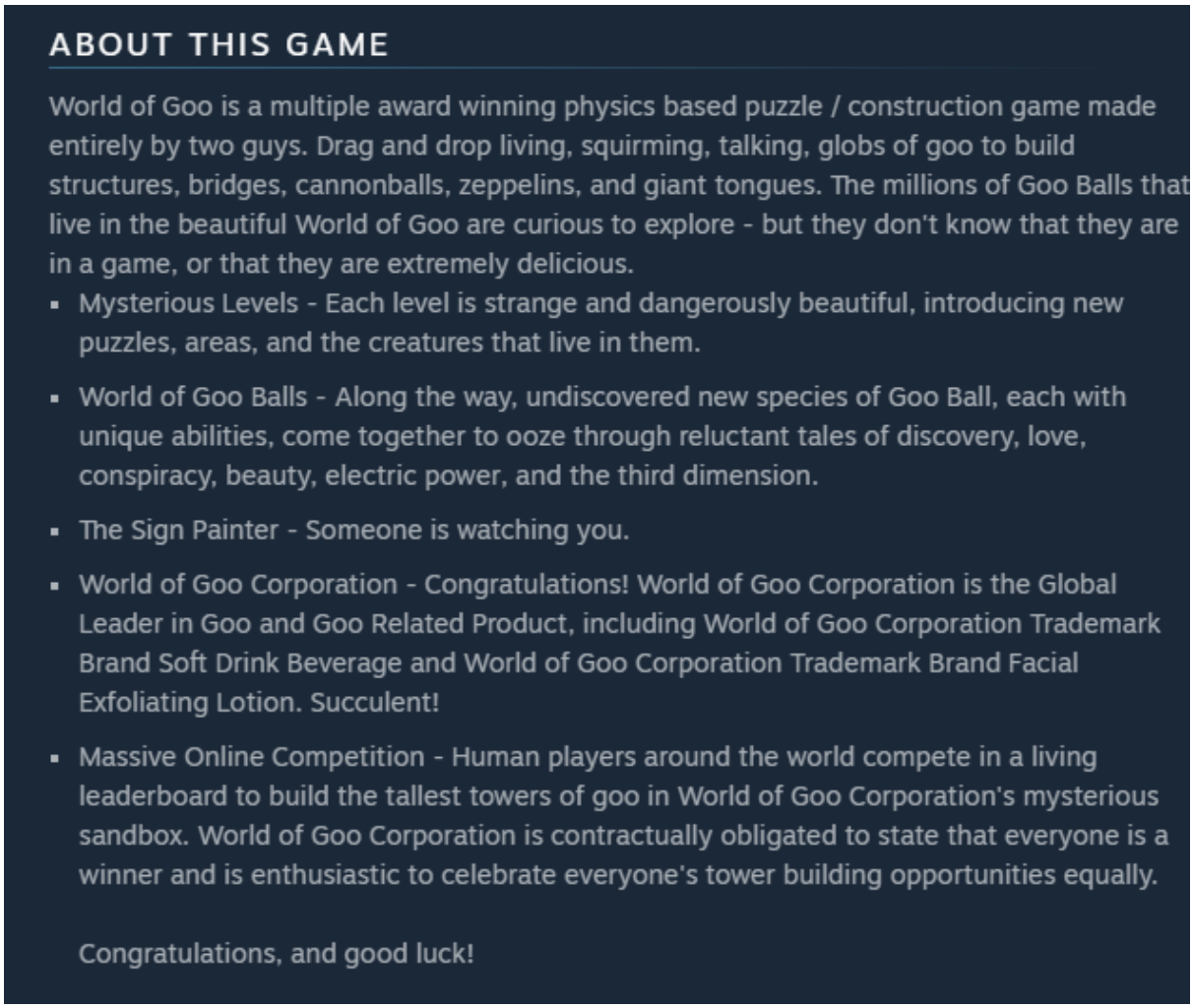
Aspects of Braid that I will take into account

- Puzzles getting progressively harder – this will improve the gameplay as the player can first get used to the controls and how the game plays before trying to complete harder levels.
- Simple controls because it allows the player to focus all their attention towards the puzzles and not the controls.
- Simple AI mobs that can add character and a different aspect to the game – E.g. if the player hits a monster it adds 5 seconds to the timer.
- Don't make every level pure puzzle based, add some other aspects to the game to keep the game fresh.
- Add objects for the player to collect to be able to 100% the game to add more depth to the game.
- Create a hint system that allows the player to get some help completing a puzzle if they need some extra help.
- Create a storyline that matches with the gameplay and adds an atmosphere for the player to relate to the main character.
- Nice graphics and a good design is highly attractive to gamers and so I will try to make Rune Raid eye-catching.

World of goo



Summary – From Steam



ABOUT THIS GAME

World of Goo is a multiple award winning physics based puzzle / construction game made entirely by two guys. Drag and drop living, squirming, talking, globs of goo to build structures, bridges, cannonballs, zeppelins, and giant tongues. The millions of Goo Balls that live in the beautiful World of Goo are curious to explore - but they don't know that they are in a game, or that they are extremely delicious.

- Mysterious Levels - Each level is strange and dangerously beautiful, introducing new puzzles, areas, and the creatures that live in them.
- World of Goo Balls - Along the way, undiscovered new species of Goo Ball, each with unique abilities, come together to ooze through reluctant tales of discovery, love, conspiracy, beauty, electric power, and the third dimension.
- The Sign Painter - Someone is watching you.
- World of Goo Corporation - Congratulations! World of Goo Corporation is the Global Leader in Goo and Goo Related Product, including World of Goo Corporation Trademark Brand Soft Drink Beverage and World of Goo Corporation Trademark Brand Facial Exfoliating Lotion. Succulent!
- Massive Online Competition - Human players around the world compete in a living leaderboard to build the tallest towers of goo in World of Goo Corporation's mysterious sandbox. World of Goo Corporation is contractually obligated to state that everyone is a winner and is enthusiastic to celebrate everyone's tower building opportunities equally.

Congratulations, and good luck!

The World of Goo is similar to Rune Raid in the aspect of it being a puzzle game however, the game also revolves around physics. Therefore, the way the game works will be slightly different from Rune Raid because not all levels in Rune Raid will involve physics whereas, the World of Goo, physics is a key part of the game.

Gamer reviews

The reviewer's key points about the World of GOO are:

- Great design and graphics.
- Quirky and humorous. – Humour makes the game more fun because laughable and funny aspects of games create a happy and light hearted atmosphere to allow the gamer to just enjoy it to the fullest.
- Challenging levels for both strategic and quick thinking minds. – This allows for both mind-sets to have an enjoyable time playing the game.

This gamers key points are:

Pros

- There is built in comedy where the player watches their failed attempts of trying to complete the level. Their bridge that they just spent time on creating just collapses completely when the level is played.
- There is a huge satisfaction when a level is completed – it's not that you have worked out the answer it is the fact that you have used your ingenuity to build your own unique solution to the puzzle.

Cons

- The controls are a hindrance at times – it's difficult to sometimes interact with the game object correctly and so this causes unwanted movements or clicks that cause the player to repeat the level.

- Some of the humour doesn't work and makes the game slightly out of sync with its other aspects.
- One of the themes is not very relatable and so isn't well delivered.

This gamer's key points are:

- The game is very enjoyable, enough that this gamer has played through the game twice.
- The levels are challenging – prevents the player from getting bored.
- The style is goofy – this is great for younger gamers but also it appealing to some older audiences who can relate it from their childhood.

My own view on the World of Goo

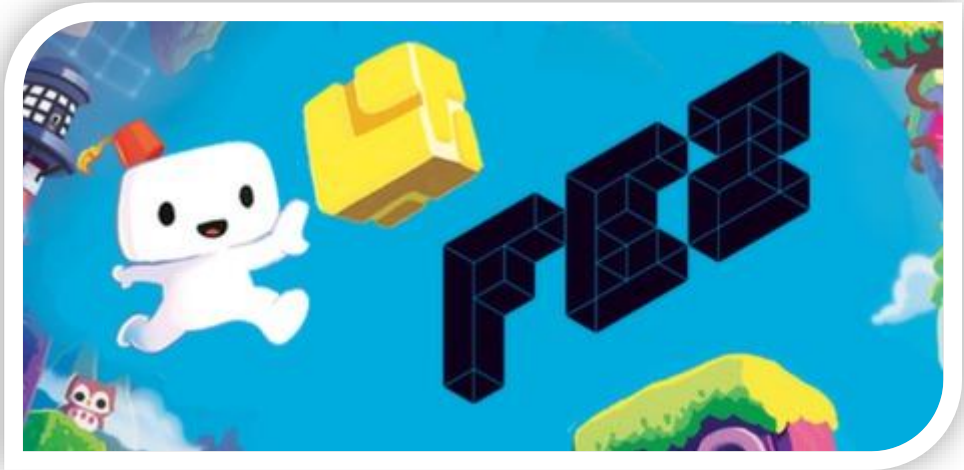
Again as I don't own a copy of the game I yet again decided to watch some videos of other gamers playing the game to get an understanding of how the World of Goo worked. It is true that the entire game revolves around a gooey theme and heavily involves physics to complete each level. This is a nice aspect of the game because it adds restrictions to the puzzles in the form of structure, weight and resources to force the player to use the resources they are given efficiently, it prevents the player from just going wild and building endless bridges which will ruin the aspect of the game and also impact the FPS, too many objects may eventually crash the game. This prevents gamers from getting technical advantage in the game if they were they have much be computer specifications. The controls only consist of a cursor that is used to click and drag objects to the desired place, which is only suitable in this kind of game genre of object building.

Overall, I very much like the game and how the puzzles are implemented, they physics aspect add an extra variable to the game and it looks very fun and challenging which I enjoy very much as a gamer myself.

Aspects of the World of Goo that I will take into account

- The physics aspect of the puzzles are very fun and entertaining to watch and so I may take inspiration from this and implement some physics into some of Rune Raid levels.
- Creating a humorous side of the game adds a nice atmosphere but it has to be relatable humour for multiple gamers and not just a specific type of humour such as dry.
- Puzzles which can be completed uniquely makes the completion of a puzzle more satisfying for the player.
- Make sure to fully test controls of the game to prevent annoying bugs that ruin the player's attempt to complete a puzzle.

Fez

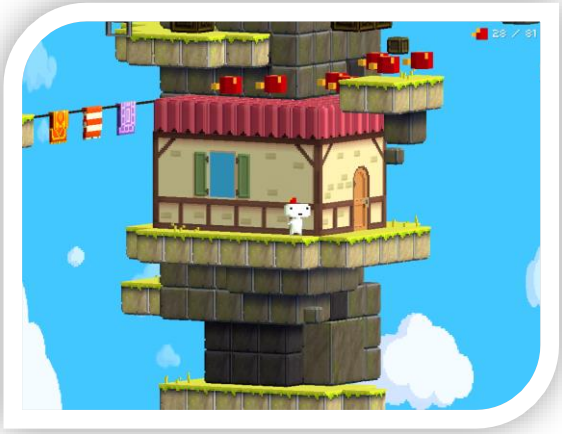


Summary - From Steam

ABOUT THIS GAME

Gomez is a 2D creature living in a 2D world. Or is he? When the existence of a mysterious 3rd dimension is revealed to him, Gomez is sent out on a journey that will take him to the very end of time and space. Use your ability to navigate 3D structures from 4 distinct classic 2D perspectives. Explore a serene and beautiful open-ended world full of secrets, puzzles and hidden treasures. Unearth the mysteries of the past and discover the truth about reality and perception. Change your perspective and look at the world in a different way.

Fez is a very unique game because it breaks the 2D platform genre by adding 3D into the game but keeping it still platform, this is called 2.5D. This is don't by the rotation of the game camera to different angles of the world of fez, at first this is very hard to get used to because it messes with you mind, as it's hard to understand how to play the 2D game whilst 3D still exists. The puzzles are often solved by rotating the camera around the world to get to a different view of the world.



This is how to different dimensions work, it has the 2D view but the camera can change between 2D and 3D to get different views of the world.

This gamer's review on Fez was:

- It is very atmospheric and a challenging game, at the start they puzzles are simple but they get progressively harder.
- Near the end of the game the puzzles are extremely hard to complete which majorly ups the concentration and thought processes needed to compete the challenge.

This gamer didn't enjoy Fez because:

- The puzzles get too difficult further on in the game to where trial and error doesn't work and so no progress could be made.
- The map design is useless because it makes it impossible to navigate to places you haven't progressed to yet, it takes way too long to find out where to go.
- Combined with not knowing where to go and the puzzles being way to difficult it creates a negative atmosphere for those who don't have enough energy and patience.

13 of 15 people (87%) found this review helpful

'b
297 products in account
36 reviews

Recommended
8.9 hrs on record

POSTED: 29 JULY

Positive:
- I love the pixel art and cartoony look of the game. It's simplistic, yet very beautiful. Oh, and the fez is just adorable.
- The cutscenes are amazingly trippy.
- Rooms are brilliantly designed. I really liked the music and glitch rooms.
- Great music, the OST is very relaxing.
- FEZ is unique because of its 2.5D puzzle platforming. The idea of rotating the screen 90 degrees to change the perspective of a primarily 2D game is just ingenious. It's simple, challenging, and fantastic.
- The map is very helpful in the sense that the boarder of the room will turn golden if you collected everything in said room.
- You will fail a lot but you respawn quickly.
- A very clever detail the developers added to Fez: when you are facing a door, you can see the room that the door leads to in the background.

Negative:
- I never finished FEZ in my first playthrough because I often felt like I did not have enough information to solve certain puzzles or figure out what to do next. If you look up a guide it becomes clear but I don't think people should have to resort to outside help. Maybe I missed something glaringly obvious but in my experience the game did not do a great job of providing information. I just tried to explore and collect as much as I could.
- This game is extremely cryptic. Many puzzles require too much time and thought to solve. Frustratingly so.

Conclusion:
Despite my criticism of FEZ, it still is an amazing game. It is just filled to the brim with charm and brilliant game design. I have never played a game like FEZ before and have yet to find anything similar.

EDIT: I managed to finish FEZ with the help of a walkthrough. I would have never been able to figure out some of the puzzles on my own. The platforming was excellent but the code puzzles were way too cryptic for me.

This review's ley points were:

Pros

- The graphics and design of the whole game is beautiful.
- Great soundtrack, it's very relaxing.
- Unique because of its 2D/3D aspect of a platform game.

- Very well designed world so that every object is detailed and each aspect of the world is not missed out.

Cons

- Not enough information to solve the puzzles which prevents progress in the game. You have to resort to looking up guides on the internet to solve some puzzles which defeats the aspect of a puzzle game.
- The game is too cryptic for a normal minded person to understand and so many puzzles require too much time and thought to solve which is very frustrating.

My own view on Fez

Yet again I don't own a copy of Fez so I watched other gamers playing the game to view the gameplay. The graphics and design of Fez is beautiful and very eye-catching, which I like, It gives off a nice gentle vibe and atmosphere. The aspect of making a 2.5D platform game is amazing, I've never seem another game like it which gives Fez a very unique gaming experience.

However, it is true that that puzzles further on are way too cryptic and I know I would have a very hard time to work out the puzzles. This in turn would frustrate me a lot because I wouldn't be able to progress unless I solved the immensely difficult puzzles. It I good to have difficult and challenging puzzle but not to this level.

Aspects of the fez that I will take into account

- The graphics and soundtrack mixed very well and creates an amazing atmosphere.
- The 2.5D aspect of the game is very unique and I like it a lot, however it wouldn't be very relevant to implement in Rune Raid.
- Make challenging puzzles that get progressively harder but don't make the puzzle too cryptic to the level of near impossibility to complete without 3rd party guides. Make sure I myself would be able to complete the puzzles without any other further knowledge about the level that Rune Raid doesn't give the player.

Explanation and justification of features of your product

After all that research into Fez, the World of Goo and Braid I've picked out aspects of each game that I very much liked and will implement in Rune Raid but I will also be disusing aspects of each game that I decided to not implement and the reasons why.

I have also thought of some other features that I will implement into Rune Raid which I've come up with myself or took inspiration from other games, not all video games.

Aspects I will be using

- There will be a login screen for Rune Raid to be able to login as a unique account or create a new account. This will be used to display the user's results on the leader board with the corresponding username. There will be a password for login to prevent people from accessing an account which is not theirs. The login will be used so the user can log back into the game using their previous account to try improve their result on the leader board.

- Some of the puzzles will involve physics, which I have taken inspiration from the World of Goo because I think it adds a different aspect to a game and makes the game fresh as I don't want Rune Raid's puzzles to be repetitive.
- The puzzles will get progressively harder the further on in the game. This is to let the player at first get used to the controls and how the game operates with the puzzles before they delve deeper into more difficult puzzles. It also prevents the player from getting bored as each level will be a different difficulty and forces the player to use their brain even more the further on in the game they are.
- The soundtrack will be relevant to Rune Raid's storyline and gameplay to make the game more immersive.
- The controls will be simple because the 3 games I've discussed all had simple controls which allowed for 100% focus on solving the puzzles and not any on trying to figure out the controls. Also, for puzzles there isn't many controls actually needed, all that's needed is movement from arrow keys/WASD, a cursor to interact with in-game objects and maybe the spacebar if jumping is involved. There will be no need for extra controls such shooting or throwing because I'm not adding that aspect into Rune Raid.
- There will be a timer that will time how long it takes the player to complete each level ad this will then be recorded on the leader boards with their username attached to it. This is to create a competitive atmosphere between the gamers that play the game to compete for 1St place on the leader board.
- On a certain level I will add the aspect of collecting certain objects whilst completing the level which will at the end reduce the time on the timer to get a higher place on the leader board. This creates an extra aspect to Rune Raid to make it even more competitive. This is inspired from Braid with collecting jigsaw pieces to 100% complete the game.
- There will be monster on certain levels which will have simple AI similar to that in Braid, which if they hit the player it will add extra time on the timer, making the player get a longer time score. This is to create a sense of fear or dread about hitting the monster because it will sabotage their attempt to get higher on the leader board.
- I will create a hint system which allows the player, if they so desire, to be given hints on how to solve the current puzzle. This will prevent them from getting frustrated, however if a hint is given it will add time to the timer so that the player can't abuse the hint system.
- There will be a storyline which I will create that will be related to the gameplay to add more depth to the game which is liked by many gamers in the community.
- The graphics and designs will be simple but eye-catching to comply with gamers that like a game with nice graphics and designs.
- There will be a constant theme throughout Rune Raid that will be linked to the storyline to add even more depth to the gameplay and make the story relevant.
- There will be multiple different answers to certain puzzles which gives the player some freedom to choose solve a puzzle different from another player. This is to make the puzzles more unique and more satisfying when solved.
- The player will be able to push objects around on certain levels, done by putting the character in contract with the object and moving in the desired direction until the player is happy with its position.

Aspects will not be adding

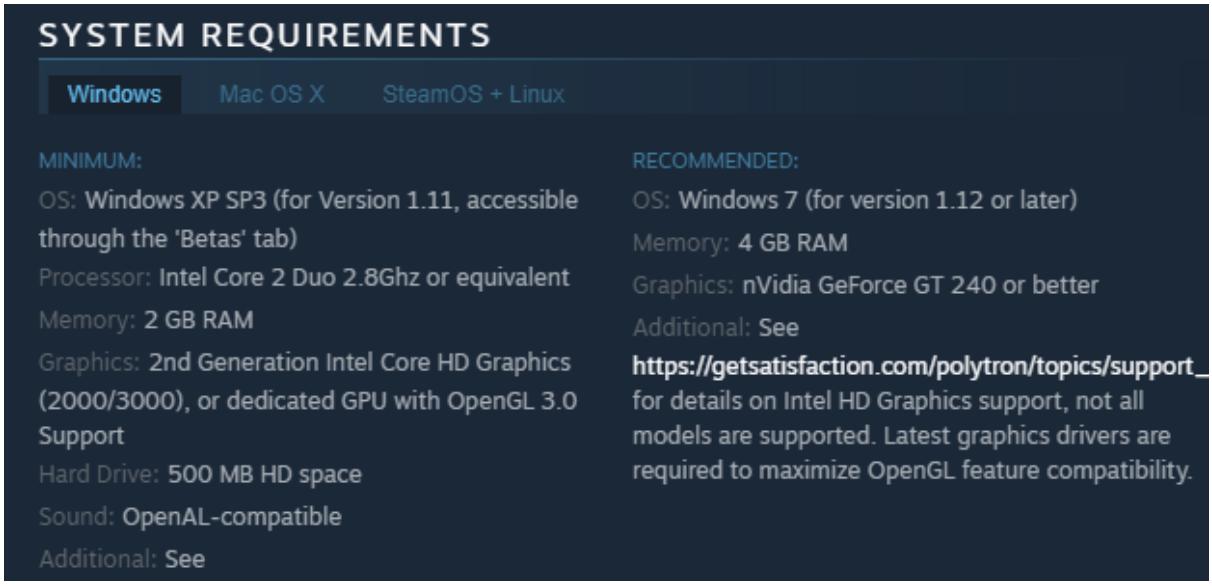
- I will not be making Rune Raid into a 2.5D platform game because I feel it wouldn't fit within my game because it's based inside a ruined temple and so getting a different view wouldn't give many more options. As well Rune Raid will be designed as a face down platform game and so 2.5D wouldn't work well.

- I will not be adding much humour into Rune Raid because the main genre is puzzle which I don't believe can be matched well with humour, unless it's like in the World of Goo which the player is building structures that can collapse. However, Rune Raid will not be similar to the World of Goo in the bridge building aspect and so that humour wouldn't be very well portrayed and would be out of sync with the rest of the game. Therefore, there will be little humour or even no humour depending on the players humour tastes.
- The puzzles will not be similar to Fez because they became too cryptic for the players to solve and they had to resort to guides to complete levels. I don't want to introduce this into Rune Raid because I want the player to be given everything they need to solve the puzzle within the game because that is what I believe a game should do.
- One of the reviews for Braid said that the game was too focused on puzzles and there were no other aspects to the game. I will be taking this into account by changing up each puzzle to be different but I will be sticking with every level being in a form of a puzzle which the player must solve to pass the next level or complete the game. This is because I believe a puzzle game should be heavily focused on puzzles as the genre suggests.

Identification of limitations and scope

As Rune Raid will be only on the PC platform there will be less limitations than on Smart Phone or console, such as it can run on a higher FPS or it can have more processing power to deal with more objects in the game. To get an idea of the minimum hardware and software needed to run Rune Raid I've looked into the minimum specifications needed to run similar games like Fez, the World of Goo and Braid.

Fez system requirements



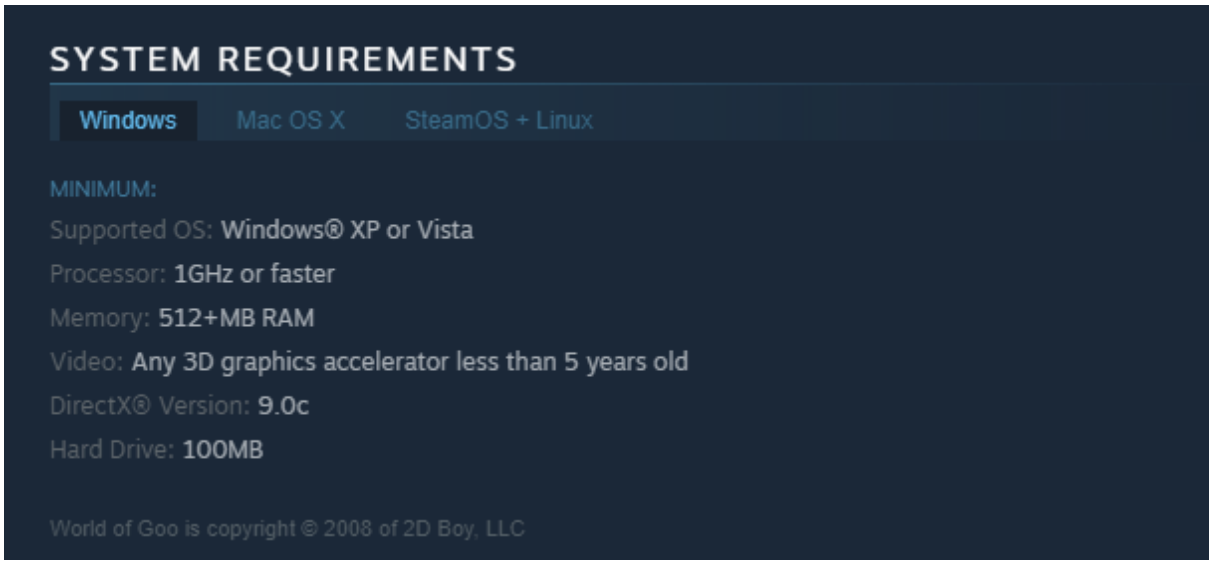
Fez is the largest game of the three in terms of size in bytes, however 500MB is still very small for PC games and so this is no problem. The Memory needed is also the largest of the three games at 2GB but still this is not a lot of RAM in terms of PC games and the majority of PC gamers will have desktops or laptops with 2GB or more of RAM. However, it is recommended to have 4GB of RAM to run Fez smoothly which some gamers may not have if their desktop/laptop is an old model, new models often have 4GB or more.

