A LEVEL PROGRAMMING PROJECT

Owen

CONTENTS

ANALYSIS

OUTLINE OF THE PROJECT

My project is based on a deserted island where you'll have to use science, skills and chance to survive. I am using my personal knowledge from my past science lessons, new scientist magazine, friends/siblings, personal gaming knowledge and the WWW (World Wide Web) to gather my information. For example, using the process of distillation (separating a compound into different substances depending on their boiling point) which I would use for separating salt and water allowing the player to hydrate instead of having pure sea water what will cause dehydration. This idea can be explored in many different ways due to the massive spectrum of possible items to find, entities to kill and mainly science processes to explore. There will be a brief story to the game but will have alternative endings depending on what paths you take.

THE PROBLEM

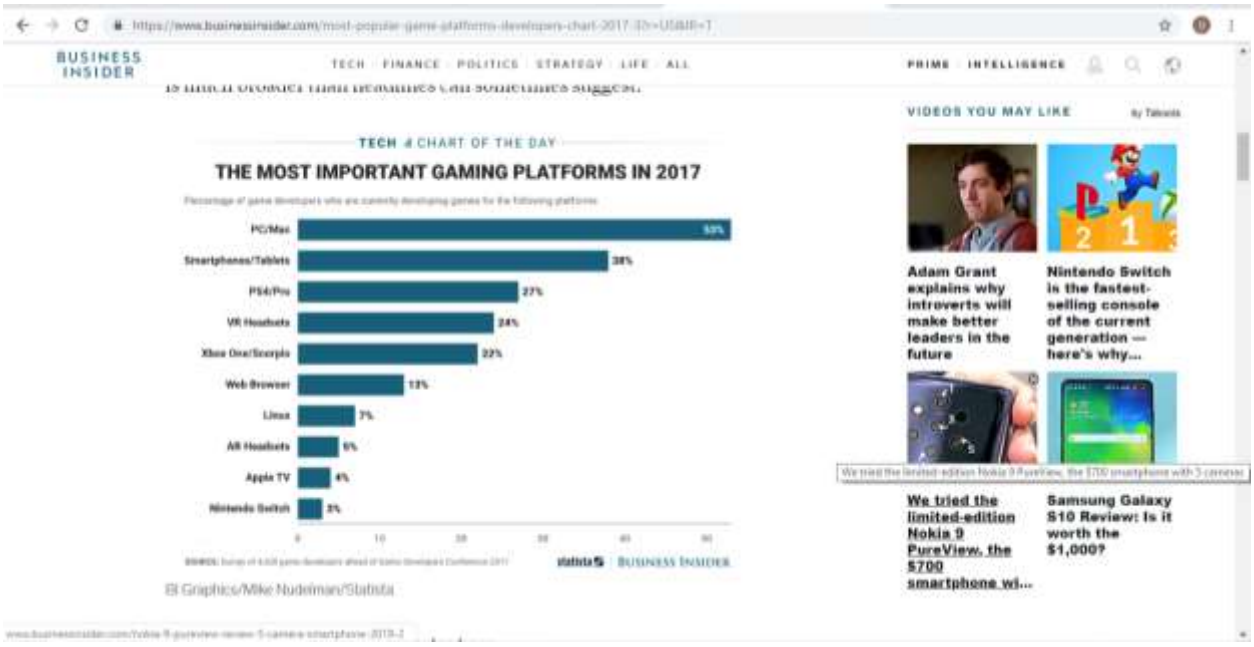
After many hours of playing multiple different game types (Adventure, Real-Time Strategy (RTS), Puzzle, Action, Stealth Shooter, and Combat) I have realized that the majority of survival adventure games are made to survive entities, and not take into account other dangerous such as savior heat stroke, dehydration, tides closing/opening and dramatic temperature plummets. This is where I thought of using science to survive and focus on real world problems you would face if deserted on an island, making this game unique and different to other survival adventure games. Films/series have looked deeply at this idea (Bear Grylls "Man vs Wild") and have been very successful with the outcome as well as very popular books ("The knowledge") but never have produced in a game.

IDENTIFICATION OF STAKEHOLDERS

My key stakeholders for this project would be 16-20-year old. I have come to this conclusion due to most people around the age of 16-17 understanding the key concepts of science experiments allowing them to understand and develop their knowledge on experiments used to survive. As I am 17 years of age with A's in Biology Chemistry and Physics (GCSE)as well as a 19-year-old sister undertaking a biomedical degree (A*'s in all GCSE sciences and A's in biology and chemistry in A level) at Sheffield University I'll be able to know what different age groups will know and understand the concepts of science I will be looking at. This will allow me to focus on this age group and know what they'll enjoy and dislike about different science experiments. I have also specifically picked out one of my class mates due to my prior knowledge of him enjoying science classes where he understood every topic and wasn't afraid to voice his opinion on matter allowing me to alter my game to the user needs. My first stakeholder is Luke Callison Bailey who is a computer scientist and game designer going onto studying Game development at university. This therefore shows a keen insight to knowing what user's would like in a game and would could be improved. As a hobby he spend most of his time trying a different genre of games as well as being in the correct age bracket at the current age if 18 going onto being a perfect candidate for one of my stakeholders

THE TARGET PLATFORM

My target platform is the PC (Personal Computer), I came to this conclusion after conversations with my stakeholders and research into similar games to what platform they're played on. Business Insider also shows that in 2017 Pc/Mac was the most popular gaming platform, therefore the game I'm producing will have a bigger audience compared to other gaming platforms. One of the biggest reasons I came to this overall conclusion was due to the simplicity of creating a PC game. Compared to other platforms such as the Xbox, PlayStation, Nintendo, iPhone and android creating a game for the pc is so much simpler because of the research out there to create entities, items, maps and the software used, which has been worked on so much that you can create basic games with little knowledge to complex games using the web. X code used to make IOS apps also had a vast amount of knowledge on making a game but the game I wanted to produce couldn't run on the average iOS device.



USER NEEDS

Questions- I created a survey using survey monkey and sent it to my computer science class to ask the final questions on the user needs. Below are the questions and the results of the survey. I chose my computer science class as my audiences due to the majority being heavily into games and loving playing them. They would have played the most modern and classic games and tried a variety of different types of games allowing me to get an accurate response. They also perfectly fit the age bracket I am looking at making them the ideal audience to send this survey.

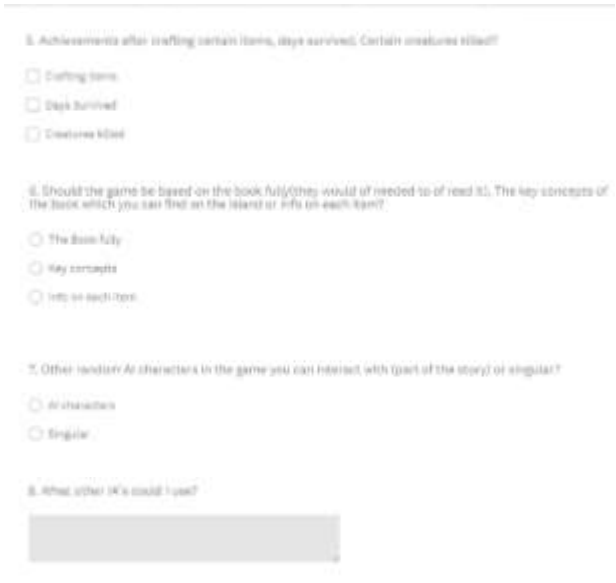
THE QUESTIONS USED FOR THE SURVEY

1.1-Health bar or lives? I asked this question to find out which GUI is less like a game and more like the real anatomy of a person. To see which simulates a human life more

2.1-A Minecraft crafting interface or craft item button? This was to decide if getting specific materials in a certain shape made the game more realistic or was unnecessary and a simple craft button with required materials would do.

3.The character wakes up on an island, how did they get there? To see if user's wanted a backstory to the game and if so what backstory would they find most engaging.

4.A large island or a small remote island? This question was picked as either survival games from research are small, therefore lots of detail in a small area or large making the gameplay more difficult. I was separating users to which they preferred as too large could become too hard and small could get tedious.



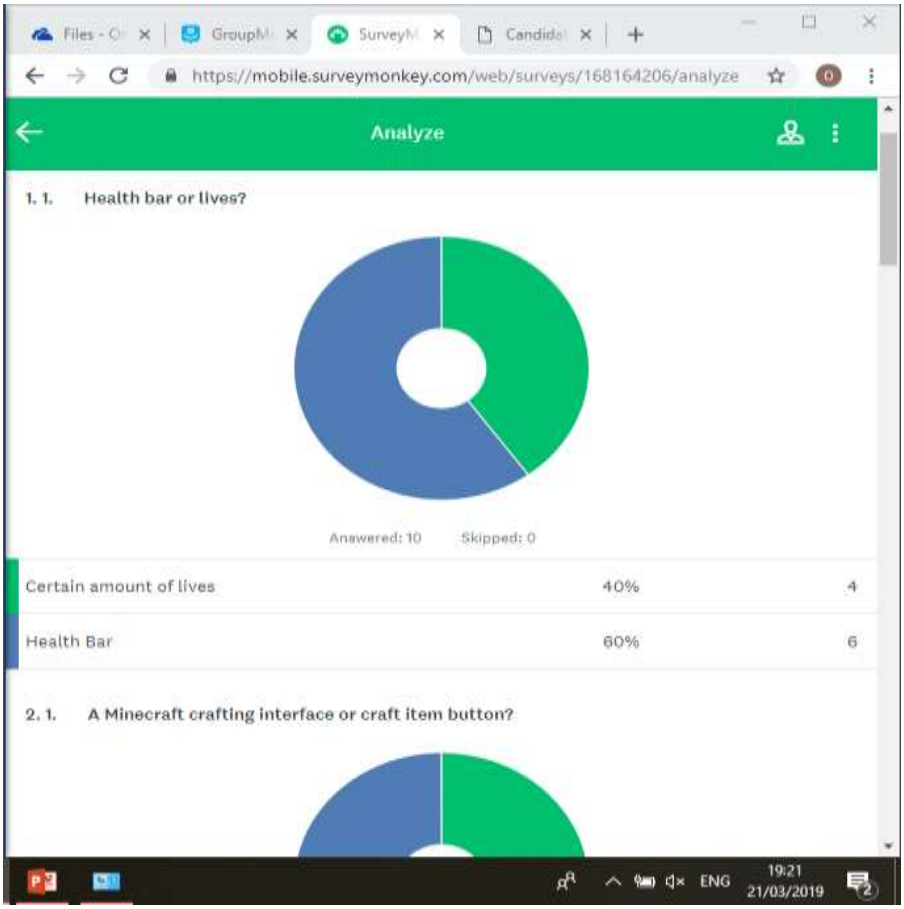
5.Achievments after crafting certain items, days survived, certain creatures killed? Doing his allowed me to see which achievements player wanted to work towards and compare with against other players.

6.Should the game be based on the book fully (they would of needed to of read it). The key concepts of the book which you can find on the island? Therefore asking this question helped me decide if notes around the island should be added for more enjoyment or the game (so that the book doesn't need to be read) or to be completely based on the book making it more challenging for the user. I added a third option to disregard the book and find infomation on specific items as I thought users who play these types of games don't rely on third part resources

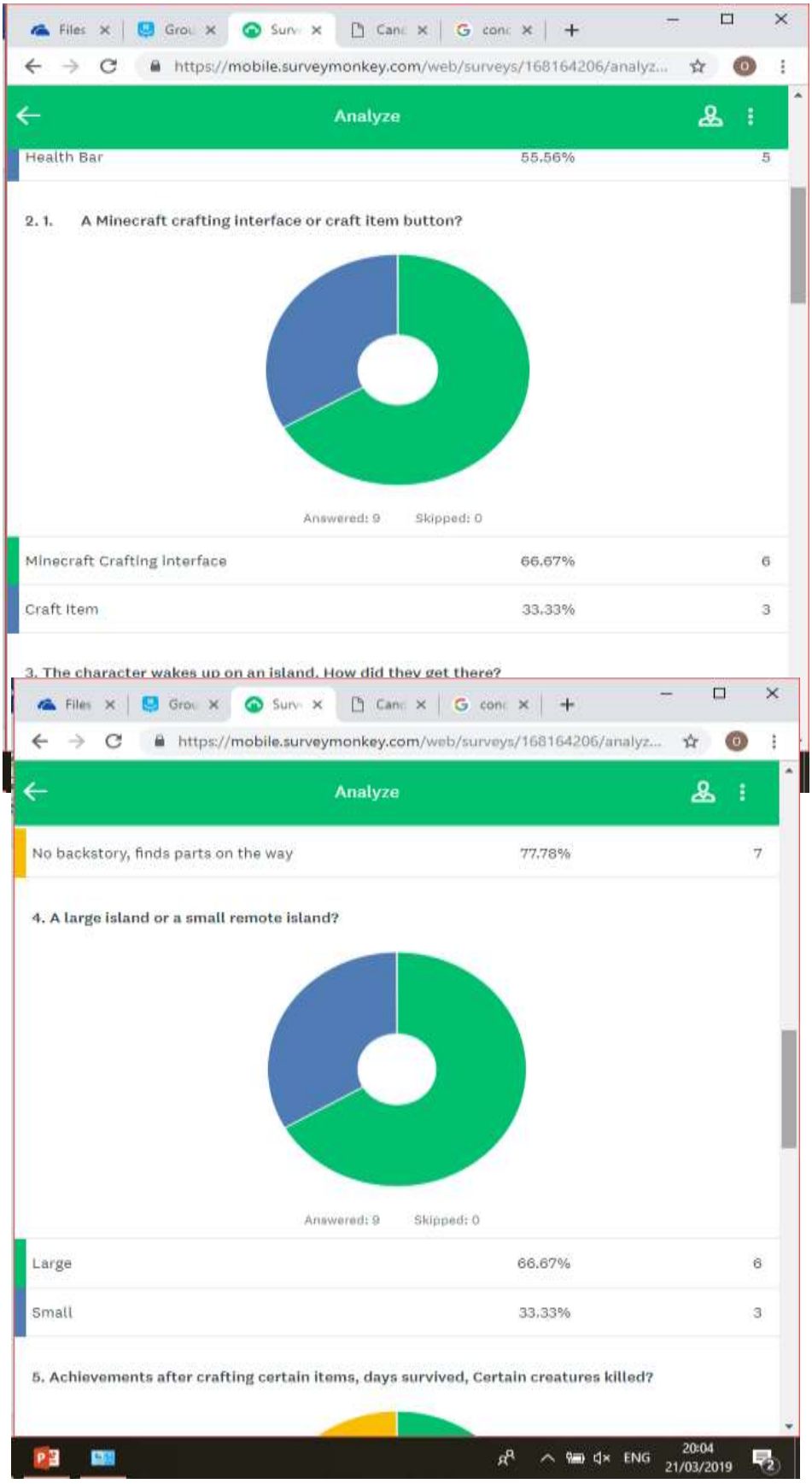
7.Other AI characters in the game you can interact with (part of the story) or singular? I asked this question because a survival game is normally based on one person's objective to survive and not interacting with other people. However, to keep the game interesting I asked this question which broadens it to animals attacking you as well and if user would want other characters involved not only for difficulty but as part of the story.

8.What other IA's could I use? This elaborated on my previous questions to which AI's they would want in the game if what they picked previously was to

ANSWERS TO THE QUESTIONNAIRE

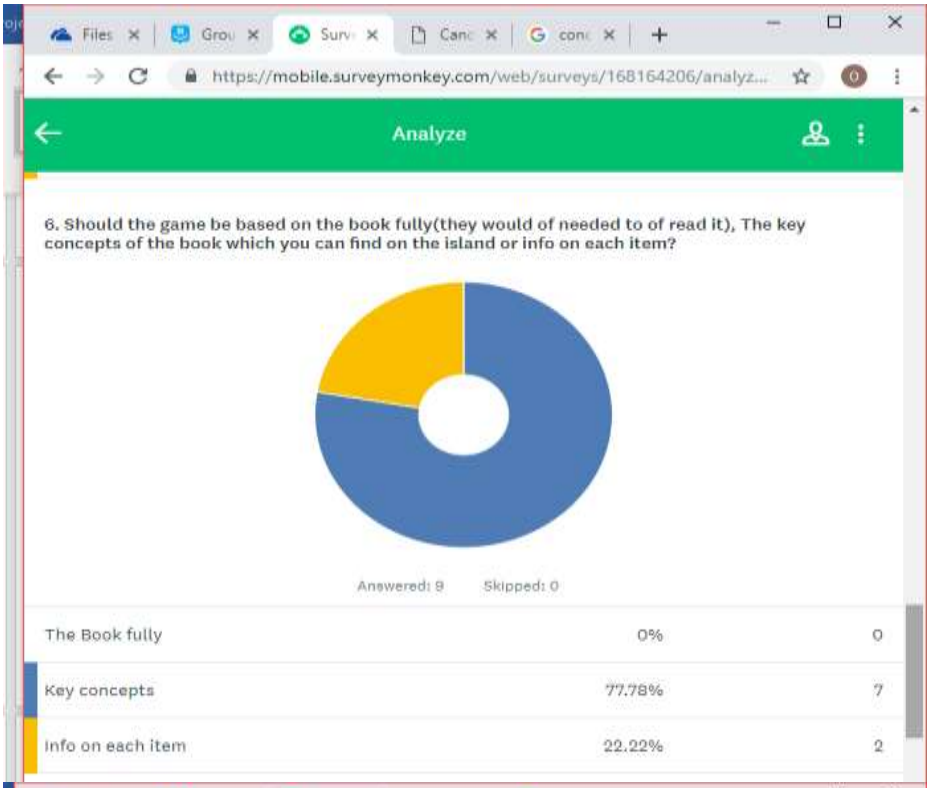


1.1Health bar or lives? This showed health bar to have the majority 6/10 -due to th8s being nearly a 50:50 split I will make my own decision on this expect f the game.

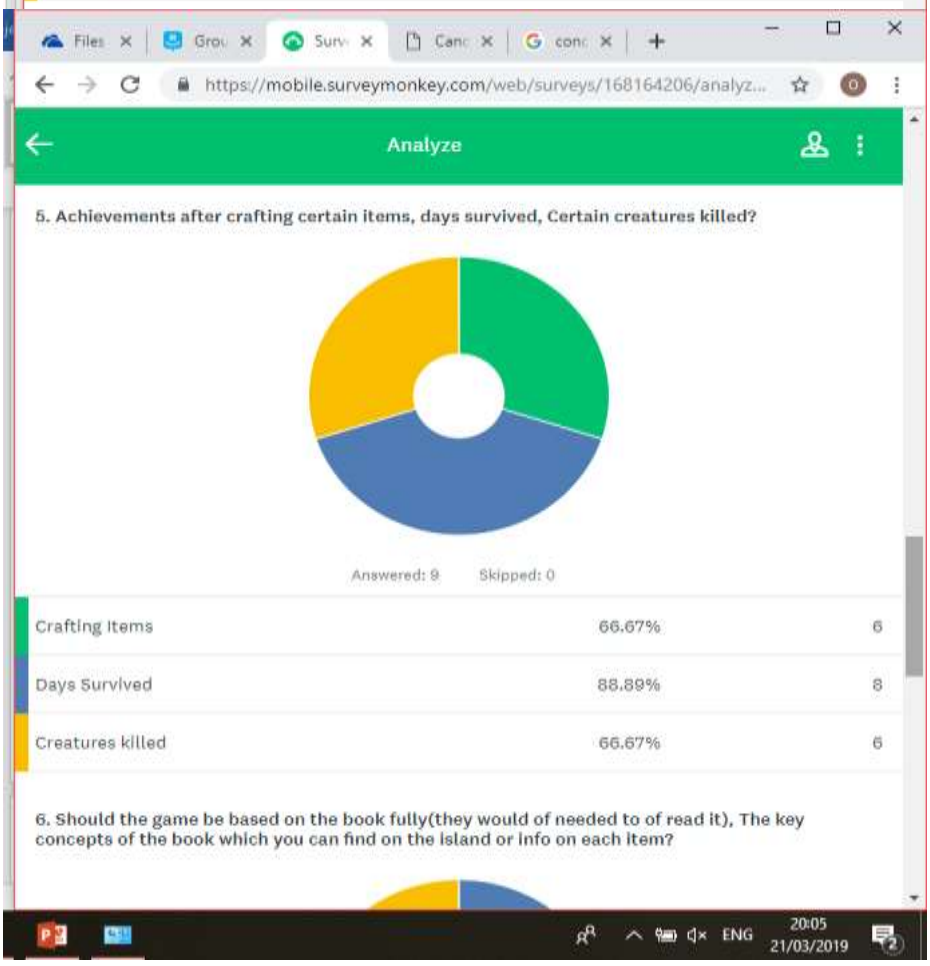


2.1-A Minecraft crafting interface or craft item button?-This showed Minecraft crafting interface to have the majority 6/9 -Minecraft interface

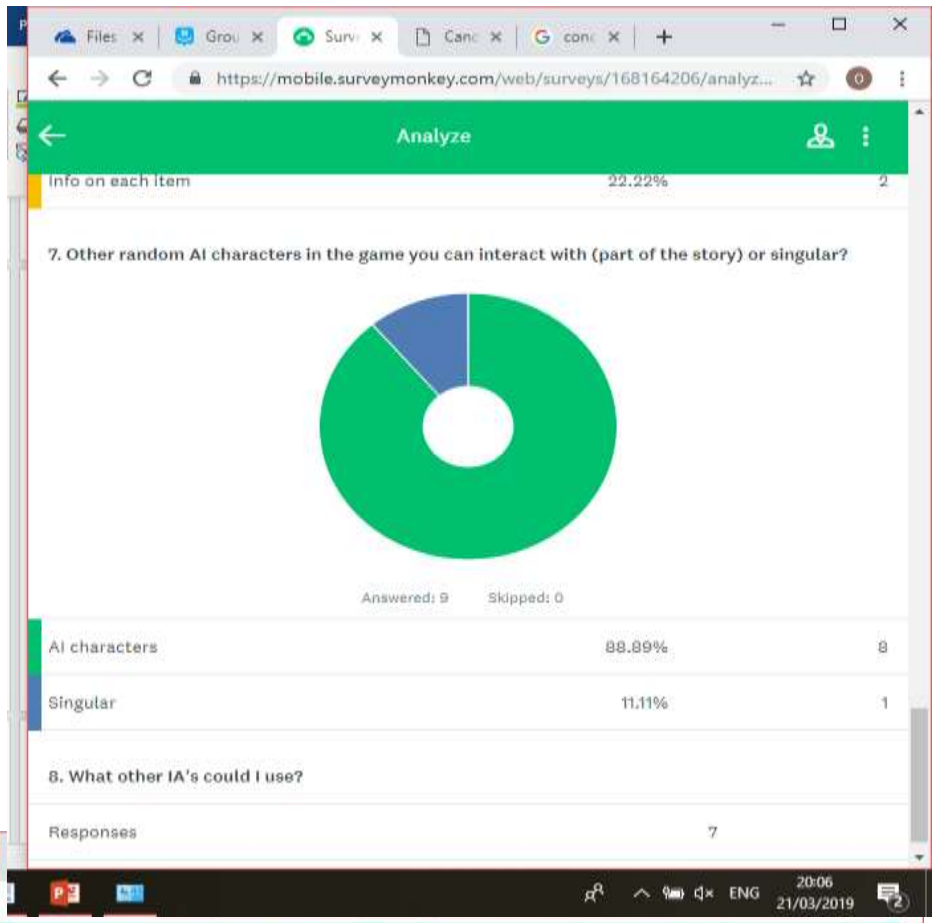
4.A large island or a small remote island? This showed a large island to have the majority 6/9 - large



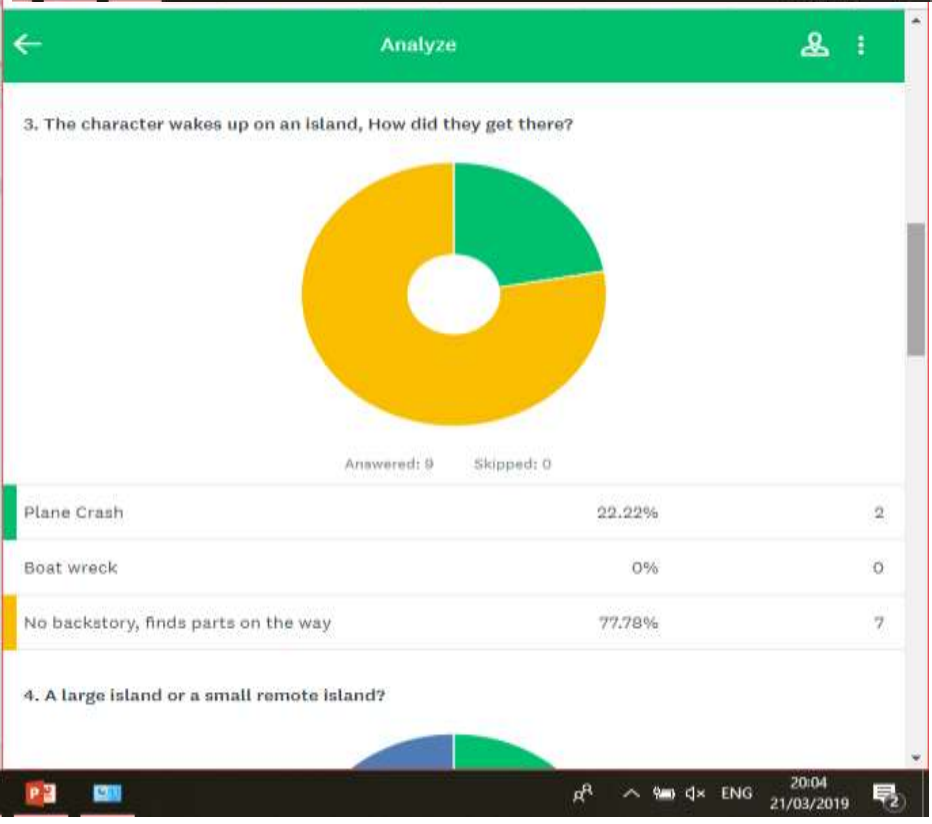
6-Should the game be based on the book fully(they would of needed to of read it), The key concept of the book which you can find on the island or info on each item?-This showed key concepts to have the majority 7/9 - key concepts