The operating system needed is no trouble because it can run even on windows XP which is very old and the majority of desktops and laptops these days are on Windows 7, 8 and 10 therefore, there is no problem. The graphics card needed is not too advanced but very old models of computer may not be able to run Fez due to the slightly advanced graphics. The processor needed is basic and the majority of computer models released in the last 5-6 years will have this processor or have a more advanced one.

Rune Raid will require less advanced specifications than Fez because there will be less hard drive space needed as the game will be less bytes due to lower graphics, less levels etc., the graphics will be less detailed than fez and so lower graphics card specifications, the processor needed will be less advanced because Rune Raid will need less processes per second than Fez and the Memory size needed will be less because less data will be stored in RAM as there will be a lower amount of data needed to run Rune Raid. The operating system required for Rune Raid will be the same as Fez's, this is because versions of Windows before Windows XP are irrelevant because they most likely won't be able to run any game on the market apart from extremely old games before 2001.

Therefore, Rune Raid system requirements will be lower than that of Fez's apart form the operating system, meaning that the majority of gamers will be able to play Rune Raid due to the majority of gamers owning better specifications on their desktop /laptop than Fez's requires.

World of Goo system requirements



These specifications are extremely basic and the majority of desktops or laptops will be able to run World of Goo. The only slight issue may be the video card needed because it needed to be at least a 3D graphics accelerator less than 5 years old. However, in term of years, 5 years is a very long time in the computer industry and many people change/upgrade their specifications every 3-4 years and so the majority of gamers will be able to run World of Goo.

Braid system requirements

These requirements are between the levels of World of Goo and Fez and so again the majority of gamers will be able to play Braid on their desktop or laptop because these requirements are very basic for computers. Old computers may not be able to run Braid mainly due to the 768 MB of RAM needed, old models of computer may only have enough space for 500MB of RAM which would mean the device cannot run Braid.

Rune Raid system requirements

In conclusion to the system requirements of Braid, Fez and World of Goo, Rune Raid will most likely be the same as World of Goo requirements. This is because Rune Raid will not have that many processes going on compared to Fez or Braid and the game size will be much smaller due to Rune Raid only consisting of 3 levels which is significantly less than the other 3 games. The graphics of Rune Raid will be similar to Braid but will be less detailed because I don't have enough time to spend making Rune Raid have beautiful graphics because it is more important to spend the majority of my time developing the algorithms/code to create the gameplay for the game.

My own system limitations

The device I'm using to develop Rune Raid is HP ENVY Phoenix 810-340nf, this is a high performance desktop therefore, there are very few limitations on what my desktop can't run. I own many games myself on this desktop and I can run each game on maximum setting without dropping below 60 FPS and so there will be no problem in developing and running Rune Raid, especially because there are many higher performance games that it can run smoothly. An example of a game that my desktop can run smoothly that is similar to Rune Raid is Shadow Tactics: Blades of the Shogun, this game has far higher graphics and processes than I plan to implement for Rune Raid. However, there would be performance issues for my desktop if I were to develop Rune Raid as the highest quality of graphics and processes on the gaming market that could be created but this is irrelevant because there are no games on the market of this calibre. This is because either they cannot be created yet due to technology limitations or that not enough devices would have the specifications to run such a game.

In conclusion there will be no limitations on developing and running Rune Raid due to my desktop specifications, apart from if I made Rune Raid have the greatest graphics on the gaming market, which is near impossible for me to do on my own.

Limitations due to the low system requirements

Because Rune Raid will not require high specifications, each level will have to be separate from the others in different modules of code and they will never all run at the same time. This will done by only loading one level at a time and the other levels will not be loaded until the player wants to load that level. This will save on processing power and memory because less operations will be running at one time. Also this will allow for a higher FPS **(frames per second)** because of the less processes being run, leading to a smoother performance of Rune Raid.

The graphics will not be able to be amazingly detailed because it will not be able to be run by simple graphic cards and so the graphics of Rune Raid will be simple but still eye-catching and pretty to still comply with many gamers want for a good looking game.

The game mechanics will have to be simple and not require too much processing power to adhere to the minimum system requirements I've set. Therefore, on each level there won't be many objects moving as moving objects take up a lot more process than stationary objects. There will only be a few monster on certain levels to keep the FPS and overall performance smooth, too many monster moving in Run Raid may crash or slow down the game due to limited processing power of some computers with the minimum requirements.

The animations in the game will be also very simple to keep performance smooth, meaning that the character and other moving objects will be limited to a certain movement speed and FPS but this won't greatly affect the gameplay because it won't prevent the puzzles form being solved, it will just reduce the amount of aspects going on at one time in the level.

However, my desktop has very high specifications so there are no restraints on the development of Rune Raid as a 2D platform game but I will have to make sure it can run on desktops/laptops with lower specifications. I can do this by using my laptop once I have built the game because my laptop has much lower specifications than my desktop.

Time constraints

As I have a limited amount of time to develop and document Rune Raid I will have to be realistic in what I can create in this time window, as I'm the only developer and I have a specific deadline. The biggest factor that I have to take into account for time constraints will be the amount of levels that I will have to develop in Rune Raid. This is because each level will be unique and so for each level I will have to create and use different scripts of code and algorithms which will take up a lot of time. Therefore, I've decided to create 3 levels for Rune Raid as I fell this is enough levels for the game to not feel empty and too short. I plan to create each level so it takes 5 to 10 minutes for the player to complete and so overall it should take the player 15 to 30 minutes to complete however, it can take much longer depending on how good the player is at solving puzzles.

As it will take a long time for to create every algorithm for Rune Raid, I plan to use Unities Engine's built in libraries, if they are available, that contain the algorithms that I will need to create my game. However, if there is not a pre-made algorithm that I need, I will have to create them myself but using Unity's algorithms when I can will drastically reduce the time it takes for to develop Rune Raid. This is also what the majority of developing companies do because it takes too much time to create code and algorithms from scratch and so they often use pre-built module code in which they can just change the variables to get their own desired

output.[15]Although, the company does need to have the right to use the code, otherwise it's illegal due to the Copyright, Designs and Patents Act of 1988.[16]

The design and graphics of Rune Raid will mainly consist of imported templates, this is due to the extensive amount of time it would take for me to create my own graphics and object deigns. As a result it isn't worth making my own because it is much more important to spend the majority of my limited amount of time on code development, level structure and documentation. However, I will make sure that the templates and imported images will be relatable to Rune Raid's environment and storyline to keep the gamer immersive.

The amount of monster on each level will be at a maximum of 4 because making each one will take time and I feel 4 monsters on a level will be sufficient enough to make the Rune Raid satisfying. As for the soundtrack, that will be imported because it would take way too much time to create a soundtrack of my own and so I plan to import one that is relevant to Rune Raid. I will try and make the storyline as interesting as possible however, I will not be able to make it extremely in-depth and a storyline that would rival top selling games. There are two reasons for this, one I'm not a brilliant storywriter and two I won't have enough time. Therefore, the Rune Raid's storyline will be relevant and pleasing but not to the standard of an amazing bestselling novel.

Money restraints

I'm not planning to spend any money on Rune Raid and so everything I create or use will be free therefore, there are many restraints for doing this. However, if I were to sell Rune Raid, either by selling it to a company or to individual gamers on the market I would have to upgrade my licence for Unity Engine. This is because Unity Engine only allows free usage for development when the project is not to be sold, if the game were to be sold on the market then the developer(s) would have to own a paid license of the Unity Engine.

Other than a Unity License, if I were to sell Rune Raid, I have no plan to purchase anything else. Therefore, I will either have to get pre-made template algorithms for free or make them myself. If I were to purchase algorithms from either Unity Engine or on the internet, then algorithm will most certainly be more efficient than one that I create myself, this is because it would have been developed overtime by an experienced developer(s). As a result of my own algorithms being less efficient will lower performance of Rune Raid compared to bought algorithms however, there will not be a sufficient enough drop in performance for the player to care about. And so making my own algorithms, whilst using some free templates will be sufficient enough to create Rune Raid to a good standard.

In conclusion, if I were to spend money on developing Rune Raid then the quality of the game will most certainly be of a higher standard. However, I believe I can still create Rune Raid to a good standard for free, that gamers will play and enjoy.

[15] https://en.wikipedia.org/wiki/Code_reuse - Date accessed 11/10/17
[16] https://www.legislation.gov.uk/ukpga/1988/48/contents - Date accessed 12/10/17

Identification and explanation of the computational methods

Abstraction and Visualisation

As Rune Raid's environment is an abandoned temple deep mountains of Bhutan, there are many things that need to be abstracted from the real world as they are irrelevant to Rune Raid.

Here are some examples of things that will be abstracted:

- The sky
- Vehicles
- Weather
- Rivers
- Temperature
- Buildings other than the Temple
- Humans other than Charles the player
- Above ground terrain e.g. hills
- Weapons
- Other 2 levels whilst playing the third.

Abstraction is the removal of objects that are deemed irrelevant when solving a problem, this is to ensure focus only on things that are needed to solve the desired problem.

Key objects needed in Rune Raid

- Charles, the player's character
- Walls
- Moveable objects
- Monsters
- Collectable runes
- Temple artefacts
- Timer
- Floor
- Hint text boxes
- Storyline text boxes

Other aspects

- Rune Raid's game logo
- There will be two font sets, one set for the login screen and then one for in-game text.
- There will be temple visuals to match the theme and location
- Each object will need its own look and design

Thinking ahead

Inputs for Rune Raid

- Mouse clicks on in-game objects
- Keyboard Keys WASD or arrow keys for movement
- Keyboard keys for username and password

Outputs for Rune Raid

- Sound through speakers/headphones/earphones.
- Visual movement through monitor.
- Game environment visuals through monitor.
- Text for storyline or hints trough monitor.
- Leader board of scores through monitor.

Thinking procedurally

The game will have 6 specific game states that cannot be run at the same time as any other, they are; Login screen, Level 1, Level 2, Level 3 and ending screen. The reason why these game states cannot be run at the same time is because if they were all being run at the same time it would severely decrease the FPS because there will be to many processes being done to keep each aspect running. Therefore, splitting them up into game states and having only one run at a given time will improve the FPS because it will cut down the number of processes needing to be done by 1/5.

Each game state will be different but will all have specific constant parameters passed to the modules when loaded up. This is to ensure that each level has been given the correct parameters needed to load the level correctly and no matter how many times the same game state is loaded, it will always be identical.

The Login game state will only consist of the login screen where the Rune Raid logo it presented and an option to create a new account or to login to a pre-existing account suing a username and password.

Each of the three level game states will be different as each level will have a different design and puzzle involved. However, there will be some constant objects such as the timer and Charles the player character that will be loaded into all the 3 levels in the same way.

The ending game state will consist of some text congratulating the player on completion of the game and also some ending to the storyline, with the Rune Raid logo shown as well

The order of game states that will needed to be loaded in will be:

1. Login screen – Loaded when the game is first loaded up.
2. Level 1- Loaded when the player's account has been created or login into.
3. Level 2 – Loaded when level 1 is completed
4. Level 3 – Loaded when level 2 is completed.
5. Ending screen – Loaded when level 3 is completed.

There will be some reusable modules that I will implement into Rune Raid multiple times such as;

- Player movement
- Object collision
- Physics
- Monster AI
- Timer

Re-using coded modules is highly efficient in code development because it saves time as duplicate aspects are not needed to be re-written multiple times, they are only needed to be written once and then just called upon when needed.

Thinking logically

Throughout the game there will be multiple critical if statements that will be needed to ensure the game runs properly.

A few examples;

- Collision checking
- Level complete checking
- Movement checking for animations – IF movement > 0 then play animation of direction of movement.
- Time score checking to ensure the leader board scores are in the correct order of time score (quickest time is 1st place, slowest time is last position)

Without these IF statements then Rune Raid wouldn't be able to run properly.

As well as critical if statements there will also be critical loops which will be needed to also ensure the game runs properly.

A few examples;

- Multiple sprite character and monster frames being run in a loop while movement – this is so animations can be made, e.g. animation of moving in desired direction until movement == 0.
- Movement – E.g. IF left arrow key or A key pressed then move -1 in X axis until key released.

Without these Loops then some aspects of Rune Raid wouldn't work because iteration of certain lines of code is needed to work properly. Such as if there was no loop for movement then pressing and holding an arrow key would only move the player 1 position and to move 10 positions then they would need to press the key 10 times. With loops they would only need to hold down the key until they are at their desired position.

Thinking concurrently

Throughout the game there will be multiple aspects of the game running at the same which will needed to be taken into account and tested to ensure that they don't clash, causing Rune Raid to crash or glitch.

Examples are;

- Background music and sound effects.
- Player movement and animation.
- Object movement and player movement.
- Multiple monster moving at the same time.

Identification of hardware requirements

For Rune Raid's development I will be using my own desktop, the model is HP ENVY Phoenix 810-340nf. This desktop is fully capable to develop and run Rune Raid as it's a platform game and my specifications are of high standard and designed for high performance.

The amount of memory needed to run Rune Raid will be 500MB of RAM, this is because I've researched multiple games similar to Rune Raid and as I've discussed before I came to the conclusion that Rune Raid only needs 500MB to run smoothly. Having 500MB or more of RAM is extremely common for computers and so there will be no problem surrounding memory capacity issues. The amount of disk space needed to store Rune Raid will be between 50Mb and 100MB. This is because of the reasons I've discussed before, similar games to Rune Raid need between 100MB and 500MB and I believe that Rune Raid will be less or no larger than the minimum 100MB because it will only consist of 3 levels which is less than the games I researched. 100MB or less is a very small amount of disk space and so the majority of computers will have enough allocated free space to store Rune Raid.

A processor of 1GHz or faster, this is because the similar games I've researched all had either required higher than 1GHz or the same. Therefore, as I'm developing Rune Raid to be no more/less advanced than these other games the requirements for a devices processor will be no more than 1GHz.

For graphics cards it will be any 3D graphics generator less than 5 years old. This is because Rune Raid will not have greater graphics than World of Goo and so it is realistic to pick the same graphics requirement.

100MB of hard drive space will be required because the smallest game that is similar to Rune Raid that I've researched is World of Goo and so Rune Raid will require no larger than World of Goo in terms of bytes.

As I've discussed in depth before about the reasons for Rune Raid's software and hardware requirements I will not repeat myself again. All the reasons for the requirements are on pages 22 to 24 however, I will list here the requirements of Rune Raid in a clear format.

Processor	1GHz or Faster
Memory	512MB
Graphics	Any 3D graphics generator less than 5 years old
Hard Drive	100MB

Identification of software requirements

Operating System	Windows XP , Vista or later versions
DirectX version	Direct3D11

The reason for this specific operating system requirement is because the three other games that I researched could all run on Windows XP / Vista or later versions and so Rune Raid will also be the same as its no more advanced that these 3 other games. As for the DirectX, this is the API needed to run Rune Raid as I've set my Unity Engine to use DirectX API version Direct3D11.

Another similar API is OpenGL, it's a graphics-oriented API suite introduced in 1992, that's open source and in continuous development by the Khronos Group technology consortium. However, DirectX is native to Windows operating system and so I chose DirectX instead of OpenGL.[17]

DirectX is an API that easily determines the hardware capabilities of a computer and then sets the program parameters to match. This allows multimedia software programs to run on any Windows-based computer with DirectX compatible hardware and drivers and ensures that the multimedia programs take full advantage of high-performance hardware. Its takes are related to 2D and 3D vector graphics, rendering video and playing audio on the Windows platform.[18]

Specifically, the Microsoft Direct3D API, which Unity Engine uses, provides an interface to the 3-D rendering functions built into most new video adapters. Direct3D is a low-level 3-D API that provides a device-independent way for software programs to communicate with accelerator hardware efficiently and powerfully. Direct3D includes support for specialized CPU instructions sets, providing additional acceleration on newer computers.[19]

Identification of utilities/libraries

As I am developing Rune Raid in Unity I will be using many of Unities own libraries such as lighting, camera position, collision, physics etc. These are all built in libraries that are free to use in the Unity Engine and are very efficient to use instead of making my own algorithms that would take a lot of time and be much less efficient. Each of these libraries will provide an important aspect of Rune Raid and help me to develop it into a fully functional game.

I need to have a database that will store the time scores of each player that plays Unity, without a database I will not be able to make Rune Raid a competitive game.

However, Unity Engine doesn't provide a database system and so I will have to import a pre-made database system or make my own to store each player's scores on Rune Raid's leader boards. I've done some research and decided that I will use SQLite to create my own database that will store the data I need to form a leader board for Rune Raid.

I will also need to store username and passwords for the player to login to their account to have their times recorded into the leader board database and so I will also have to create a database for this.

Pre-made database https://forum.unity.com/threads/iboxdb-lightweight-embedded-database.217097/

[17] http://www.techradar.com/news/gaming/directx-12-what-is-it-and-why-it-matters-to-pc-gamers-1318636 - Date accessed 13/10/17
[18] Ibid
[19] https://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/what_is_directx.mspx?mfr=true – Date accessed 13/10/17

Success criteria

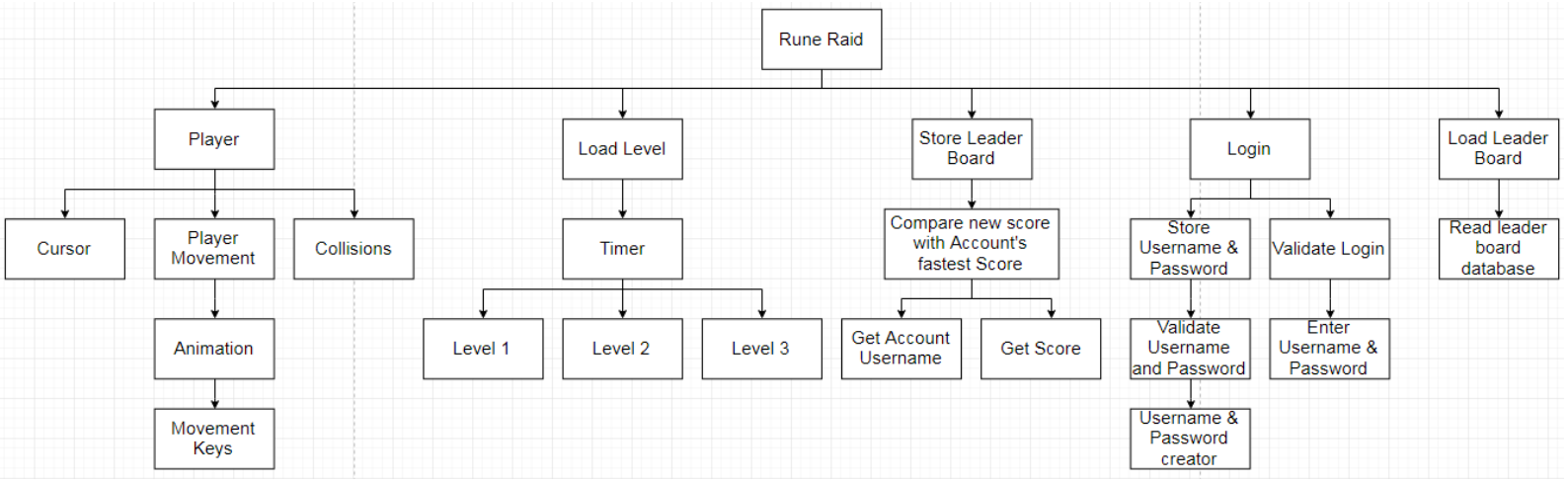
Success Criteria	Reason	How's it going to be measured?
Fully functional leader board that contains each player's best individual time scores for each level.	This to make Rune Raid a competitive game as each player can compete for 1st position on the leader board. Without a leader board to record time scores there will be no way to record each player's time scores accurately and be able to compare with other player's times.	When a player logs into Rune Raid and completes a level by solving the puzzle, the time it took them to complete the puzzle will be recorded and stored on an accessible leader board. Each time score should be ordered quickest to slowest**.** The player's best score will only be stored for each level on the leader boards, their slower times will be appended by their faster time or not added.
Rune Raid must have three unique levels for each player to play on.	To make Rune raid more interesting as it won't be repetitive and also having 3 levels is to have enough levels for the game to not feel empty and too short.	There will be three levels in which the player can play on and fully complete. Each level will have a unique puzzle not similar to the other levels.
A user friendly login interface.	To make sure each player has a unique username to be displayed next to their time score for each level. It will also allow for players to come back and log back into Rune Raid to improve their time scores to get a better position on the leader board. Having a password will protect each player's scores so no other player can get into their account.	Each player will be able to create a unique username with its own password. The user can log back into their account to play Rune Raid again with the correct details. Each player's username will be recorded next to their own scores on the leader board.
Score Timer.	To record how long it takes the player to complete each level, to then be stored on the leader board.	There will be a timer that acts like a stopwatch, each level it will start from 0:00 and then run until the level is completed by the player.
Documentation of error trapping.	This is to prevent invalid data from being stored and all valid data is stored correctly. This is to ensure Rune Raid operates correctly.	I will code each of Rune Raid's modules that require inputs to restrict them to certain valid inputs. Invalid inputs will be coded so the game doesn't crash but either asks for another input or ignores the input. To make

		sure it works I will input extreme, normal and invalid data to see if the game still works as desired.
Documentation of testing.	This is insure that Rune Raid is bug free and doesn't give undesired outputs or create in-game glitches. A gamer's worst nightmare is when games are full of bugs because they can't play the game properly.	Each aspect of Rune Raid will be tested to make sure each input gives the correct output. Also I will test Rune Raid in-game making sure each object is made and coded correctly e.g. make sure player's character can't go through walls or be invincible.
Documentation of errors whilst in development or testing.	To make sure the errors I come across whilst developing Rune Raid are documented and solved correctly. This can also help in further development of Rune Raid if I come across a similar error because I can easily fix it as I've already come across it before and documented my previous solution.	I will have screenshots of each error that occurs explaining the problem, then proceed to explain my solution to the problem. After the problem is fixed I will screenshot the solution implemented and the correct outputs being outputted.
The player will be able to fully interact with all of Rune Raid's aspects.	To make sure the player can fully use each aspect of the game to its full potential to create an enjoyable game.	Each aspect of Rune Raid will be used and proved to be working to its pull potential by using screenshots of in-game or in line code.
All algorithms will be as efficient as I can possibly make them.	This is to make sure Rune Raid can run smoothly at a high FPS. In efficient algorithms will increase the processes needed and so lower the FPS.	I will use Big O notation to identify any inefficient algorithms and try to make them more efficient.
All decisions made will be shown and justified.	To make sure each decision is not made on a whim with no reason, to prevent the use of unneeded code or objects.	Every aspect of Rune Raid that I implement will be explained and justified why I chose to implement it. Also, if relevant giving other possible solutions that I didn't implement and why not.

Rune Raid's puzzle must be challenging for the player.	This is because after researching similar games, the majority of the reviews were about the great puzzle gamers were the aspect of challenging puzzles.	I will play the game fully myself after completion and also get some second opinions from my friends who are also into games to give me their opinion of the level of challenge each puzzle presents.
All the code is split up into specific functions within modules.	This is to make the code easier to read and understand, makes each module re-usable and so I can save time by not re-writing the same code and it also frees up memory space because instead of all the variables being global they will be mainly local and so not use up memory when the module is not being executed. This will therefore improve the performance of the game as less memory is needed to be used up.	I will make sure to decompose all of my code into specific functions. I will do this by going through each aspect of my code and making sure that all the code specific to an aspect of Rune Raid is group into its own individual function. I will also make sure to re-use modules when possible, checking I've not repeated code unnecessarily.

Design

Structure of the solution



The hierarchy chart above is the decomposition of my project, Rune Raid and will allow me to split up my computational problem into smaller modules. Doing this it has many benefits; it splits up a large problem into smaller more understandable problems that can be solved individually, allows each module to be reused multiple times and each module can be tested separately and so it is easier to troubleshoot as an error would be confined to a specific module and easy to locate.

Decomposition of the project

Explanation of each module

Movement keys: The keys used for movement will be WASD or four arrow keys, each time one of the keys are pressed the player sprite will move along the specific axis by 1 every specific time the key is pressed for. Example, if the W key is pressed for 2 seconds the player sprite will move along the Y axis for 2 seconds which will be +4 in the Y axis if the specific time pressed is set to, every 0.5 seconds the axis moves by 1. For this I will be using Unity Engine's built-in dictionary of functions for movement as it does exactly what I want the function to do, it even allows me to modify the keys used. All I need to do is put the movement function within a line of code that will set the other variables such as the specific axis and speed to go along with the movement function. Without this function the user wouldn't be able to move the in-game sprite to play the game.

Animation: This will be dependent on the Movement Key function because once the Movement Key function runs the Animation function needs to run along in parallel, this is to create the aspect of the player sprite moving along the axis with a walking animation. This will work by the animation function detecting when the player sprite is moving and the direction it's moving in. Example, if the player sprite has started to move in the negative x axis (moving left) then the Animation function will detect the movement and the direction which will be left. As there is a movement in the negative x direction then the left player sprite animation will start to run. This will create the visual of the player sprite walking left smoothly. There is also a built-in function within Unity Engine that can detect movement and the direction it is in. This is perfect to be used and so I will link this function to trigger the player sprite walking animations when movement is detected. Without this the player sprite would move along the axis but the player sprite would be a static visual.

Player movement: This is just the combination of the Movement key function and the Animation function, they will both be run in parallel but the Movement key function must be run first for the Animation function to be executed (the Animation function can't run unless the Movement key is run first). This combination of the two functions will provide the visual of the player sprite walking smoothly in the desired direction of the key pressed. Without this the player movement and animation would be out of sync or even not play together at all, this could create the visual of the player sprite moving with animation but there would be no movement along an axis or the player sprite could be moving along an axis but the animation wouldn't play.

Collision: This module is to set the boundaries of the player movement, in other words to not allow the player to move through objects that I don't want them to move through e.g. a wall boundary or a solid object. This will be done within the Unity Engine using their built-in collision function. This is very simple to use, when making each object in the game there is an option to set what the object can collide with and what it doesn't collide with. Example, a ball can be create and set to collide with physical objects, the floor object is not a physical object and so the ball can move through the floor (technically over it) however, a wall that is set to a physical object will collide with the ball and so the ball will not be able to move through the wall. If this function wasn't implemented then the player or objects inside the game would have no restrictions and could move outside the map zone, move off screen or just move through walls to complete the puzzle incorrectly.

Cursor: This function will be used to detect a cursor click on certain aspects of Rune Raid, for example when the cursor clicks inside the username box on the login screen then the username bar will be activated, once there is another cursor click located outside of the username box then the username box will be deactivated. There will only be a few areas in Rune Raid where a cursor click is valid such as the login screen or clicking next on the next box. Without this the player wouldn't be able select certain things within Rune Raid such as move on the story in the text box.

Player: This is the collection of all the functions specific to the user whilst in game. Any other function outside this module will not be directly linked to the player's actions but may be indirectly linked such as the leader board.

Level 1: This is the function that contains the first level of Rune Raid, this will mainly consist of visuals from the Unity Engine but it will still contain some code implemented such as the AI for the monsters. This will all be coded and linked within the level on Unity Engine as so it can be concatenated as one function. If this function was not implemented then level 1 would never load when the player logs into Rune Raid because there would be no level to load.

Level 2: This is very similar to the level 1 function but the level design will be different in the Unity Engine, the level will still function in the same way as the other levels but will have some new aspects involved such as objects involved with physics, which will be coded within Unity Engine and not as a separate script. Therefore, there will be little coding actually involved as its all built-in within the Unity Engine which can be easily used and manipulated. If this function was not implemented then level 1 would never load when the player complete the level 1 puzzle and wishes to proceed to level 2 because there would be no level to load.

Level 3: Again this function will be very similar with the other 2 levels but will have slight differences because of the different puzzles they contain, it will still have the majority of its aspects built within the Unity Engine such as visual designs or Unity's own functions from their dictionary but will contain a few other scripts that will be linked with the level. If this function was not implemented then level 1 would never load when the player complete the level 1 puzzle and wishes to proceed to level 3 because there would be no level to load.

Timer: This function is the timer that will be implemented within all three levels of Rune Raid, it will be located at the top of the screen and when each level is loaded the timer resets to 0:00 and then starts until the player completes the level and proceeds to the next level (or proceeds to the end screen). When this happens the time currently on the timer will be stored for later use. It will always be displayed whilst the player is playing the level as a live timer and can be used to tell the player how well they are doing. For example, if the player is trying to beat the top score of 6:36 then the timer can give an indication of how they are doing against this time, whether they can beat the time or whether they are too slow and need to start again. Without this timer,

there would be no way to tell how long a player took to complete a level, to input into the leader board and also if the timer wasn't displayed live in-game then the player would have no indication of how long they have been playing the level for.

Load Level: This will be the function that will load one of the three specific levels and the timer together to form the completed level. It will only execute when the player logs into the game with their account or they complete a level and wish to continue to the next. Example, when the player has completed level 2 and chose to enter the door to level 3 then when the player enters the door it will execute the level loader for level 3 and the game will transition to the third level. This is a relatively easy function because it can be done in Unity Engine as Unity can load levels. Without this function each level would not load because there would be no way to transition between login screen, level1, level 2 and level 3.

Get Account Username: This function will be used to get the player's Username to be implemented later in the leader board. This is so that the player's time scores for each level can be linked to their own specific Username.

Get Score: This module will be used to get the player's time score for each of the levels they have competed, this so that it can be compared later with the previous scores and if quicker then added to the leader board.

Compare New Score with Account's Fastest Score: This function will be used to compare the player's new time score with their accounts fastest score however, if the players score is slower than their previous then it is not stored and ordered on the leader board, this is because the leader board will only contain the fastest scores of a player. Example, a player scored a time of 5:49 on level 1 but their fastest time for level 1 is 5:11 and so their new time is compared with their fastest score and because their new time is slower it will not be stored as 5:49 > 5:11. However, if the player has never played the level or has a faster new time then it will be stored.

Store Leader Board: This function is where after the new time score is validated to be the fastest it is concatenated with the player's Username and added to the Leader Board in order. If the Username already has a time score on that certain level then it will be overwritten and re-ordered. The order of the Leader Board will be fastest time in 1st position and the slowest time score will be in last place. Therefore, the new time score with a Username that needs to be added will be compared with the current time scores and placed in the correct place. Example, Gemma 4:50 will be compared with the Leader Board list and placed between the next slowest score which may be 4:45 and the next fastest score which may be 4:55 (if first place there will be no next faster). If this function was not implemented then when the accounts with scores are stored in the leader board there would be no score ladder as it would all be random and hard to understand. When it's ordered properly it will be easy to understand which accounts hold the quickest times. Also, if old scores were not over-written then there would be redundant data in the database because there would be multiple scores for Usernames that are slower than their fastest time.

Username and Password Creator: This module will be used to create a new account for a player with a specific Username but any Password. It will be placed within the login screen at the start of Rune Raid under the open to create a new account. The user will input their desired Username and Password into the reserved text boxes. This needs to be a function because otherwise the users wouldn't be able to make an account in the first place and so the leader board would only consist of time scores but no link to a Username. Therefore, there would be no way to tell which player is in 1st etc.

Validate Username and Password: If the username is already taken then it will tell the user to input a different Username until a new unique username is inputted. Also the username and password will have limitations on the characters that can be used because some certain symbols or characters will not work in Rune Raid. As well there will be a character minimum and maximum limit to usernames (6 to 16 characters) and passwords and must contain a number to make sure the passwords are not extremely weak. Therefore, if the username contains an invalid character then it will tell the user the problem and then ask them to input a different username with valid characters and if a password is less than 8 or more than 20 characters or doesn't contain a number then it will tell the user the problem and tell them to input a new password. If this function

was not added then password for users could be extremely weak and so easy to crack, usernames may not be unique and so cause confusion on the leader board as well as the login because it would not tell which user was logging in a username could have multiple passwords and usernames and password could contain characters that may break the functions. Therefore, it is important to validate both usernames and passwords.

Enter Username and Password: This will be used for when a user already has an account and wishes to log back into their account. Without this function users wouldn't be able to log back into their previously made accounts and therefore, they would be forced to create a new account each time they close and re-open Rune Raid, this would be incredibly inefficient and very annoying for players wanting to play on their account again to improve their score.

Store and Username and Password: This function will store the valid Username and Password for each unique account into a database to be used later for either login or the leader board.

Login: After the Username and Password has been validated, this will be the function that will log the user into their account that they either just made or have previously made by setting all parameters to their unique account. It will then load the level 1 function. This function is essential because without it, logging into or creating an account would be useless because level 1 would never load linked to an account.

Read Leader Board Database: This function will be used to open and read the database that contains all the data on each player's fastest time for each level in the form of a leader board. This function is needed because otherwise the leader board database wouldn't be able to be loaded as it wouldn't be opened in the first place.

Load Leader Board: After the Read leader board database function is executed, load leader board will execute which will then use the read database and display it in a GUI for all players to see each player's fastest score for each level. This will allow for competiveness in Rune Raid as player's can look up and compare scores with other players. Without this function there would be no other way to show the leader board as Unity doesn't support databases and so it much be a separate imported script.

Justification of my approach to the solution

The way that I will be approaching this computation problem will be to utilise the hierarchy chart to its full advantage, this way I can slowly build up Rune Raid function by function, testing each function during its production and at its final stage to make sure that all the that all the code that I implement will be relevant and error proof.

The way I will do this will be to code each function individually in the correct order such as, coding the account and score modules before the leader board functions, this is because I can't test the leader board functions without account and score modules already implemented.

The benefits of this approach to this solution are:

- Easier to manage.
- Easier to troubleshoot.
- Can test functions individually which is much more efficient and easier to isolate problems in the code.
- Modules can be re-used which saves time.
- Easier to build a large project as functions can be added together one by one once they are completed and tested.
- Modules are easy to modify as they are independent from other modules.

The majority of my development will take place within the Unity Engine, this is because Rune Raid is a game and the majority of game developer's time today is spent within Engines like the Unity, Unreal

Engine is another, this is because it's much more simple, as the majority of game development foundations are on these engines that are easy to understand and implement.

However, there are some aspects that are not within the Unity Engine, an example is that Unity Engine doesn't contain an in-built database but it does allow a database structure to be imported and linked within Unity. Therefore, I will be creating the database for the leader board separately from Unity and when it's completed I will just import it within Unity to link it to Rune Raid.

I will be developing the levels and all the functions that are needed to run each level first before anything else, such as the leader board, this is so I can focus solely on the foundation of Rune Raid to make sure I've got the correct structure of the game and then later when it's all working, implement the addition aspects to make up the full game. This way I will not get confused and frustrated while developing Rune Raid because the function that I'm coding will have all the parameter functions already made, meaning I can fully test the function with its real parameters.

Algorithms in Pseudocode

Player Movement

- (Note that this is just movement key and Animation functions added together- I'm not doing separate Pseudocode for them.)

```
FUNCTION playerMovement (Movement):
    IF W key pressed:
        WHILE n = True
            RETURN Movement = 0.1
            EXECUTE Backward animation
            Move Player +1 in Y axis
            IF W key pressed:
                n = True
            ELSE:
                n = False
                RETURN Movement = 0.0
                EXECUTE Idle Backward animation
        END WHILE
    ELIF A key pressed:
        WHILE a = True
            RETURN Movement = 0.1
            EXECUTE Left animation
            Move Player –1 in X axis
            IF A key pressed:
                a = True
            ELSE:
                a = False
                RETURN Movement = 0.0
                EXECUTE Idle Left animation
        END WHILE
                b = False
                RETURN Movement = 0.0
                EXECUTE Idle Forward animation
        END WHILE
```

Continuation of Pseudocode

```
    ELIF S key pressed:
        WHILE b = True
            RETURN Movement = 0.1
            EXECUTE Forward animation
            Move Player –1 in Y axis
            IF S key pressed:
                b = True
            ELSE:
                b = False
                RETURN Movement = 0.0
                EXECUTE Idle Forward animation

    ELIF D key pressed
        WHILE c = True
            RETURN Movement = 0.1
            EXECUTE Right animation
            Move Player +1 in X axis
            IF D key pressed:
                c = True
            ELSE:
                c = False
                RETURN Movement = 0.0
                EXECUTE Idle Right animation
        END WHILE
    ELSE:
        Ignore
    END IF
END FUNCTION
```

Collision

```
FUNCTION Collision (physicalObjects, playerSprite, Movement):
    IF playerSprite IN CONTACT WITH physicalObjetcs == TRUE THEN:
        IF Right animation == TRUE
            RETURN Movement = 0.0
            EXECUTE Idle Right animation
        ELIF Left animation == TRUE
            RETURN Movement = 0.0
            EXECUTE Idle Left animation
        ELIF Forward animation == TRUE
            RETURN Movement = 0.0
            EXECUTE Idle Forward animation
        ELIF Backward animation == TRUE
            RETURN Movement = 0.0
            EXECUTE Idle Backward animation
        ELSE:
            Movement = 0.0
            RETURN Movement
        END IF
    ELSE:
        RETRUN Movement
    END IF
```

Cursor

```
FUNCTION Cursor (Mouse, hintButton, storyButton, userBox, passBox, loginBox):
    IF hintButton CLICKED THEN:
        EXECUTE nextHint
    ELIF storyButton CLICKED THEN:
        EXECUTE nextStory
    ELIF userBox CLICKED THEN:
        EXECUTE enterUser
    ELIF passBox CLICKED THEN:
        EXECUTE enterPass
    ELIF loginBox CLICKED THEN:
        EXECUTE Login
    ELSE:
        Ignore
    END IF
END FUNCTION
```

Level Functions

These cannot be written in Pseudocode because they are built-in objects within Unity which are just loaded when called.

Timer

```
FUNCTION Timer (Level1, Level2, Level3, Username):
n = 1
Time = [dc : ba]
        WHILE n == 1:
                EVERY SECOND a = a + 1
                IF a == 10 THEN
                        b = b + 1
                ELIF b == 10 THEN
                        c = c + 1
                ELIF c == 10 THEN
                        d = d +1
                ELIF Level 1 or Level 2 or Level 3 or Login EXECUTED THEN
                        n = 0
                 DISPLAY Time
                END IF
        IF Level1 == TRUE THEN
                STORE Time + Username in Level1 File
        ELIF Level2 == TRUE THEN
                STORE Time + Username in Level2 File
        ELSE:
                STORE Time + Username in Level3 File
        END IF
END FUNCTION
```

Load Level

```
FUNCTION loadLevel (Timer, Level1, Level2, Level3):
    IF Level1 EXECUTED THEN:
        LOAD Level1 AND Timer
    ELIF Level2 EXECUTED THEN:
        LOAD Level2 AND Timer
    ELIF Level3 EXECUTED THEN:
        LOAD Level3 AND Timer
    ELSE:
        Ignore
    END IF
END FUNCTION
```

Compare new score with accounts fastest score

This Pseudocode function will be combined with the 2 functions below in the hierarchy chart (Get Username and Get Score), this is because they will all be coded together in the same function because to get a variable and not use it is pointless in its own function because the compare function will need to get the variable again anyway.

```
FUNCTION Compare (Time, fastestTimeL1, fastestTimeL2, fastestTime3):
    IF Time < fastestTimeL1THEN
        fastestTimeL1 = Time
        RETURN fastestTimeL1
        storeLeaderboard()
    ELIF Time < fastestTimeL2THEN
        fastestTimeL2 = Time
        RETURN fastestTimeL2
        storeLeaderboard()
    ELIF Time < fastestTimeL3 THEN
        fastestTimeL3 = Time
        RETURN fastestTimeL3
        storeLeaderboard()
    ELSE:
        Ignore
    END IF
END FUCNTION
```

Store Leader Board

```
FUNCTION storeLeaderboard (currentUser, fastestTime1, FastestTimeL2, fastestTime3):
        IF fastestTime1 == TRUE (If the new fastest time has come from level 1)
                STORE currentUser + fastestTime1 in Level 1 Leader Board Section
        IF fastestTime2 == TRUE (If the new fastest time has come from level 2)
                STORE currentUser + fastestTime2 in Level 2 Leader Board Section
        ELSE:
                STORE currentUser + fastestTime3 in Level 3 Leader Board Section
        END IF
END FUCNTION
```

Username and Password Creator

```
FUNCTION userPassCreator ():
        Username = input ("Enter Username")
        Password = input ("Enter Password")
        RETURN Username
        RETURN Password
        Validate ()
END FUCNTION
```

Validate Username and Password

```
FUNCTION Validate (Username, Password):
        IF Len(Username) > 16 OR <= 6 THEN
                OUTPUT ("Please enter a username of 6 to 16 characters")
                userPassCreator()
        ELIF Username == Username in stored list THEN
                OUTPUT ("This username has already been taken, please enter another username")
                userPassCreator ()
        ELIF Username CONTAINS any character other than alphabet OR integer OR _ OR – OR blank THEN
                OUTPUT ("Invalid username, please enter another username")
                userPassCreator ()
        ELIF Password > 20 OR <= 6 THEN
```

```
            OUTPUT (“Please enter a password of 6 to 20 characters”)
            userPassCreator ()

        ELIF Password CONTAINS any character other than alphabet OR integer OR _ OR – OR @ OR . OR THEN
            OUTPUT (“Invalid password, please enter another password”)
            userPassCreator ()
        ELIF Password DOESN’T CONTAIN an integer THEN
            OUTPUT (“Please enter a password with at least 1 integer”)
            userPassCreator ()
        ELSE:
            storeUserPass()
        END IF
END FUNCTION
```

Store Username and Password

```
FUNCTION storeUserPass (Username, Password, pointerID)
        AccountID = currentID + 1
        pointerID = AccountID
        RETURN pointerID
        STORE Account in Account database
        STORE Username AND Password within AccountID column
END FUCNTION
```

Validate Login

```
FUNCTION validateLogin (Username, Password, AccountID):
        Open Account Database
        IF ANY AccountID CONTAINS Username AND Password THEN
            Login()
        ELSE:
            OUTPUT (“Incorrect Username or Password, Please try again”)
        END IF
END FUNCTION
```

Enter Username and Password

```
FUNCTION userPassLogin ():
        Username = input (“Enter Username”)
        Password = input (“Enter Password”)
        RETURN Username
        RETURN Password
        validateLogin ()
END FUCNTION
```

Read Leader Board Database

```
FUNCTION readLB ():
        OPEN File leaderBoard
        READ File leaderBoard
        loadLB()
END FUNTION
```

Login

```
FUNCTION Login (Username):
        currentUser = Username
        GET fastestTimeL1 for currentUser in Account database
        GET fastestTimeL2 for currentUser in Account database
        GET fastestTimeL3 for currentUser in Account database
        RETURN fastestTimeL1
        RETURN fastestTimeL2
        RETURN fastestTimeL3
END FUNCTION
```

Load Leader Board

FUNCTION loadLB ():

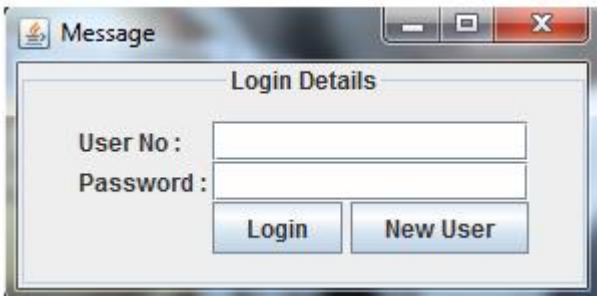
DISPLAY leaderBoard in Format (Displaying the leader board in a desired format e.g. different areas for each level)

END FUNCTION

Usability Features

Explanation and justification of the design of the user interface

Login UI



[20]

This is a simple design of the way my UI login will look like, as the key features will be:

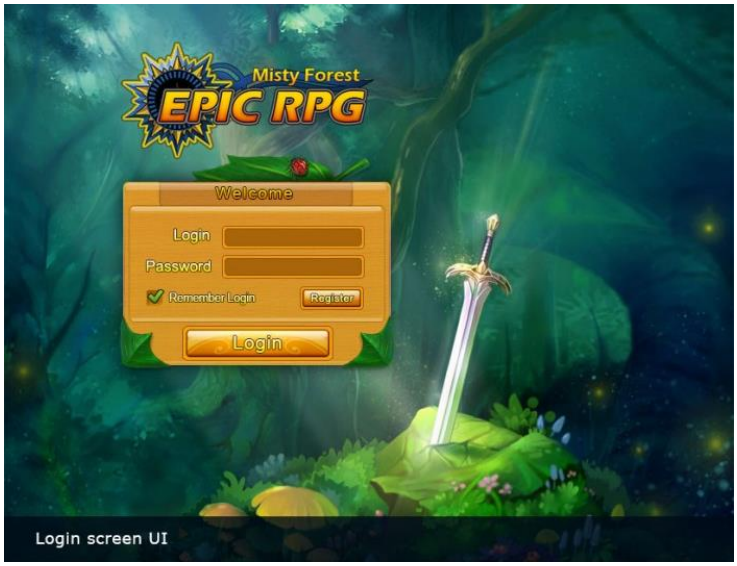
- Login interface screen separate from the levels of Rune Raid
- A Box for entering a username and password
- A login button for users who already own an account for Rune Raid
- A new user login button which will load an identical UI but the function behind it will be different because their Username and Password will need to be validated and stored, rather than just checking if the Username and Password match a stored account.

However, as this is not a very appealing UI I will be making Rune Raid's much more eye catching and will implement a logo and name of the game "Rune Raid" within the UI but the foundation of Rune Raid's UI for login will be the same.

[20] https://stackoverflow.com/questions/14979647/background-image-hides-all-gui-design-components - Image of GUI – Date Accessed 30/11/17

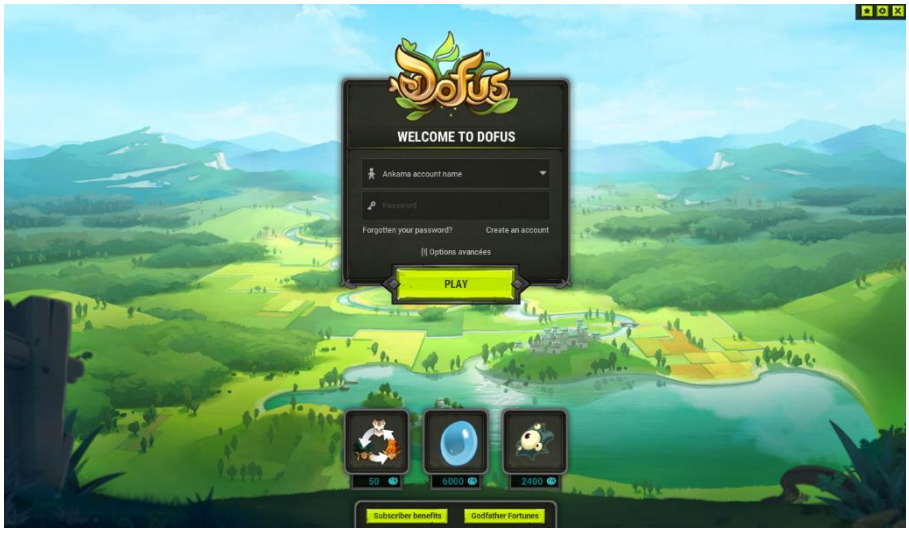
Similar login designs

21



22

23

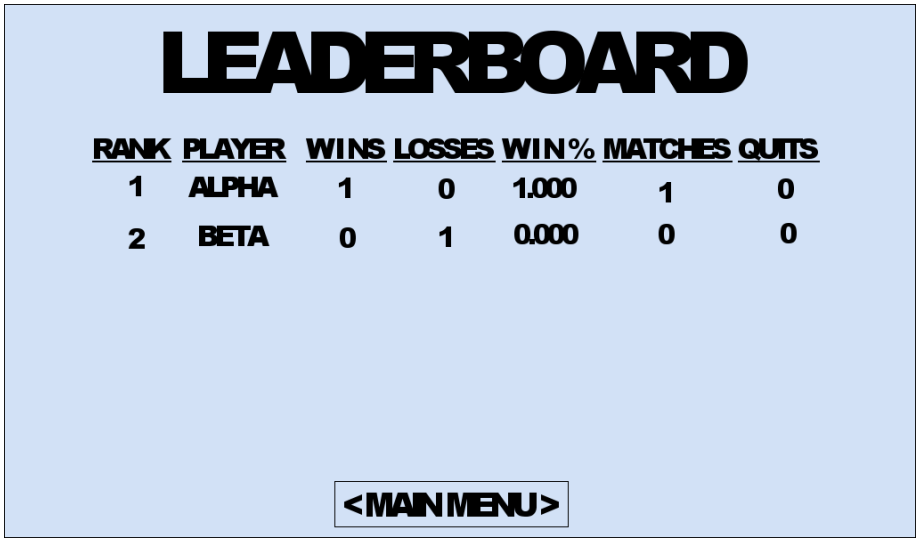


[21] http://www.supergameasset.com/epic-rpg-ui-game-asset.htm - Image 1 - Date Accessed 30/11/17
[22] http://www.playragnarok2.com/news/article/50 - Image 2 - Date Accessed 30/11/17
[23] http://www.playragnarok2.com/news/article/50 - Image 3 - Date Accessed 30/11/17

The Reason for this Login UI is because I want to make Rune Raid user friendly and eye catching. The majority of games that require Usernames and Passwords will have their own login screen, like shown above, this is so that the game is inviting and appealing to the user and make them want to play the game. If it was just a command line structure, it would come across as cold and uninviting towards the player which is undesirable for game developers. Also, it is highly important for there to be a login UI as it is the first thing the player will see when they load the game. Therefore, I will be following the same format and making sure Rune Raid's login UI is attractive and inviting towards the player.

Leader Board UI

[24]



[25]

[24] https://openlab.citytech.cuny.edu/sp13-group1/2013/04/19/battleship-analysis-phase-gui/ - Image 1 – Date Accessed 30/11/17

[25] https://trovesaurus.com/mod=2768/enhanced-leaderboards - Image 2 - Date Accessed 30/11/17

These UI's above have a similar structure to what Rune Raid's will be:

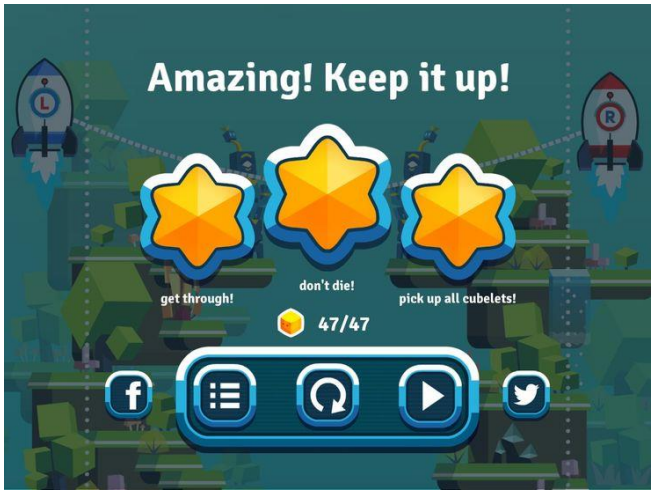
- Username, to show the player that is lined to the rank.
- Score, to show how well the player did and give a benchmark for themselves or other players to beat.
- It will have a rank/position E.g. 1st, 2nd …. 10th…. and so on, to provide a competitive environment.