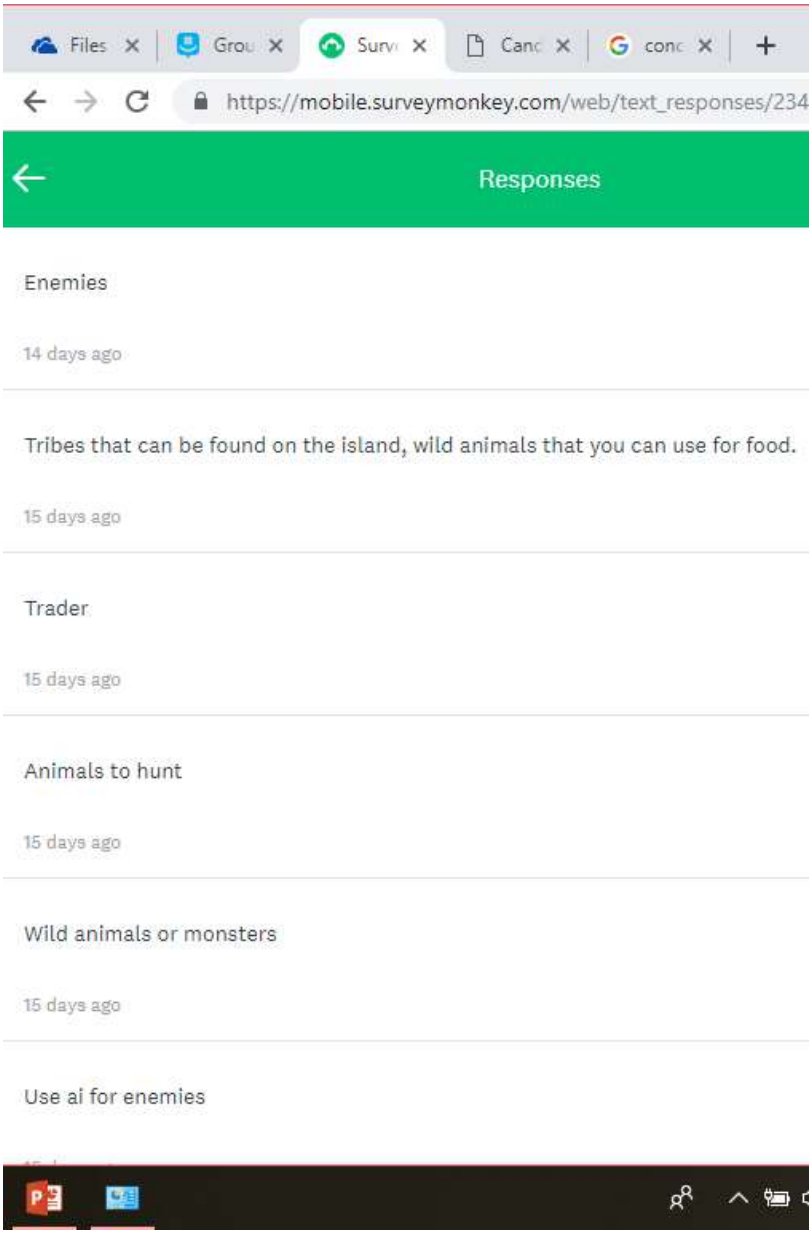
5.Achievments after crafting certain items, days survived, certain creature killed? This showed days survived to have the majority 8/20 -Days survived

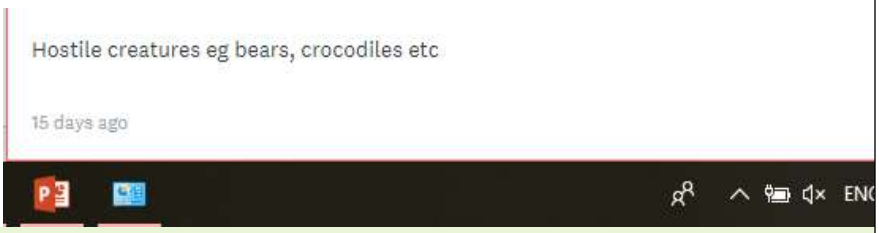


7.Other random AI characters in the game you can interact with (part of the story) or singular?- This showed AI characters to have the majority 8/9 - AI characters



4The character wakes up on an island, How did they get there? This showed a no back story find parts on the way to have the majority 7/9 - no backstory





8.What other IA's could I use? Enemies were repeated twice as it is a survival game I presumed the users meant animals as enemies or the different aspects of the weather ect. 34more users specifically said an animals (one asking about monsters as well) such as wild boars tigers bears crocidiles ect. 2 separate user's mentioned about traders like in Minecraft or tribes to interact with to be allowing the story to develop in the game.

RESEARCH INTO EXISTING SIMILAR PRODUCTS

The main the problem with current existing games is how inaccurate they're to r
you would need to survive in the real world) the games Minecraft, Ark: Survival E

to what I'm developing, they either incorporate the survival aspect of the game, the layout/map or the game type but none of them have the science used to survive in these situation which I and many others would enjoy exploring and playing . This makes my game unique to other games in the market allowing me to undertake ideas which have been brought to life in books (which have had pre-research into what attracts an audience and leaves them recommending the book to others) but not pre- existing games. I have researched and played all the listed games and specifically looked into the mechanism of the movements, the advantages, disadvantages, the platform, how they're controlled, and the view on the character, if the view is interchangeable as well as reading the book (which is heavily based on the game) and highlighted the key aspects I'll be using in the game. This allows me to bullet point what the gamers like and dislike, what should be in the game what I shouldn't and many other key features I would use from other games.

Minecraft

One of the most similar games is Minecraft, this game as been defined by the Telegraph as a "Lego style adventure game" which "The video game puts players in a randomly-generated world where they can create their own structures and contraptions out of textured cubes", one key aspect of Minecraft is the game mode survival "In 'survival' mode players must find their own building supplies and food. They find resources to craft tools while avoiding moving creatures (mobs) such as zombies and giant spiders". The main reasons why this game is similar to the game I'm producing is because it's an adventure game, its graphics isn't too complicated, you use raw materials to craft items and the process's used are very similar to the real world (the main feature of my game).

Good Features

For example in the game you can cook raw pork chop in a furnace to get cooked pork chop, this focuses on the idea of food poising. If you eat it raw then there's a chance your hunger bar starts to decrease while turning green however if you eat the cooked pork chop your hunger starts to increase (You also need to kill a pig to gain this raw pork chop). This shows there's some vague ideas of using science in the game to survive. This is very similar to the game I would like to produce due to using real world problems you would need to consider if stranded on a deserted island. In this situation you use the ore coal (other fuels can be used) to power the furnace which needs to be mined using crafted items, in Minecraft you have to start by gaining raw materials to create a pickaxe which then is used to mine harder to gain items to then craft an another item (pyramid scheme). This is very similar to game I would like to produce due to using a pyramid scheme where you will need basic items to create more complicated items which in turn will help you survive.

Bad Features

However, the game incorporates aspects, which are highly inaccurate such as teleporting to different dimensions, using potions to gain unrealistic expectations (flying, fire resistant), fighting dragons and wearing diamond-coated amour. These features wouldn't happen in the current era which defeats the object of the game being produced (everything in the game will be accurate to the extent you could do what you do in the game but in real life). For

example wearing diamond coated amour would be unrealistic due to the weight of the diamonds and the rarity of getting a bigger enough diamond to then cut the diamond down into amour shape using a diamond cut drill. This in turn doesn't come close to making a diamond set of amour in the game.

Limitations

Due to the game being based on simple graphics as shown below (A) you will only be able to recognize items if you already know them to exist (due to the picture being painted with pixels). Minecraft tries to fix this issue by showing what an item is if you hover it, however it still limits the game. The simplicity of the game also limits its crafting abilities, this is shown in the crafting table where it has 3 blocks by 3 to craft an item this means there is a limited amount of different combinations when creating an item so making the rough shape of an item using certain resources may be similar to another item so it can't be used twice.

Inspirations from Minecraft

Minecraft's crafting table and brewing stand GUI (B) gave me the idea to use a similar GUI in the game I will be producing. This way I will be able to combine raw materials to create tools used to survive instead of spending too much time on the graphics on building items. This feature of the game was one of the player's favorite to use, this was because it allowed people to experiment with different combinations or research items which lead to newly discovers.

Ark

Ark: Survival Evolved is based on deserted island with only the player some tree's (deserted island objects) and entities present, which is very similar to how my game is structured however Ark is based in the dinosaur era and focuses on retrieving (riding and capturing) different species of dinosaurs. Ark also has an interface like Minecraft that is used to craft items using different pieces of raw material, my game will also incorporate this making it easier to building items to help you start different science experiments. The official description of Ark Survival evolved is "As a man or woman stranded, naked, freezing, and starving on the unforgiving shores of a mysterious island called ARK, use your skill and cunning to kill or tame and ride the plethora of leviathan dinosaurs and other primeval creatures roaming the land. Hunt, harvest resources, craft items, grow crops, research technologies, and build shelters to withstand the elements and store valuables, all while teaming up with (or preying upon) hundreds of other players to survive, dominate... and escape!".

Good Features

Ark is an open world free roaming game, allowing players to explore the world straight away without focusing on specific missions in a specific order. My game uses a similar idea where the world is open based however a story is incorporated in it but not in a specific order (the game can also be played without the story). However, this doesn't restrict the player to what task they can accomplish. Ark also has a Day and night cycle which will influence the gameplay of the game, this would mean that night dinosaurs appear when its night and the daylight dinosaurs are more likely to spawn in the day making the game more complex but more challenging. My game will also have a day and night cycle however the changing factors will not be what entities are spawned but the different health problems you would face in the real world such as the player getting hyperthermia. Ark also uses important features such as chopping down a tree to gain wood, to then build a shelter to protect yourself from entities, this works parallel to the game being produced based on the idea of using real world science to help you to survive. Ark also follows the idea of waking upon a standard island naked and freezing once again relating heavily to the game being produced. You also must grow crops and research technology's in order to survive which would be needed that situation making this game one of the most similar games to what is being produced.

Bad Features

In the game you can tame and ride dinosaurs, this makes the game straight away very unrealistic due to theory/aspect coming close to impossible (if some way someone had travelled back in time to the dinosaur era and tried this). There are also some dinosaurs which have never existed (ant, Titanomyrma's) making the game even more inaccurate taking away the accuracy of the time era. To add onto this there's a force field surrounding the island what would never happen in the real world however due to the complexity

of the game the only way around this is repeating the map (once again losing its authenticity) or making it go on forever generating new land to the game (this would be done through a heavily memory oriented algorithm). Both methods would also cause the game size to increase dramatically which can have a very big effect on the sales of players, purchasing the game.

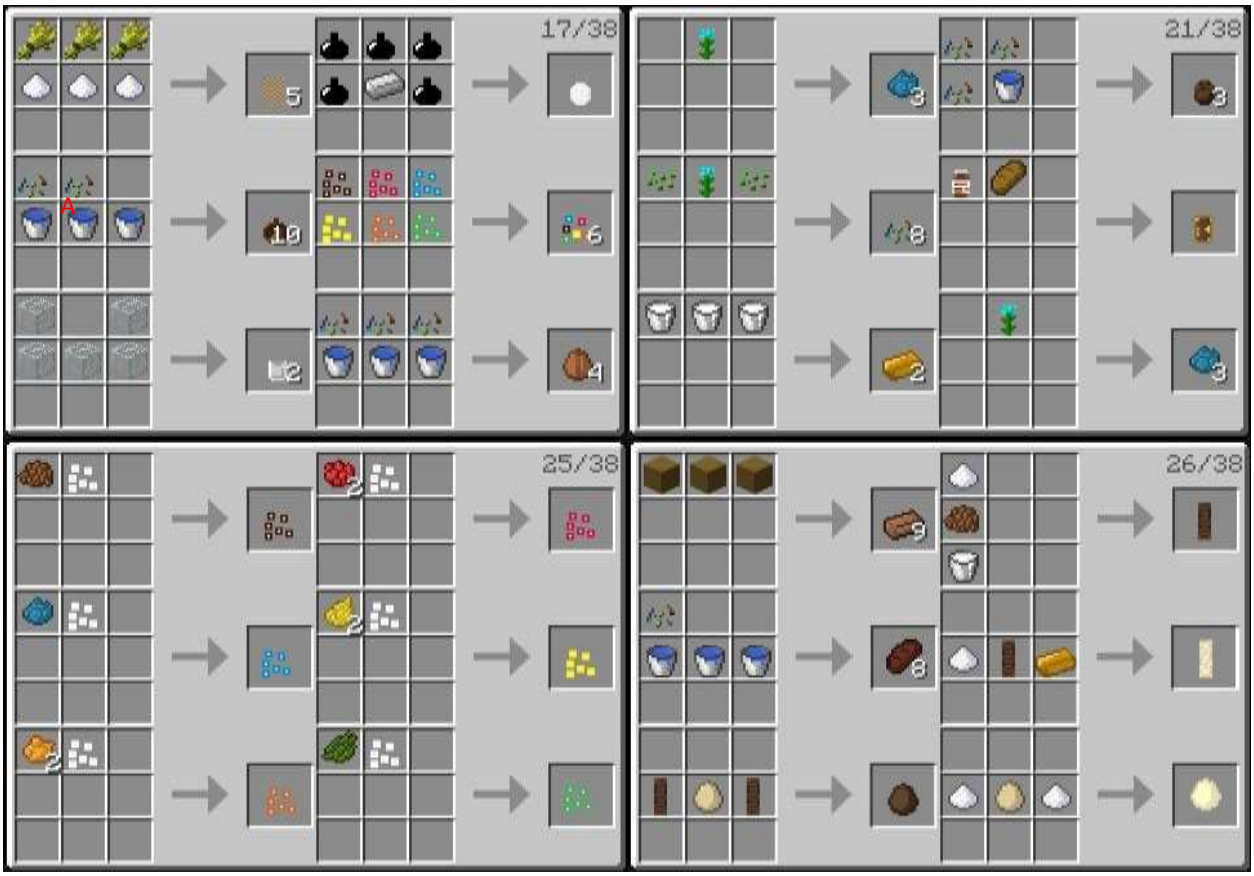
Limitations

The major difference between Ark and the game being produced is the time era! Ark is based in the dinosaur era ("**dinosaurs** lived between about 245 and 66 million years ago, in a time known as the Mesozoic **Era**") which automatically make the game focused on trying to survive dinosaurs. This limits what animals can spawn and the terrain used and only being able to update the game based around this fact. As mentioned earlier there must be an end to the game map causing the game to become inaccurate, the only ways around this (mentioned previously) would take a lot of memory up in the game. Ark uses a force field to show the edges. This one of many examples which most games are limited in such as time, arks food e.g. meat spoiling in 10 minutes which would be a lot longer in the modern day however if this was to be the case, it would make the game very dull. As mentioned in the Minecraft research there will be only a limited amount of crafting spaces to build your items (ark has a similar crafting interface).

Inspirations from Ark

Ark uses a similar interface to craft items as Minecraft due to its simplicity and time to create it. I will use a similar first person looks and will have a very similar storyline as Ark does at the beginning of the game. I will base my help descriptions of similar processes used in the game allowing me to spend more time of more complex parts. The terrain and structure of the map will be heavily based on the game due to the similarity of the story line and how both games are based on waking up on a deserted island and trying to survive.

Minecraft Java edition- PC Game- Crafting bench screenshot- Used to see the layout and materials used to create and item.



Day Z is based on survival, but instead of crafting items, they are found in the map due to it being based on a large abandoned city which has been overtaking by zombies. The survival aspect of this game links very well with my game using different items to survive however these items need to be crafted unlike Day Z.

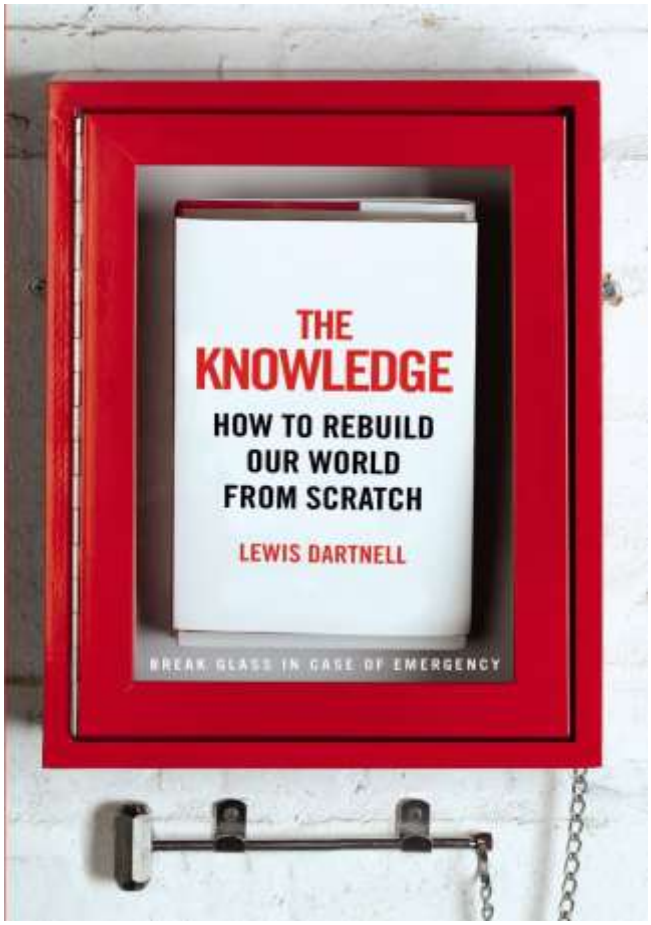
Rust is a very similar game to what I'm producing such as using an abandoned island, crafting items to survive, and killing entities however doesn't focus on more detailed survival techniques as mentioned earlier. Rust also incorporates buildings that have been left behind which the game I will be producing will not, but instead the items left behind will be purely based on the story line(depending on what story you choose at the beginning).

Finally the majority of my ideas on survival is based on the book called The knowledge by Lewis Dartnell is describes key concepts on survival by using natural resources around him and what you would need to do to reboot society and survive. This book uses key science concepts to create fire, Reserve food, purify water and create fuel to power old and new machines used to survive as well as many other things |, which I will be able to incorporate in my game accurately. The book is based on virus that wipes out humanity but everything is left untouched however it doesn't specify the specific details due to having no story line instead what you would need to do in that situation. This is different to my game but will include some outside objects and does involve a storyline unlike the book.

Ark survival Evolved - PC Game-Creatures in this instinct dinosaurs as AI's-screenshot- used to see the view of the user when holding a bow and arrow and how the AI's act

Rust -PC Game-Starting up lighting up and setting up a fire for survival-screenshot- used to see the set up of the fire and how the fire was created

Rust -PC Game-Using a hand pickaxe (crafted) to destroy a tree -screenshot- used to see how many swings it takes to break down a tree and how it is cut down



The knowledge - Book - used to see how the player can survive using everyday object's found in the world based on true methods

EXPLANATION AND JUSTIFICATION OF FEATURES OF YOUR PRODUCT

- I will have a health bar instead of a certain amount of lives (this may change depending on the difficulty rating)
- They'll be a Minecraft Crafting Interface instead of crafting button allowing items to be created and processes to happen
- Th character will find their story on the way depending on what paths they take
- A large island will be used instead of a small remote island allowing more to be explored
- The main achievement of the game is to see how many days you can survive on the island
- You will know how to survive with key passages of the book (The Knowledge) hidden around the island
- Other IA characters will be around the island (part of uncovering your backstory) what you'll be able to interact with

THE LIMITATIONS AND SCOPE

To make this game the best it could be I would focus on particle effects, new graphics on scientific processes as well as servers to play internationally. However due to my limited amount of time and only me making this game making it international with servers would be too expensive and I would need to learn more professional skills to handle a sever to play on multiplayer. I will be using some particle effects but not pacifically made for this game because of the time and skill required to do this (the mark scheme also states that no marks will be given for graphics) so I will be using similar existing ones and possible adjusting them to suite my game more. This is the same with the graphics I will be using on my players foe example however I will be combing different avatars from different games to make the game more unique.

THE COMPUTATIONAL THINKING

This problem can be solved using computational methods quite easily, below I've stated the methods I've used to overcome various problems. Due to the game being based on today's science I can use basic/simplistic algorithms to solve the problem from previous made algorithms (Pattern Recognition). I can turn these real-life situations into a game based layout using these stated methods. This clearly shows me that this problem is suited to a computer programmer.

Drop Down

I'm going to be using a top down layout to show the key objects in my game. This allows me to create each section and go back to this diagram to make sure I haven't missed any key points. It also allows me to abstract the route nodes making a basic layout of the game and only go into each individual hierarchy if I have time allowing the main points of the game to be made first. Therefore a priority order is made

Abstraction

I have used abstraction in my game to allow the basics/ main features of the game to be created. This is a key computational way to think because it takes the most important aspects of similar games (Minecraft in this example) and picks which is necessary for the game being produced. Abstraction also considers the time I have to create the game due to it focusing in the key points instead of extra unneeded parts of the game that isn't needed and will take up my time.

- Killing entities for Raw food
- Cooking Raw items to avoid food poising
- Crafting/finding items to craft/create machines or items
- Using a bed to sleep through the night
- A toolbar to hold your items
- Basic textures
- Basic animals such as sheep, Pigs, chicken etc.
- The movement of the player
- The island
- Sound effects

Thinking Ahead

This is used to get a brief layout of the game in the future therefore removing the aspect of not knowing where to start on the game. This allows the programmer to save time and build a foundation of a game before the details are incremented. It also reduces the chance of a problem occurring in the future due already having an idea how you're going to solve the problem and not getting to it and two ideas not able to co-operate with each other causing a redo of part of the game.

The Controls: I will be using ASDW as my forward (w) backwards(s) left (d) and right (a) controls due to this being the most popular in most genres of games so players will be already used it and most comfortable playing this way. My toolbar can be quickly

accessed by my numbers (1234...) and the space bar to jump. The mouse will be used to attack an item or use an item as well as the movement on the screen.

Outputs on the screen: Assembling items will make a possible noise with some particle effect, the weather will change effecting the brightness of the screen and your condition of your health if the weather turns to a more dangerous state.

Map: An island surrounded by water and trees with a beach and a texture on the edge giving a rock effect

Logically

This way of thinking is used, due to the time restriction of making the game. Without this a simple problem what can be solved by using common sense has be picked on what computational method to use, then must be designed in a certain way to be solved by the chosen computational method which leads to too much time taken up. When looking at a problem what are the most logical ways of solving it such as when you click the start button to the game("Play") you move to the gameplay area and delete GUI (Graphic User Interface) showing you the starting buttons. This computational method also looks at other things in the game such as damage, this occurs when the player has fallen from x height or hit by an entity then the health bar decreases, or the time of day turns to night the screen brightness is reduced. It based all around common sense and some problems become more complicated when designing an algorithm or formula where sometimes its more sensible to think of it logically. This is where thinking logically comes in.

Procedurally

Thinking procedurally is used in the process of making the game to allow the game to make run smoothly and be as life like as possible. It states what else needs to happen when a function/procedure is run. Game States: Dehydration for when the character hasn't had enough water the game screen will start to lower its brightness and possible start getting dizzy, Injury after fighting an entity or falling from a high distance. A noise will be simulated as well as your health being lowered and possible effects on the characters looks, Hunger after not eating enough food to maintain survival (will have the same effects as dehydration).

Concurrently

This way of thinking will allow the game to be more efficient by doing different process at once (pipelining) such as updating the health bar as an entity makes a sound effect. It also relates to making the game run more smoothly by being as life like as possible

COMPUTATIONAL METHODS

Backtracking

I used Backtracking to help me code the game, when trying to find realistic values for certain aspects of the game this method was used. This method creates solutions incrementally one by one but then goes onto removing failed solutions. For example, when adjusting variables in the code to make the game more lifelike, if the changed value is too high too or too low the value has failed and deletes a solution. This the creates another solution and repeats until the correct value is found and is used in the code.

Data Mining

Data Mining is used in the game to make it more appealing to the target audience by finding relevant data and making this game as enjoyable possible. This method finds common patterns in large sets of pre-existing data, allowing predictions to be made and know what users are more likely to want in a game. I can use this method to see what main items, objects and map layout are most commonly used, I can go onto differentiate between the most popular of these games and the least popular, therefore find what players prefer. This allows me to create of lost of unsensual things in the game and a foundation of what to start on when first developing the game.

Heuristics

This was used to allow the time to be saved when making the game, Heuristics solves this by using other methods when classics are too slow .Heuristics definition is " a technique designed for solving a problem more quickly when classic methods are too slow, or for finding an approximate solution when classic methods fail to find any exact solution". Using this method saves lots of unnecessary wasted time, where using complex methods (which require a mass amount of time to create setup and run) can be a lot slower. Where basic common sense can be used instead/more simplistic methods to create an approximate solution what will work just as good and can be a lot more affiant in the long run.

Pipelining

Pipelining is used to increase productivity allowing multiple process's to be run at once therefore more time is saved, making the process of creating the game more affiant. I will use pipeline when controlling the day and night cycle as well as controlling the player movements as they would need to be at the same time. As well as this sounds effects would need to be heard as the player moves. This method can't always work as some process's need to have has other process's run first therefore not at the same time.

Visualization

I would use a Visualization method to help me solve a problem which is easier to understand when visualized, therefore the solution makes it a lot clearer on how to be solved. This eliminates me trying to solve a problem using algorithms and other methods which the problem can't be modeled on. This prevents me having to find a completely different solution to a task instead of solving the problem. I could use this when entities attack me I could see what looks the best and seems the most life like.

HARDWARE REQUIREMENTS

A Monitor must be used to view/display the game itself/its graphics, this could be separate such as a desktop or attached like a gaming pc. A mouse is needed, this is used to move the camera around in the game/break items/ select items/select settings etc.. A mouse can be attached to a PC/laptop via a cable or Bluetooth connection or be built in such as a mousepad. Speakers are used as sound effects to the game such as fighting an entity or the waves on a beach. They can be, built in, attached, or connected via Bluetooth or wires but need to be loud and clear enough to understand what each sound effect represents. You'll need roughly 25-30GB of secondary storage to download and keep the game in your memory, most pc's/computers come with 500GB+ of secondary storage space making this size very manageable. Modern day PC's/computers come with a minimum of 8GB of Ram which is perfectly fine for this game, it doesn't run 4k or ultra-graphics meaning it won't need how amounts of RAM to run unlike some other games such as Mass Effect. Finally you will need a powerful Processor due to the high demand that games requires such as an i5 4690 3.5GHz processor this allows for a nice game feel and no lag so the player can fully enjoy the game. The table below represent the hardware requirements for Visual Studio (the software I'll be using to create the game.)

SOFTWARE

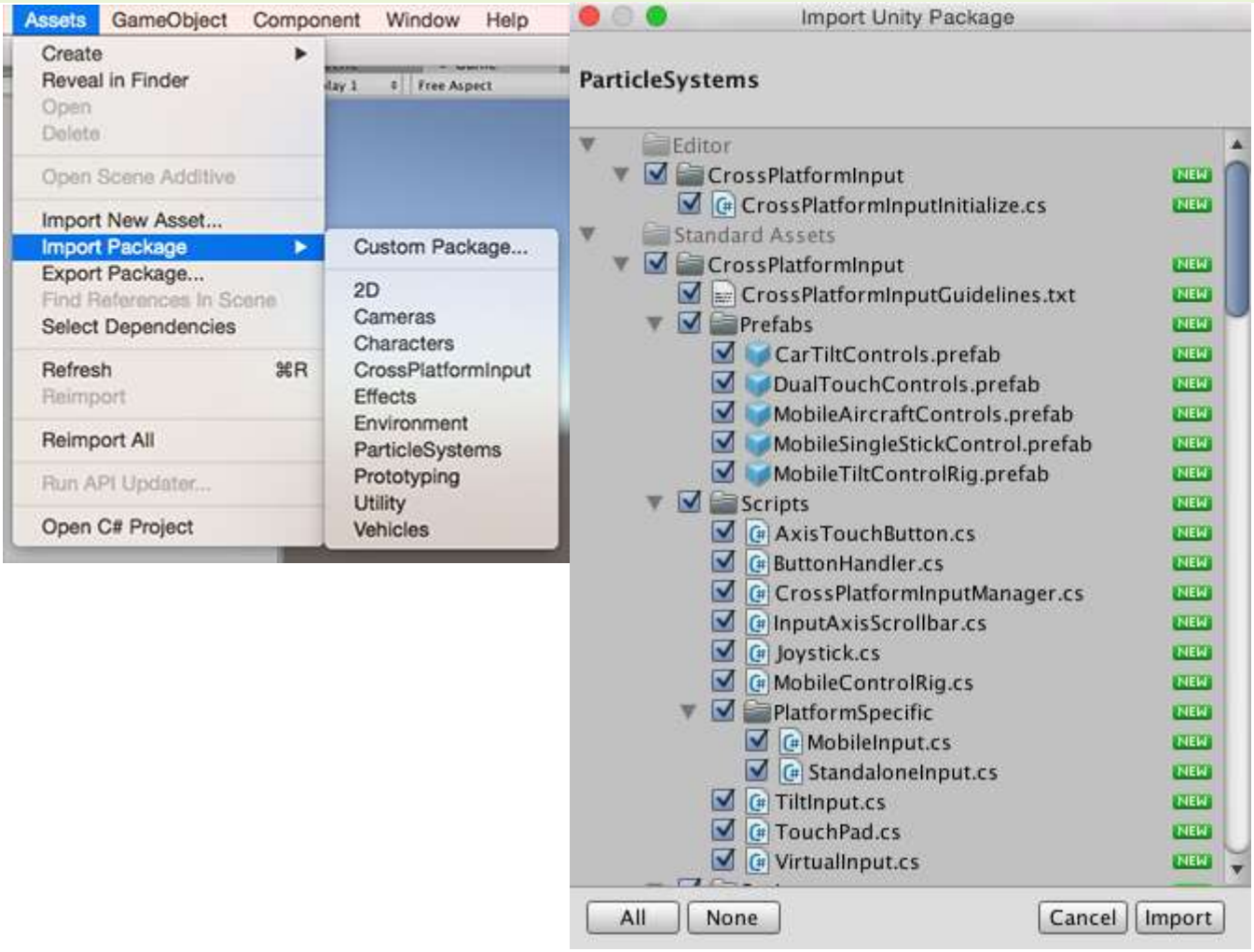
Requirement	Visual Basic	Visual C++	Visual C#	Visual J#
Processor	PC with a Pentium II-class processor, 450 MHz Recommended: Pentium III-class, 600MHz[1]	Same	Same	Same
RAM	Windows 2000 Professional — 96 MB; Windows 2000 Server — 192 MB; Windows XP Home — 96 MB; Windows XP Professional & Windows Server 2003 — 192 MB *Recommended*: 128MB for 2000 Professional, 256 MB for 2000 Server, 160 MB for XP Home, 256 MB for XP Professional & Windows Server 2003[1]	Same	Same	Same
Available Hard Disk Space	750 MB on system drive, 2.5 GB installation drive[2]	Same	Same	Same
Operating System	Windows® 2000, Windows XP, Windows Server 2003, or Windows NT 4.0[3, 4, 5]	Same	Same	Same
CD-ROM or DVD-ROM Drive	Required[6]	Same	Same	Same
Video	800 x 600, 256 colours Recommended: High Colour 16-bit	Same	Same	Same
Mouse	Microsoft Mouse or compatible pointing device	Same	Same	Same

REQUIREMENTS

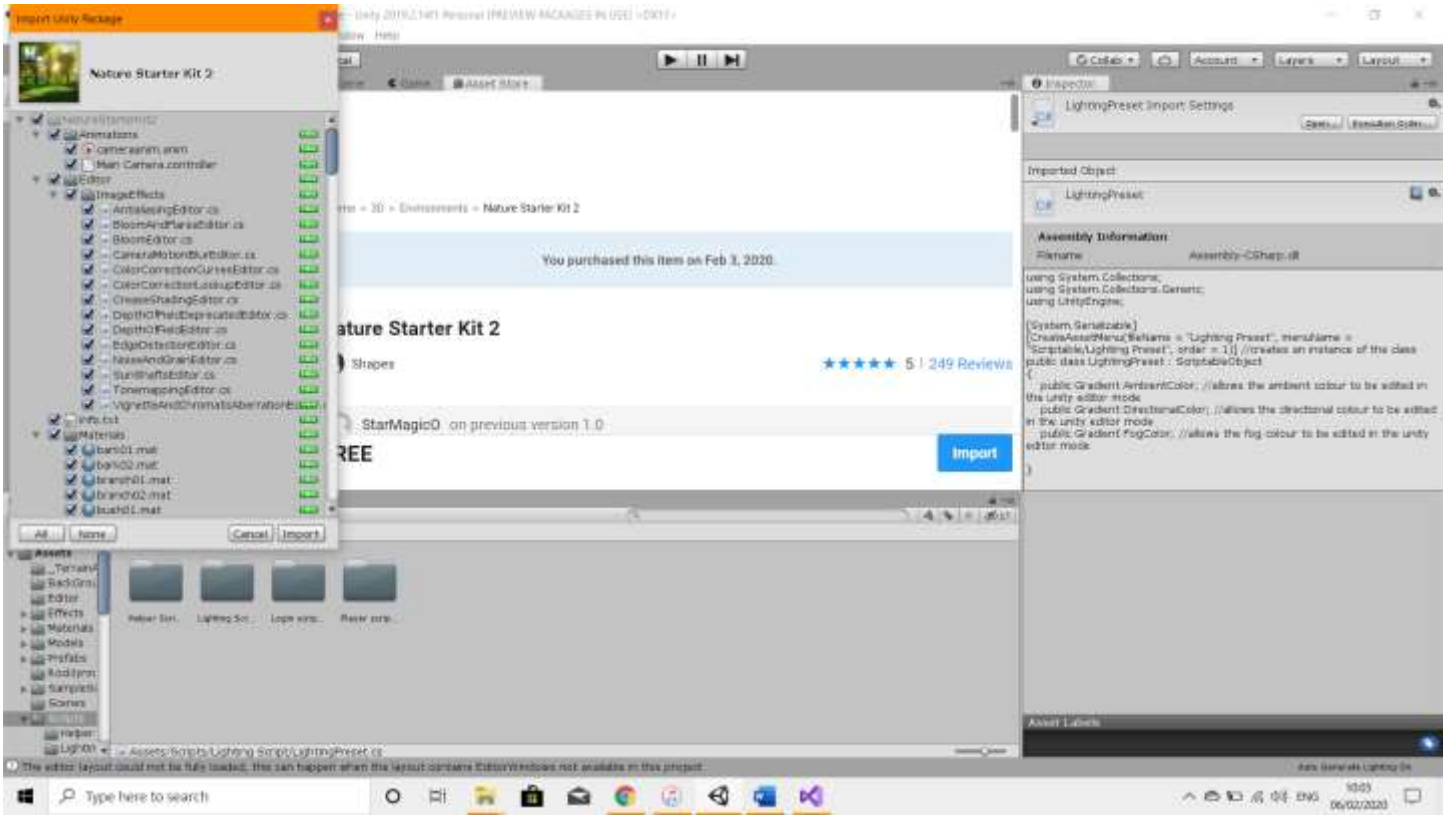
What Development language will you be programming in and why? I will be programming in C#, this is because I will be devolving my game in unity which uses c# as its main coding language. After many assignments and games being made for practice at this language, I am now confident I will be able to make my game using it. If any problems to arise as I make my game my computer science/game design teacher will be able to help me with any techniques I am struggling with due to his vast knowledge with programming. There is large amount of data online on this language, it's highly recommended and easy to program once basic instructions are understood.

Software	Justification
C# run-time library	This allows the game to be run without using the software to test the game (Unity), this overall saves storage space and stops the user from seeing the source code (prevents copyright)
If a windows powered desktop is used	
OpenGL API for graphics rendering	"(Open Graphics Library) is an application programming interface (API) designed for rendering 2D and 3D graphics" This allows me to render my game and put all the graphics together
OpenAL API for audio	"OpenAL (Open Audio Library, Open but not as in open-source) is a cross-platform audio application programming interface (API)"This allows for sound effects and music to be played in the game.
OpenAL Windows drivers	
Microsoft Visual Studio 2019	

THE UTILITIES/LIBARIES



The pictures show the standard packages for unity, this must be downloaded or added externally to the unity project. It comes with different parts such as 2D graphics which contains parts such as animation materials scripts and sprites, Camera's for view (script included), vehicles and many more. I could use the SDK library which Unity uses to interact with Facebook, this allows the game to post live feeds of the best scores of the game and another way to log in via Facebook allowing a faster access to the game. Another way to post on social media is using twitter, this can be done by downloading an external asset called "Let's tweet in unity". The GetAccessTokenForNetwork is a utility specialized in unity to "get the client's access token for a particular network, if it has been set." There is also other similar utilities used such as GetSourceId (Utility function to get the client's SourceID for unique identification) or SetAccessTokenForNetwork (Utility function that accepts the access token for a network after it's received from the server). These are all incorporated in unity without external downloads needed and can be used in many different ways.



Re-look at

INDENTIFICATION OF THE SUCCESS CRITERIA

This section specifies the key points of the game that initially was thought to be added to the game as well as what was added due to research, using the results from survey and additional extra parts that would help me have a better understanding of how the game should be produced. I've used a tick box system to show what was implemented and what I didn't have time for or was too hard to implement. These have been colour coded and a kept on the left side to help people understand the table more. The table will have 5 columns, the first to number each concept allowing the reader to see what an initial idea was and what was an extra part. The second to show the concepts/ideas what ties in with the third that justifies this concept/idea why it would be good idea to add and why this feature would make the game better to play. The fourth is too show why it was thought of in the first place (from the book, survey, internet research etc.) allowing the reader to get a reference. The final column is where the tick box is allowing the reader to see what has been added and not (plus extra details as mentioned earlier).

Number	Features	Justification	Reference
1.	Startup screen	Allowing the player to adjust the settings load into a game a previous game or start new one	Previous Gaming knowledge- Similar layout to Minecraft
2.	Settings-Startup Screen	This allows the player to adjust different layouts of the game making it easier to play	Previous Gaming knowledge- Similar layout to Minecraft

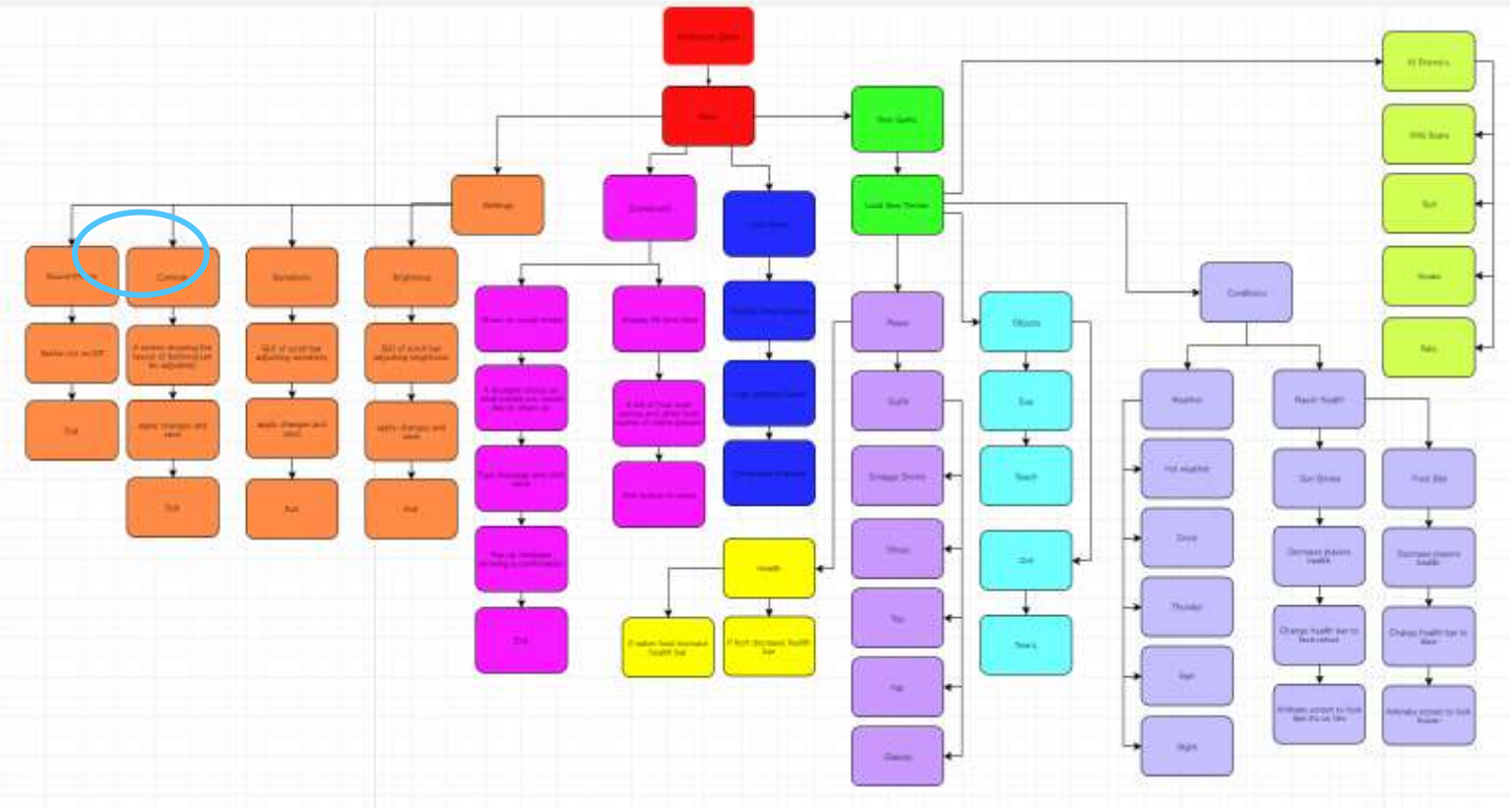
3.	Load into games-Startup Screen	So you don't lose progress of your previous game and continue from where you left off	Previous Gaming knowledge- Similar layout to Minecraft
4.	Start new games-Startup Screen	To start a fresh game	Previous Gaming knowledge- Similar layout to Minecraft
5.	Map Layout-Island	Without this feature the player wouldn't be able to move around the world due to their being no world. This feature makes the game unique to different games (starts to separate them into different categories)	Previous Gaming knowledge as well as more in-depth research- Similar Map layout to Ark survival evolved
6.	The sand-Map Layout	This allows me to create the atmosphere of an abandoned island. Using the sand as a beech of the island.	Previous Gaming knowledge as well as more in-depth research- Similar Map layout to Ark survival evolved
7.	The Sea-Map Layout	This allows me to create the atmosphere of an abandoned island. Using the sea to complete the beach image.	Previous Gaming knowledge as well as more in-depth research- Similar Map layout to Ark survival evolved
8.	Rocks-Map Layout	To create the atmosphere of the island. To possibly use this as raw materials to create machines/tools	Previous Gaming knowledge as well as more in-depth research- Similar Map layout to Ark survival evolved
9.	Tree's-Map Layout	To create the atmosphere of the island. To show this has a jungle layout. To possibly use this as raw materials to create machines/tools	Previous Gaming knowledge as well as more in-depth research- Similar Map layout to Ark survival evolved
10.	The character's clothing	This will emphasis the fact the character has been standard and left to rot	In depth research from the internet and Gaming magazines. Similar character looks to

			the character in rust
11.	Pickaxe-Tools	Allowing the player to gain raw items by mining them as they would do in the real would	Previous Gaming knowledge- Similar layout to Minecraft
12.	Part Of Books	This goes back to the storyline where a book is left from how the player got there. This book helps the player know how to survive by reading passages from this said book.	Reading of the Book The Knowledge
13.	The Character Itself	Allows the player to move around with an entity instead of being invisible what would defeat the whole purpose of the game.	N/A
14.	Monkeys-Entities	Monkeys will give the game character, by adding a combat aspect into the game as well as bringing back the idea of science to kill and eat the monkey to survive.	Research into this type of game genre
15.	How many day's survived on the island-Layout	Using the day's survived as the scoreboard it will make the game more interactive and not just a play once kind of game.	Unique- Survey Results
16.	Heath Bar	Without a health bar the player wouldn't know how much health he has left before he starves/dehydrates/etc to death.	Unique- Survey Results
17.	Sound effects	This gives the game some more character allowing the player to be more immersed into the scenario the character has been put in.	N/A

DESIGN

STRUCTURE OF SOLUTION

The Top Down model below shows all the features that I would like to implement in my game. Starting from the route node (Adventure Game) it then follows onto the main menu which then leads into the foundations of the game, these include: Settings, Scoreboard, Load Game and New Game. This structure was created based of the previous data found such as user needs users and requirements. This allows me to create the basics of the game first and remaining time go onto each branch and implement features which aren't as important but would make the game more enjoyable. However some of the branches state how that node can be



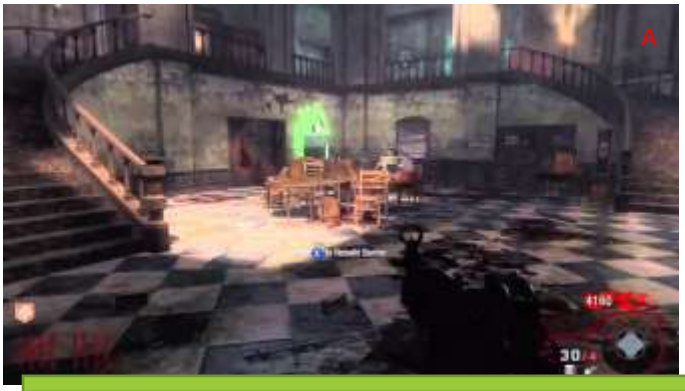
structured such as the sensitivity bar stating a scroll bar and needing an apply and save button. This helps me know what is essential when creating that node and again gives me a foundation for that specific function/procedure. I have colour coordinated each group making it easier to read and understand. These groups are based on classes making the algorithm/code more affiant and allow an external programmer to be able to understand more easily.