However, the leader board will be split up into 3 different systems, one for each level, this is because there may be different players on different positions for each level E.g. John is 1st on level 1 but 5th on level 2 and 3rd on level 3. Therefore, the 3 different levels must have their own separate areas within the overall leader board to show the ranks for each level, to not confuse the players.

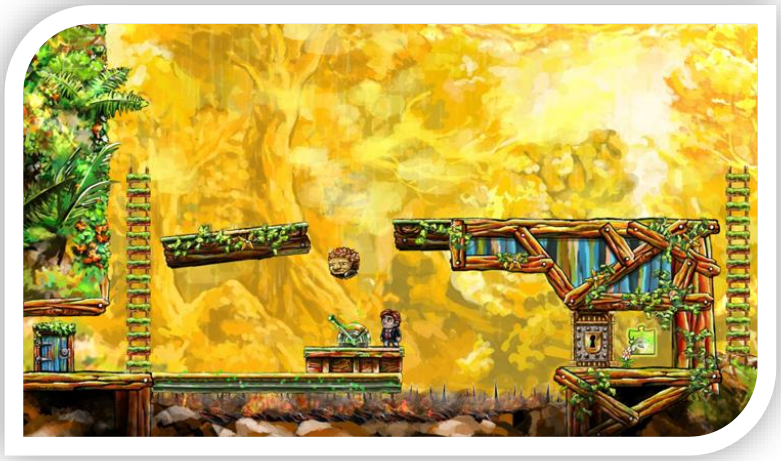
In Game UI

The entirety of the in-game UI will be built with the Unity Engine using pre-defined objects that are simply dragged and dropped into the design view and manipulated to my liking. It will be very simple to use because I don't want a difficult UI for the players, as the puzzles is what I want them to be focusing on. A confusing UI would divert their attention from the puzzles which may frustrate the players because they would have to learn the UI as well as try solve the puzzles, while at the same time be against the clock. Therefore, having a simple to use UI is very important for Rune Raid to create a fun, competitive environment.

[26]



[27]



[26] https://www.pinterest.co.uk/timoa/mobile-ui-games/ - Image 1 – Date Accessed 03/12/17
[27] https://mubi.com/notebook/posts/braid-everything-and-nothing-as-a-video-game - Image 2 - Date Accessed 03/12/17

The UI will display the current level with everything within it, timer, runes collected, story text, hints, etc. When each level is finished it will display the player's final time, runes collected, time penalties (if any occurred) and give the player the open to replay the level, exit the game or play the next level.

Description and justification of the usability features

The usability features within Rune Raid are as follows:

- A restart level option whilst playing the level (also replay at end of level) – This is important because if the play is unhappy with their progress whilst playing any of the three levels, then they have the option to restart from the beginning of the level, with the timer reset. Without this option then if the player is unhappy with their progress, then they would still have to complete the level to try again. This would be very frustrating for players who are competitively aiming for 1st position in the leader boards because these types of gamers often restart the level immediately after they make a mistake because they know that they won't be able to achieve 1st place with their current progress. An option to replay the level will also be available at the end of the level once it's completed. This will be the same as the restart option and will therefore, allow the player to quickly replay the same level again, without needing to exit the level, go to the the main screen and choose to play the level again.#

- Pause button – This is implemented in the majority of single player games and because Rune Raid's levels are single player only, then a pause button is expected, as nice addition to the gameplay. This is extremely useful for players who are often busy with real life scenarios such as, helping out friends/family. It is also very useful for players who just want to pause to eat breakfast, lunch or dinner or something as simple as wanting to change their music that they are listening to. The pause button will allow players to do all these things without being penalised for being away from their keyboard (AFK), as the elements within Rune Raid will all pause and essentially the timer will freeze. As a gamer myself, this feature is something I feel is essential for me to play single player games, as I often need to leave my computer at multiple random unpredicted intervals throughout the day. If this feature was not added within Rune Raid, it would annoy me because every time I leave my computer I would be forced to restart the level because of the timer not pausing.

- Leader Board – This feature will be the core of the competitive environment between players, as it allows players to compete for high positions on the leader boards. Players will be able to show off their skills and achievements for each level, to their fellow gamers, with the leader board as their evidence of their own time scores. Without this feature it would significantly reduce the competiveness of Rune Raid because there would be no easy way to compare scores between players. The only way would be if players took a screenshot of their time scores and posted them on social media. This would be extremely inconvenient and the majority of players would chose not to do this. There would also be no easy way to find out the quickest time for each level, as all times posted on social media would need to be gathered. As well, a player may have achieved the quickest time but didn't post it and so the quickest recorded time would be inaccurate. A further problem could be that players may fake their time scores by editing their own screenshots, to fake a quicker time score. Therefore, to ensure that every player's score is recorded accurately, a leader board is needed. This feature will be easily accessed upon loading up Rune Raid, located on the login screen, as a separate button.

- Next level button – This is an fundamental feature within Rune Raid, as it allows the player to simply choose to proceed to the next level, once they have completed their current level. It will be located on the end screen for level 1 and level 2, on level 3 the button will be slightly different, as it will load the main screen of Rune Raid instead of the next level because there is no 4th level. Without this feature, at the end of each level, the game would then have to load back to the main screen, to then choose the next level to play. With the next level feature within the each level, it allows for a smooth and simple transition between levels.

- Storyline – This feature will be displayed within a text box, whilst each level is played and also before and after the game is played. This feature is not essential within Rune Raid for it to function but it is crucial in creating an immersive atmosphere for each player to enjoy, if they so wish to read the lore and current story. It is designed towards the casual gamers who enjoy a good storyline to play, alongside the gameplay. As a game developer, it is important to address the competitive side of gaming but also to not forget that there are gamers who just want to enjoy playing a challenging game with a nice storyline. By adding a storyline, Rune Raid can achieve both of these gaming desires without infringing on each other.

- Hints – This feature will be designed to simply give the player a slight push in the right direction, if desired, when solving each puzzle. This will prevent players from getting frustrated if they cannot make any progress with their current puzzle. However if a hint is given to the player, it will add time to the timer so that the player can't abuse the hint system, by using it to proceed quicker without a time penalty. The hints will be located on the screen as a small button, which when pressed, opens up a text box displaying a hint to the current puzzle.

- Timer – This is one of the main features within Rune Raid, as it is the only way to determine each player's final time score, upon completion of any of the three levels. Without this feature, Rune Raid's leader board would be an empty database, as there would be no time data to store in it. Also, having the timer displayed live on the screen for each level is helpful for competitive players. Players who are aiming for a speed run to achieve 1st place can use the live timer as an indication of their progress. They can use it to determine whether they need to restart their current level due to being too slow or if they have made good progress to have a chance to gain 1st place.

Description and justification of validation required

- Username – Upon creation of a new username, it will need to go through a process of validation to ensure that the input for the player's username is valid. Without any form of validation some functions may not function properly, such as storing usernames and displaying them, due to unknown/impractical characters being used which may cause crashes or cause formatting errors. It will also prevent duplicate usernames from being used, which will confuse the database, as it will treat the identical usernames as the same account. Therefore, if the username is already taken then it will tell the user to input a different Username until a new unique username is inputted. The username characters will be limited to ensure that no undesirable characters are used in any username. There will be character limitations also, which will be a minimum of 6 characters and a maximum of 16 characters, this is because less than 6 or more than 16 characters would create a formatting error when displaying the username. Thus, if the username contains an invalid character then it will tell the user the problem and then ask them to input a different username with valid

characters and is the character limit is invalid then it will inform the user to comply with the character limit and input a new username.

- Password - Upon creation of a new password, it will need to go through a process of validation to ensure that the input for the player's password is valid. This is to ensure that all passwords created comply with the character limitations and requirements needed to create a strong password. Each password will need to contain at least one number and must contain between 8-20 characters. This is because if a password doesn't contain numbers and is very short (say 3 characters), then the password would be extremely weak and so easy to crack. Therefore, all passwords must contain a number and have between 8-20 characters, to make ensure the passwords are not extremely weak. If a password is less than 8 or more than 20 characters or doesn't contain a number then it will tell the user the problem and tell them to input a new password.

Key variables and structures

Identification of the key classes and attributes

- Monsters – In one of Rune Raid's levels there will be multiple monsters that will be used to hinder the player. Each time the player is hit by one of these monsters it will add extra time on the timer, making the player get a longer time score. This is to create a sense of fear or dread about hitting the monster because it will sabotage their attempt to get higher on the leader board. Using a class for this means that I can use the monster class as a blueprint and edit it for each monster so that each monster is unique.

- Pathfinding Algorithms – This will be implemented for the monster within the levels. This is so that each monster can navigate the level, moving to and from their designated positions. As the pathfinding will be a class, each monster can have a slightly altered pathfinder. This is needed because each monster will have their own positions to move to.

- Player Movement – The movement for the player will be defined as a class because the "blueprint" will consist of changing the axis of the player sprite whilst a specific button is pressed down. This class can then be used as a "blueprint" for modification for each axis used, in this case only X and Y. The specific buttons e.g. button A moves negative in X-axis, meaning the player sprite will move left. However, because this class is a built-in function within the Unity Engine, I will not need to create the class but just modify it.

- Physics – The physics will also be defined as a class because the foundation of the physics will just be to when not in constant with a solid object below in the Y-axis, then the object will move negatively in the y-Axis until it meets another physical object. Or if the gradient the physical object is in is too steep then it will slide down the other physical object until it meets a physical object of stable gradient. There are many other examples which will cause movement for objects, these two examples are just a few. The Unity Engine provides a built-in Physics class which is what I will be using and so I won't have to create the class, which is good because coding physics is very complicated and time consuming. I will just be modifying the Physics class to my own objects with their own scenarios.

- AI opponent – I will be creating the AI opponent as a class so that I can easily adapt, modify and reuse depending on my requirements. As I will not know 100% what my game will have implemented within it because in my beta testing, my beta players may want another element with the AI to be added. Therefore, I will making the AI as a class so that if I chose to add extra elements with a similar AI structure, then with a class AI this can be easily done. The class will consist of the foundation AI to be implemented so that it can just be slightly altered depending on the requirements.

- Timer – This will be a class so that there will be a foundation of code for the timer which for each level will be slightly altered to match up with the level requirements. This is because on one level it will have monsters that will add time to the timer if they come into contact with the player. Another level will have a different requirements such as if a mistake is made with the puzzle it adds time to the timer. As each level is different it is more practical to create a class timer instead of having one function with all the 3 level requirements implemented. Also it is good code development practice to make the timer a class rather than a stand-alone function.

- Game objects – There will be a class for the basic functions of an object within each level. This will be the base of the building blocks for each level and because each object will be slightly different but fundamentally the same. Therefore, it is more practical to create an object class to then be used to modify to create each unique object.

Explanation and justification of the data structures and key variables

Key Variables

VARIABLE NAME	TYPE	EXPLAINATON	EXAMPLE
USERNAME	String	This will be the variable that is stored within the account database.	“Blurry”
CURRENT USER	String	This will be the username that the current user possesses.	“Reflex”
PASSWORD	String	This will be the variable that is stored in the account database to be used to verify logins to accounts.	“Basket@uk99”
ACCOUNTID	Integer	This is the unique ID for each account.	“02”
TIME	String	This will be the current live time for the timer. It will consist of four variables, a, b, c and d. Each variable will be an integer and they will	06:57

		concatenate together with ":" in the middle to form the time.	
POINTERID	Integer	The current position for the unique AccountID. This will allow the database to keep track of which AccountIDs are available to be used for creating a new account.	"23"
FASTESTTIMEL1	String	This will be the accounts fastest time score for level 1. This will be stored in the leader board database. It will also be used to compare with new end Time variable for level 1 to see if the user has accomplished a new time best record.	"12:45"
FASTESTTIMEL2	String	This will be the accounts fastest time score for level 2. This will be stored in the leader board database. It will also be used to compare with new end Time variable for level 2 to see if the user has accomplished a new time best record.	"5:05"
FASTESTTIMEL3	String	This will be the accounts fastest time score for level 3. This will be stored in the leader board database. It will also be used to compare with new end Time variable for level 3 to see if the user has accomplished a new time best record.	"09:32"

PHYSICAL OBJECTS	List of string	All the physical objects within Rune Raid, used to detect physical contact, thus causing collision.	[Wall, Ball, Pipe, Door, Floor]

Data Structures

DATA STRUCTURE	EXPLAINATION
ACCOUNT DATABASE	This will be the database where all account login details are stored; AccoutID, Username and Password. This database will be used to verify logins and also a place to store new unique accounts when they are created. When logging into an account this database will provide the necessary data to identify which account is being logged into and whether the password is correct, to identify that it is the user is the verified user of the account.
LEADERBOARD DATABASE	This will be the database where all account times for each level is stored. It will be set up so that each AccountID has their own three rows to store the current fastest time for the user for each level. The database will be used to supply the required data to form the leaderboard display.

Test data for development

Identification and justification of test data to be used during development

Throughout my development I will be rigorously testing Rune Raid to identify any errors I may have made or to see if what I have created works. When each module/function/class is created I will run it to make sure that it works, providing evidence of this with screenshots of the results. This way I can quickly identify errors that are within the code and fix them, just after I have created them. The benefits of this are that I will have the fresh knowledge of knowing exactly what I'm trying to implement and what everything does, instead of having to remind myself what each variable does, at the end of the first stage of development.

Also, testing and fixing any problems with the current function/module/class that I have developed will mean that there will be less issues later on in development. An example would be, that if I create a function, test it and find a problem, I can fix it there and then. However, if I left it and tested every function at the end of the first stage of development, I might find a fundamental problem that I made in the first functions of Rune Raid, which I then implemented within all the next functions I created. Meaning that because of one incorrect function, I would have to fix every function later made with faulty element, which will significantly increase my time spent on fixing problems. As I want to reduce the amount of time I spend on fixing problems as much as possible, testing after each function/module/class is made is the sensible thing to do.

Example of testing

TEST	INPUT/CONDITION	EXPECTED RESULT
CREATING A NEW USERNAME	Inputs: “James99” or “James 99” or “JamesMatthewAlexJones99”	Username stored if “James99” is not already stored in Account database. Output (“Username already taken, please enter another username”) if username is already taken. Output (“Invalid characters, please enter another username”) if invalid characters are within the username. Example (white space is not allowed). Output (“Please enter a username between 6 and 16 characters”) if username is out of range.
CREATING A NEW PASSWORD	Inputs: “Hello@fish.com9” or “matthew” or “XYR@hello.matthew.password.com99999” or “pass” “qwerty`99”	Username stored in Account database if password meets all validation. Output (“Password needs to be between 6 and 20 characters, please enter another password”) if password out of range. Output (“Invalid characters, please enter another password”) if invalid characters are within the password. Example (` is not allowed). Output (“password needs to contain at least 1 number, please enter another password.”) if a password doesn’t contain a number.
CHECKING TIME AGAINST FASTEST TIME L1	Input “09:32”	Overwrite FastestTimeL1 if Time < FastestTimeL1. So FastestTimeL1 is now “09:32”.

		Discard Time if Time > FastestTimeL1. So FastestTimeL1 remains the same fastest time for level one for the account.
LOADING LEVEL 1	Condition: Login Successful	After the login is successful then it should load level 1. The level must be fully loaded and working completely, with the timer running in parallel.
LOGIN	Inputs: Username and Password; Username: “James98” Password: “Snowing@santa99”	If the username and password are the same as any AccountID within the Account database then, it will set the currentUser to the username entered and login. If there are no matches for the username and password to any AccountID within the Account database, then it will output (“Incorrect Username or Password”).
DISPLAY LEADERBOARD	Condition: Leader board button clicked	When the leader board button is clicked then, it will load up the leader board with all the correct fastest time scores for each user who has completed a level. The leader board will be split up into 3 fragments, one for each level, showing the fastest time in position 1 and the slowest time in last position. Any new best time scores achieved for an account will be updated in the leader board database and therefore, the displayed leader board should be updated with the new times as well, replacing the old times.

COLLISION	Condition: player sprite/ physical object in contact with another physical object.	The objects/ player sprite will not be able to move through the object(s). When the player moves into contact with the wall, it will prevent any extra movements in the axis of the physical object. Example, the player sprite moving left and then comes into contact with a wall. Once the player sprite comes into contact with the wall then it will prevent any more movement in the negative x-axis. Therefore, not allowing the player to move through the wall.
PLAYER MOVEMENT	Inputs: Arrows keys, W, A, S, D keys.	When any of these keys are pressed, it should move the player sprite in the axis designated to the key. Example, pressing the left arrow/A key will move should move the player sprite in the negative x-axis. Once the left arrow/A key is released, it should stop all movement in the negative x-axis. Whilst the player sprite is moving in the desired direction, it should trigger the animation for the direction the player spire is moving in. Example, if the player sprite is moving right then, the right walking animation should execute until the right arrow/D key is released. Once the player sprite stops moving it should execute the idle animation for the

		previously direction of movement. Example, if the player sprite was moving upwards and then stopped, it should execute the upwards idle animation, until another movement key is pressed.
TIMER	Condition: Loading level, Monster contact, pause, hint displayed and mistake.	If a level is loading then, the correct timer should load with the clock set to 00:00. Once the level Is loaded the timer should increase by 00:01 every second passed. Once it reaches 00:09 then, the next should be 00:10 and so on. If the player sprite makes any contact with a monster then it should add time to the timer, e.g. hitting a ghoul should increase the timer by 00:10. Making a mistake with a puzzle should also increase the time on the timer. E.g. dropping an object will add 00:20 to the timer. When Rune Raid is paused by the player then, the timer should freeze and unfreeze when the pause is ended. Example, the player pauses at 03:43 then it will freeze at that time. When Rune Raid is eventually resumed, the timer will resume adding 00:01 every second to 03:43. When a hint is loaded for the player, the timer should increase by 01:00.
LOADNG HINTS	Condition: hint box is pressed.	When the Hint button is pressed by the player with their cursor, the hint system should load up, displaying

	the hint for the specific area of the level. When a hint is loaded, it should notify the timer function to add time to the timers.

Test data for beta testing

Identification and justification of further data to be used in post development phase

Once I have completed the development stage for Rune Raid, I will be implementing a stage of beta testing. This is because beta testing is an extremely useful way to identify any faults within a program that has slipped through the developer(s) testing. It is even easier for beta testing games because there are many gamers who are willing to test games in beta for free.

Examples of beta testing games are;

- Battlefield series [28]
- Battlefront series [29]
- Call of Duty series [30]
- League of Legends (PBE) [31]

The reasons why beta testing is so successful is because the testing involves many more users testing the program than the number of developers and so more people testing a program, usually means that more faults are found if there are any. Another advantage of beta testing is that, the beta testers will be testing the program/game in the genuine environment that the program/game will be used in. E.g. Rune Raid beta testing will be testing in the environment of actual gamers, playing the game exactly how it should be played. Once all reports of errors are fixed, it helps to solidify that the program/game is ready for release on the market. This is because the beta testers will be testing the program/game to its limits, trying to break the program/game and using it normally. All these aspects will make sure that when fixed/proven robust, the released program/game will not have any major problems and be of professional standard level.

One more advantage of beta testing is that, it allows the beta testers to give feedback to the developers on aspects of the program/game that they liked, disliked and any aspects they would like to be implemented. This way the developers can interact with the consumer market to understand what they want and check if it's viable to implement. Implementing consumer market wants is an effective way to increase the attractiveness of a product and increase sales. It is also a great way to improve the quality of the program/game, making consumers happier.

[28] https://www.battlefield.com/en-gb/beta - Date Accessed 08/12/17
[29] https://www.ea.com/en-gb/games/starwars/battlefront/battlefront-2/events/beta - Date Accessed 08/12/17
[30] https://www.callofduty.com/uk/en/wwii/beta - Date Accessed 08/12/17
[31] https://support.riotgames.com/hc/en-us/articles/201751904-Public-Beta-Environment-FAQ - Date Accessed 08/12/17

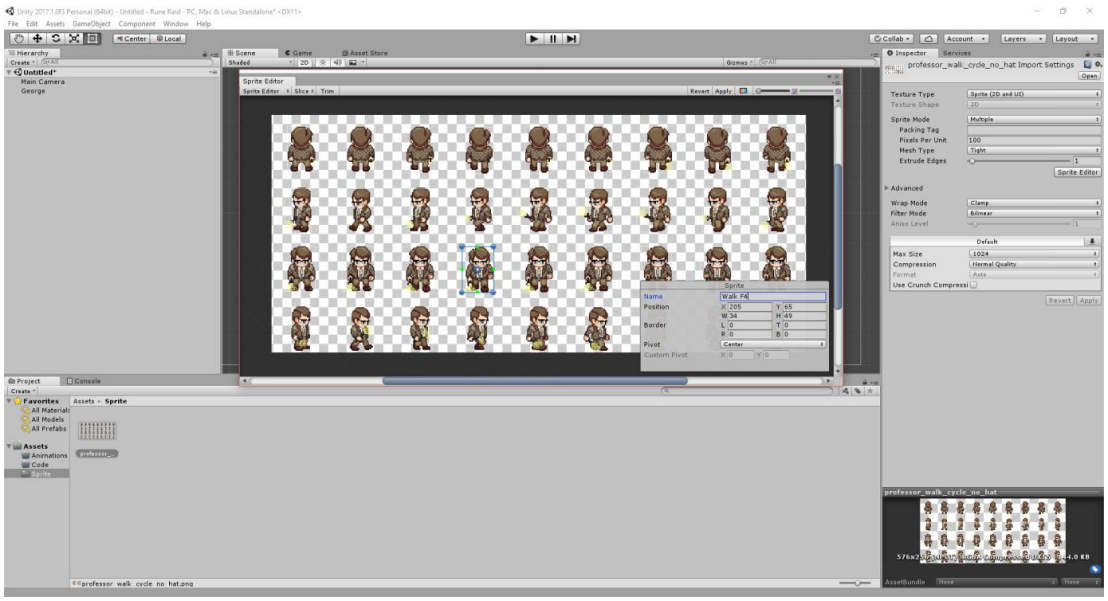
The way I will be implementing beta testing for Rune Raid will be by distributing the game to my fellow peers who also enjoy a variety of different computer games. Allowing them to try reveal any errors, with the intention of trying to break my game. They will also be giving me feedback on what they thought of Rune Raid, what they liked, disliked and any aspects they thought would be good to add. This way I will have enough test data to review, fix and implement any new aspects to the game which I feel would be beneficial. Making Rune Raid more enjoyable for the gaming community and to ensure that my game is as fault free as possible for its official release.

Development Part 1

Sprite

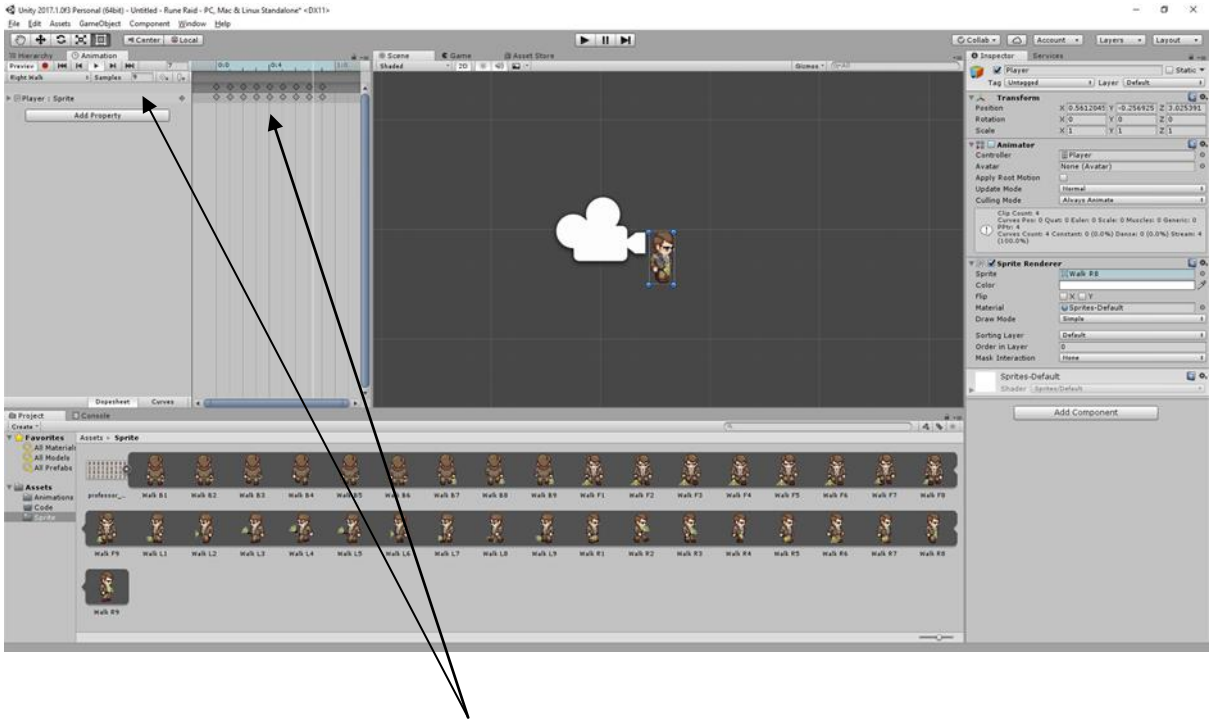
To start off my game I first needed a character sprite which would be controlled by the user. As my game genre is puzzle and is based in an old ruin of a factory I decided I needed a fitting sprite for this. After some thought I decided to import a pre-made sprite for my player model as making one from scratch would be very time consuming especially if I were to make it detailed and so importing an already existing sprite was the most efficient option.