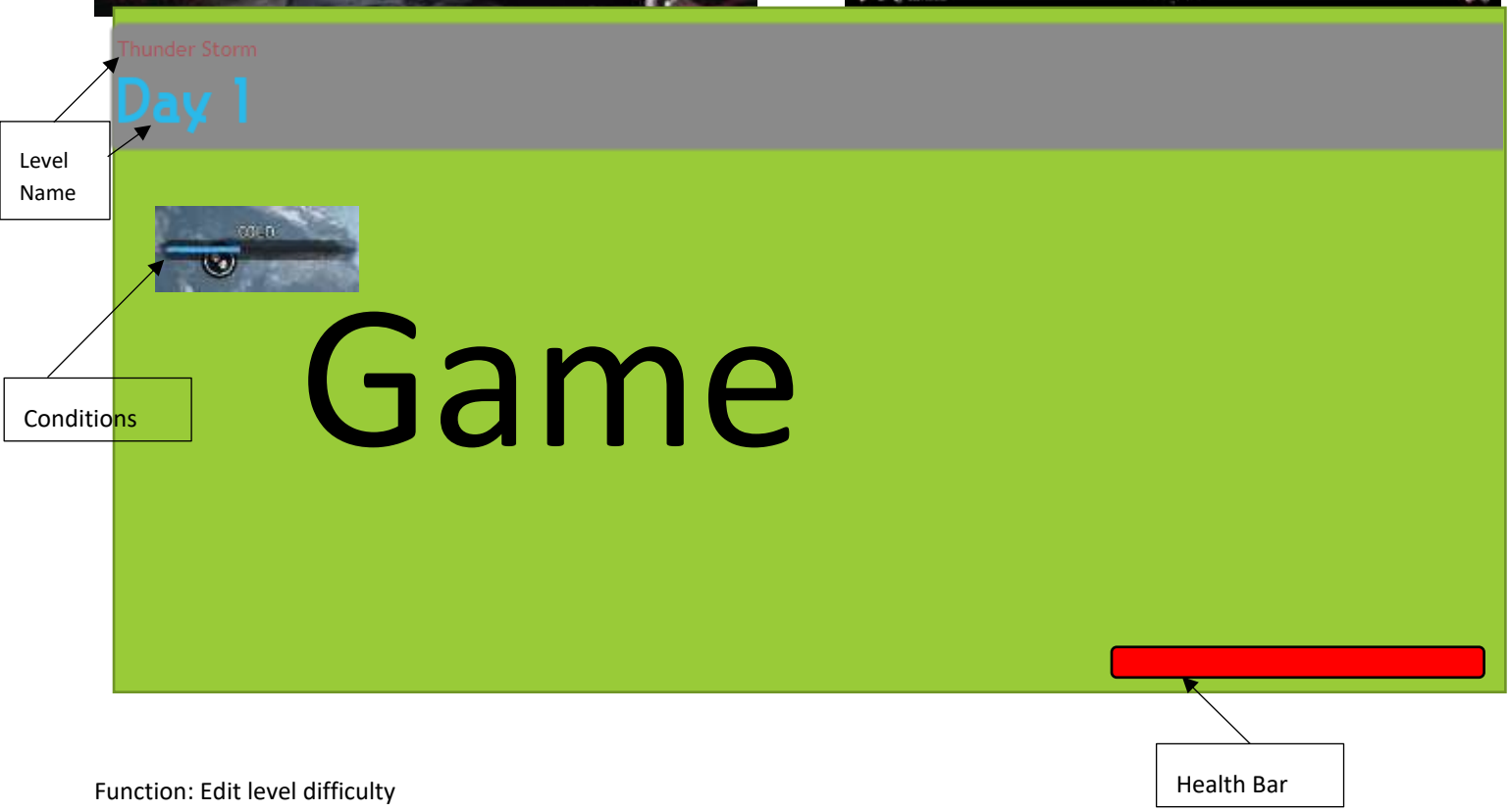
EXPLANATION OF EACH OF THE MODULES, METHODS, PROCEDURES AND FUNCTIONS REQUIRED

COMPUTER GAMES

Procedure: Read level

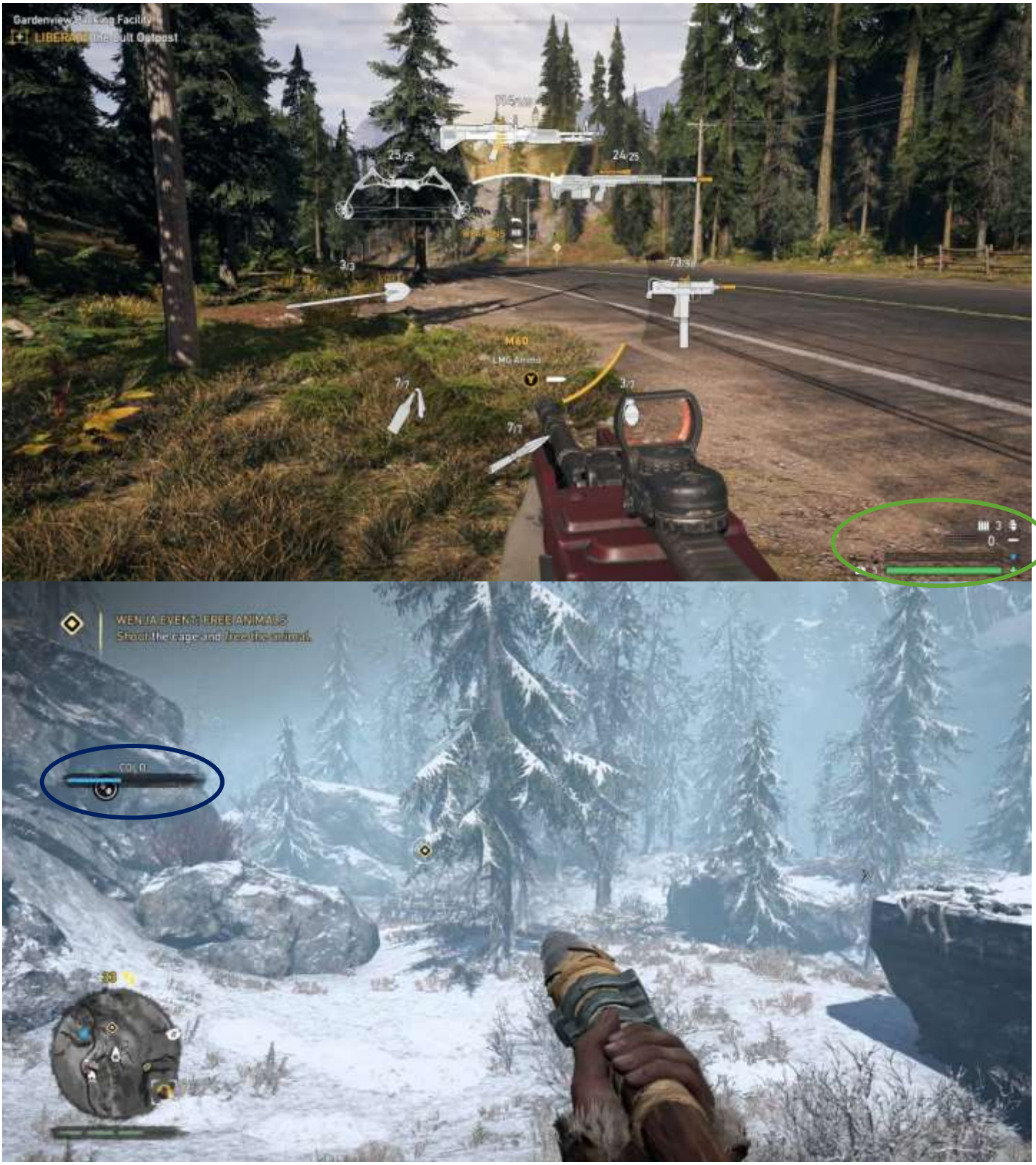
All the levels are put in .txt files labelled with which day it is. This procedure looks for a certain day (a variable depending on what part of the game is currently being played at) in a folder full of the levels and reads all the contents of the said file. The file contains the Day, the name of the day (the current level), the difficulty of the enemy's and the weather. This is then given to the GUI to display all the data above on a screen for a couple seconds before the game starts like in COD (Call of Duty) black ops zombies (A) or Dead rising (B).





Function: Edit level difficulty

This function looks inside all the level .txt files and edits the game difficultly depending on what has been chosen on the GUI. This will be done by displaying what the current level difficulty is (by using the Read level procedure on all the levels), by displaying this



Information on the screen and text boxes next each level allowing the player to edit this information. After a confirmation of the user changing the settings, the inputs will be changed to variables allowing each level to be written to the new difficulty selected.

Procedure: Read Player Status

This status is read from the characters .txt file, this contains the players: starting health and any updated health if loading into a pre-existing game, any conditions they are currently facing and what gender they are. This is kept in the main folder of the game and is needed to display their health and any conditions affecting them.

Procedure: Weather

This randomly generates a number (python-Get Random) where each number is associated with a weather condition. After the number is generated, the game time is checked using an IF else, statements if it is between a certain amount of time then either day or night will be displayed by returning this value. This continues to return what weather condition the game will change to (every time it changes from day to night or night to day)

Function: Item pick

Goes into the bag pack .txt file and uses an If/Else statement to see if there is any space, if not the console will return will "Bag Full" using an (C#-Console function) and return a null value back however if not the item will be added and the item count of the bag to be incremented by 1 for the next item is picked up to see if it's full or not.

Function: Item Remove

Goes into the bag pack .txt file and finds the selected item, which is wanting to be removed, this is then amended to delete the line plus an extra delete, so the format is kept tact without corrupting it for next time this file is read.

CLASSES, ATTRIBUTES AND METHODS

I have created 6 separate different class's: Food, Player, Animals, Conditions, Equipment and Weather for my adventure game. These are the class's a chose when developing my drop down diagram using specific groups in different colours each class is chosen down to how many individually process a class would have meaning if 2 or more process would work under that class, a class would be created.

Food
Plants()
Meat()
Food_Spawn()
Seeds()

Plants()-This cross references with an external library what plants been picked, how much health it will it give the player and gives out whatever seed it makes(unless no seed is produced). The amount of seeds given back is based of a random number generator.

Meat()-This cross references what type of meat has been taken from an animal and then uses this to see how much health needs to be gained. However if the meat is raw and hasn't been cooked the food_poisoning() method will be called upon

Food_Spawn()-At certain time in the game this is always called to generate all the food spots to be spawned again allowing a constant flow of food allowing the player to survive

Seeds()-This is called upon when the player wants to plant some seeds for creating crops to eat or eat them straight away (both ways will gain more health unless the players already full) by either clicking left to eat the seeds (small amount of health will be gained) or clicking right plant the seed.

Player

- Player_Move()
- Player_Attack()
- Use Equipment()
- Storage()
- Player_Health()
- Day Count()

Player_Move()-Using the ASDW(space) keyboard inputs will allow the player to move. This method will check what has been inputted and respond will the appropriate response. For example if the keyboard input W was pressed the center location of the player will be noted then appended by adding x amount to x/y depending where the player is facing giving the simulation that the player is moving. When (space is pressed) only the z axis will increase or decrease depending on the position.

Player_Attack()-This will be used when in combat with an animal. A possible gesture will be made by the player and depending what item the player is holding (separate procedure to check) the animal's health will be decreased (call the Animal Health method)

Use Equipment()-The equipment being clicked on will be cross referenced with an external library to see what particle effects will be used and if any other function needs to be called upon

Storage()-This will start to check what's been picked up and then be proceeded to be displayed on the screen

Player_Health()-This method will be constantly called upon to keep updating the health bar displayed on the screen by appending the number and changing out images to what is most suitable for the amount of health left.

Day Count()-When a certain time is met this will be called upon appending the date on the screen as well as displaying for a coupe seconds the level type and which day it is in much bigger writing

Animals

- Animal_Move()
- Animal_Attack()
- Spawn()
- Animal_Health()
- Skin()
- Animal_Death

Animal_Move()- This will cross reference with what type of movement the specific animal chosen will do. It can be either random movements in particular order at slow medium or fast speeds or to follow the player as well as attacking the player (if this is the case the Attack() method will also be called upon).

Animal_Attack()This will be used when in combat with a the player. Using the cross referenced data the code will know how strong the animal is and know how much health is needed to be deducted every time the animal attacks. A certain gesture might be used or a different skin might also be applied to the animal.

Spawn()-this chooses specific places where to spawn animals. This will differ depending on how many has been killed, the more killed the more endangered it will get o the less will spawn (possibly to extinction)

Animal_Health()- This will be used when the player uses the attack feature, from the Player_Attack() method. From this method a certain amount of health will be decreased depending. Once The player health has reached 0 the Animal_Death() method will be called upon.

Skin()- Depending on the animals chosen a skin will be found in the libraries and used on the animal so the player knows what type of animal it will be facing

Animal_Death-The animal will be removed from the game and replaced with the items it leaves

Conditions

Dehydration()

Starvation()

Hyperthermia()

Food_poisoning()

Sunburnt()

Dehydration()-this is called upon when the player has drank too much sea water or not enough water. An extra health bar is added with the colour blue instead of red to representing how dehydrated the player is, slowly reducing until the player drinks some water. Once the blue health bar is at 0 the players health is reduced at certain time intervals depending on the current level, until pure water is given it wont's stop. As soon as it does the extra health bar will disappear and the health will stop dropping.

Starvation()-this is called upon when the player hasn't ate anything . An extra health bar is added with the colour green instead of red to representing how hungry the player is the player is, slowly reducing until the player drinks some water. Once the green health bar is at 0 the players health is reduced at certain time intervals depending on the current level, until food is eaten. As soon as it does the extra health bar will disappear and the health will stop dropping.

Hyperthermia()-this is called upon in cold weather where there's no fire or heating present. An extra health bar is added with the colour blue instead of red to representing how cold the player is, slowly reducing until the player is heated up. Once the blue health bar is at 0 the players health is reduced at certain time intervals depending on the current level, until heat/ a fire is present it wont's stop. As soon as it does the extra health bar will disappear and the health will stop dropping.

Food_poisoning()-this is called upon when the player has ate raw meat and the random number generator has made he player have food poisoning . An extra health bar is added with the colour green instead of red to representing the toxins the food has given the player, slowly reducing until the a certain time is met (randomly chosen). Once the green health bar is at 0 the players health is reduced at certain time intervals depending on the current level, until the time Is up. As soon as it does the extra health bar will disappear and the health will stop dropping.

SunBurnt()-this is called upon when the weather is sunny and the player hasn't used a sun cream substitute. An extra health bar is added with the colour yellow instead of red to representing how the sunburnt the player is, slowly reducing until the player uses a sunburnt substitute. Once the yellow health bar is at 0 the players health will go straight to 0. As soon as it the player goes in the water the sunburnt health bar increases a sizable amount until it is full again. Then extra health bar will disappear and the health will stop dropping.

Equipment

Distillation()

Fire()

Distillation()-Open's distillation menu and code for how it all works

Fire()-Open's fire menu and code for how it all works

Weather

Rain()

Thunder()

Sun()

Cloudy()

Rain()- changes the background to a rainy weather look-increases the chance of hypothermia

Thunder()- changes the background to a rainy weather look with extra sound effect of thunder

Sun()- changes the background to a sunny weather look-increases the chance of sunburn

Cloudy()- changes the background to a cloudy weather look

KEY VARIABLES AND STRUCTURES

Player_health-Global Variable-Represents the health of the player. This needs to be accessed throughout the code due to constant changes from weather animal's conditions etc...

Player_Center_location-Variable-To know the center of the players position allowing the player to be moved by adding integer to their position therefore making them move.

Random_Seed_Return-Variable-This is randomly generated number through a random number generator between 3-0 to see how many seeds are produced out of a plant/food

Food_Poisoning- Variable-This is randomly generated number through a random number generator between 1 and 0/True or false. This occurs when a player has eaten raw food and will either get food poisoning or not just like real scenario

Plant_Health-Variable-After a plant has been found in a file this is used to store it and add it back to the health once eaten

AnimalX_Center_location--variable- To know the center of the animal's position allowing the animal to be moved by adding integer to their position therefore making them move. X-depends on how many animals are spawned

Animal_health_X- Global Variable-Represents the health of the animal. This needs to be accessed throughout the code due to constant changes from weather player attacks conditions etc... X- depends on which animal is being talked about

Selected_equipment-Global Variable-This allows the code to know what functions should happen when a button is clicked due to knowing whats in the players hand. This is needed throughout the code for combat placing items and most importantly what's currently being displayed on the screen.

Day_count-Variable-to know what level is next to load

Weather_sun-Variable-Allows the code to know if it's true or false so player can only be sunburnt in them conditions

Weather_rain-Variable- Allows the code to know if it's true or false so player can only be affected by hyperthermia in them conditions

Weather_thunder-Variable- Allows the code to know if it's true or false so player can only be affected by hyperthermia in them conditions

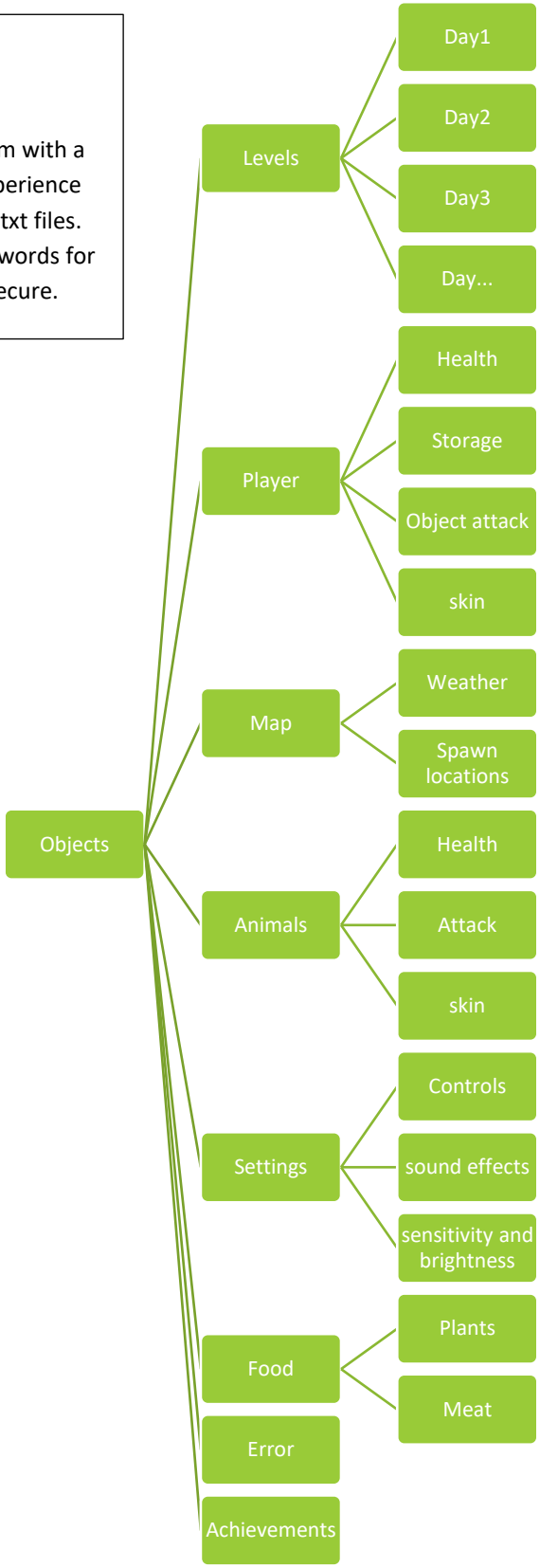
item_Damage-Variable- Represents the power of an item. This cross referenced and used to attack an animal.

Achievements-Variable- Finds all the achievements gained.

FILE FORMATS

Why will I use .txt files?

I will use .txt files due to c# being able to manipulate append delete and create them with a few lines of simplistic code. Due to my experience coding I also am very familiar with coding.txt files. They also allow you to easily encrypt passwords for the login system making my game more secure.



ALGORITHM-PSEUDOCODE

Plants()-This cross references with an external library what plants been picked, how much health it will it give the player and gives out whatever seed it makes(unless no seed is produced). The amount of seeds given back is based of a random number generator.

Search in “object/food/plants” for plant selected

If not found

return error and add to error file

If found

Add Plant_Health (below the selected plant) to the Player_health variable

Return Player_health

Find Selected_equipment

Delete selected_equipment from “object/Player/Storage”

Exit “object/food/plants”

Exit “object/Player/Storage”

Meat()-This cross references what type of meat has been taken from an animal and then uses this to see how much health needs to be gained. However if the meat is raw and hasn’t been cooked the food_poisoning() method will be called upon

Search in “object/food/Meat” for meat selected

If not found

return error and add to error file

If found

Add Meat_health(below the selected meatt) to the Player_health variable

Return Player_health

Find Selected_equipment

Delete selected_equipment from “object/Player/Storage”

Food_Spawn()-At certain time in the game this is always called to generate all the food spots to be spawned again allowing a constant flow of food allowing the player to survive

Check animal count

If animalX is 3 or 2

Spawn animal 1 animalX

Else if animalX is 1 or below

Print (animalX extinct)

Else

Respawn all animalX()

Seeds()-This is called upon when the player wants to plant some seeds for creating crops to eat or eat them straight away (both ways will gain more health unless the players already full) by either clicking left to eat the seeds (small amount of health will be gained) or clicking right plant the seed

If leftbutton clicked= True

Add 10 to Player_health variable

Return Player_health

Else

Where ever the pointer is pointed too plant the sead

Find Selected_equipment

Delete selected_equipment from “object/Player/Storage”

Player_Move()-Using the ASDW(space) keyboard inputs will allow the player to move. This method will check what has been inputted and respond will the appropriate response. For example if the keyboard input W was pressed the center location of the player will be noted then appended by adding x amount to x/y depending where the player is facing giving the simulation that the player is moving. When (space is pressed) only the z axis will increase or decrease depending on the position.

If “A” is pressed

Get Player_Center_location

Minus 5 from x Player_Center_location

Return Player_Center_location

Else if “S” is pressed

Get Player_Center_location

Minus 5 from y Player_Center_location

Return Player_Center_location

Else if “D” is pressed

Get Player_Center_location

Add 5 to x Player_Center_location

Return Player_Center_location

Else if “W” is pressed

Get Player_Center_location

Add 5 to y Player_Center_location

Return Player_Center_location

Else if “[Space]” is pressed

Get Player_Center_location

add 5 to z Player_Center_location

wait 0.3 seconds

Minus 5 from Player_Center_location

Return Player_Center_location

Player_Attack()-This will be used when in combat with an animal. A possible gesture will be made by the player and depending what item the player is holding (separate procedure to check) the animal’s health will be decreased (call the Animal Health method)

Gesture_Player_Attack()

Find Selected_equipment from “object/Player/Storage”

If found =True

Once found find the item damage it does to an animal

Remove item_Damage from Animal_health variable

Return Animal_health

Else

Remove 10 from Animal_health variable

Return Animal_health

Use Equipment()-The equipment being clicked on will be cross referenced with an external library to see what particle effects will be used and if any other function needs to be called upon

Find Selected_equipment from “object/Player/Storage”

Look for any particle effects used and if found run them

Run the code that’s attached to each item that’s been found in the description of the selected_equipment

Storage()-This will start to check what’s been picked up and then be proceeded to be displayed on the screen

i1=length of the first row of storage spots that are full

i2=length of the first row of storage spots that are full

i3=length of the first row of storage spots that are full

Array-where to pace each item image=x,y

For i1 in range(0,i1)

i1=i1+1

x=x+5

Add i1 to the coordinates(x,y)

Y=y-5

For i2 in range(0,i2)

i2=i2+1

x=x+5

Add i2 to the coordinates(x,y)

Y=y-5

For i3 in range(0,i3)

i3=i3+1

x=x+5

Add i3 to the coordinates(x,y)

Player_Health()-This method will be constantly called upon to keep updating the health bar displayed on the screen by appending the number and changing out images to what is most suitable for the amount of health left.

If Player_health is 0

Fetch achievements from “objects/achievements”

Make it into variable achievements

Print (“Achievements ”+ achievements)

Else

Go to “objects/player/health”

Find IMAGE NAME with “player_health” in

Select image

Delete current IMG in (x,y) position

Add selected image and position in (x,y)

Day Count()-When a certain time is met this will be called upon appending the date on the screen as well as displaying for a coupe seconds the level type and which day it is in much bigger writing

Day_Count= Day_Count+1

Go to “objects/Levels/STRING(daycount)”

Find IMAGE NAME with “player_health” in

Rest saved on college Computer needs transferring

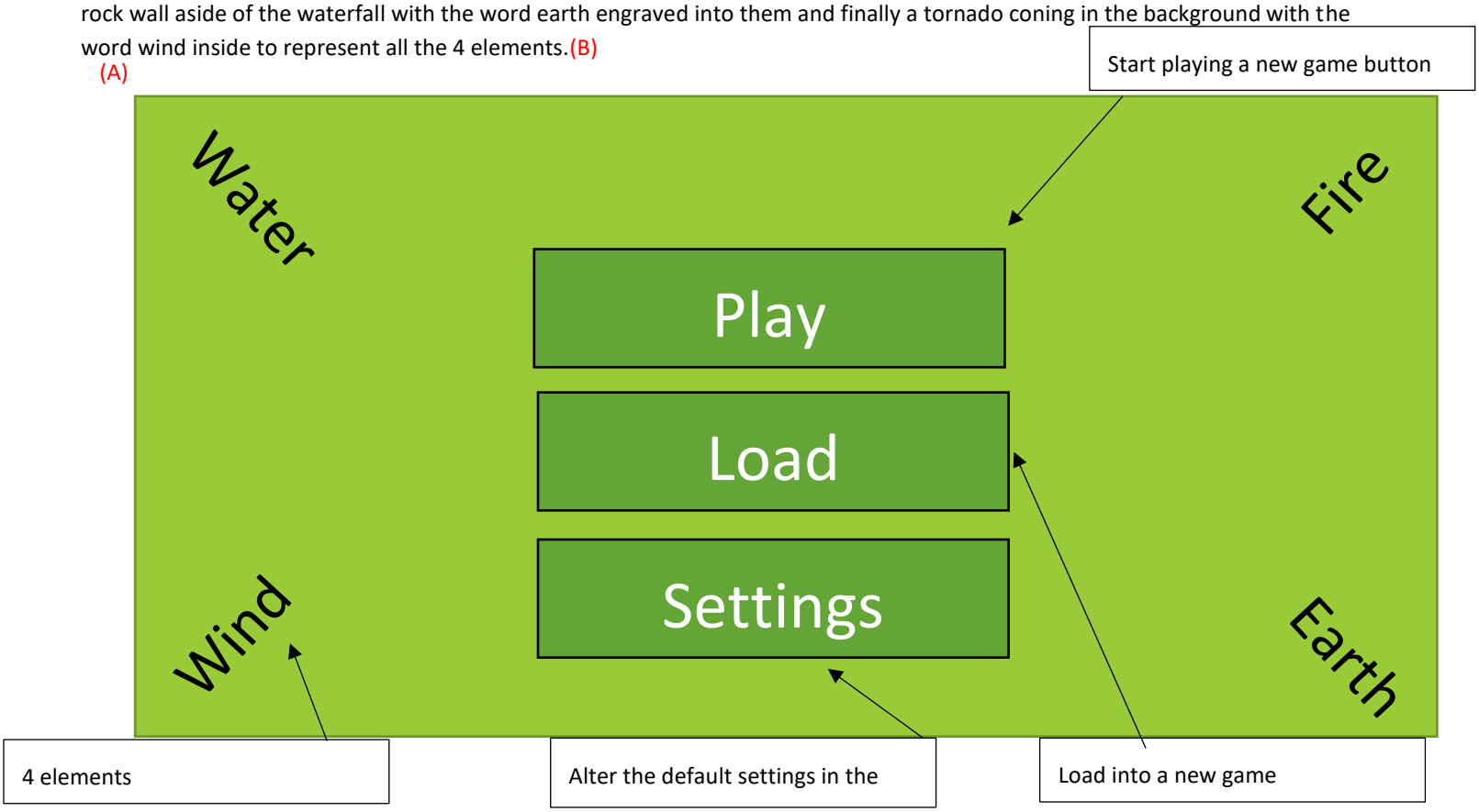
Gameplay/New Days GUI

As shown on page on 27 this is what the GUI would roughly look like when the game is starting/going onto the next level or just being played in general. First a grey bar will be animated across the screen with the level what is being currently being played or going onto to be played in bold light blue letter's, along with a brief description of what might be encountered in this level above (this will be in 60% smaller writing and in a light red). The rest of the GUI shows the rough default layout with the conditions of the players health eg hyperthermia. as well as this the actual players health in the bottom right in corner of the screen (displayed in a rectangle box which is filled with the colour red which will be reduced if the player for example gets attacked). The toolbar is also displayed in the bottom left allowing the player to scroll and pick what item they would like to use.

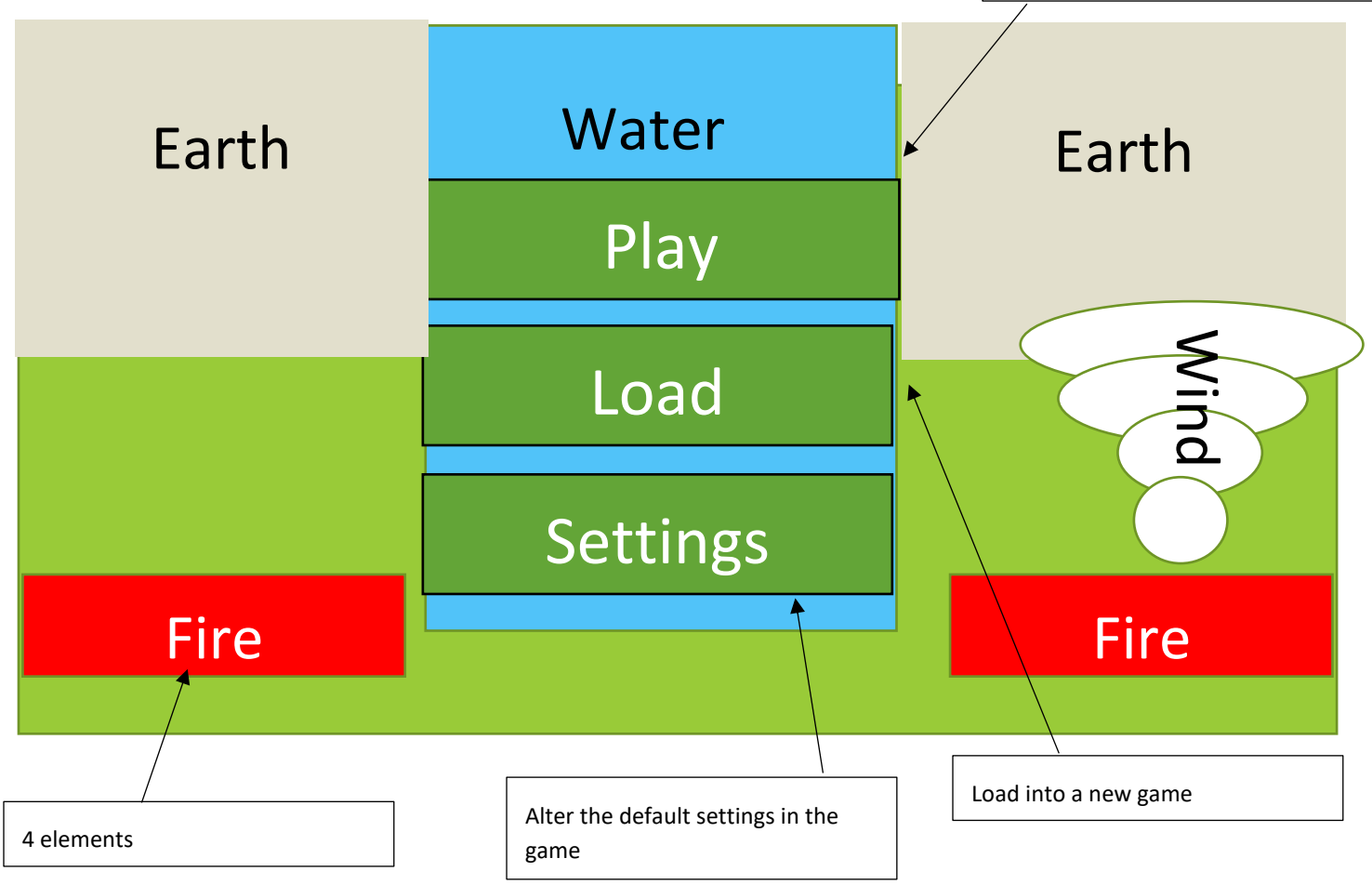
Starting Menu GUI

This will be shown as soon as the game is launched allowing the player to pick either Play, Load or Settings. The Play button will put the player in a brand new world, The load button will show a selection of previously played games allowing the player to select one and continue with there progress and finally the settings button which will allow the player to alter different components of the game so the game can be played more to their liking. These buttons are set in the traditional layout of classic looking games such as ECO and Minecraft, the 4 elements are put in the background to bring to life the survival aspect. This is either going to be done with each element in each corner with a picture of the gameplay in the background (a) or a waterfall going down the middle behind the buttons to represent water with the letters water inside, fire on both bottom right hand corners again with the word fire in them, rock wall aside of the waterfall with the word earth engraved into them and finally a tornado coning in the background with the word wind inside to represent all the 4 elements.(B)

(A)

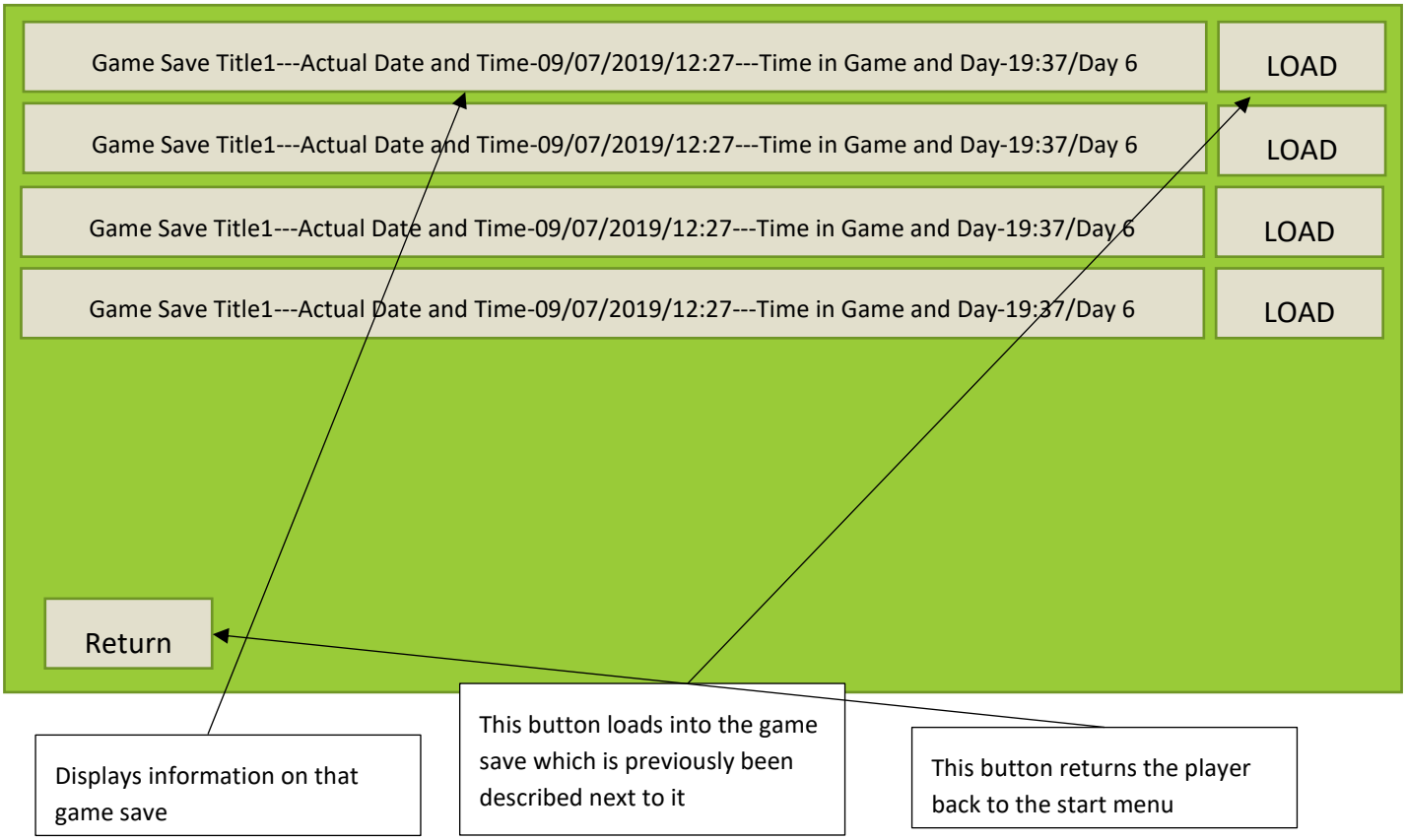


(B)



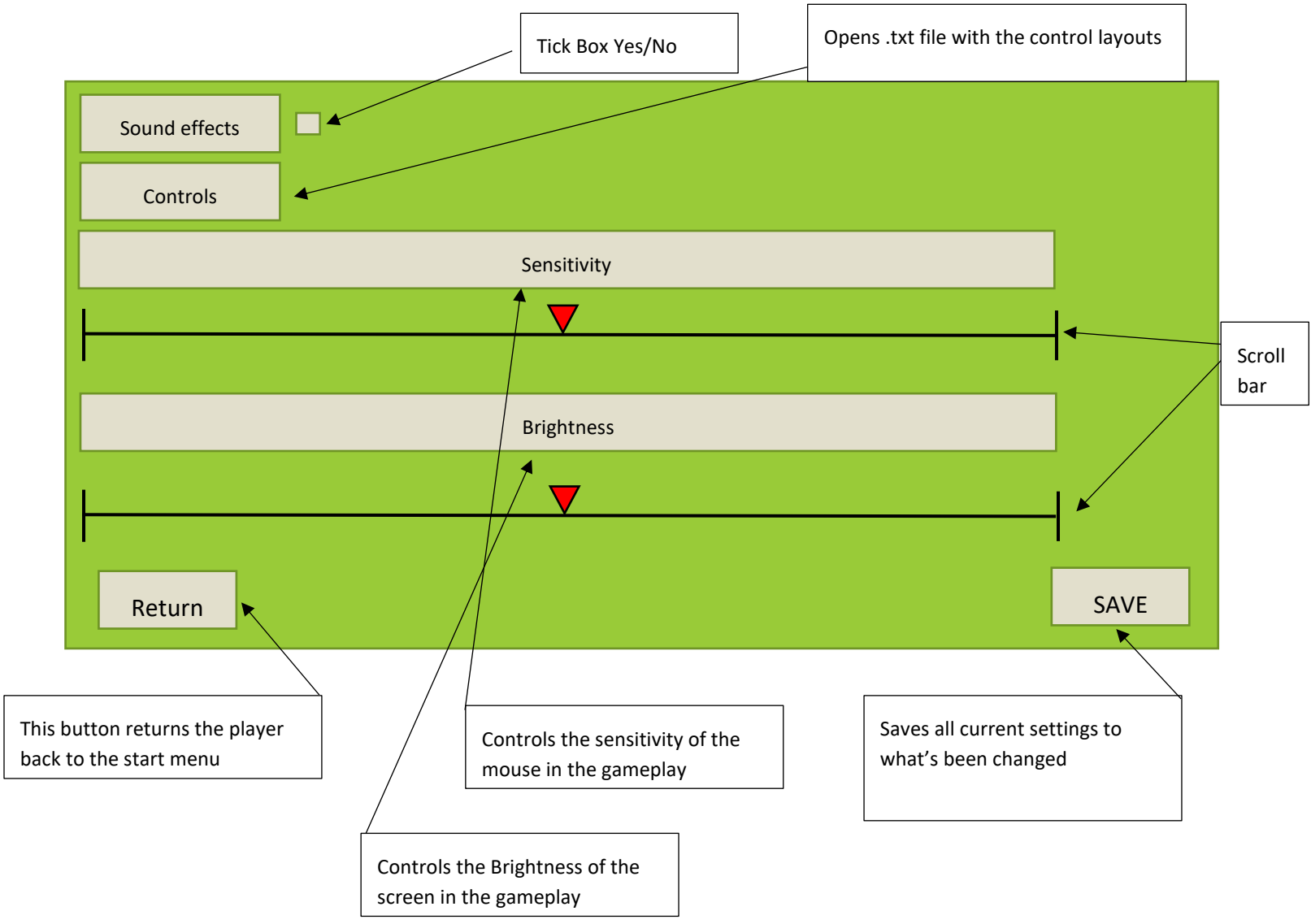
Load Menu GUI

This will be displayed when the Load button is clicked in the start menu to show a selection of previously played games allowing the player to select one and continue with there progress. Each game save will display a little information of the game this includes the custom name of the save e.g. “Game Save Title1” (currently the default) the time and date it was saved on “Actual Date and Time-09/07/2019/12:27” and the game save time and day it was saved on “Time in Game and Day-19:37/Day 6”. Next to this information on the right is a button that lets the player load into that game save as well as another button in the bottom left that lets the player return to the starts screen if miss pressed.



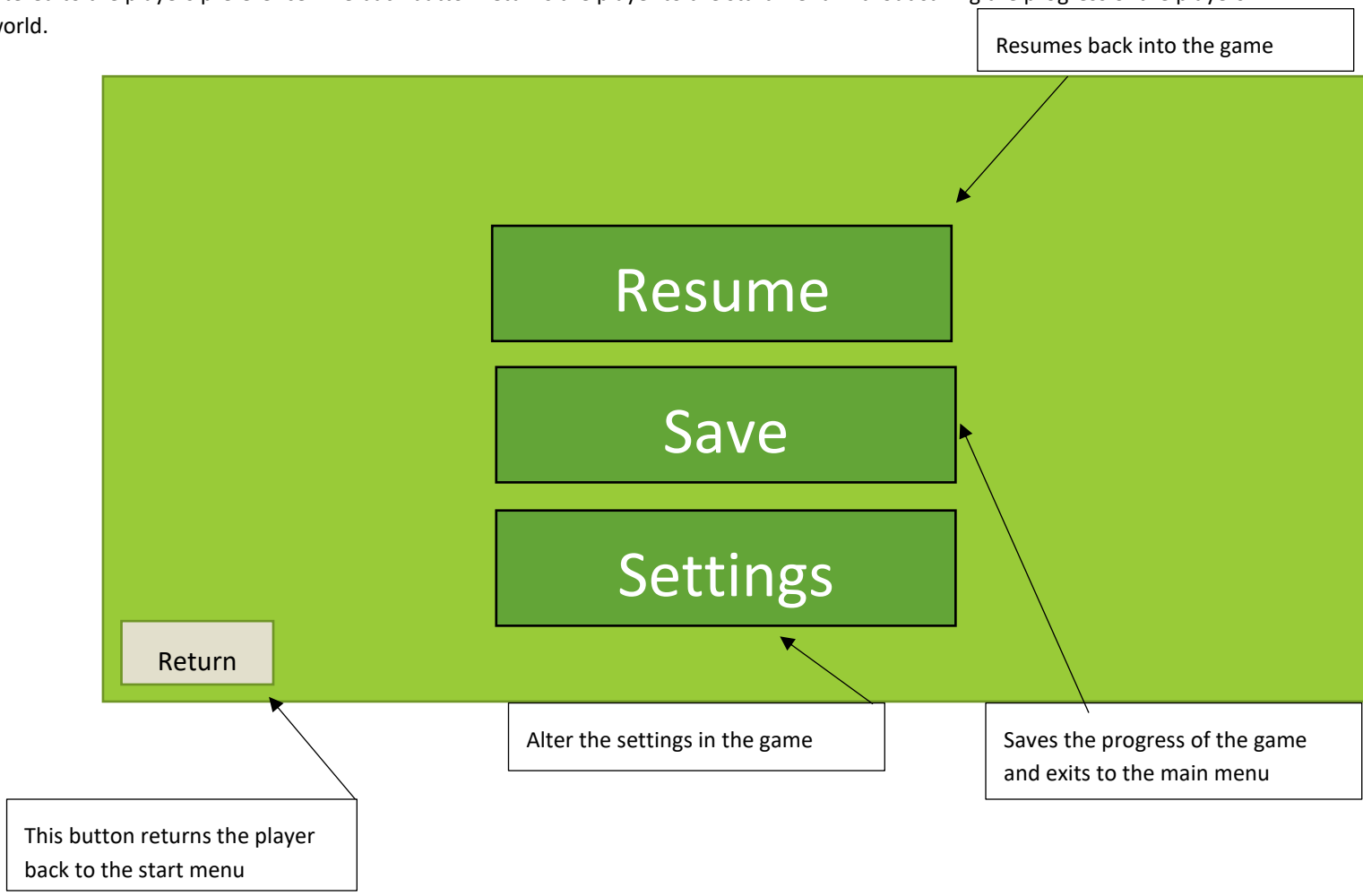
The GUI for Altering Settings

This will be displayed when the Settings button is clicked in the start menu or in game play, to allow the player to alter the game settings to their preferences making the game more enjoyable and easier to play specifically for them. In the top left there is a label that states sound effects with a tick box next to it for yes or no. If ticked the sound effects will be played if not the sound effects will be put on mute, this can be changed by a simple click for on or off by the player. Below this is a control button once clicked a .txt file will open with an easy to understand/ read format allowing the player to easily adjust the button layout. To save their changes they would need to click save as it's crossed off. Underneath this is a label that states sensitivity, this is accompanied with an adjustable scroll bar beneath with the triangle being the pivot point moving the pivot to the right will increase the sensitivity moving it to the left will decrease it. Again under this a similar principle but for the screen brightness which works the similar to the previous scrollbar (Sensitivity) as its moved to the right the brightness will increase as its moved to the left it will be decreased. In the bottom left is a return button to go back to the start menu but if the save button isn't pressed in the bottom right then all the changes to settings won't be changes (excluding the controls due to that being separate).



The GUI when the game is paused

This is displayed when the the game is paused it has the 3 buttons in the middle like the start screen which are Resume, Save and settings with an extra return button in the bottom left of the screen. Th Resume button goes back to the gameplay, the save button exits and saves the current game to be loaded and played another time and the settings button goes to the settings GUI to be altered to the players preference. The back button returns the player to the start menu without saving the progress of the players world.



DESCRIPTION AND JUSTIFICATION OF THE USABILITY FEATURES

The Inputs and outputs as default will be stated below for the game controls. This has been ultimately decided from research into different games and what my stake holders thought would be most appropriate. The moving controls are all based around one area on the default keyboard, reducing the chance of wrist restrain and complications. The left and right click of the mouse is used as the same as it would be in similar games allowing the player to not learn any new controls, this is the same with the spacebar and scroll wheel. The speakers will output sound effects like in real life making the experience of the game feel more authentic.

- A-move left
- S-move backwards
- D-move right
- W-move forwards
- (left click)-break item in front of the player
- (right click)-use item in hand
- Built in Speakers-project the sound effects
- Scroll Wheel- Allows the player to select an item specifically in the tool bar with rearranging with the mouse
- (space bar)-Jump

For player's with disabilities such as colour blindness they simply need to go the settings/controls and change the no to a yes on the colour blind section, this will alter some of colours to more beneficial colourers for them, allowing players who have this disability to be less effective when playing the game.

POST DEVELOPMENT TESTING

The table below shows all the post development testing that would be done as the game is being developed and created

Test	Description of the tests	Justification
Movement	The movement of the player	To check if the player can move around using the correct keys.
Sound Effects	The Sound effects of the player	To see if when the player moves a sound effect is followed as well. This will be the same when the player swims
Weapons	The weapons can be mounted and used	To check if the weapons work when killing enemies or collecting materials
Enemies	The enemies attack the player and spawn at random or near specific area's	Make sure the enemies attack the player and the players health is decreased when done so
Login System	The login system works, and all the functions of the buttons works	To allow specific users to have their own account
Tree's	Tree's can be chopped down	To allow the player to gain materials
Environment conditions	Certain environments can hurt and damage the player	So, the player has something to survive too and has to use science to survive (main purpose of the game)
Players health	That when a player is hurt the health decreases and when a player eats food the health is gained back	To test of when the player is drowning or dyeing in some way the health is decreased or when the player is eating the health is increased like in real life

Day and Night	Testing if the day and night cycle work	To make sure the days are realistic and work making the game more believable
Swim	If the player can swim and if the player can drown	So the player doesn't drowned and can actual swim to different parts of the island
The Map	Can't fall through the floor no glitches	Allows the player to actually play the game instead of a glitch which the map has stopped them from continuing the game

DEVELOPMENT AND TESTING

CODED SOLUTION

This bullet pointed list states how I created my game and overcome the problems I Initially faced.

- Created the structure of the files
- Created the player
- Added weapons
- Added movement first person camera
- Created the map and imported the old scripts into the new map
- Added sound effects
- Created a login system
- Added day and night cycle
- Made tree physics (hitting a tree with an axe and it falling over)

THE DIFFERENT PROTOTYPES

I created 3 different prototypes based on big jumps between what I am about to code next and the previous coded versions.

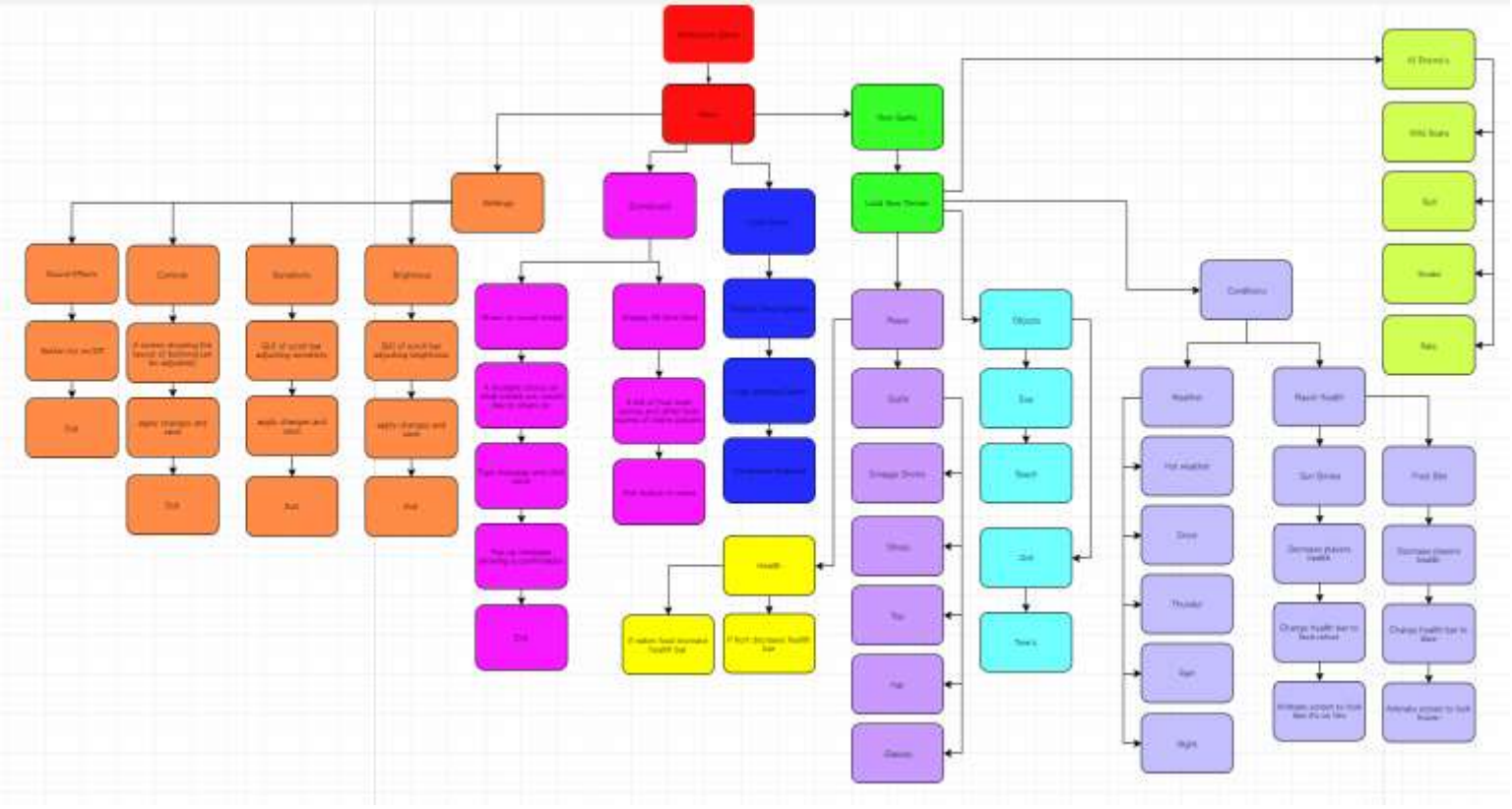
Prototype one included an imported map, coding for the movement of the player and the structure of the files. I then used this structure to import guns and drag and drop skins into the guns and hands then to position these in the correct format. I could see where they needed to go due the first person camera added here. I finally added sound effects.

Prototype two was all based around the login system. This was created by first adding a new scene to the game to go onto designing the GUI(graphic User Interface). This involved resizing 3d object adding text boxes and making password text boxes produce hash symbols when typing in. I then created a camera to over the look the 3d model making the illusion that it was 2D. I then had to state the rules meaning I had to assign buttons to a function for example make them check if the username had already been taken if the password matches to an account already created or encrypting/decrypting the password.