This is what the sprite file contained, it consists of multiple images of the same sprite from different positions. This is because each character image is a frame of movement and so if each image is played in sync then it will create an animation of the sprite walking in a specific direction. The next step was for me to cut out each character into its own individual image piece and to do this I needed to use Unity's inbuilt slicing system.



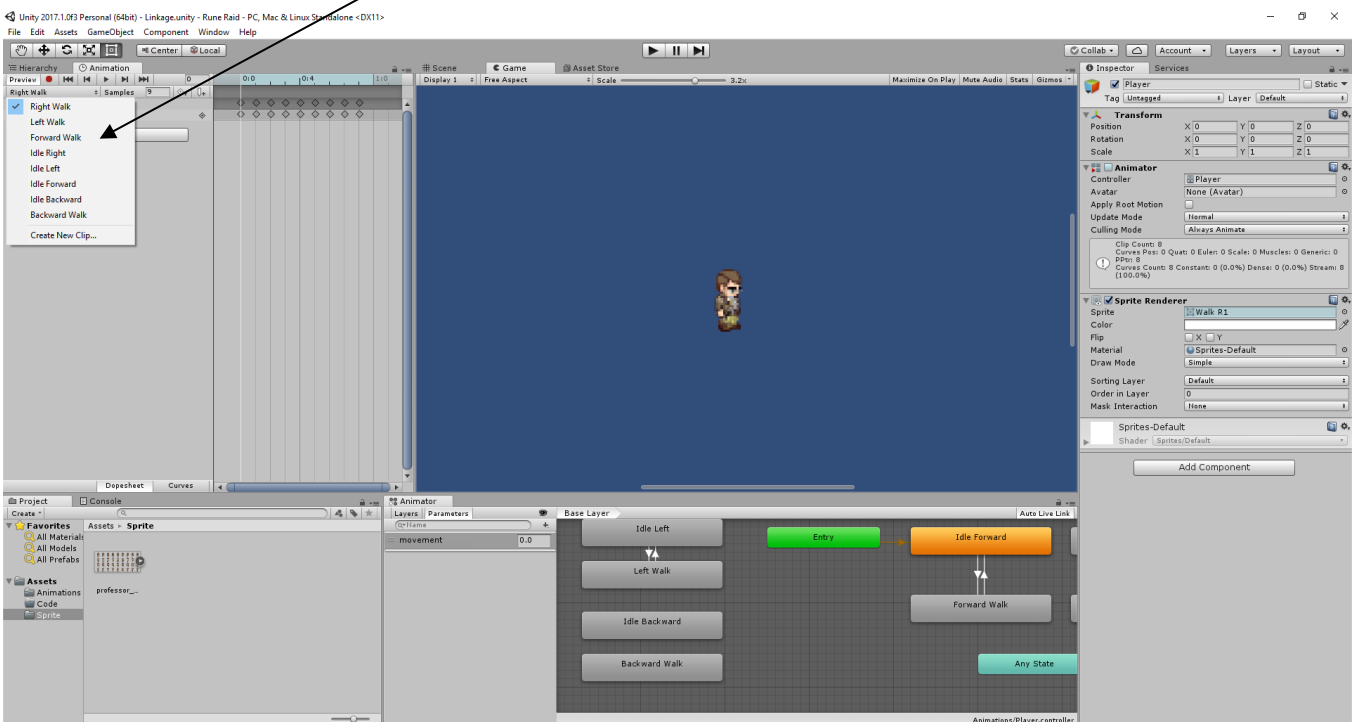
- https://www.youtube.com/watch?v=6P1ivCvofuk -tutorial for how to upload and slice sprites
- https://opengameart.org/sites/default/files/lpc-art/professor_walk_cycle_no_hat.png - detective sprite source
- http://michaelcummings.net/mathoms/creating-2d-animated-sprites-using-unity-4.3 -detective sprite source

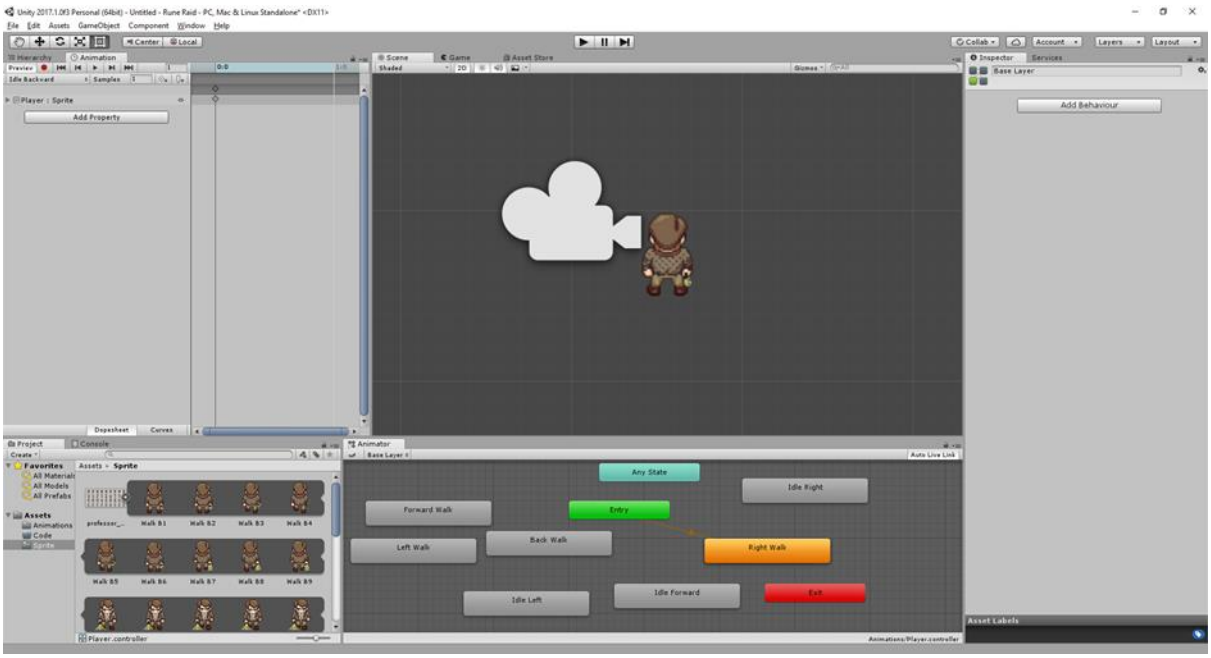
These are the individual sliced images of my sprite and to create an animation all I needed was to drag and drop each group of 9 images into the animation section in Unity. All of the 9 images have to be in the correct order to create a smooth and correct animation of walking.



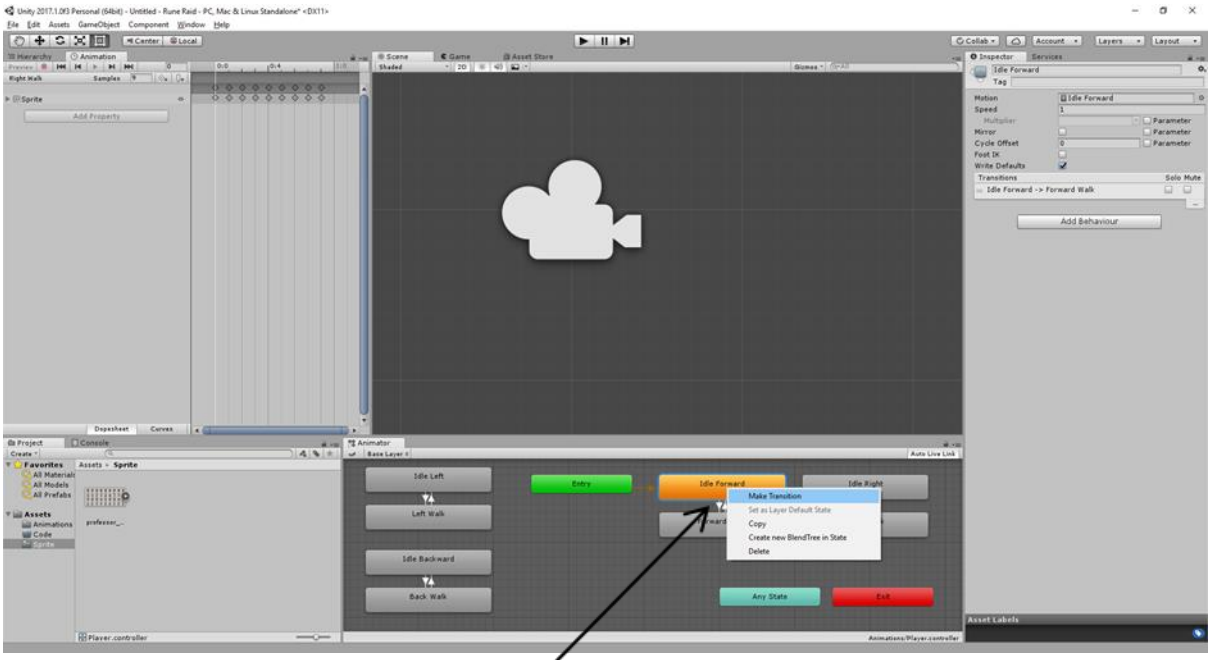
Creating 4 separate animations for movement of the player in game using the sliced images and creating animations. Each of these animations will be sampled at 9 frames per second as there are 9 images and so one complete cycle each second.

I have created 8 different animations; idle right, left, forward and backward and then right walk, left walk, forward walk and backward walk. Each of idle animation will be sampled at 1 because there is only one image and so only one image per second is needed.





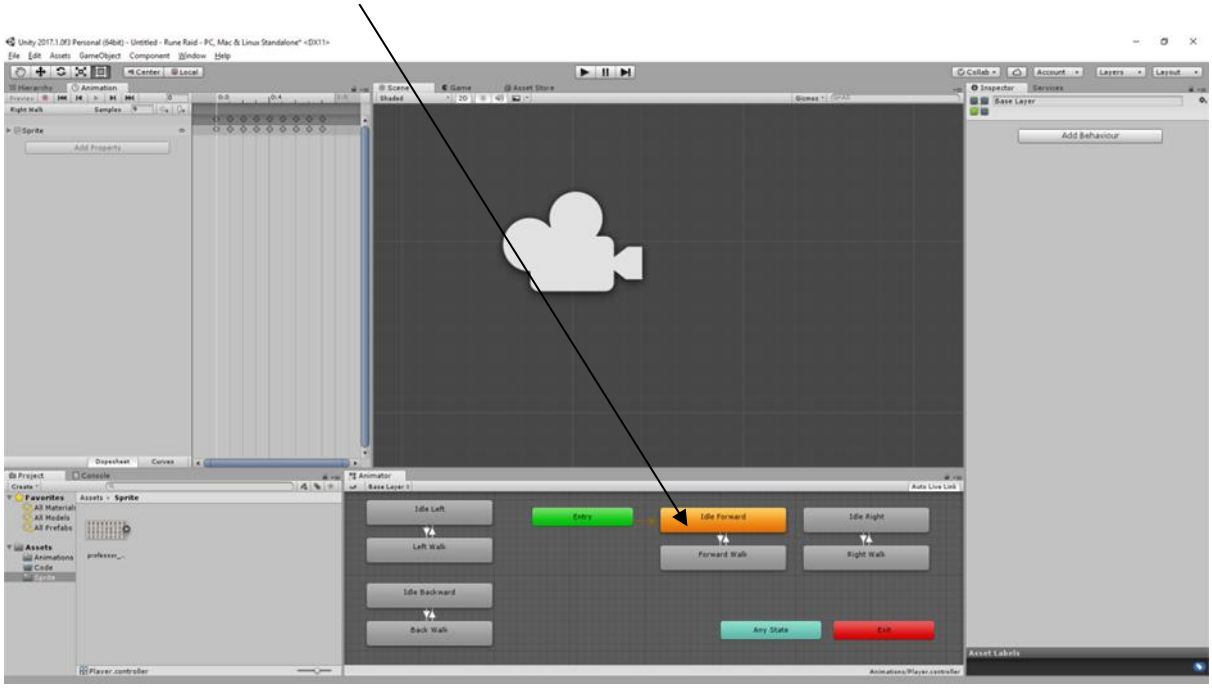
Creating linkage within each animation



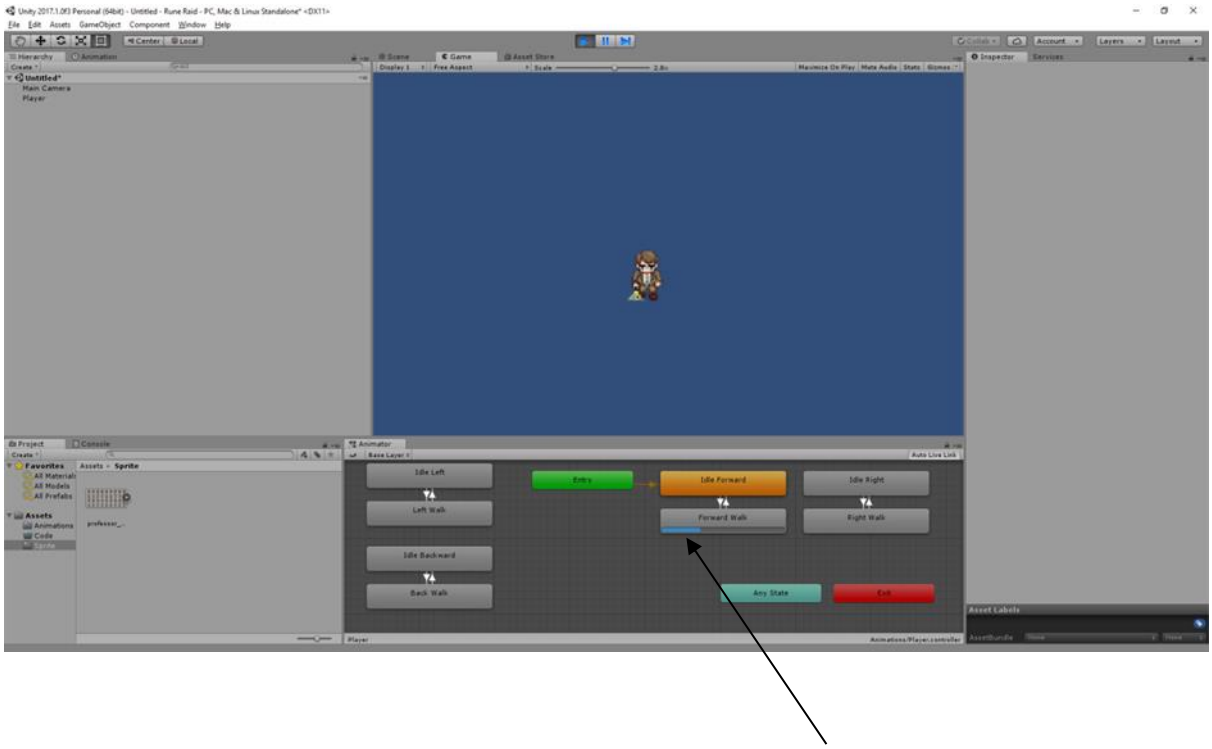
To create linkage just right click animation and click make transition

- https://www.youtube.com/watch?v=mTxmR5zuY7A—tutorial on how to link up animations to make them flow

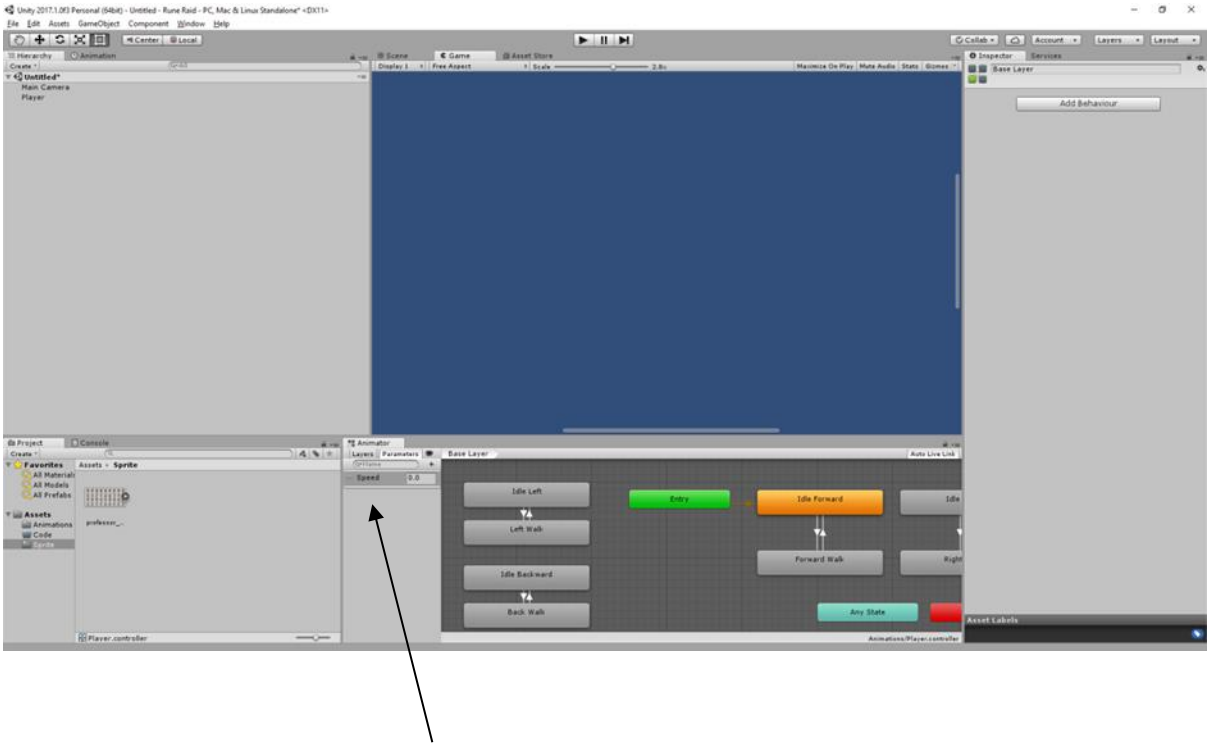
Idle forward is highlighted orange because I've set it to the default state so when the game loads in the player model its facing forward still and not moving on the stop or facing the wrong way.



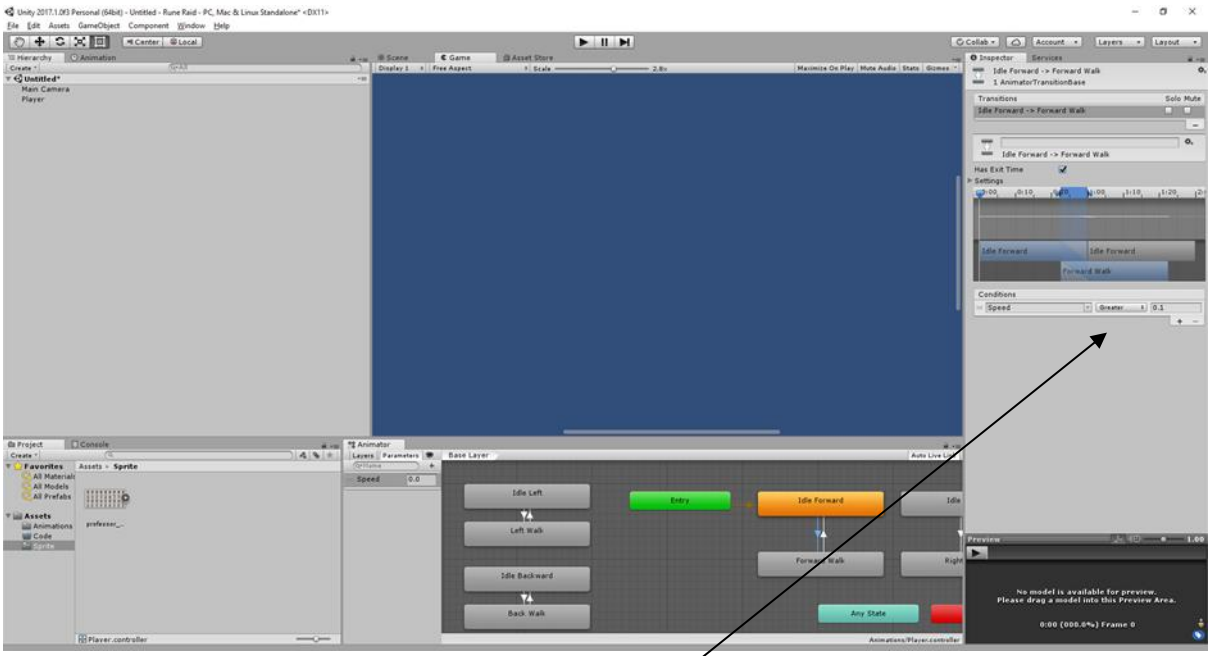
Linking each idle animation to its retrospective walking animation. Each arrow is a linkage.



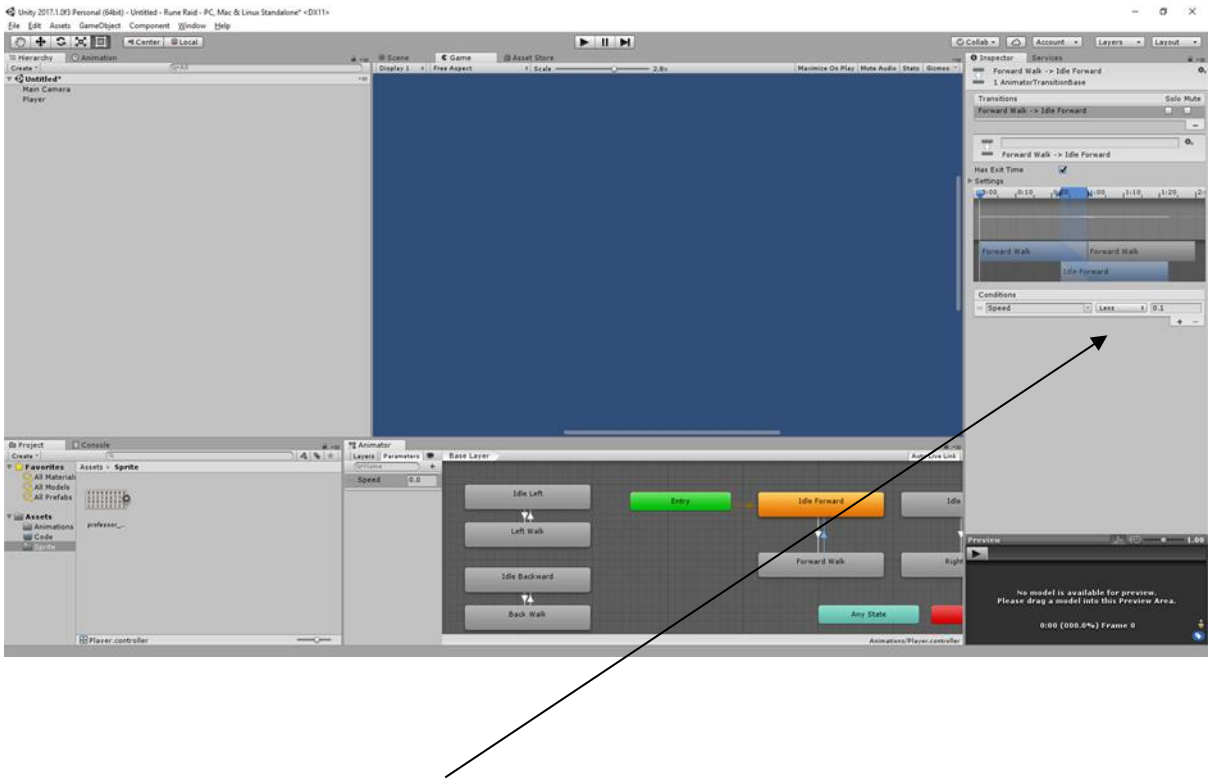
It is shown to be working because of the blue bar that loads on the idle forward and then the forward walk and the player model in the game screen the player is stood still and then the forward animation is played.



To set up each linkage properly so each animation only runs when needed, I need to create a parameter to define a condition. I have called the parameter movement and will be used to decipher which animation will be run depending on the speed of the player model. The parameter is also a float parameter to be able to use smaller decimal values for movement.



For the linkage from idle forward to forward walking I've set it so it will run only when the movement parameter is greater than 0.1.