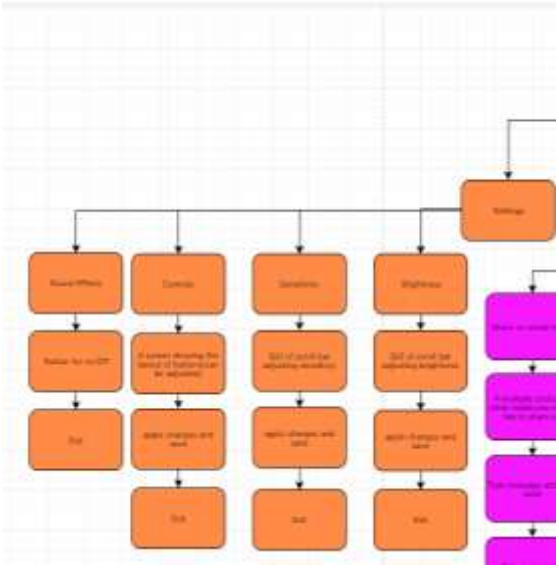
Prototype three was based around creating the day and night cycle facing the issue stated by Luke Callison Bailey (one of the stakeholders) of too many trees. I created the code for the day and night cycle using the position of the light to change and the brightness of it. I then had to create a time for each day which I worked out 10 minutes after research was the best time for full day in the game. Finally I added paths to the game so there wasn't as much cluttered tree's to fix the users issue but then decided to go onto added tree physics which required me to remove the pre-made trees and replace with the new ones. I then made it so it 1 is pressed on the key board the axe (imported from prototype one) is I n hand and three hits t a tree would make it fall then disappear after 4 seconds.

FEATURES THAT WASN'T ADDED BASED ON THE PREVIOUS DROP-DOWN MODEL

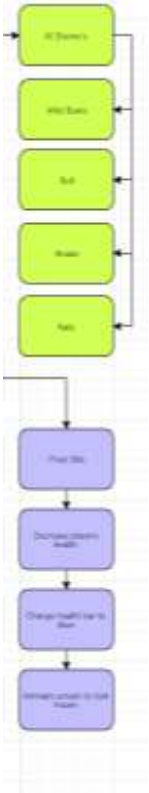
The drop-down model below is imported from the previous analysis making it easier to compare too. I am going to split each of the model into parts to show where they have fitted into creating the game.



Below is part of the Drop-Down model for the settings of the game. Due to this not being an essential part of the game, where the



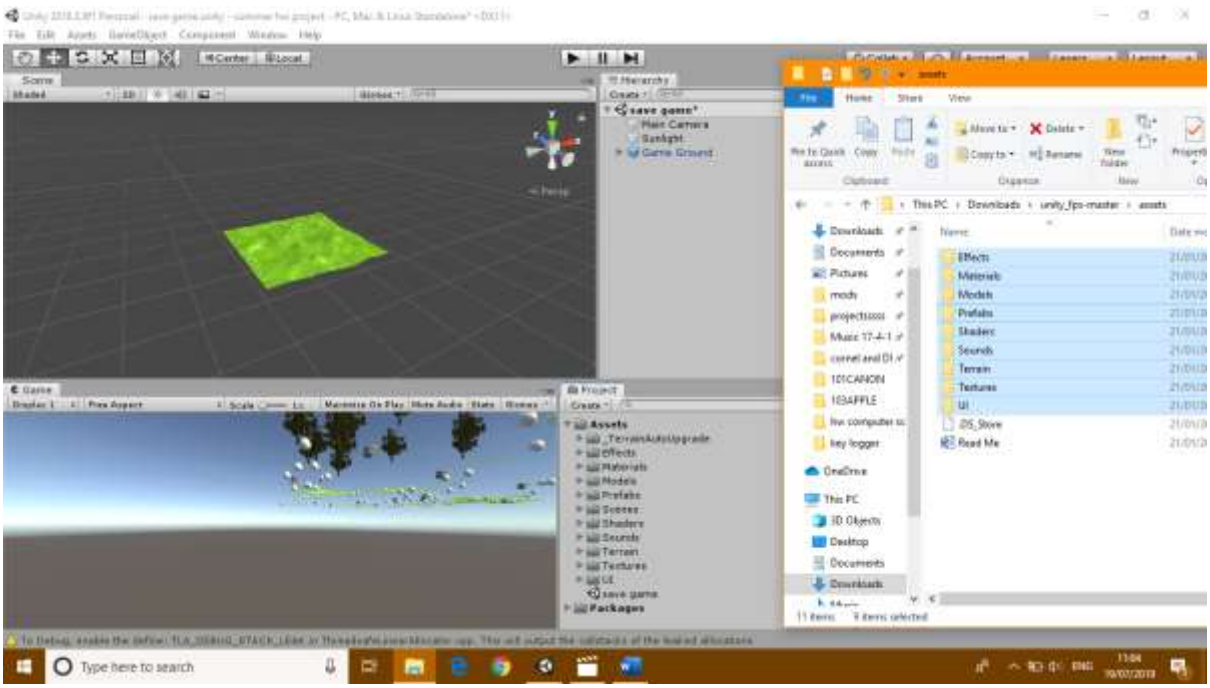
game can run without it smoothly it wasn't added (due to games time deadline). However for future development of the game this would be added, as it does allow the game to be played more to a players preferences and would be created as one of the next tasks. I made this decision based on my stakeholders and what I thought the game needed to have and didn't need.



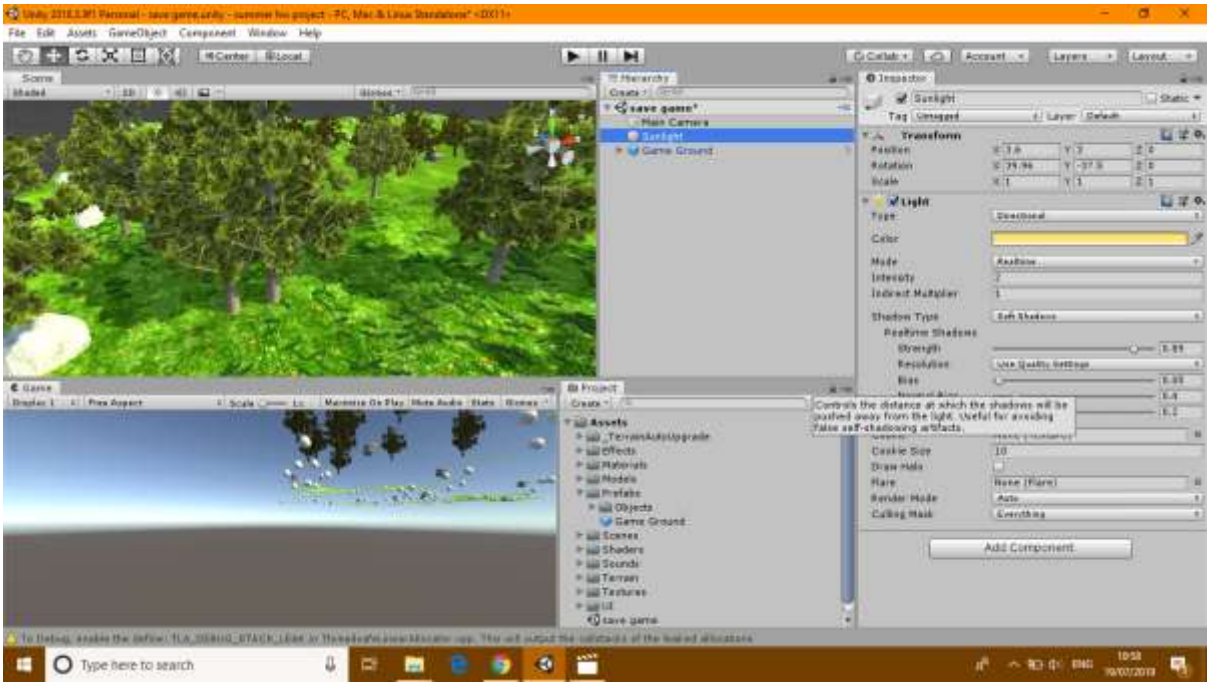
Above is another part of the Drop-Down diagram that couldn't be added due to time. This part was based around enemies and things the player would have to fight against in the game. This would of made the game more challenging for the player but wasn't added due to the game wanting to be based more around factual things to survive then fighting enemies.

The scoreboard wasn't added due to not being essential part of the game and wouldn't affect the actual gameplay, only to be used to compare against other players. This would have been added if further time was given but wouldn't be close to be creating near the current position.

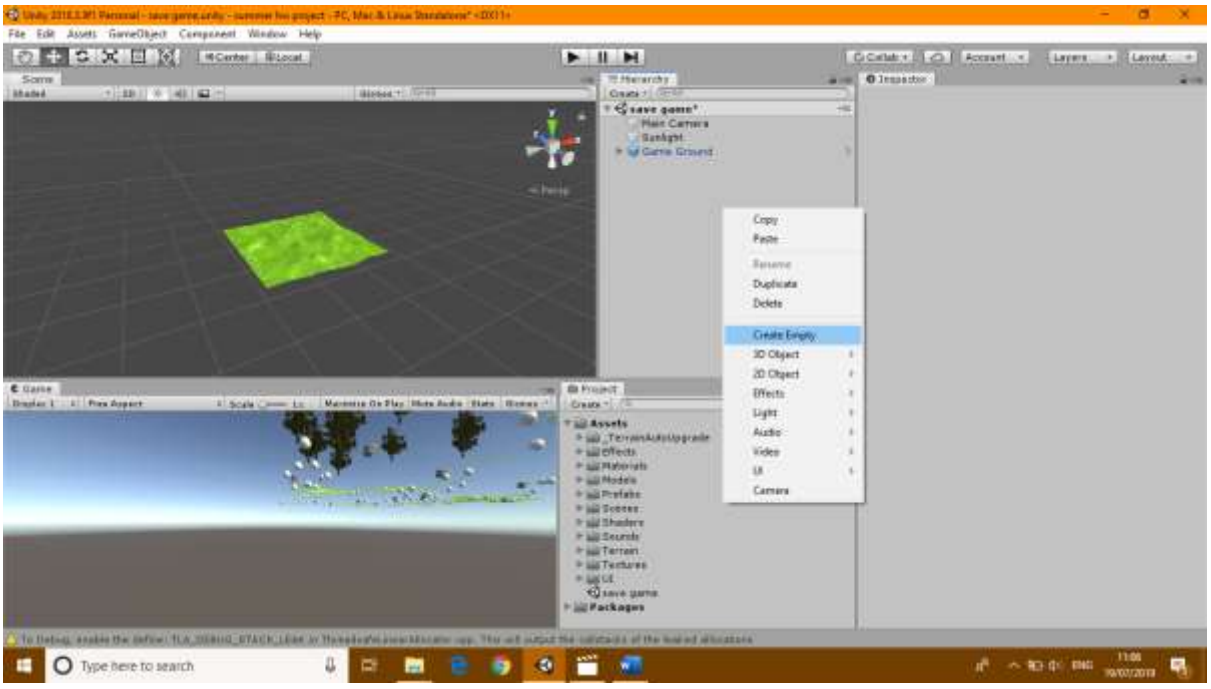
FIRST PROTOTYPE -THE BASIC'S OF THE GAME



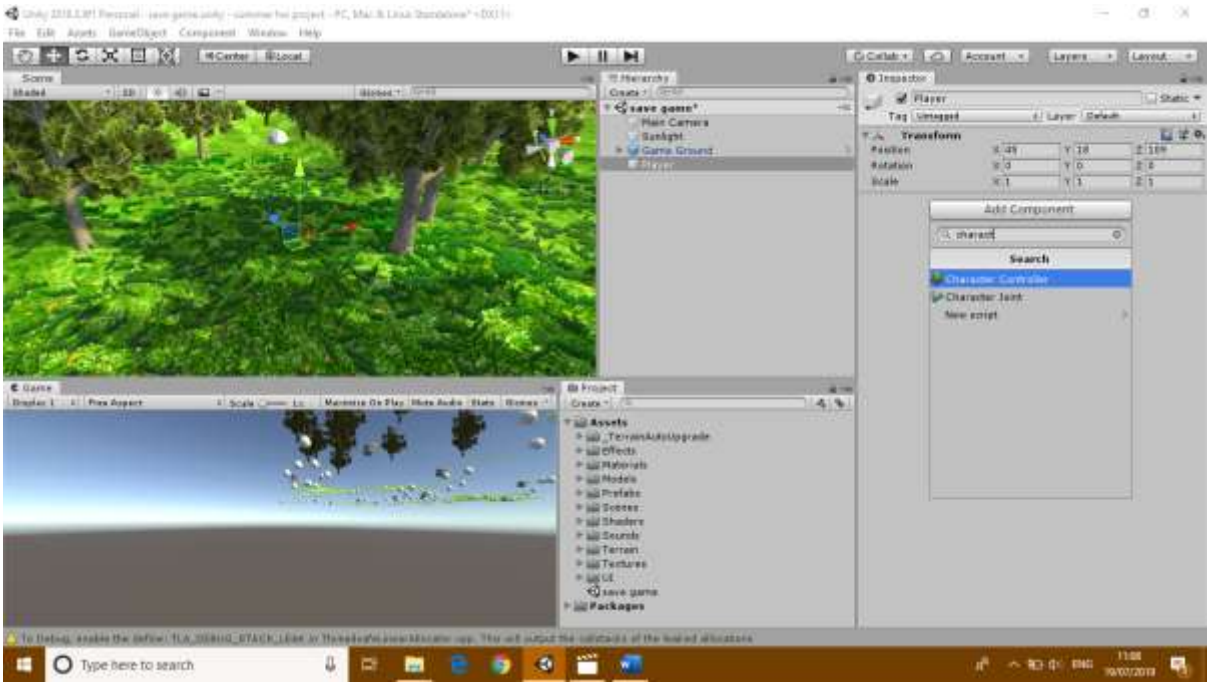
I first created a new 3D game in the unity platform Calling it "Prototype one" I found a specific tutorial for a very similar game and thought by started to use some of the tutorial's ideas would make a good foundation for my game. I proceeded to download the assets from this tutorial and drag and drop them into my assets folder to my game. This proceeded by automatically adding a pre-made map.



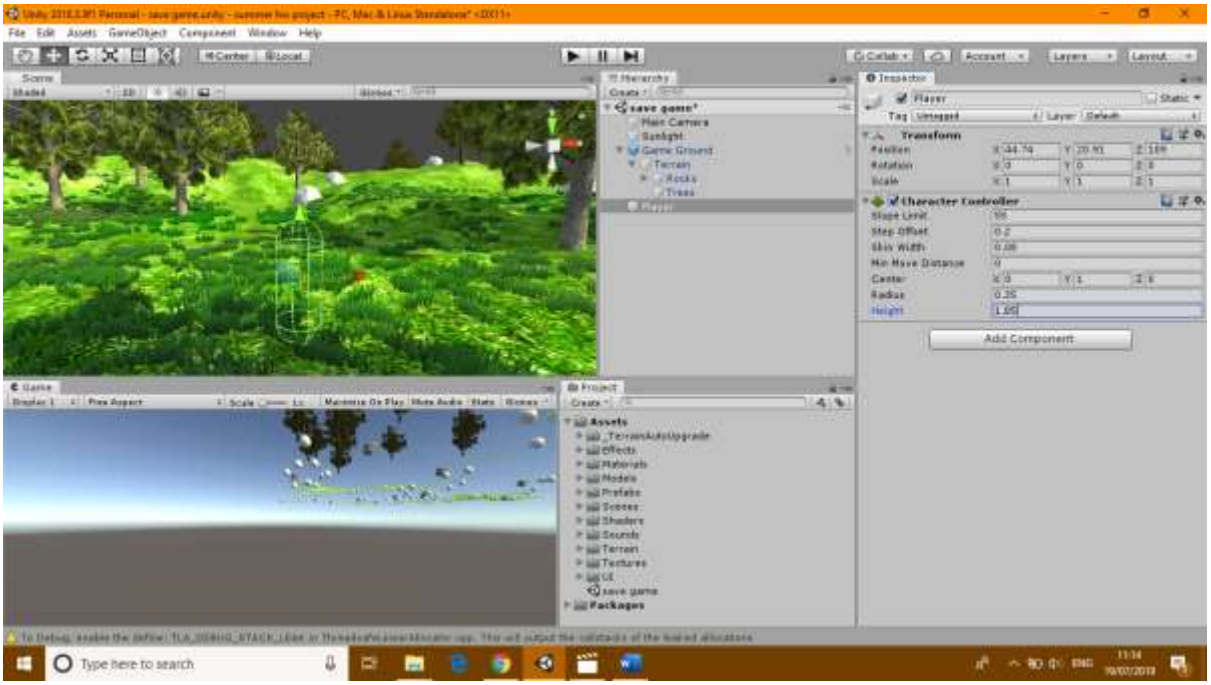
Here I changed the light intensity making the terrain look bright to represent midday I also made the shadow strength not as strong once again given the allusion that its midday. I changed the colour of the sun to the hexadecimal value of FFE56D showing the same colour of the sun. I finally changed the directional light name to sunlight.



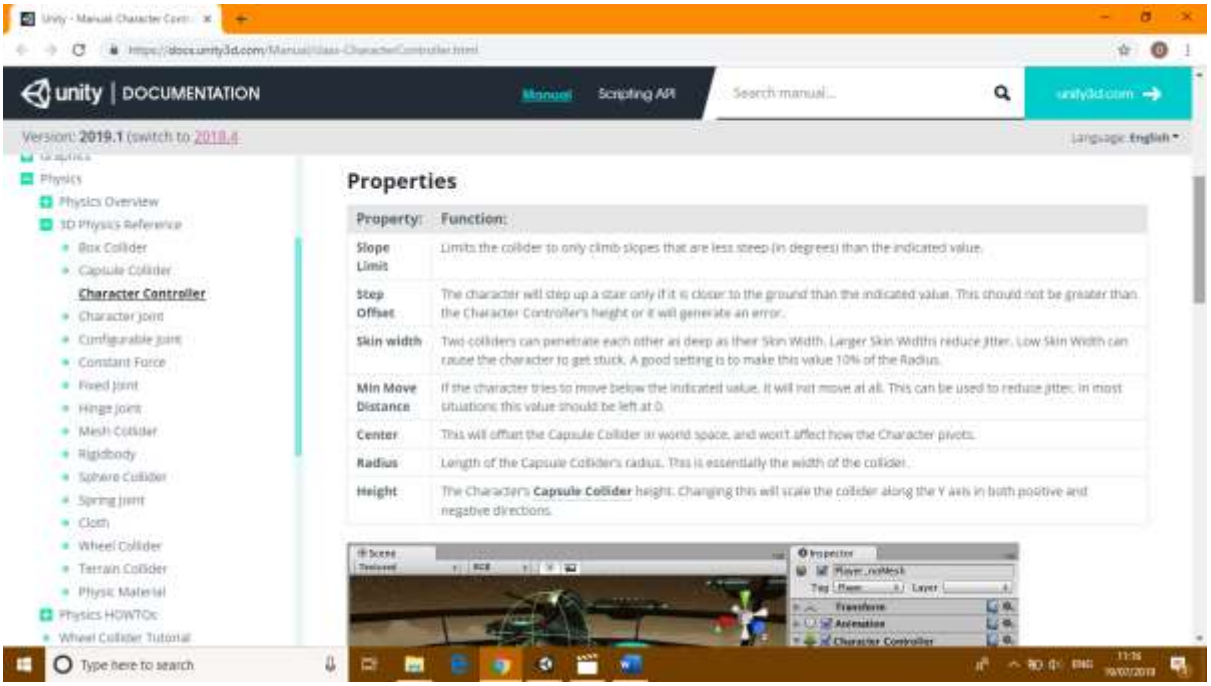
I created an empty game object renamed it player and positioned it to where I wanted to spawn my player

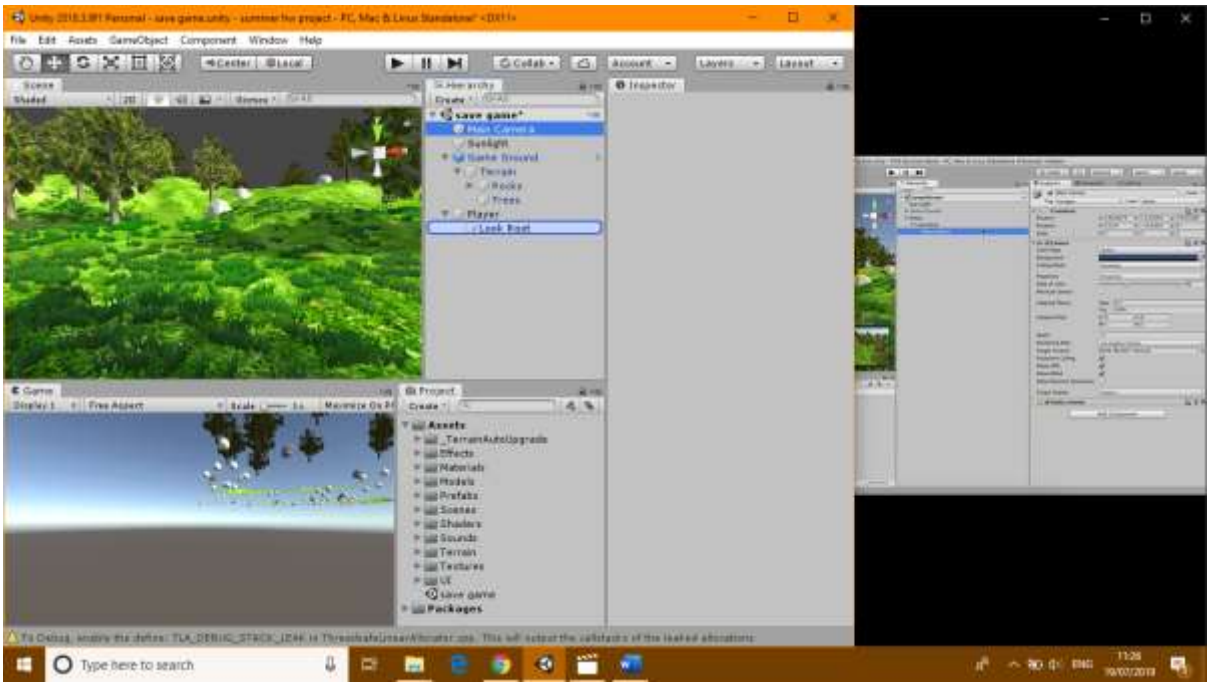


I added a component called character controller onto the player object allowing the player to move

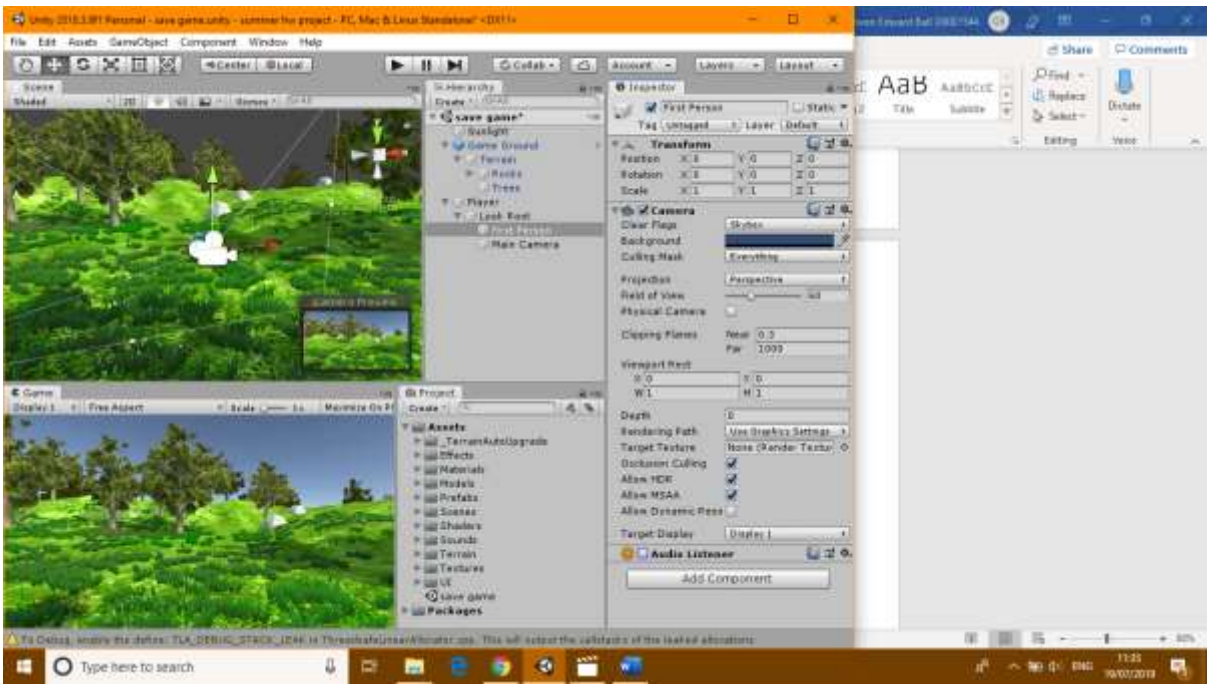


I adjusted the stats the player in the character controller using reference to the picture below