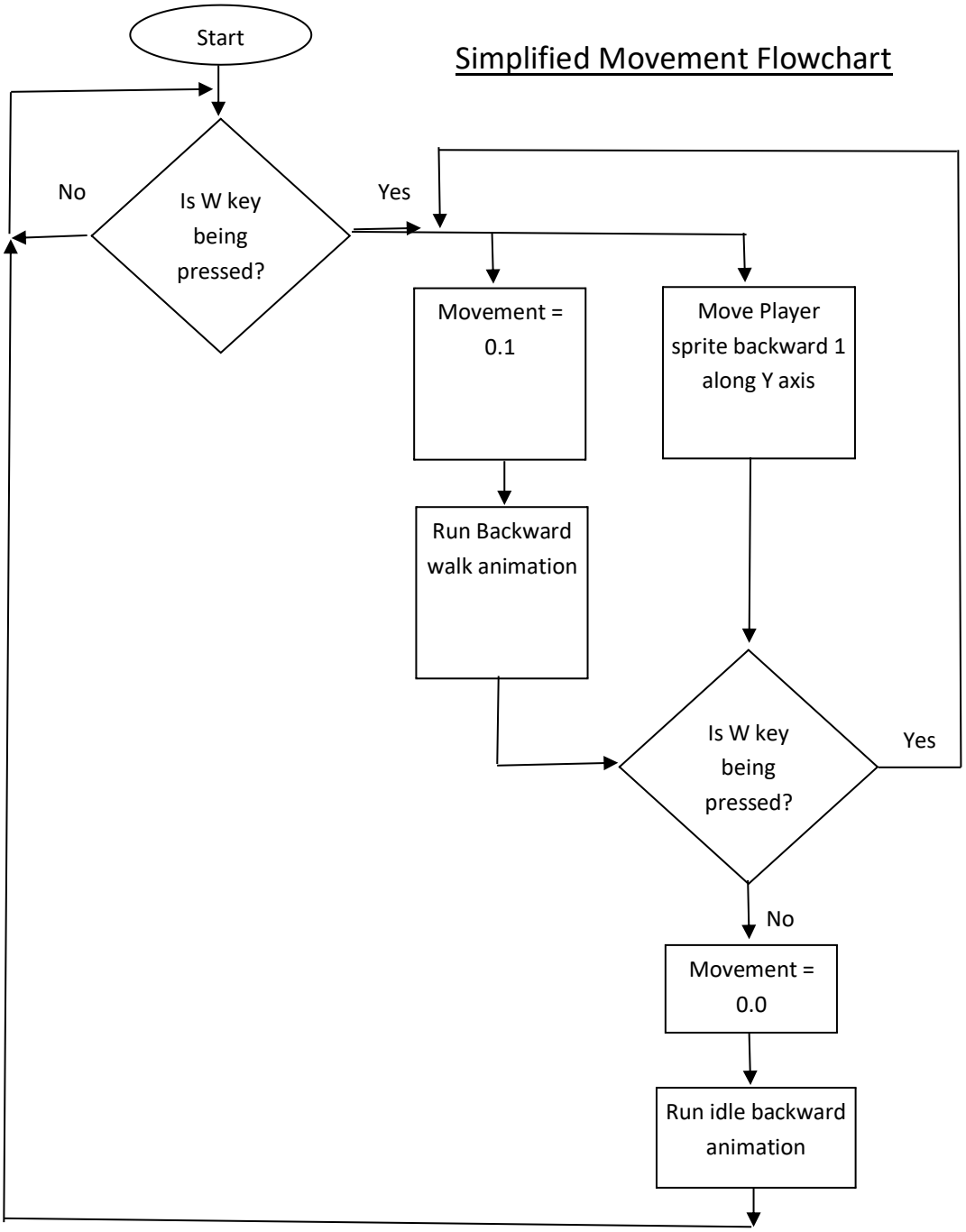
For the linkage from forward walking to idle forward I've set it so it will run only when the movement parameter is less than 0.1.

This will mean that if movement is greater than 0.1 then the forward animation will run until movement is less than 0.1 and then the idle forward animation will run.

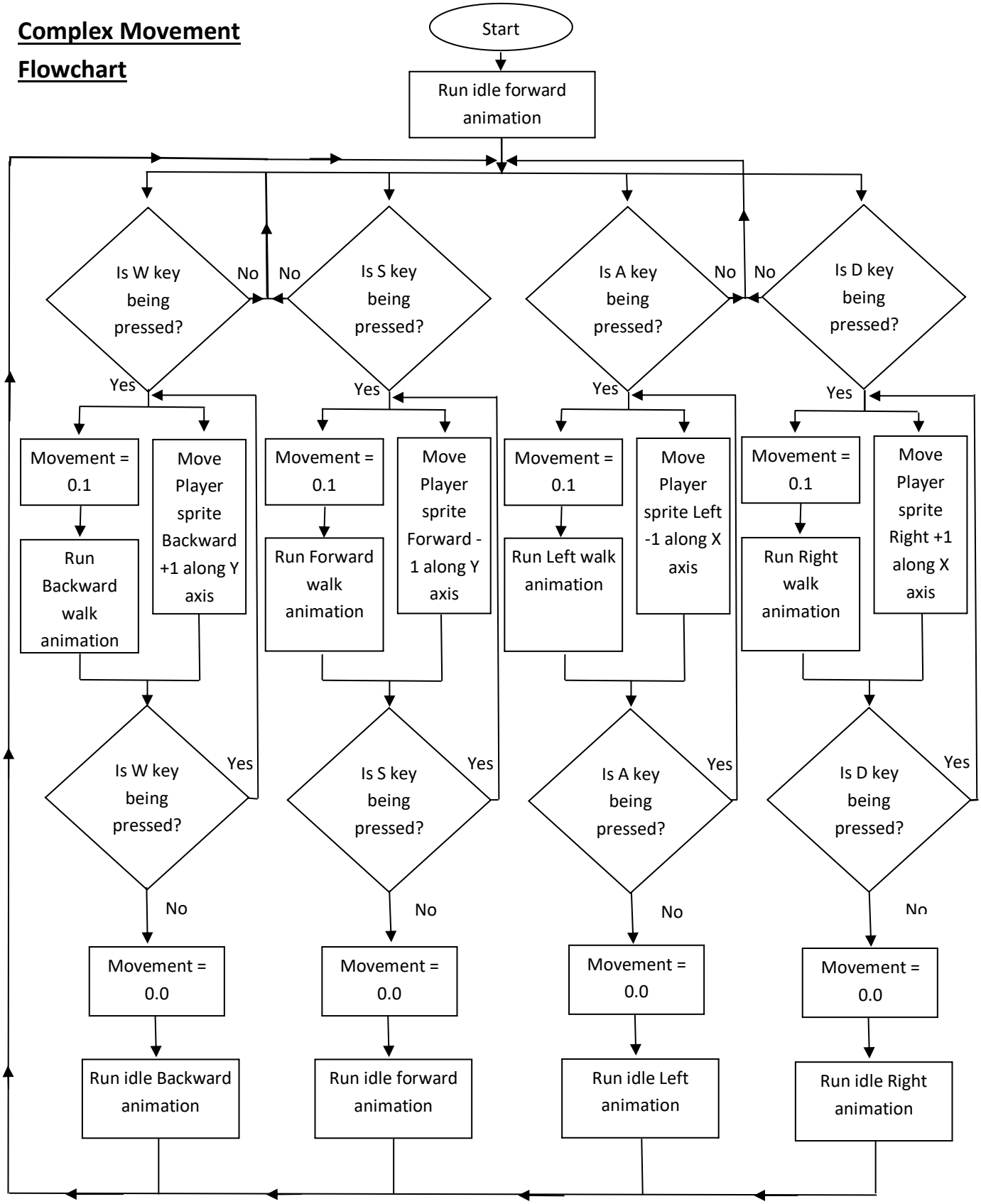
Flowchart

I now needed to create a code script to link the keyboard buttons WASD to the specific movement animations desired. Each key will also move to player model along the X or Y axis depending on what key is pressed and for how long. This will mean that when a valid key is pressed the correct animation will be played and the player model will move along the correct axis until the button is released. This will create the visual look of the player model walking.

- W key will be to move backwards and trigger the backward walk animation
- S key will be to move forwards and trigger the forward walk animation
- A key will be to move Left and trigger the left walk animation
- D key will be to move right and trigger the right walk animation



Complex Movement Flowchart



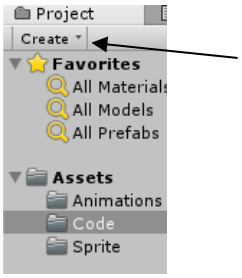
Movement Pseudocode

```
Movement = 0.0
EXECUTE Idle Forward animation
IF W key pressed:
    WHILE n = True
        Movement = 0.1
        EXECUTE Backward animation
        Move Player +1 in Y axis
        IF W key pressed:
            n = True
        ELSE:
            n = False
            Movement = 0.0
            EXECUTE Idle Backward animation
    END WHILE
ELIF A key pressed:
    WHILE a = True
        Movement = 0.1
        EXECUTE Left animation
        Move Player –1 in X axis
        IF A key pressed:
            a = True
        ELSE:
            a = False
            Movement = 0.0
            EXECUTE Idle Left animation
    END WHILE
ELIF S key pressed:
    WHILE b = True
        Movement = 0.1
        EXECUTE Forward animation
        Move Player –1 in Y axis
        IF S key pressed:
            b = True
        ELSE:
ELIF D key pressed
    WHILE c = True
        Movement = 0.1
        EXECUTE Right animation
        Move Player +1 in X axis
        IF D key pressed:
            c = True
        ELSE:
            c = False
            Movement = 0.0
            EXECUTE Idle Right animation
    END WHILE
ELSE:
    Ignore
END
```

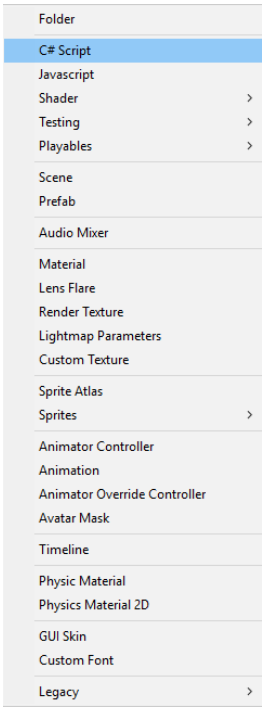
Scripts

The next step was create my code for the movement of my Player sprite. As my game is 2D I will be only using the X and Y axis as the Z axis is only for 3D. To start off I did some research and began testing how to just move my Player model left, right , up and down without the use of any keys. This is so that I understood how movement worked and then once I understood I would then implement the use of keys to run my lines of code.

To create a script in Unity and then link it to my "Player" was quite simple.

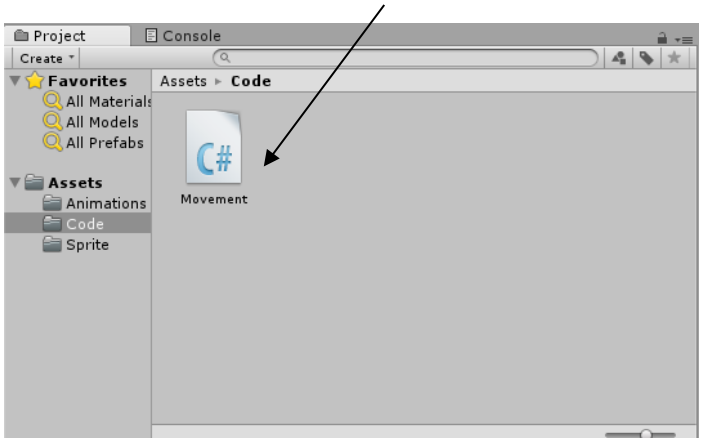


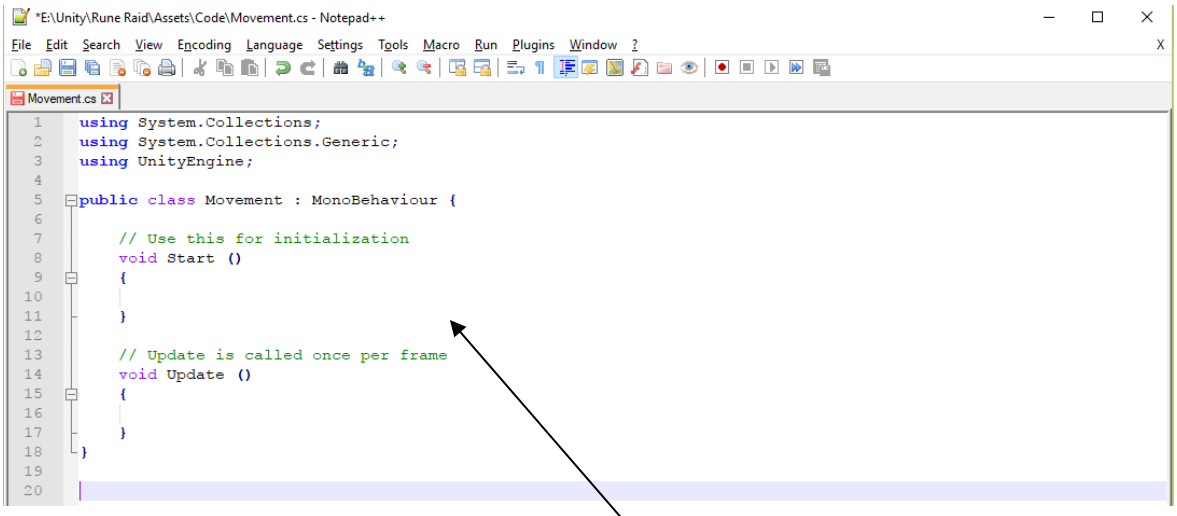
There is a small tab on the bottom left hand side of Unity which when opened looks like this.



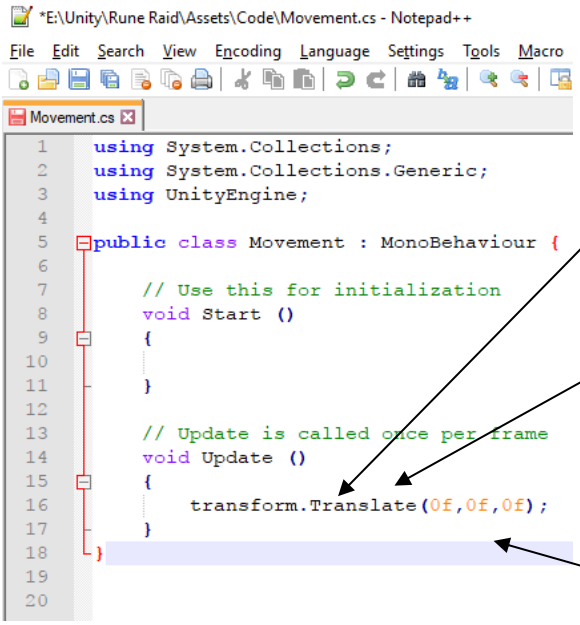
As the language I'm using is C# I chose the C# Script option

Once clicked a C# folder is created and I placed it into my subcategory of Code in my Assets and called it "Movement"





For my coding I'm using Notepad++ and so when I open the "Movement" folder for the first time this is what was presented to me.



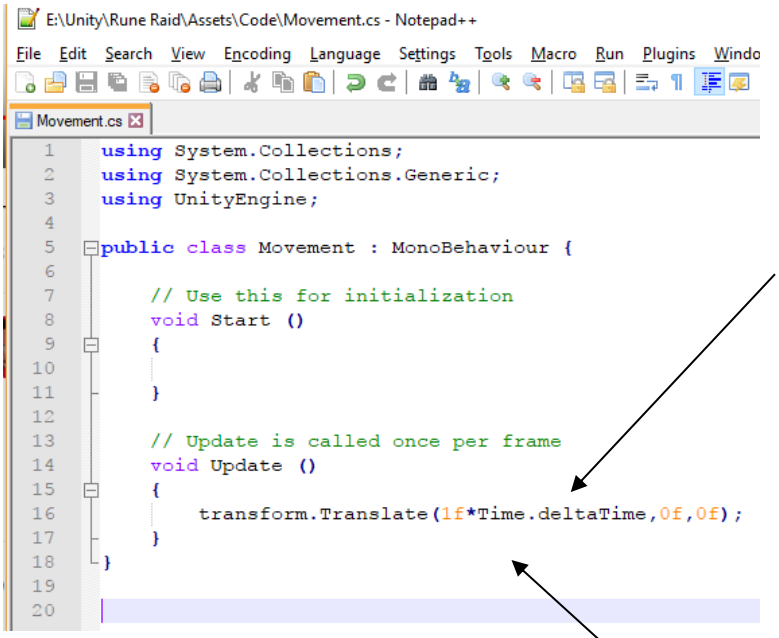
This one line of code is how to move my "Player" sprite in each axis depending on what I set in the brackets.

"transform" is the Position, rotation and scale of an object.

And so:

"transform.Translate" moves the transform by x in the X axis, by y in the Y axis and by z in the Z axis.

"(0f,0f,0f)" is the coordinates of the axis the order of x, y, z. "f" means that the value is a float which means it allows decimal numbers as well as integers to be used



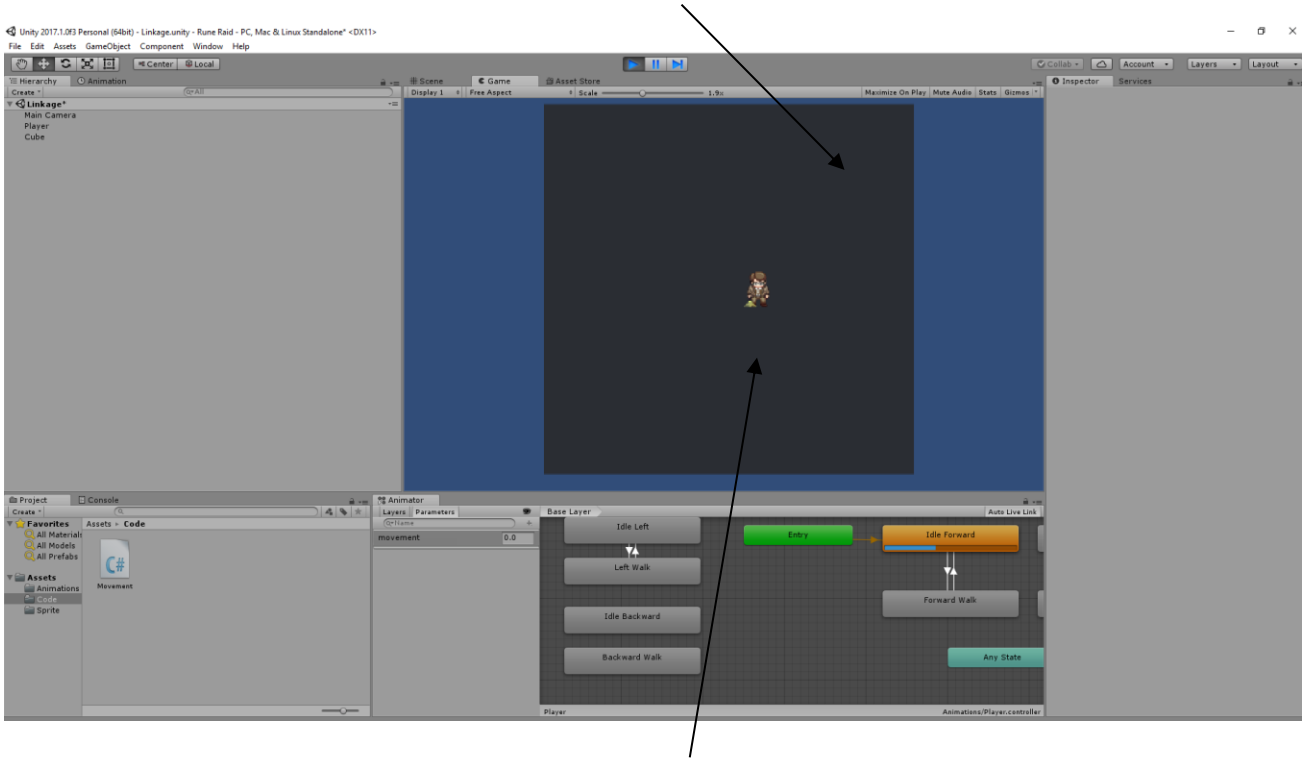
To test if the line of code works correctly I changed the x axis value to "1" but I also added "*Time.deltaTime".

"*Time.deltaTime" is used to make the game frame rate independent. This means that the frame rate of the movement will be also balanced and smooth, irrespective to the computers capabilities.

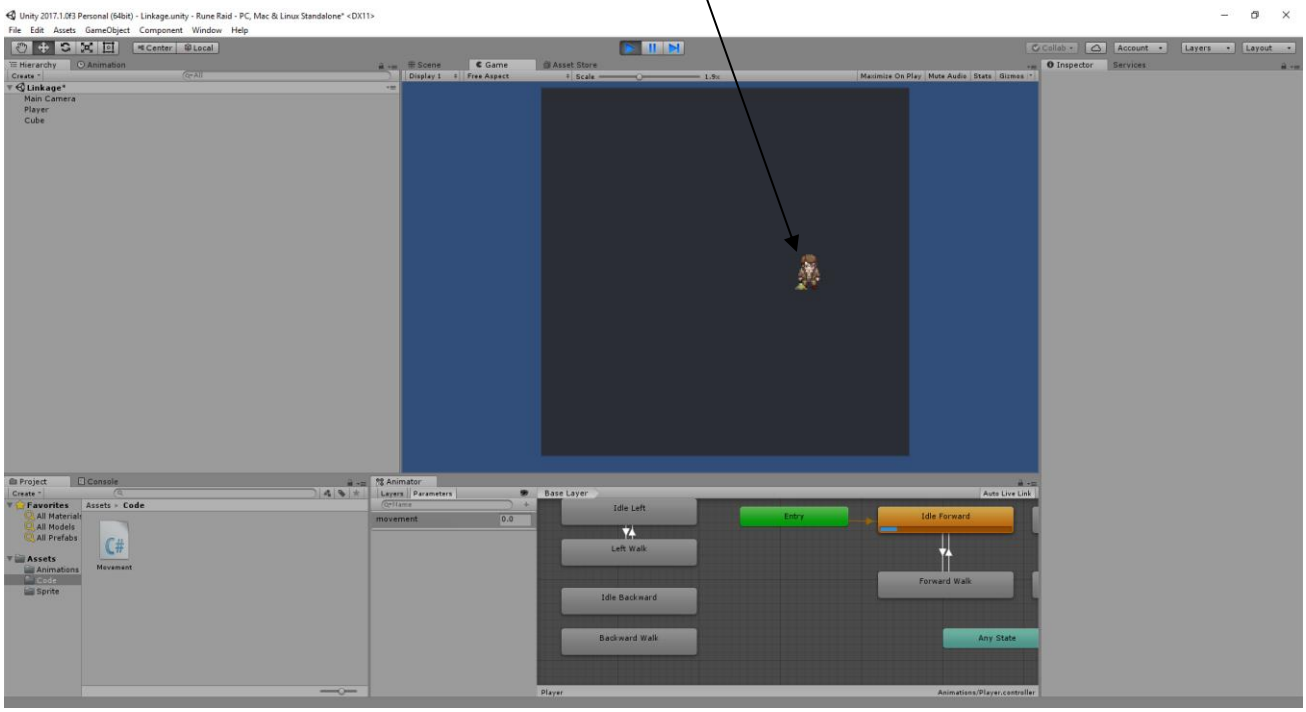
The result of this line of code should be that my "Player" sprite will move +1 in the x axis constantly until I stop the game.

- http://answers.unity3d.com/questions/182391/how-to-make-your-character-move.html - Examples of code to move my "Player" sprite
- https://www.youtube.com/watch?v=sXQI_0ILEW4 - video explaining the code needed to move my "Player" spritel in game
- https://docs.unity3d.com/ScriptReference/Transform.Translate.html - definition for transform.Translate
- https://docs.unity3d.com/ScriptReference/Transform.html - definition for transform
- https://docs.unity3d.com/ScriptReference/Time-deltaTime.html - definition of Time.deltaTime

I've also added a grey cube behind my “Player” sprite as a platform for me to work off. It’s only temperamental for now

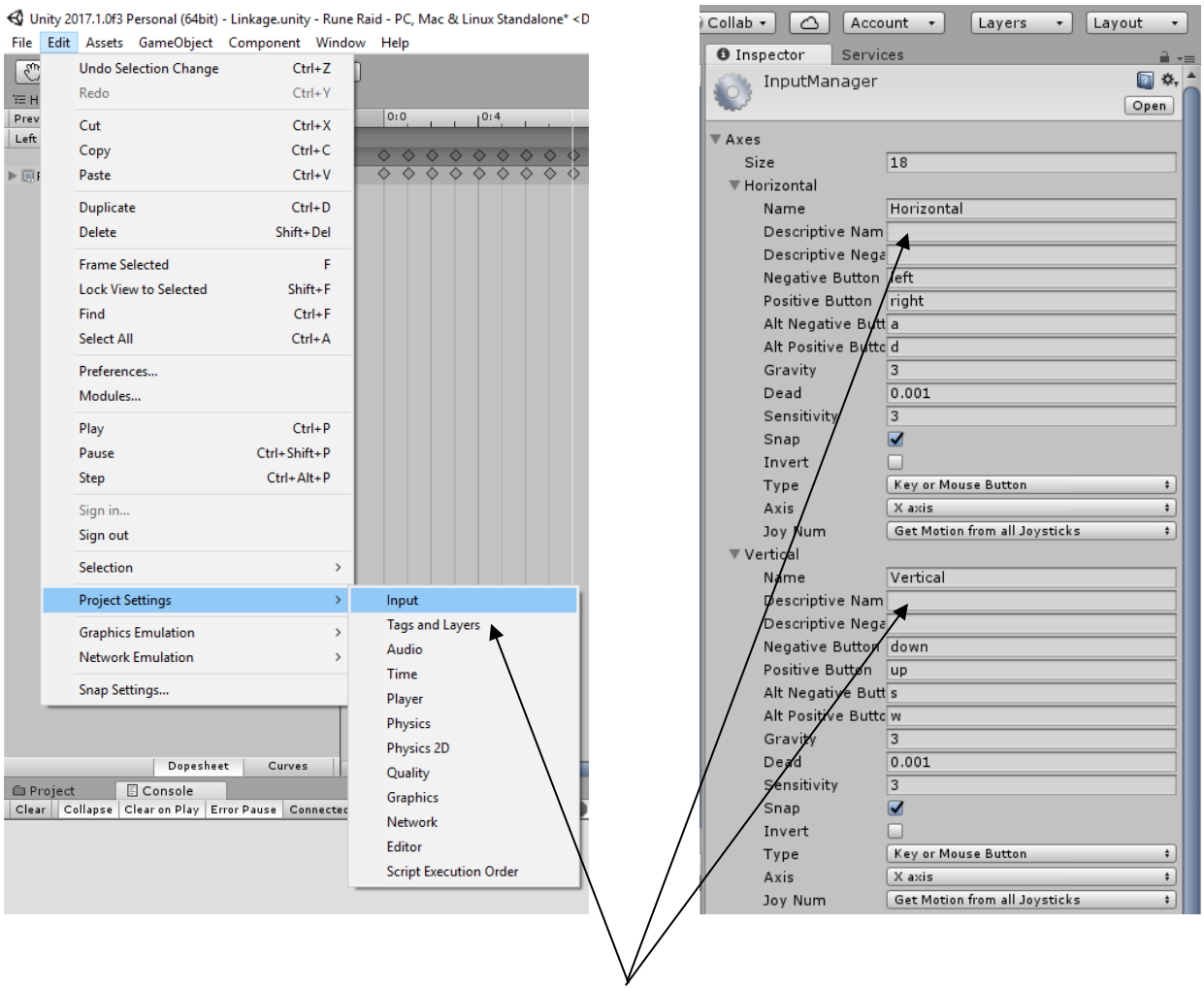


It’s difficult to show that my player sprite is moving +1 in the x axis but when I hit play my sprite does move constantly to the right (positive x axis). These two screen shots show that my “Player” sprite is moving.



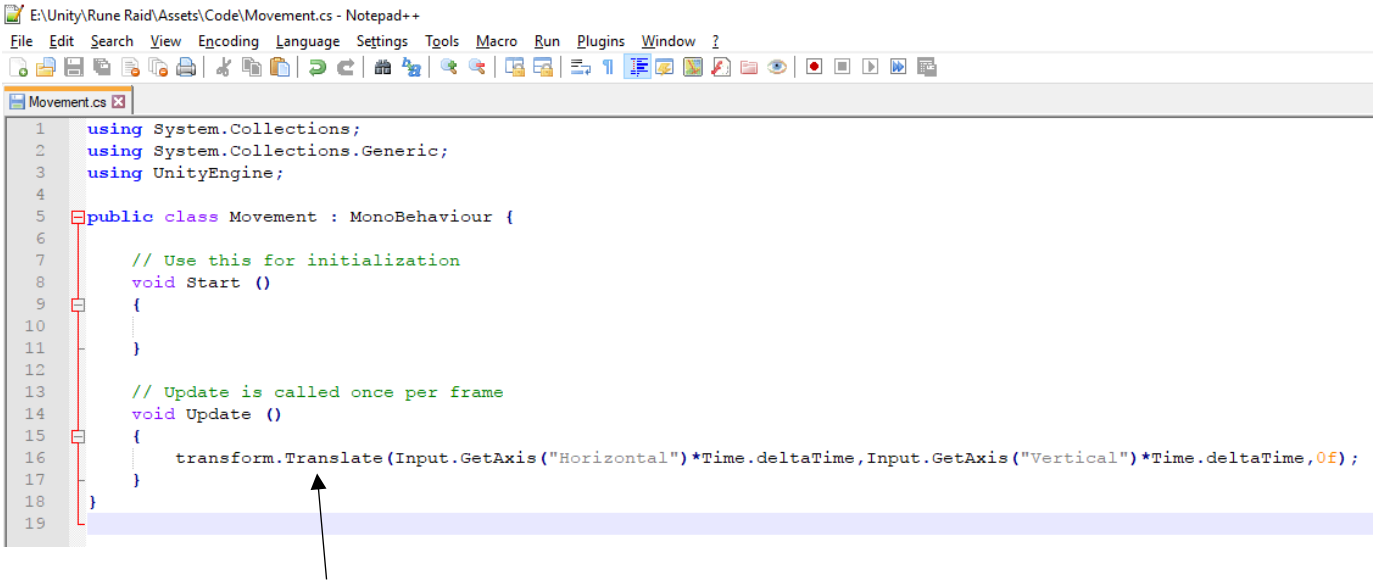
I also checked whether it works to the left, up and down by changing the code for the x axis to –1 and the y axis to 1 and –1. All of these changes worked and so that meant I could proceed to next stage of linking the movement to keys on the keyboard being pressed. I needed further research to find out how I would do this.

After some research I found out how to be able to move my Player sprite using WASD on the keyboard. In Unity there is already some pre-made script for the use of arrow keys or WASD keys and so all I needed to do was to input Unity's functions into my lines of C# from their inbuilt directory which is called "vertical" and "Horizontal".



This is where the directory for the Horizontal and vertical functions are kept. The functions can also be edited to match different keys on the keyboard or to change the desired axis. However, the functions were already set up in the way that I desired so no change was needed.

- https://www.youtube.com/watch?v=sXQI_0ILEW4 - The previous video explaining that there is functions in Unity's directory that can be used for movement.



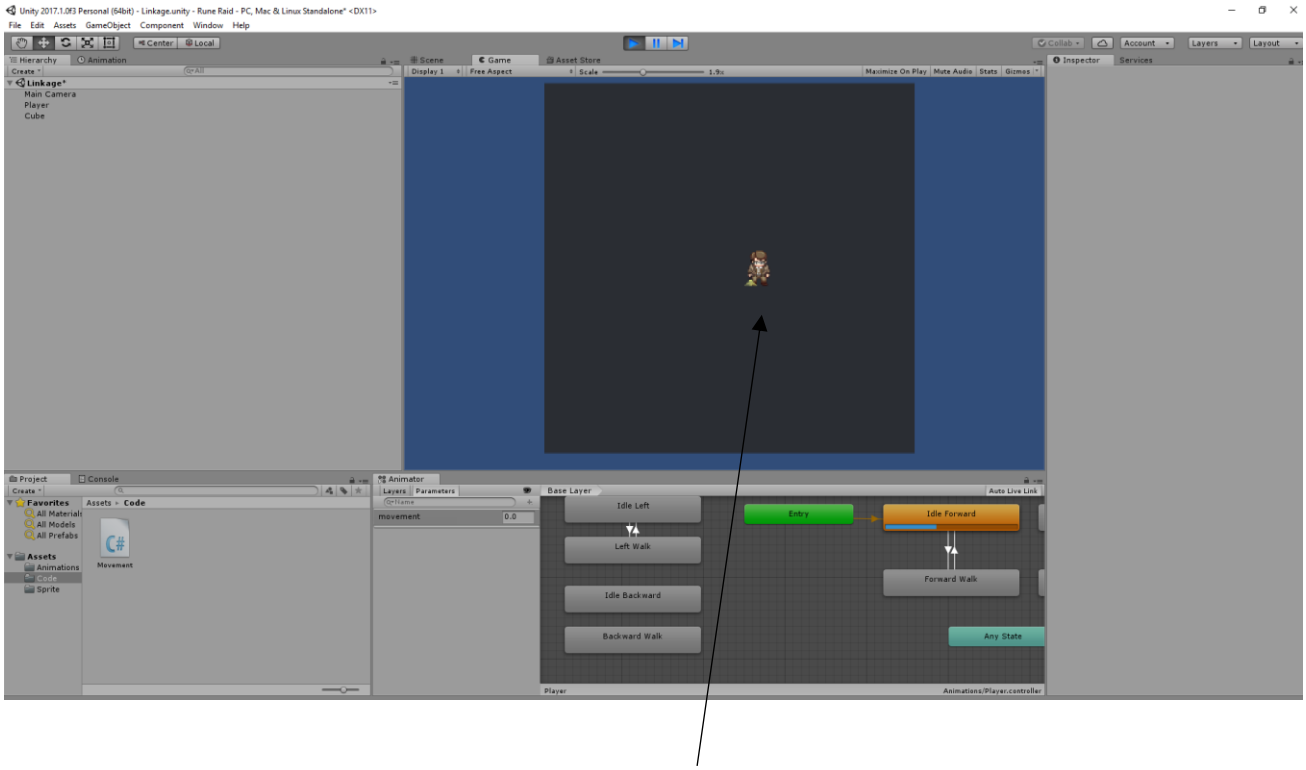
To implement these two dictionary functions within my code this one line is needed.

The line of code needs to be within the "Void Update" because it is code that needs to be run throughout the game and to allow Rune Raid to be constantly updated when the Horizontal function or Horizontal function is run.

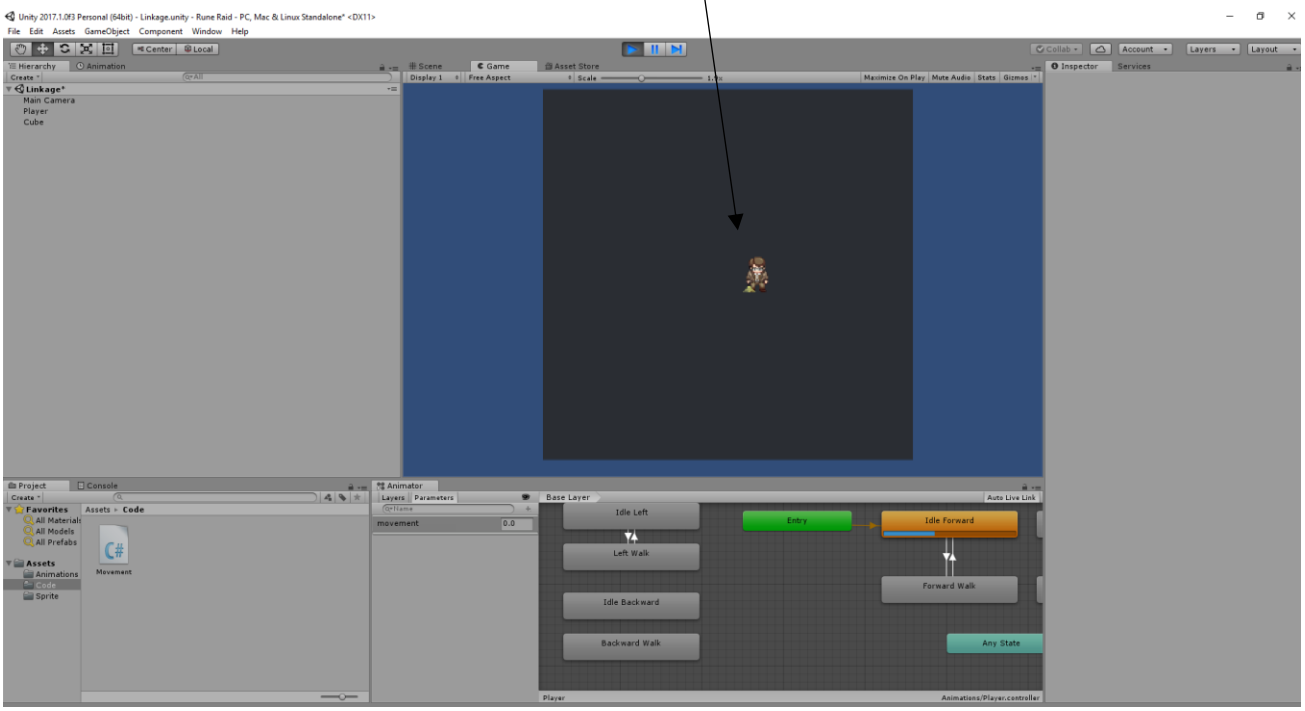
- **transform.Translate** - is used to move desired objects in the direction and distance of the translation. [32]
- **Input.GetAxis -** Returns the value of the virtual axis identified by axis Name. [33]
- **Time.deltaTime –** Explained on page. 76
- **Horizontal –** Dictionary function for movement in the horizontal axis.
- **Vertical –** Dictionary function for movement in the vertical axis.

To function of this line of code will be to move the player sprite, within Unity, in the desired direction depending on which keys are pressed, which are defined in the horizontal and vertical functions.

[32] https://docs.unity3d.com/ScriptReference/Transform.Translate.html
[33] https://docs.unity3d.com/ScriptReference/Input.GetAxis.html

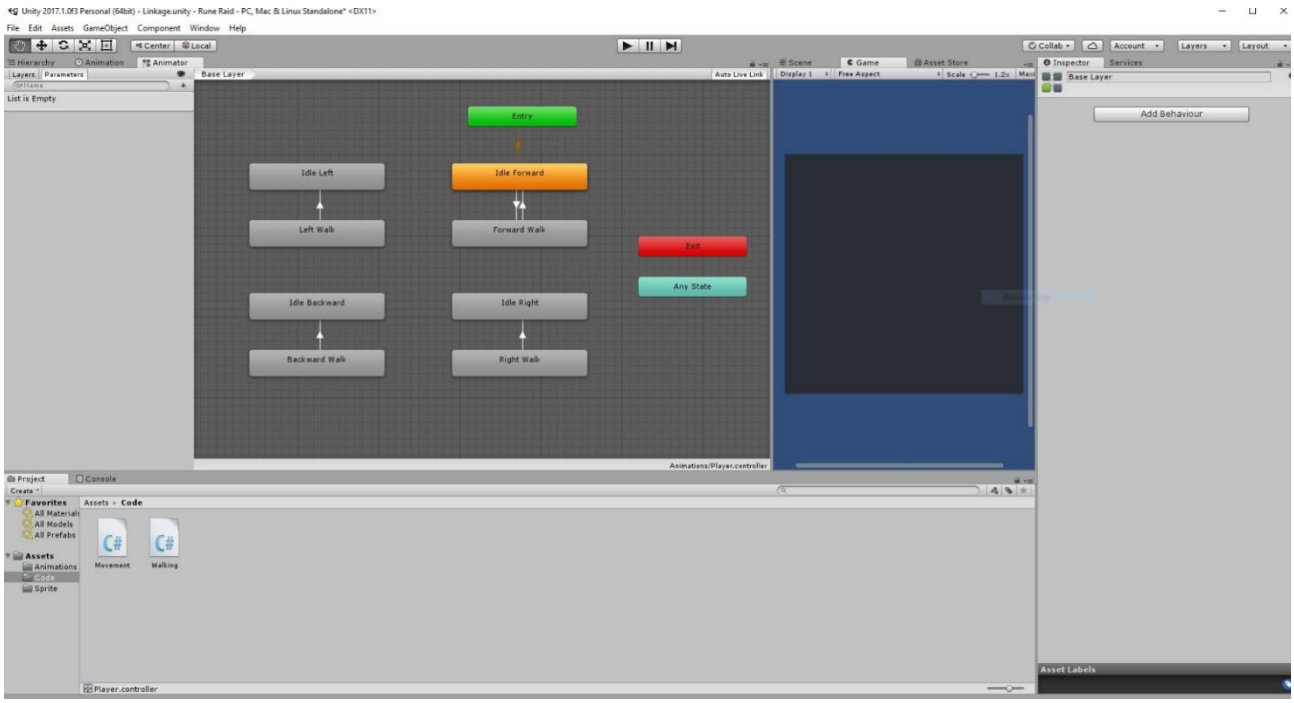


These two screenshots prove that the player sprite moves with the WASD keys.



Now that my movement functions are working, I need to combine them with my animations to make it so that when the player sprite moves via keys pressed, the correct animation will play in parallel. E.g. holding down the A key will move the player sprite left and also play the left walking animation.

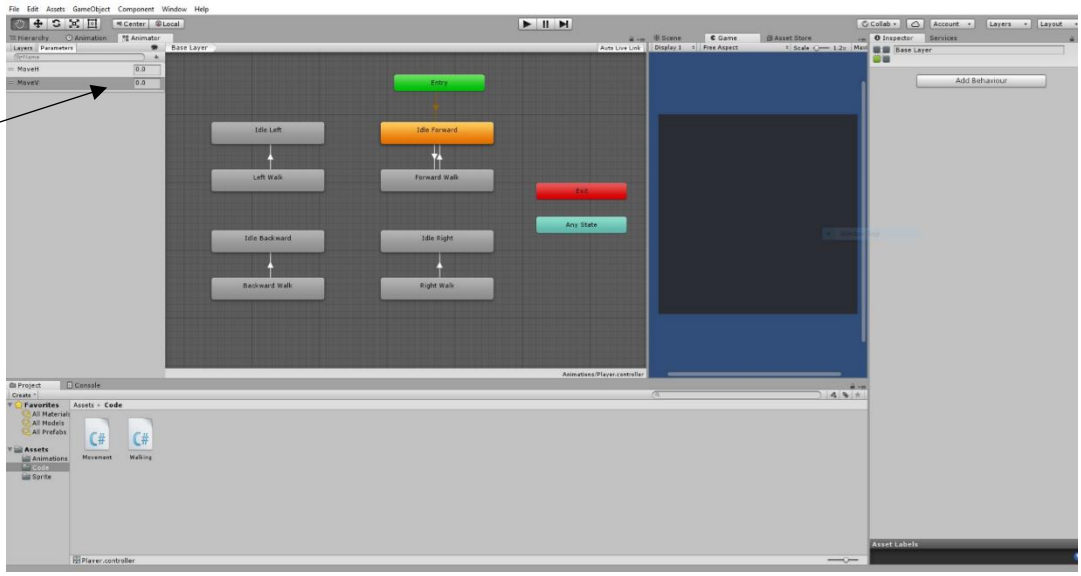
To do this I will to create some more lines of code to directly link the player movement to the animation, whilst also adjusting the linkage.



Firstly, I've linked up some of my animations together, Forward Idle with Forward Walk etc. With this I'm trying to find a way to correctly link them together. Mostly this was me just playing around, trying to figure out the system works.

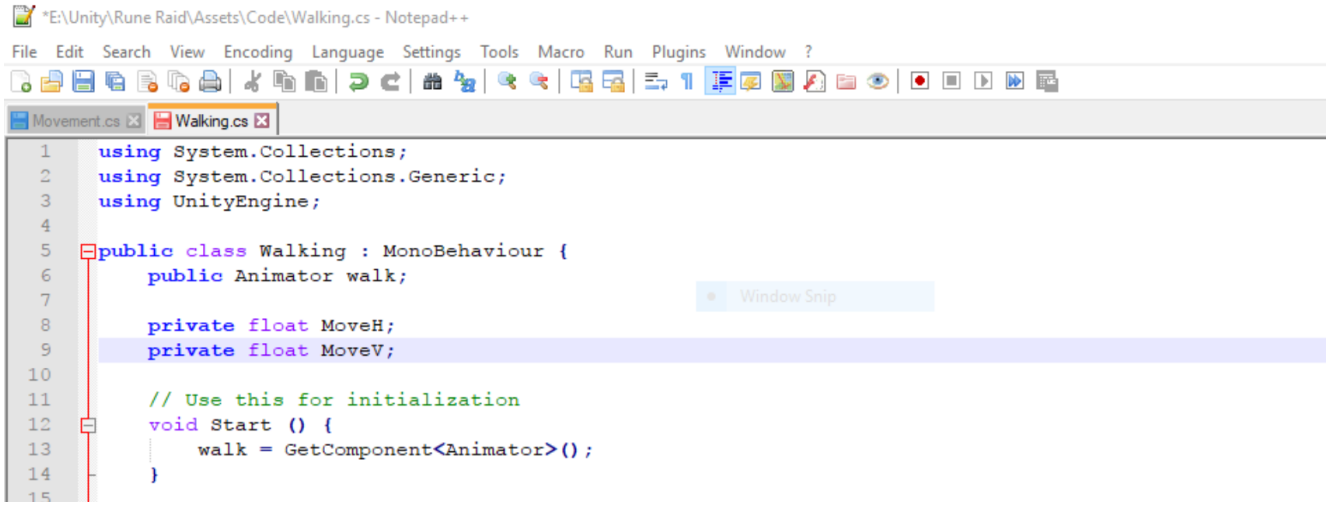
I ended up watching this YouTuber on how he implemented animation within movement. [34] From watching his video I learned a lot on how to system works and decided I knew enough to try and make my own.

I've added two parameters called "MoveH" and "MoveV". These will be used for the parameters for the vertical and horizontal axis. "MoveH" for horizontal axis and "MoveV" for vertical axis.



[34] https://www.youtube.com/watch?v=wdOk5QXYC6Y&t=1937s&list=LLDbedb7gvNzK43UXt9rTlzg&index=4 – Date Accessed 30/12/17

These two parameters will allow me to implement a constant change in animations depending on what keys are being pressed, which is needed for animation in parallel with movement. Now that vie added the two parameters within Unity I now need to implement



I created a new C# file to code the animation. This time within the class function some lines of code are needed to implement the two parameters I just made within Unity.

Public Animator walk – This line is setting the "Animator" within the Unity Engine to the variable "walk". This means whenever "walk" is used the program knows what its interacting with, this time the Animator In Unity.

Private float MoveH – This is setting up the parameter within Unity, it needs to be a float because the movement in the axis is on a slider between -1 and 1 and so it must be able to understand non integers such as -0.1.

Private float MoveV – This is the same as MoveH but with the vertical axis instead.

Walk = GetComponent<Animator>(): - This is setting the variable "walk" to get the Animator component within Unity, in other words to link the variable "walk" to the Animator to open it up, to allow modification. This is within "void Start" because it needs to be set at the start of the class and never needs to be modified again so does not need to be within "Void Update".

After setting up the variables, the next stage is to add the code within the "void Update" to manipulate the parameters and coordinates of the player sprite.

```
16      // Update is called once per frame
17      void Update () {
18          if(Input.GetKeyDown("s"))
19          {
20              walk.Play("Forward Walk",-1,0f);
21          }
22          if(Input.GetKeyDown("w"))
23          {
24              walk.Play("Backward Walk",-1,0f);
25          }
26          if(Input.GetKeyDown("a"))
27          {
28              walk.Play("Left Walk",-1,0f);
29          }
30          if(Input.GetKeyDown("d"))
31          {
32              walk.Play("Right Walk",-1,0f);
```

These lines of code are all very similar because they all do the same thing except for they need a unique input value and they execute a unique animation.

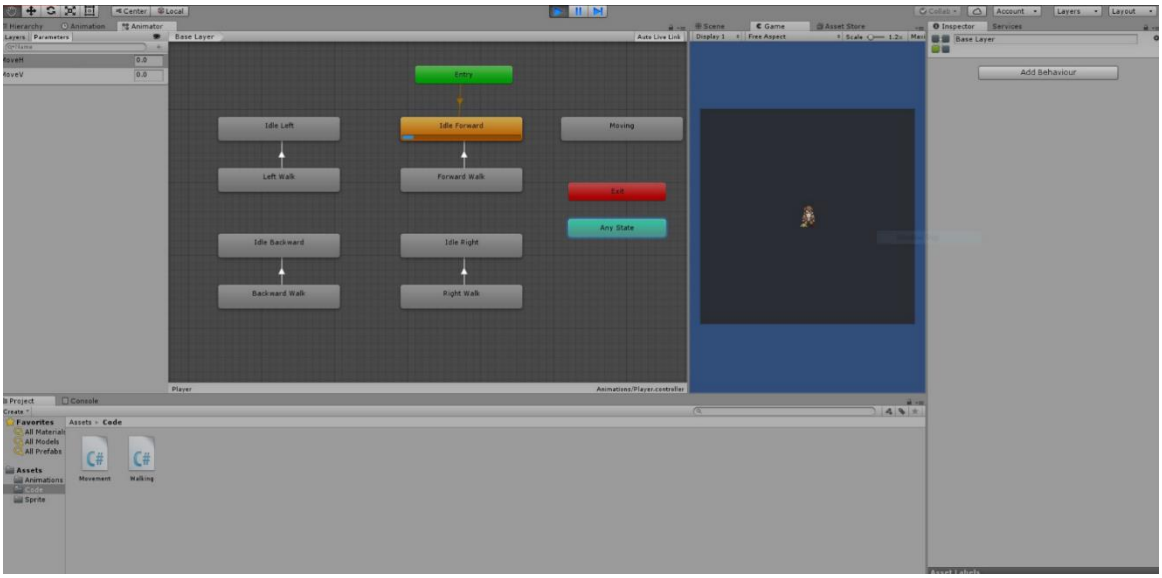
If(Input.GetKeyDown("s")) – This returns the value TRUE when the defined key is being pressed. So when the S key is pressed the function will return TRUE and execute the if statement. [35]

walk.Play("Forward Walk",-1,0f) – This will execute the Forward Walk animation within the Unity Engine's Animator. The "-1" is used to identify that the Animation is located in the base layer of the Unity Engine Animator. "0f" is used to tell the code to play the animation from that start, at frame 0.