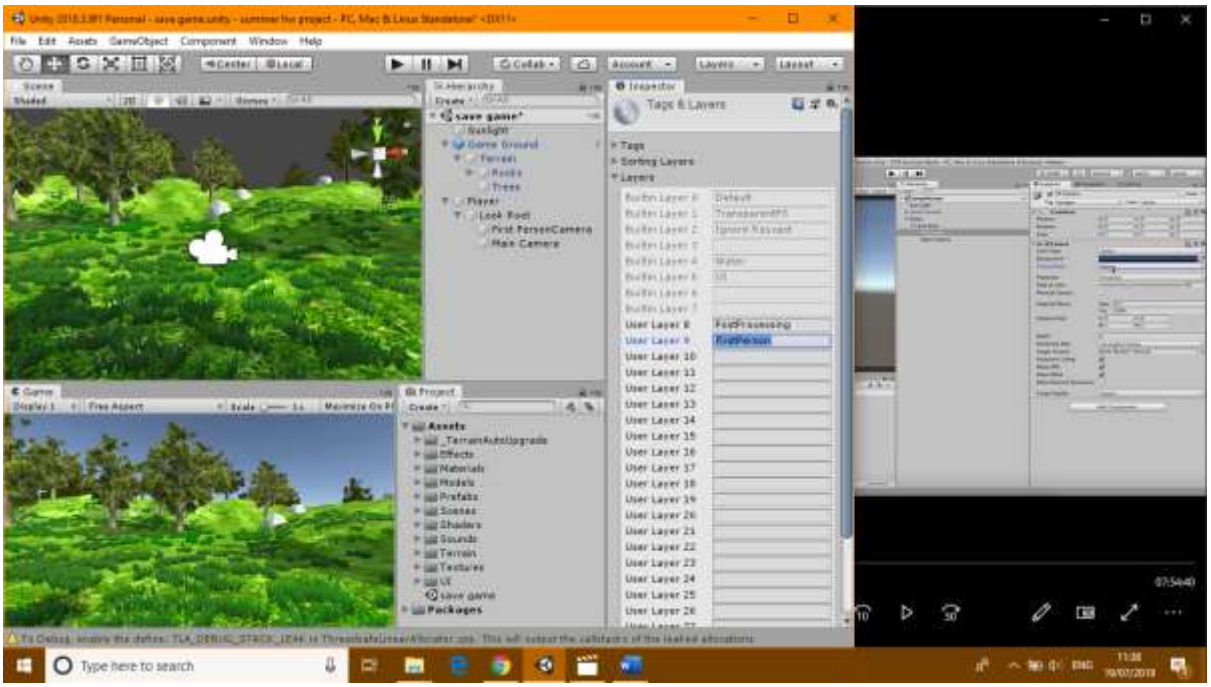




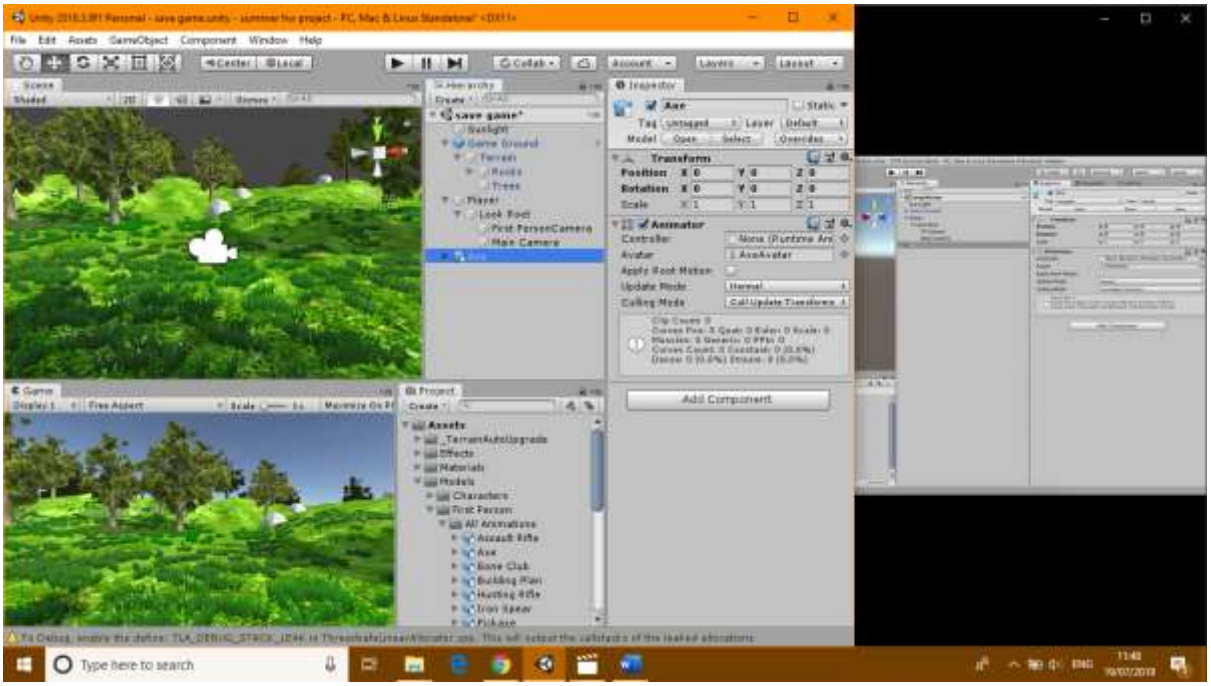
I created another empty object in the player object and renamed it “Look Root” symbolizing what the player see’s in the game. Proceeding to dragging the main character into the “Look Root” object to make this happen. Next I reset the transform so that the camera is correctly previewed in the game tab.



Another camera is added the “Look Root” object for first person view, it is then renamed to first person and is unticked for audio listener due to the main character having that ticked therefor saving processor power

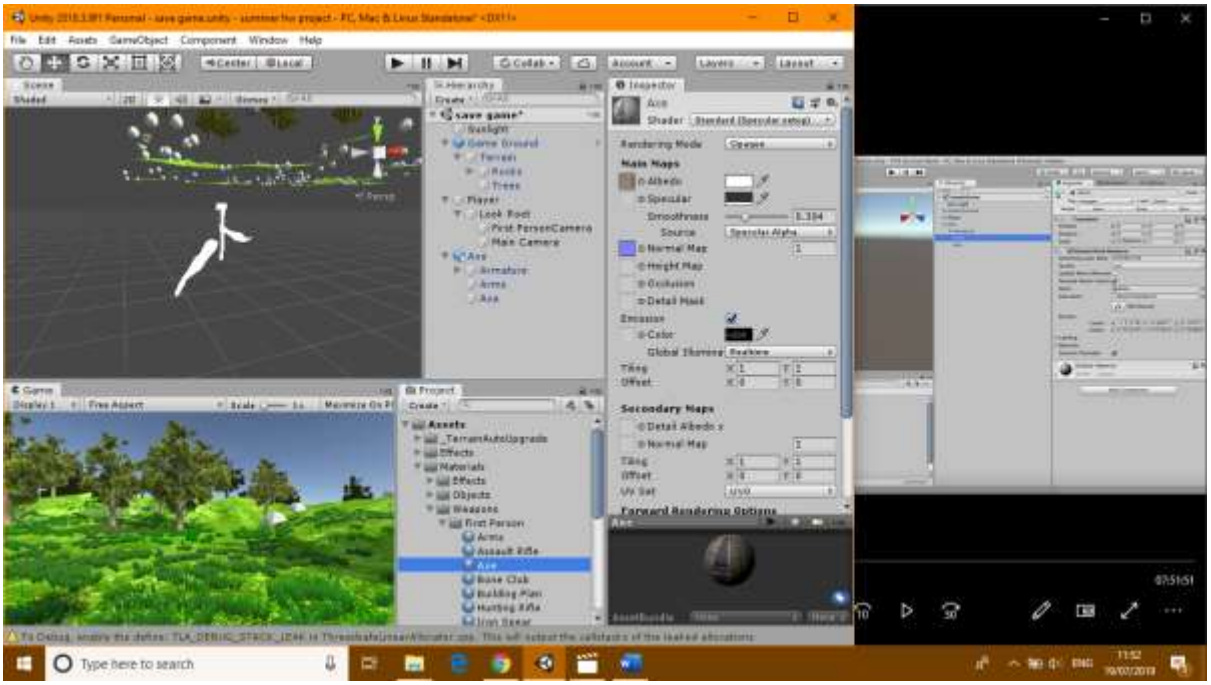


We do these following steps to allow the first person camera to use weapons. A layer is added to the FP camera changing the culling mask to nothing and then to first person, we then change the clear flags to Depth only (this makes the camera view only see the first person layer objects). The field of view I reduced to 45 near is changed to 0.01 and far is changed to 10. Finally we go onto the main camera and uncheck the “firstperson” layer so the weapon isn’t seen in the main camera view.

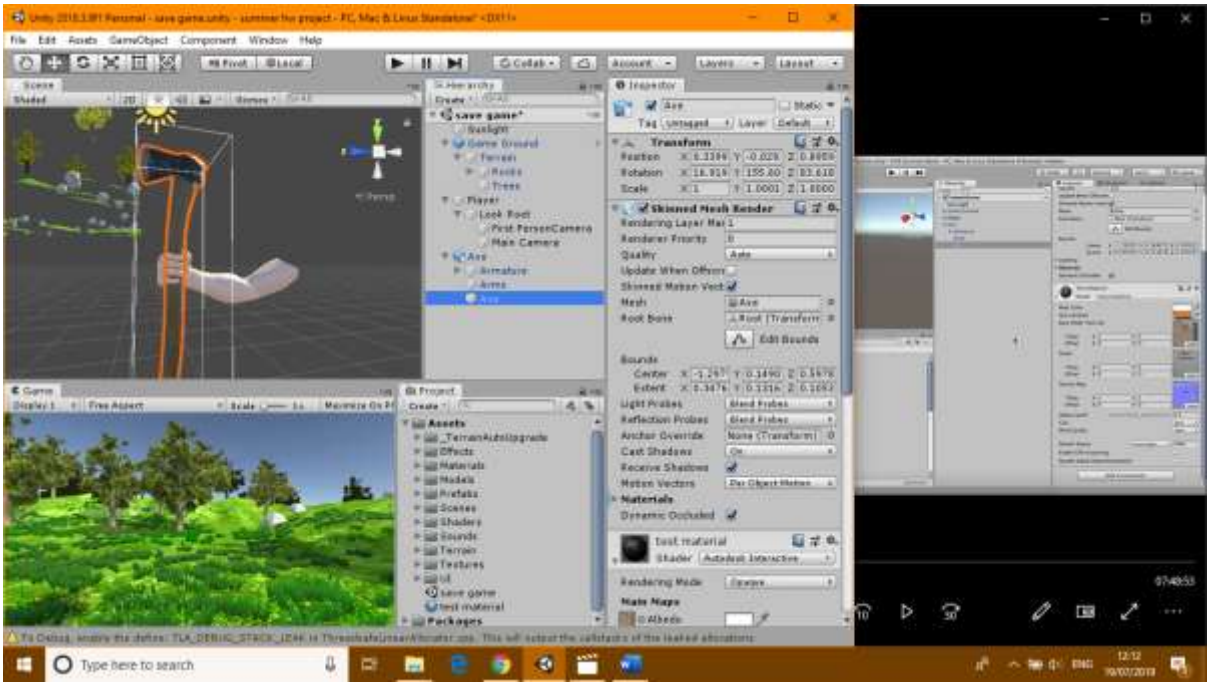


A premade weapon is added to the game e.g. an axe in this case by going into the “assets/model/firstperson/all animations.axe” then its dragged into the console allowing it be added to the game

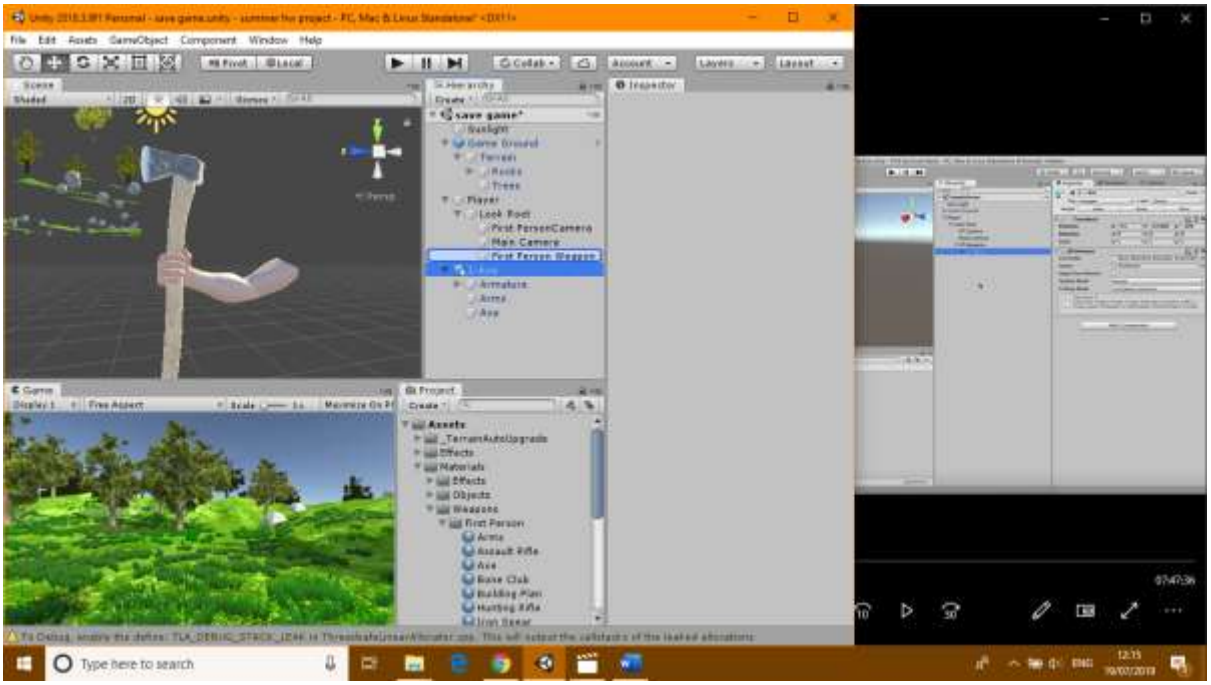
FIRST PROTOTYPE -ADDING WEAPONS

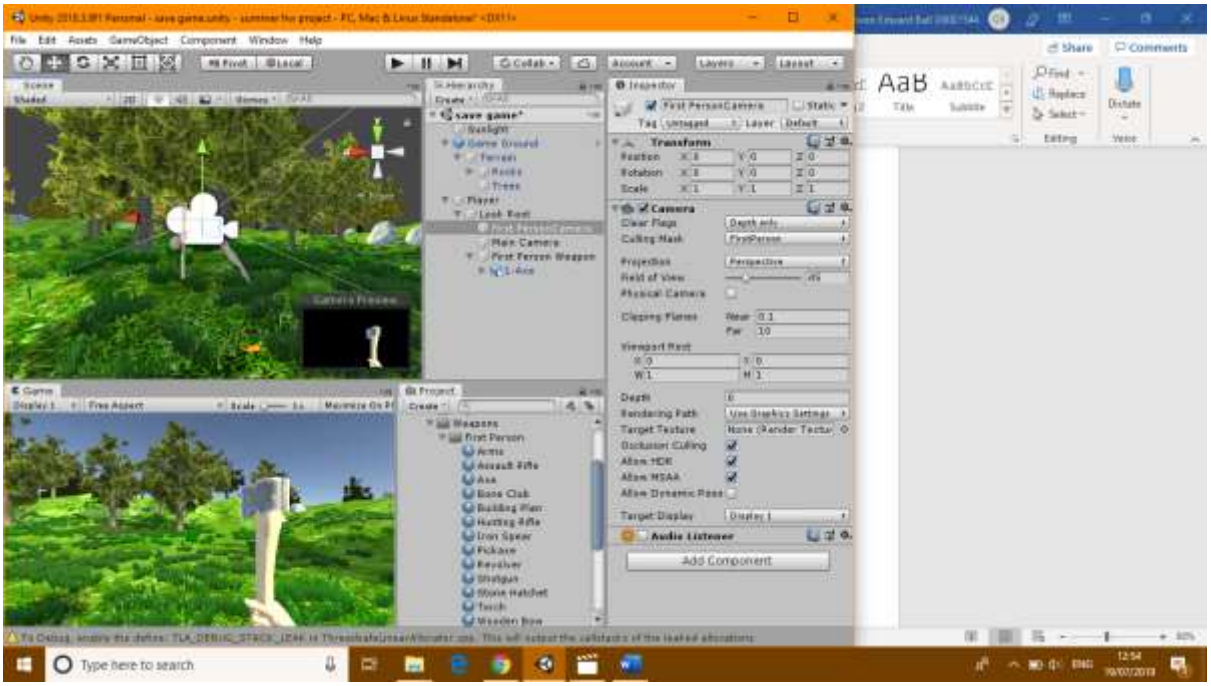


A texture is then added to the axe and arms by looking at the materials in the assets/materials/weapons/first person/Axe this then displays a wooden texture made for an axe and the arms is the same folder for the texture fir a set of arms. This is then drag and dropped into the appropriate destination

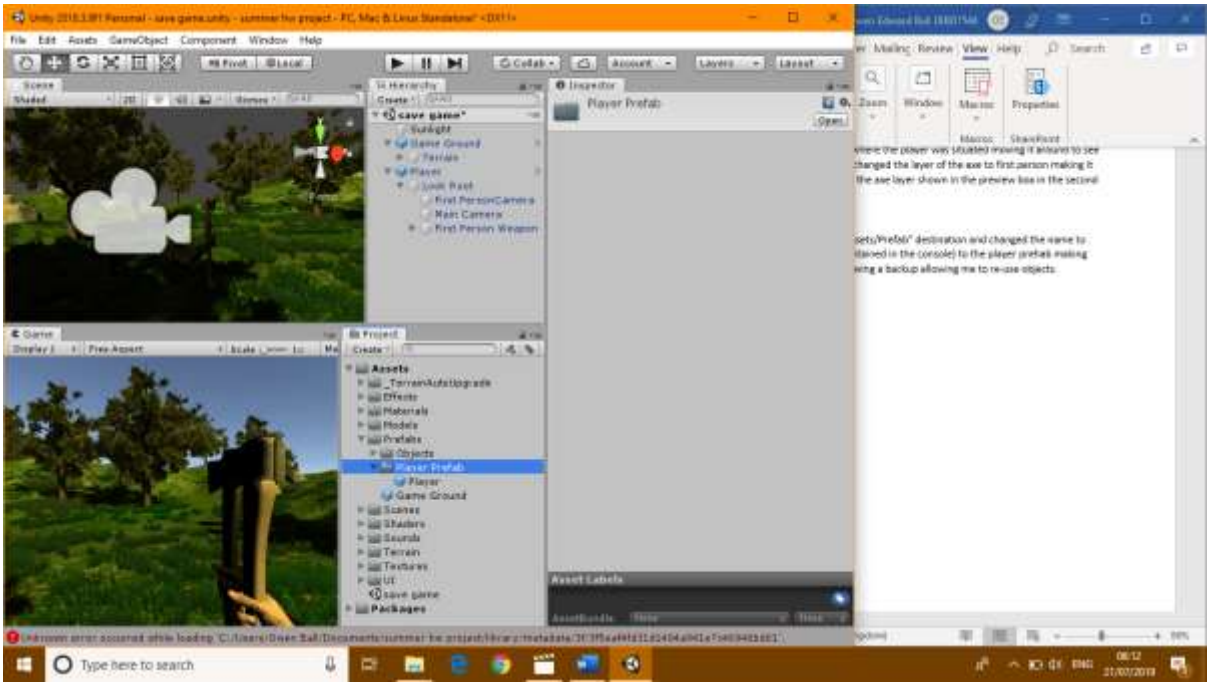


I attempted create my own material by adjusting the shaders texture etc.

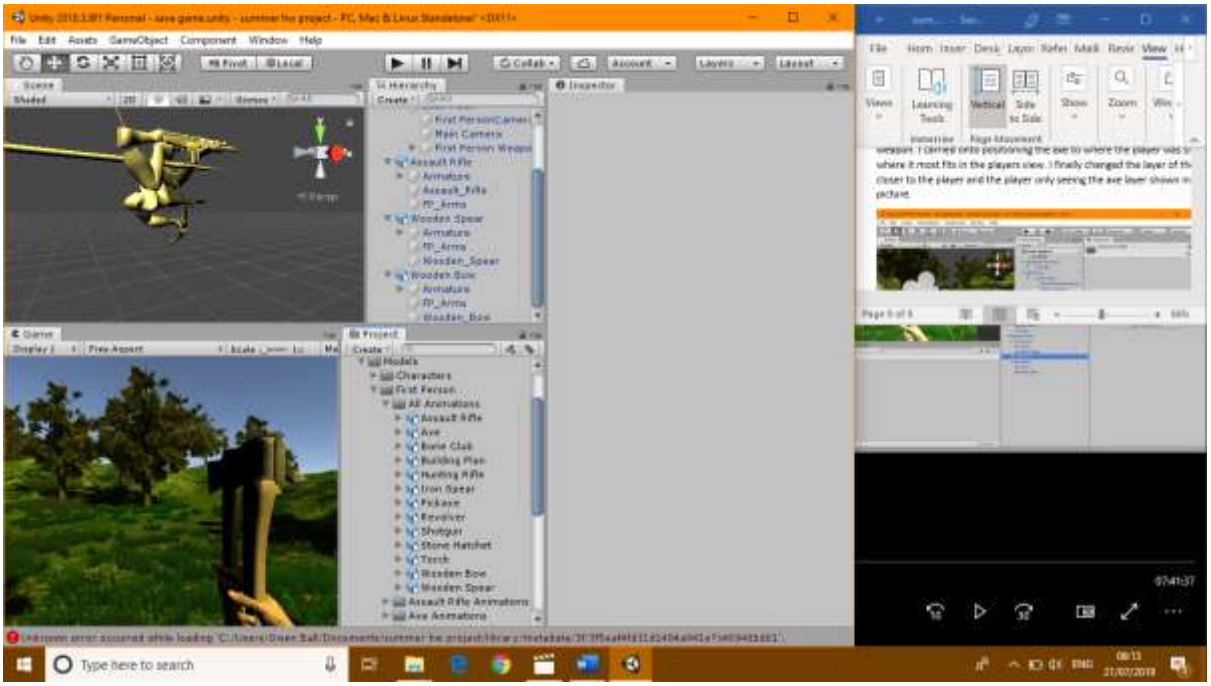


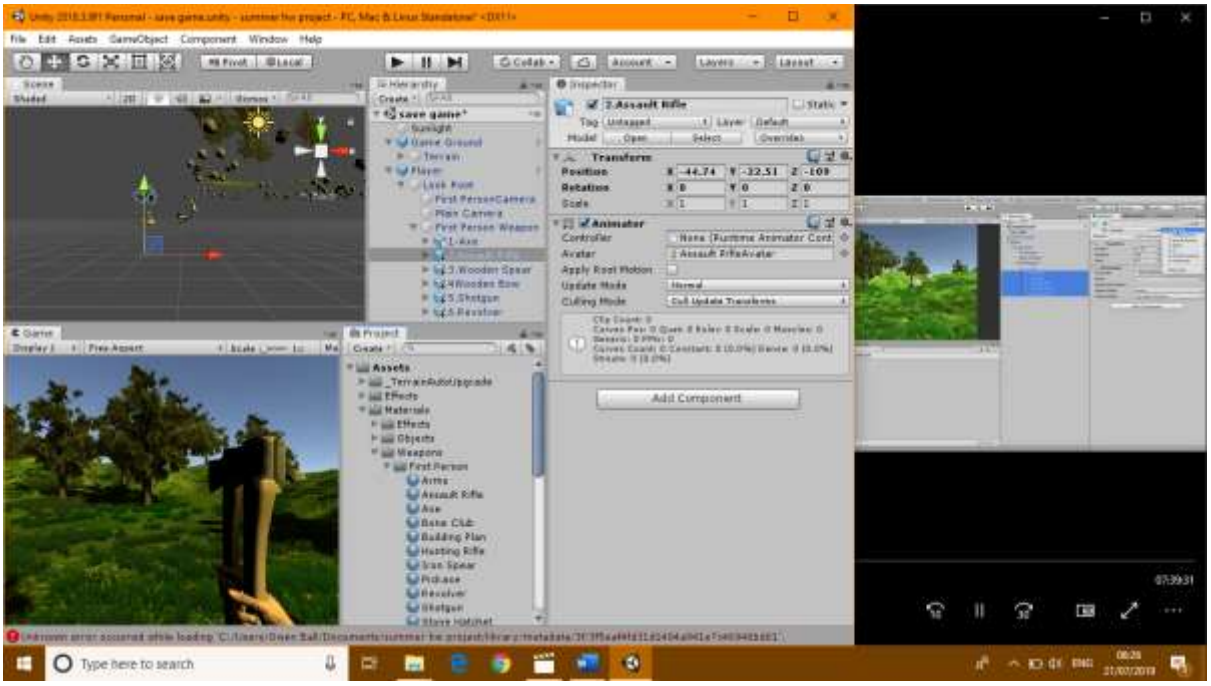


I created a new empty object in ”Player/Root Look” renamed it to ”first person weapon” as well as changing the axe name to “1-axe” for the axe item and added the axe object to the first person weapon. I carried onto positioning the axe to where the player was situated moving it around to see where it most fits in the players view. I finally changed the layer of the axe to first person making it closer to the player and the player only seeing the axe layer shown in the preview box in the second picture.