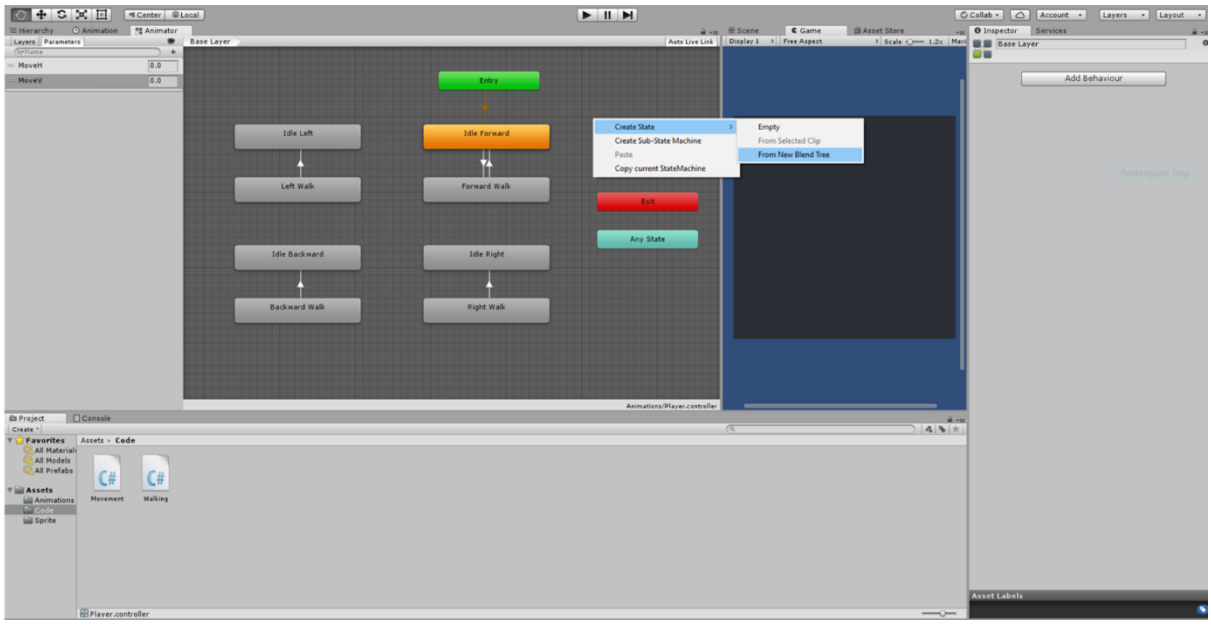
The rest at the lines are just the same but with a different key and animation.

[35] https://docs.unity3d.com/ScriptReference/Input.GetKeyDown.html - Date Accessed 30/12/17

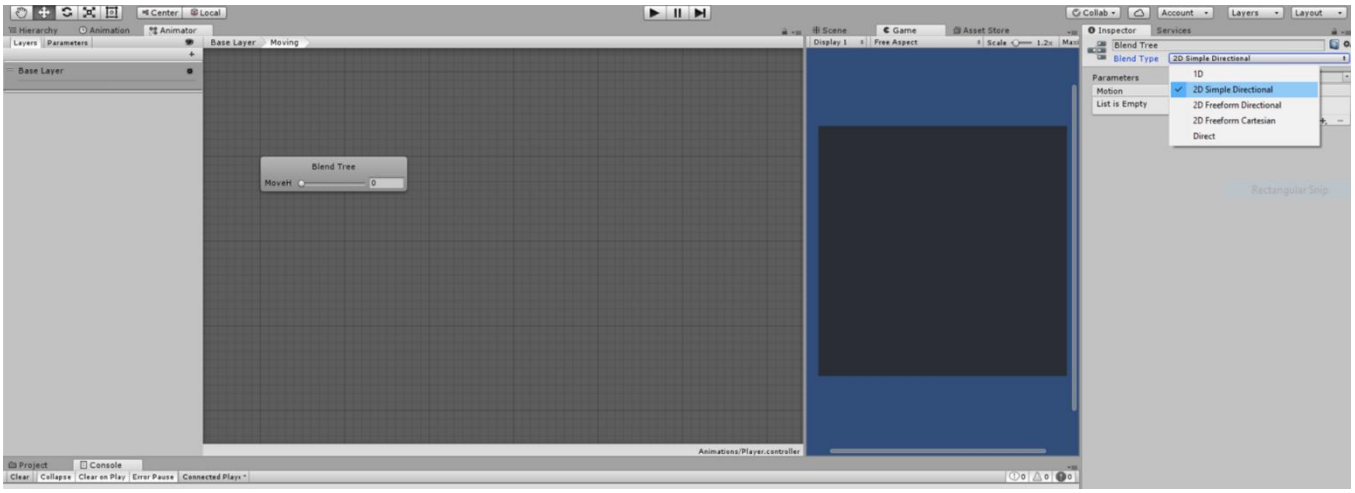
Now that I've coded the movement to be linked to the animations, I now need to structure the Animator within Unity to allow the code to work.



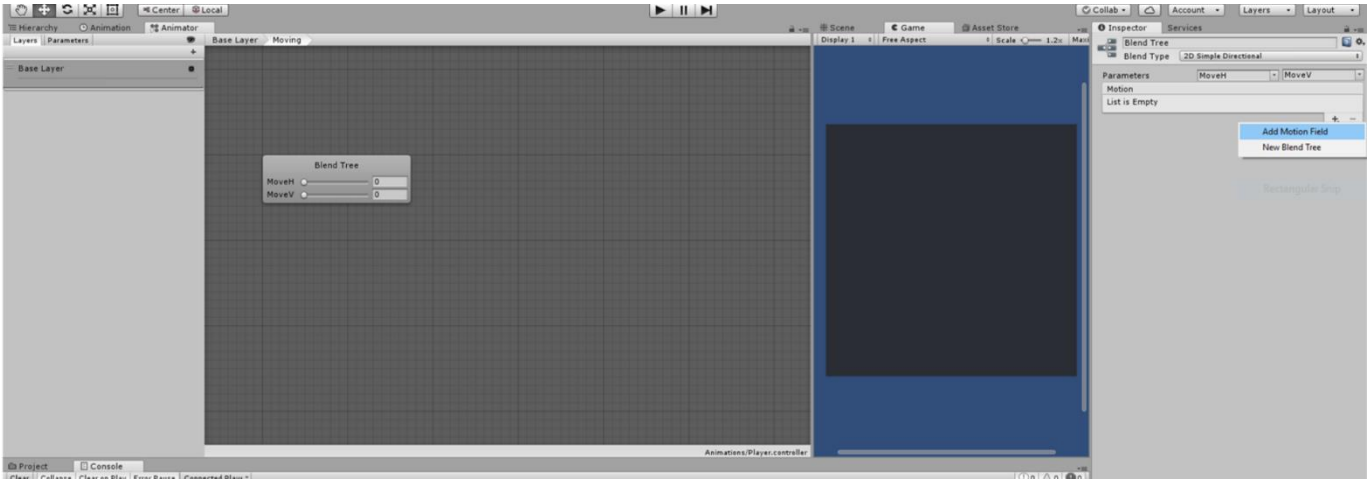
So far when I move forward by pressing the S key on my keyboard, the player sprite will move until I release the button, however the animation will only run once and then stay on idle forward. This is clearly not what I want and so it seems I've got to work out how to make it so that the animation loops until the key is released and then it plays the idle animation.



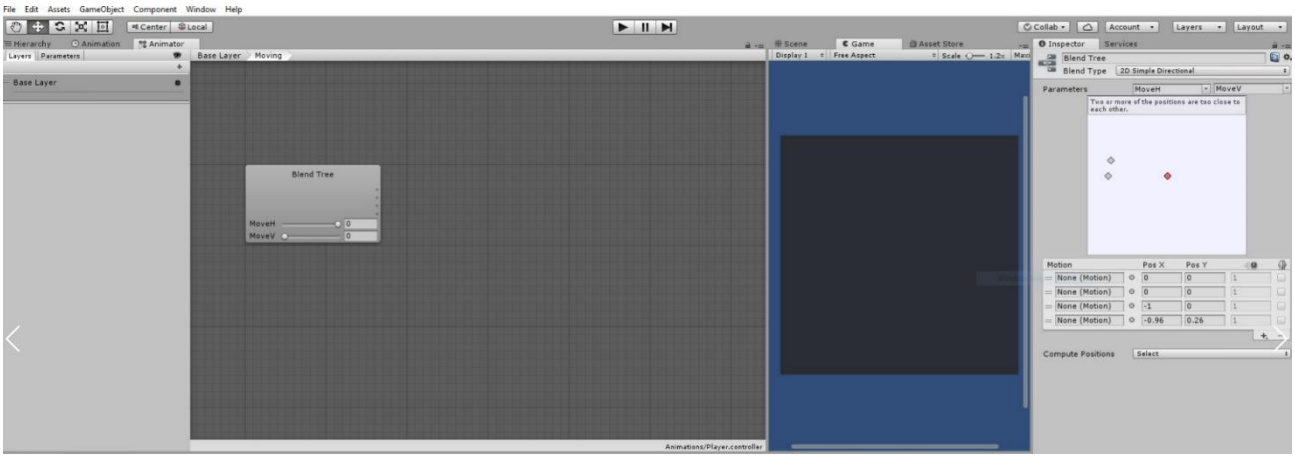
The YouTuber that I watched used a Blend tree to get his animations to work an so I've decided to use a Blend Tree as well to see if I can also use it for my animations.



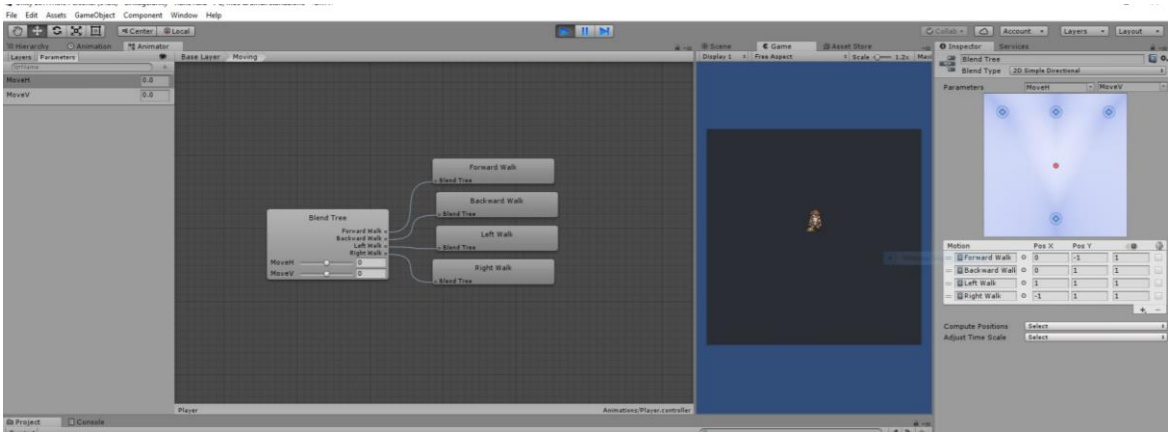
I've called the Blend Tree "Moving" and it also has to be set to "2D Simple Directional" because it is only using a 2D axis, X and Y.



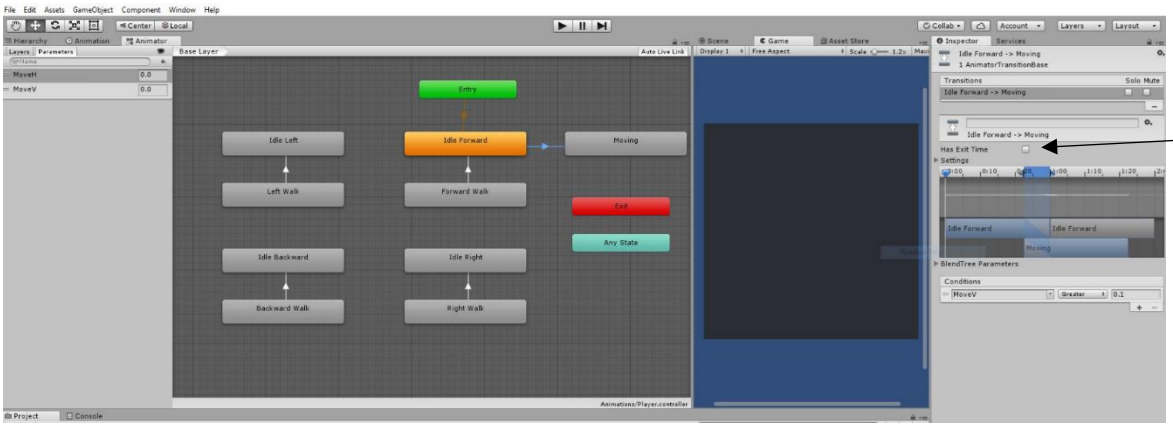
A Blend tree is modified in a new layer of the Animator, shown above, and the Blend Tree has the two movement parameters assigned to it. Next, 4 motion field's needs to be added to allow movement in all four fields; negative X-axis, positive X-axis, negative Y-axis and positive Y-axis. This is done by clicking the + icon in the motion section and click "Add motion field". Also the two movement parameters need to be added to so that they can be used within this Blend Tree.



At first this is what all the motion fields look like when added. However they need to be modified to link it an animation and define when it runs. E.g. the Forward Walk animation runs when the Y position is -1.



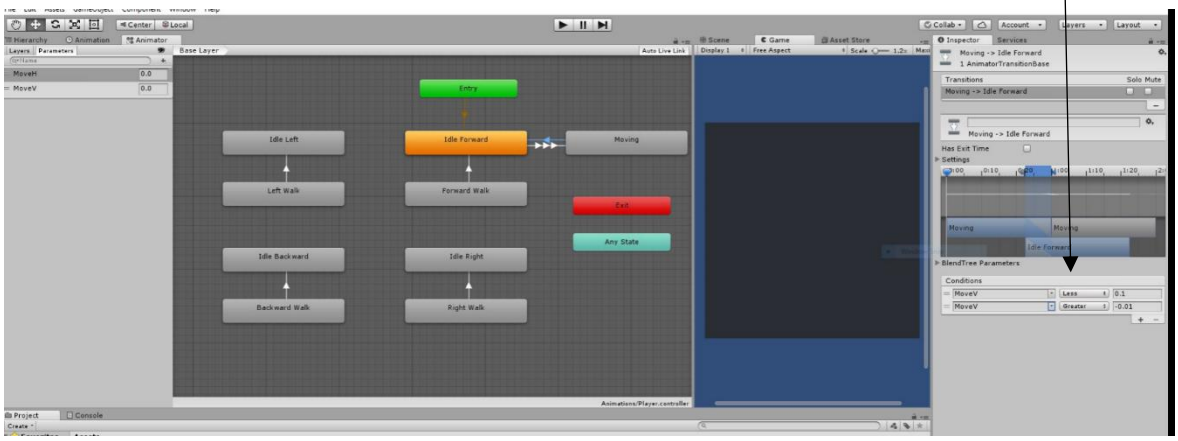
This is what it looks like when all the motion fields are set correctly, note that now the filed can now go negative to -1 and go to positive +1. This will allow for movement in all directions. The position sections are set up correctly so that the correct animation runs when the player sprite coordinates are moving.



There is a box called Exit Time that needs to be unclicked because this means that now the animation can stop at any point when another animation is needed.

The linkage now needs to be set up so that it only runs when the parameters are currently certain values, this is down by adding a link and then modifying it within the right-hand window. To do this, two conditions need to be added which are linked to the two movement parameters. The two conditions are set so that when MoveH and MoveV are a certain value it will move to the next animation via the transition link.

The animation Idle forward is set to the start animation, highlighted orange, because this is what I want to the player model to look like upon loading into a level. Then the linkage is set so that when the MoveV parameter is greater than 0.1 it will transition to the Blend Tree "Moving". Another link is also needed to be connected to the "Moving" Blend tree so that I will transition when the MoveV parameter is less than -0.1. With both of these links, the Blend Tree will run when the Y axis for the player sprite is moving. A linkage back is needed to link back to the Idle Forward animation once the movement has stopped. This is done by adding a link back to Idle Forward with two conditions being, MoveV less than 0.1 and greater than -0.1. This means that it will transition back when the positioning in the Y axis is -0.1<Y>0.1, in other words 0.



This is what it looks like when the Blend Tree is fully implemented.

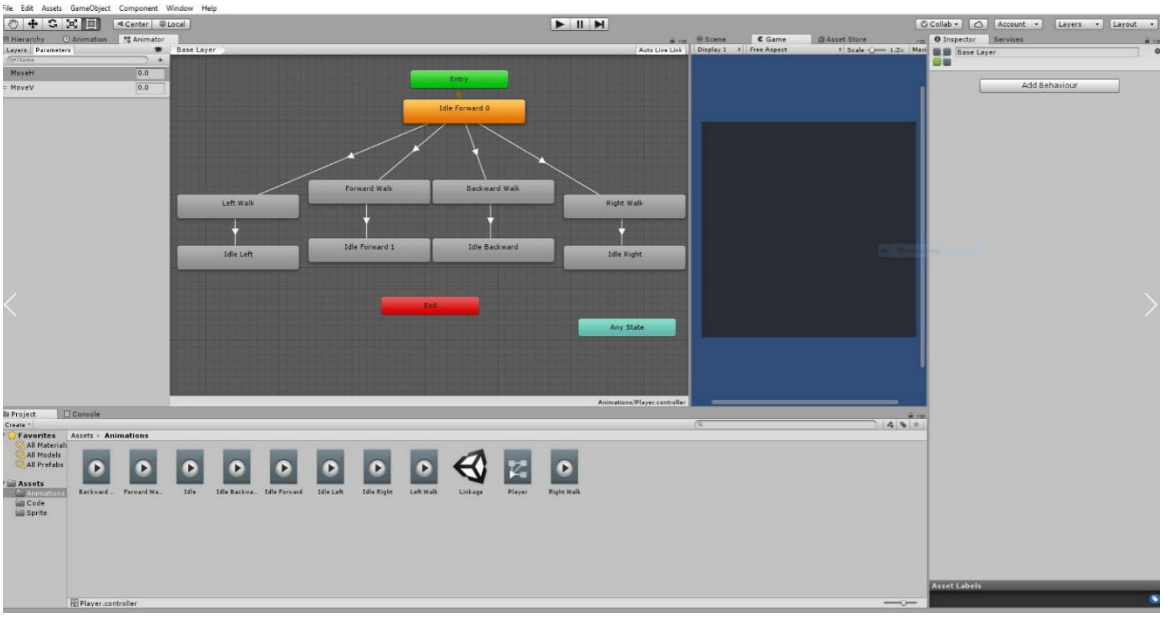
Another blend tree is needed to implement the exact same but for the horizontal axis with MoveH.

With this new Blend Tree added the other animations become redundant and so I've removed them from the linkage.

So currently, my player sprite can move in all directions and when moving it will play the correct animation and then when the movement stops it plays the Idle Forward animation. This is fine but it is not fully correct because there is a problem. When the player sprite stops it plays the Forward Idle animation every time, even if he was just walking right. I want it so that when he stops moving right it plays the Idle Right animation and not the Idle Forward animation. There is also a slight problem when a movement button is released and a new movement button pressed immediately after. It still plays the Idle Forward animation in-between the new animation of movement. This looks very unprofessional and looks like a glitch in the animation.

Therefore, I need a different solution to fix these two problems.

After some time trying to fix the problem I gave up on the Blend Tree aspect and decided to make a completely new animation structure, using all my knowledge I've gained from creating the previous structure.



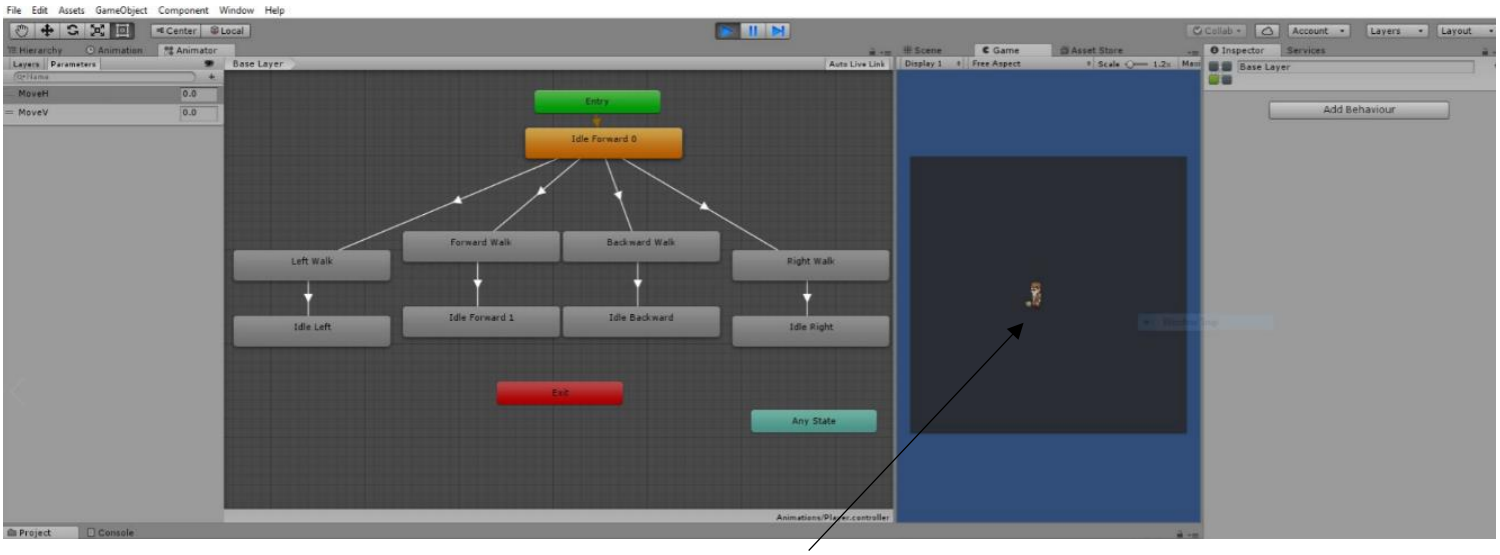
This is what I ended up creating, it starts with the Idle Forward animation just like before however, these is no links that link back to the Idle Forward animation. This means that upon loading in the player Spite will have the Idle Forward animation playing but then after any movement it will then transition to the correct animation.

I've added the four walking animations and their corresponding Idle animations. The start animation Ide Forward 0 is linked to all four of these walk animations and have their own specific link conditions. Example, the link between Idle Forward 0 and Left Walk is that is will make the transition when MoveH is less than 0. Meaning the A key is being pressed as the MoveH parameter will be changing to -1.

Each walk animation is linked to its Idle via one link. This link has two conditions which are, less than 0.1 and greater than -0.1 for its corresponding axis, in other words movement is roughly 0. Example, the link between Backward Walk and Idle Backward has the conditions, MoveV less than 0.1 and greater than -0.1.

Each link also has it so that it has no exit time so that each animation can stop at any point and transition to the new animation needed at that specific time. Example, if the A key is pressed and the Left Walk animation plays but then right after the key is released before the end of the animation, it will stop and play the idle animation. With an exit time enabled, it will mean that even though there is no movement it will still play the animation till the end, making it look like the player sprite is walking on the stop until the animation ends and then the idle animation is played, which is not what I want. Therefore, the exit time is disabled.

With this structure it will mean that the player sprite can now stop moving and then play the Idle animation of the previous movement, example moving left then stopping will then play the Idle left animation.



As shown above the player sprite can now stop moving and facing the previous movement direction, in this case left (it can also do this for the other three directions). It also doesn't play any idle animations when transitioning between the walking animations, it just plays then next walk animation smoothly.

However, when testing this structure I found out that it doesn't play the animations when using the arrow keys on the keyboard. I found out the reason was because I haven't coded any if statements for the arrow keys being pressed, I had only coded for WASD. Therefore, I need to add four more if statements, each one for each unique arrow key.

```
34      if(Input.GetKeyDown("down"))
35      {
36          walk.Play("Forward Walk",-1,0f);
37      }
38      if(Input.GetKeyDown("up"))
39      {
40          walk.Play("Backward Walk",-1,0f);
41      }
42      if(Input.GetKeyDown("right"))
43      {
44          walk.Play("Right Walk",-1,0f);
45      }
46      if(Input.GetKeyDown("left"))
47      {
48          walk.Play("Left Walk",-1,0f);
```

These are the four new if statements that I've added to allow the parameters MoveV and MoveH to be manipulated by the arrow keys as well as WASD.

Now the player sprite moves with the keys WASD and arrow keys whilst playing the correct animation in parallel.

The website I used to find out what the arrow keys are called is this.[36]

[36] https://docs.unity3d.com/ScriptReference/Input.GetKey.html - Date Accessed 30/12/17