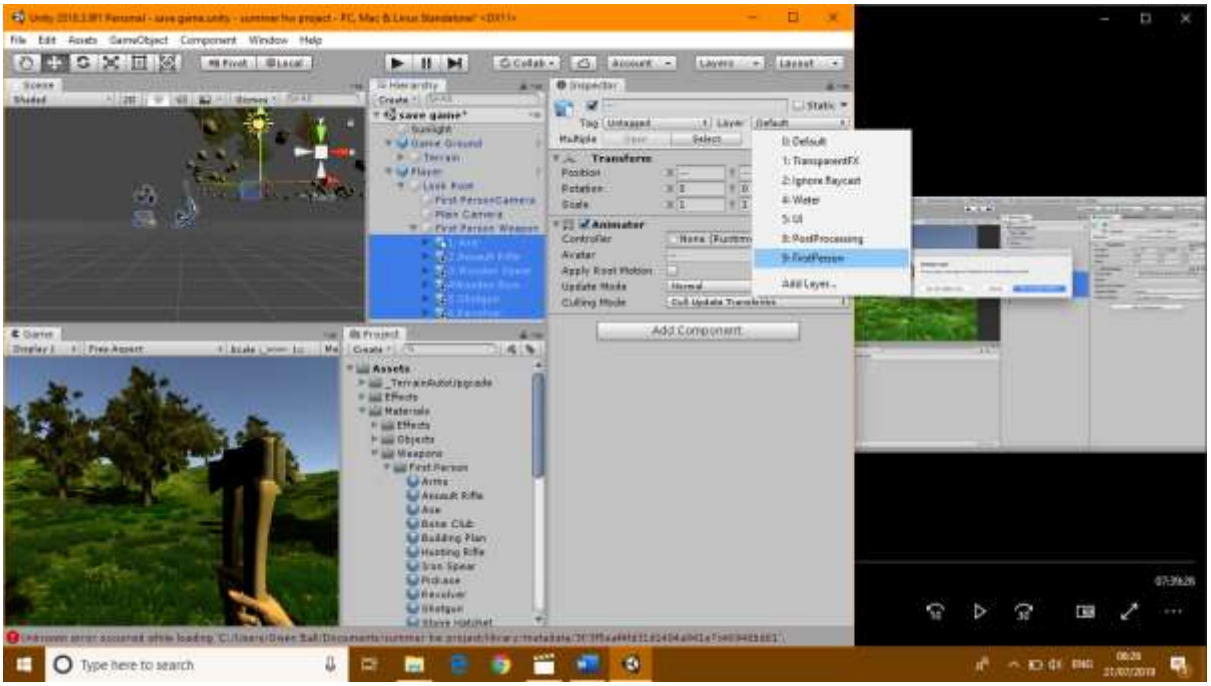
I moved onto to adding a new folder in the “assets/Prefab” destination and changed the name to “Player prefab”. I moved the Player folder (contained in the console) to the player prehab making my structure of folders more organised and having a backup allowing me to re-use objects.



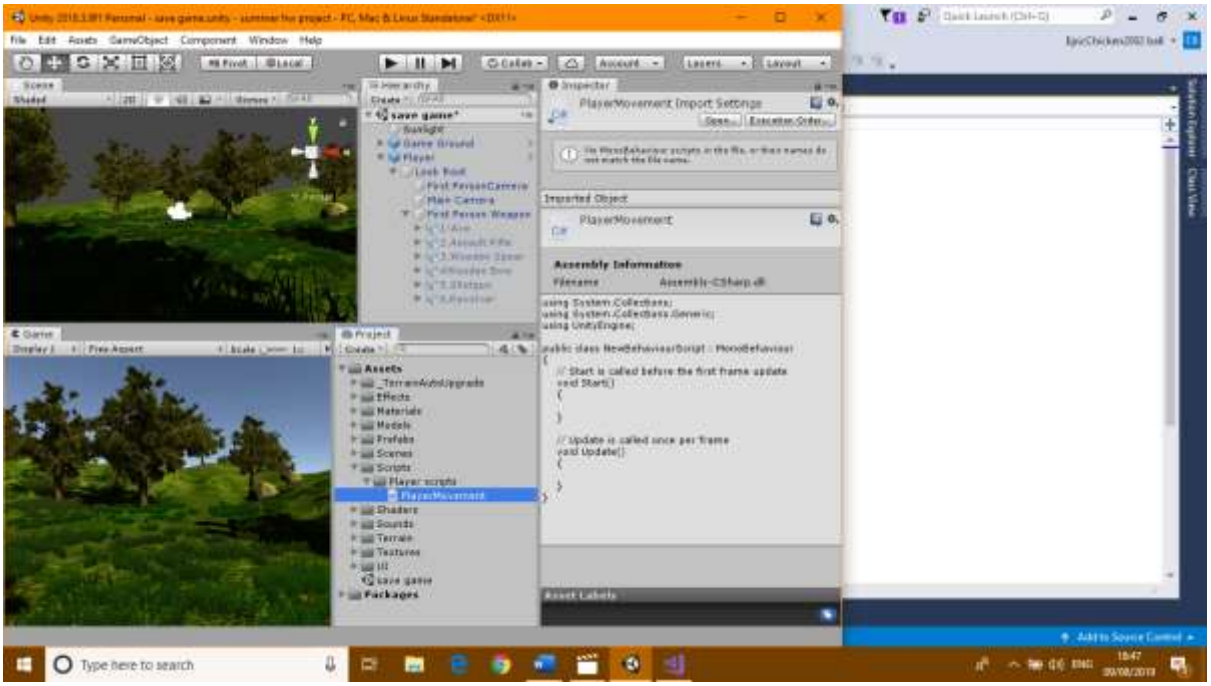


More weapons was added to the game such a spear, bow and assault rifle. This was accomplished by adding the premade weapons from the “assets/models/firstperson/all animations” dragging them into the console where the player file is then each one is added with the correct materials for each object. For example all of the weapons needed the arms skins which was gained from “assets/materials/weapons/first person/arms” and dragged into the arms object contained in each weapon allowing the skin to be shown. This is then repeated with skins of weapons like before. Finally, a renamed each weapon with a number like I did with the axe.

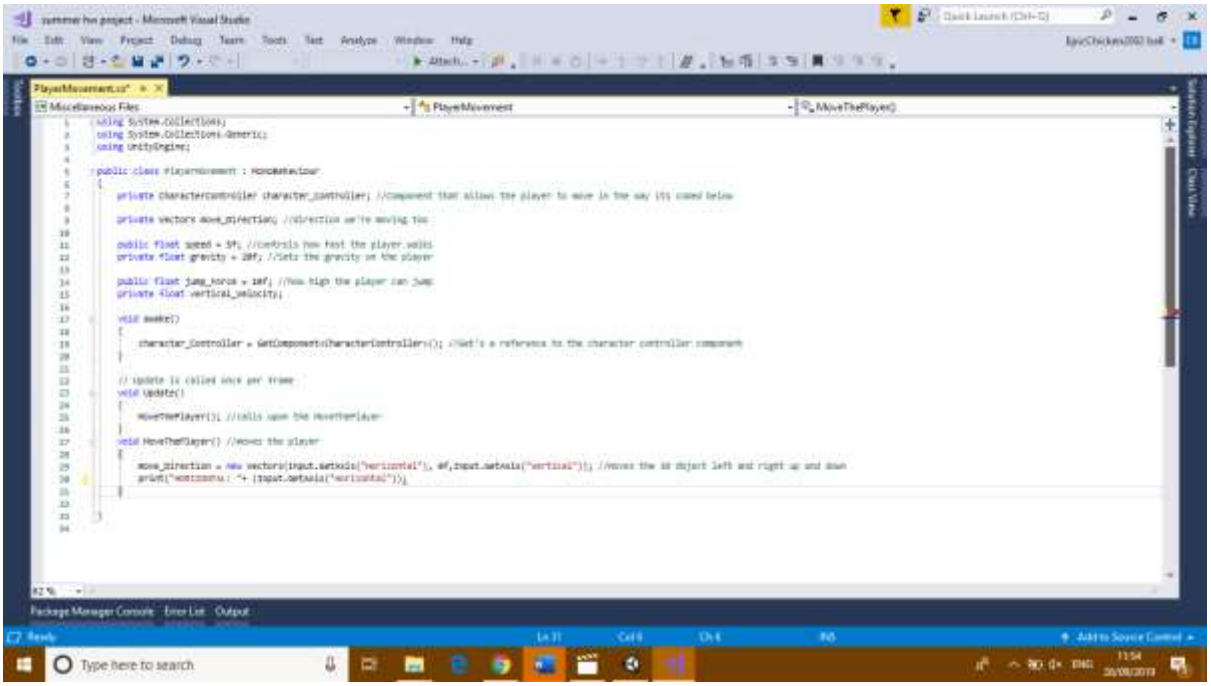
FIRST PROTOTYPE -FIRST PERSON CAMERA’S AND MOVEMENT



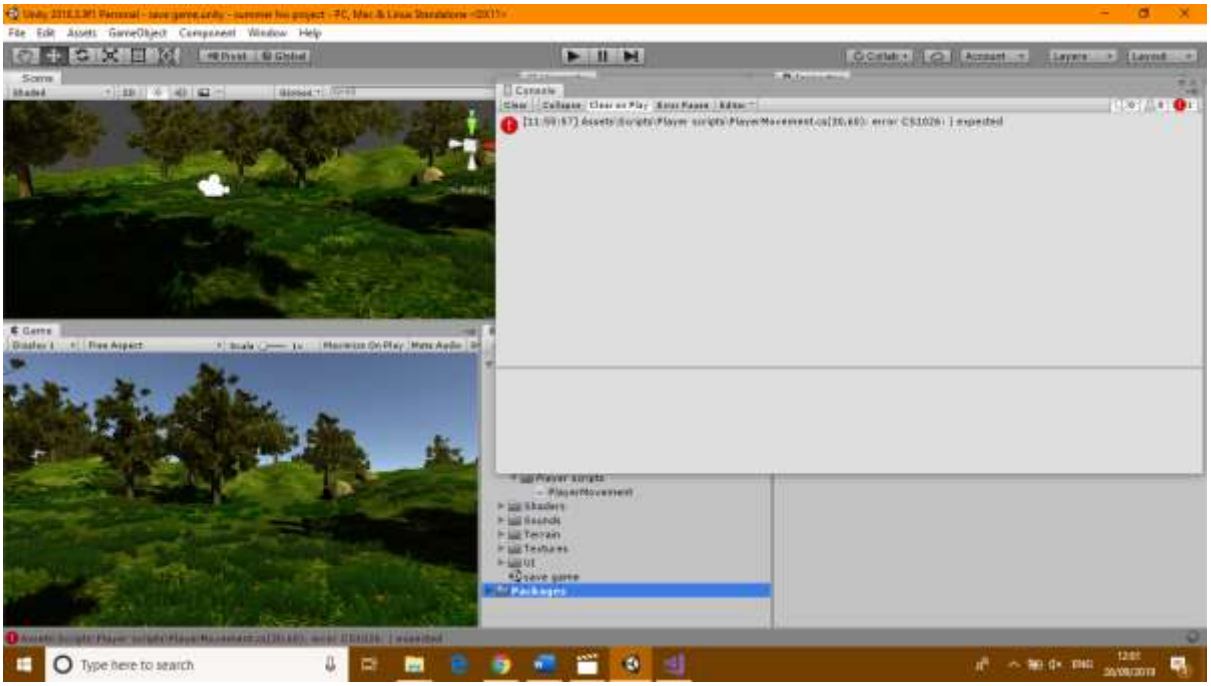
Highlighting all the weapons made and the changing the layer to a first person due to only being able to see the weapons in a first person view. Moving onto adjusting the coordinates allowing the first person camera to see the weapons instead of their default location when added. I used the co-ordinates from where the spawn point is and main camera then proceeded to adjust each weapon so the best view could be seen then apply the changes.



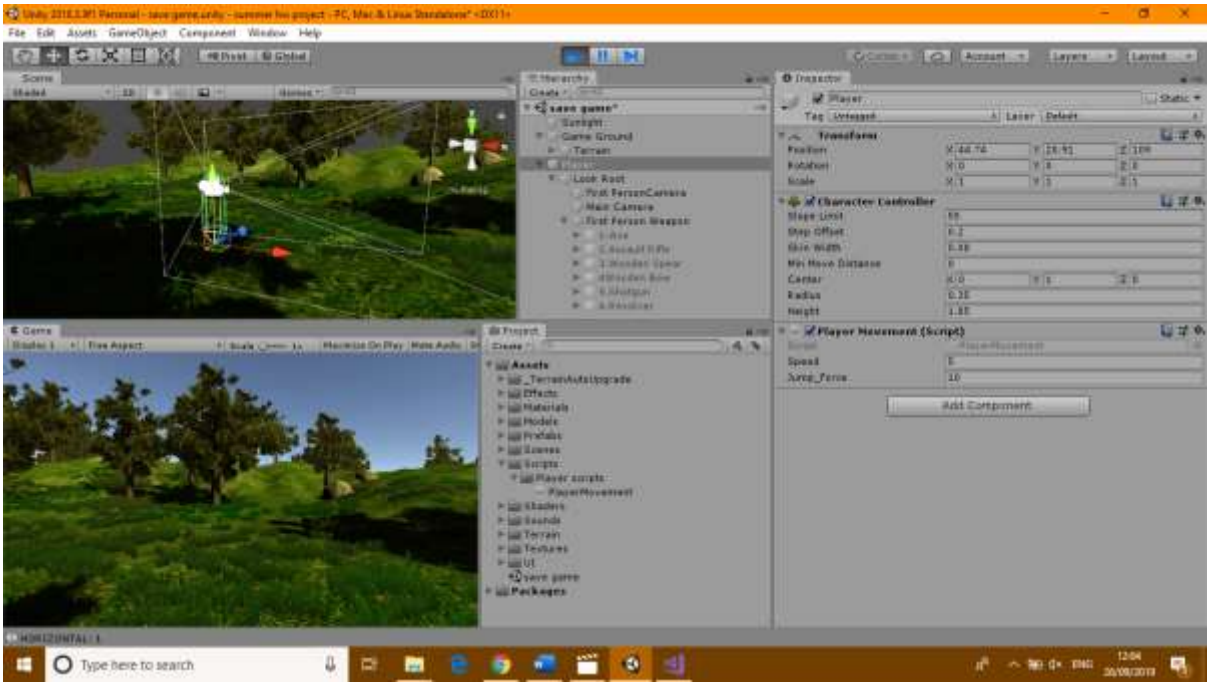
To make the layout of the folders more manageable I created a new folder in the assets folder called “script” this will hold all the scripts for the game. I then proceeded to make another folder inside this folder holding the scripts for the player. The first c# script I made in this folder was for movement.

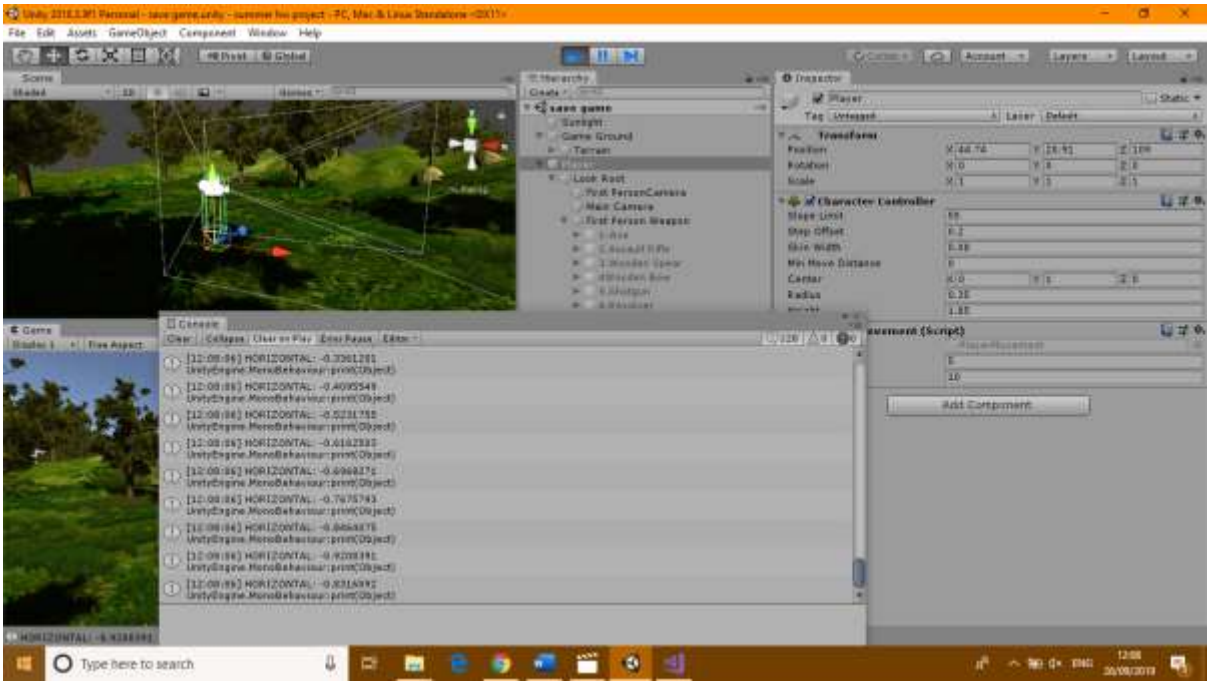


This is my script for player movement here I have made “print” statement allowing me to see if the left and right movements are working when I press them and they are printed to the console printing out constantly the coordinates of the player.

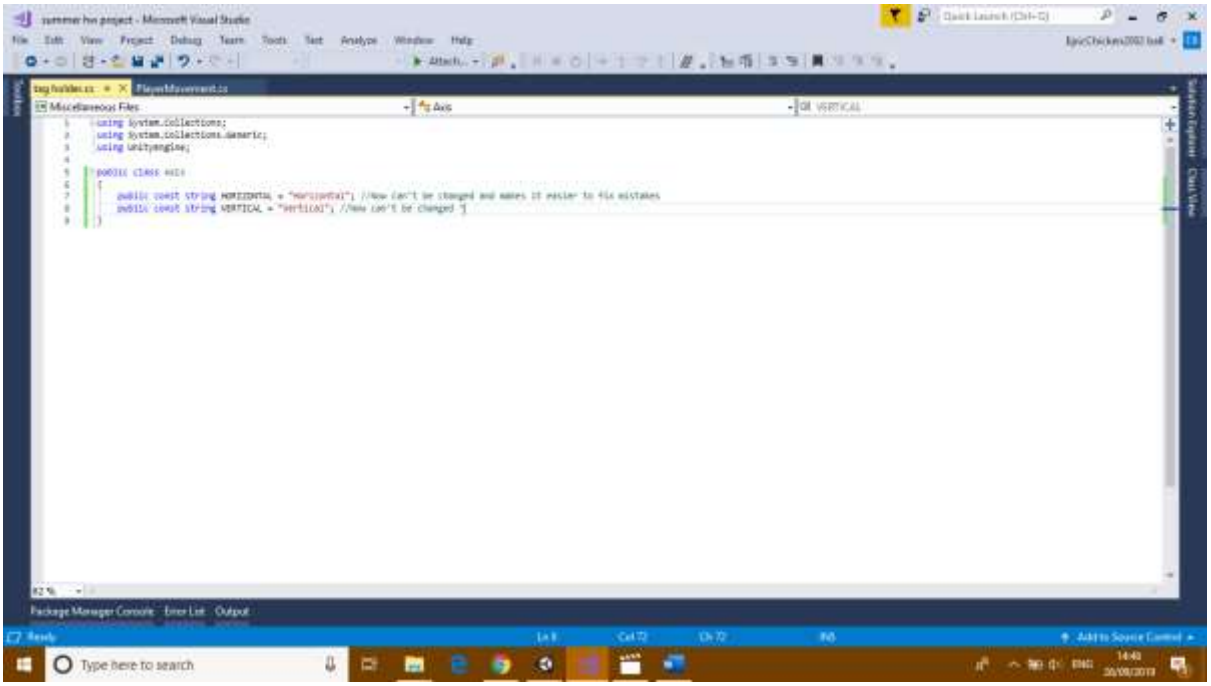


An error occurred I then looked back in my code to the IDE highlighting an extra bracket not needed

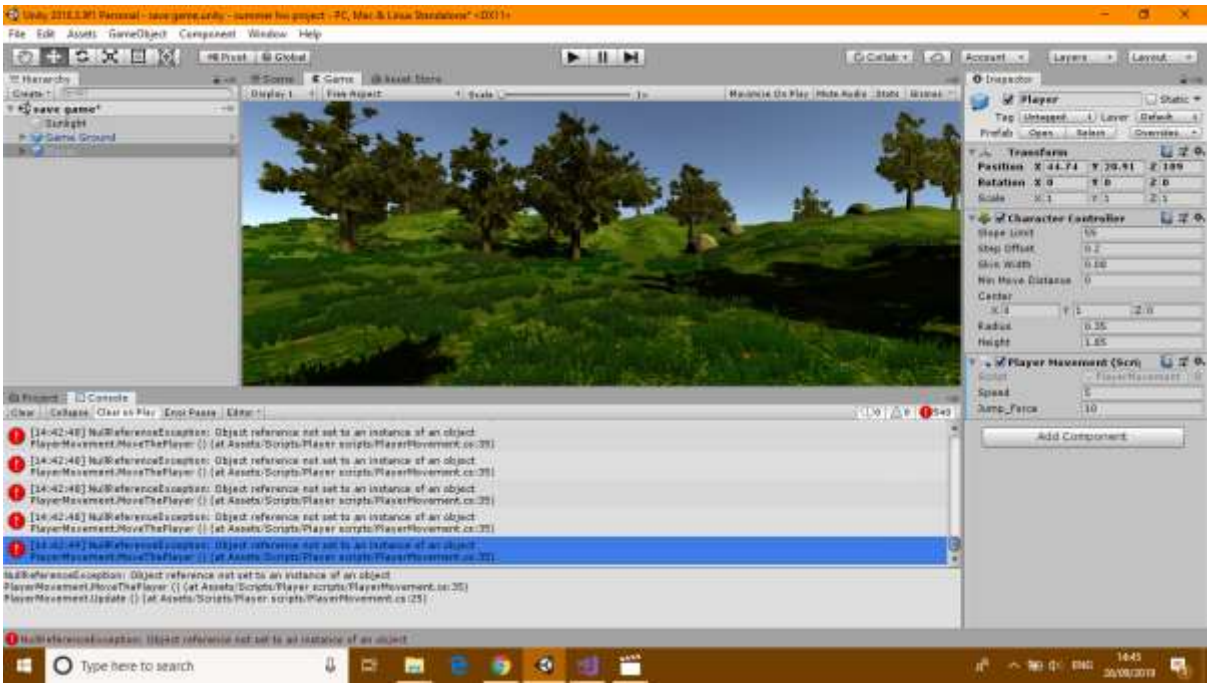




I then realised my script wasn't linked so I then linked it and tried and it was a success as shown above

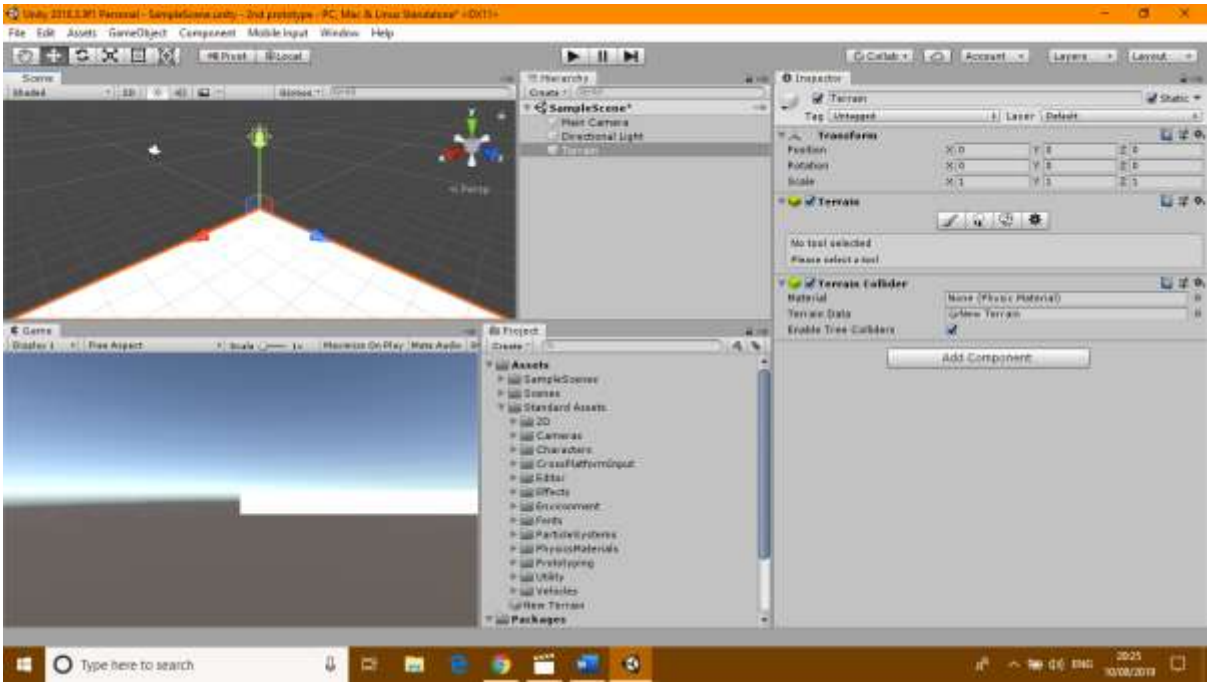


Proceeding to make a folder holding the helper scripts and a new c# script inside this called tag holder I proceeded to add public constant strings making it easier to fix if there's an error in the code

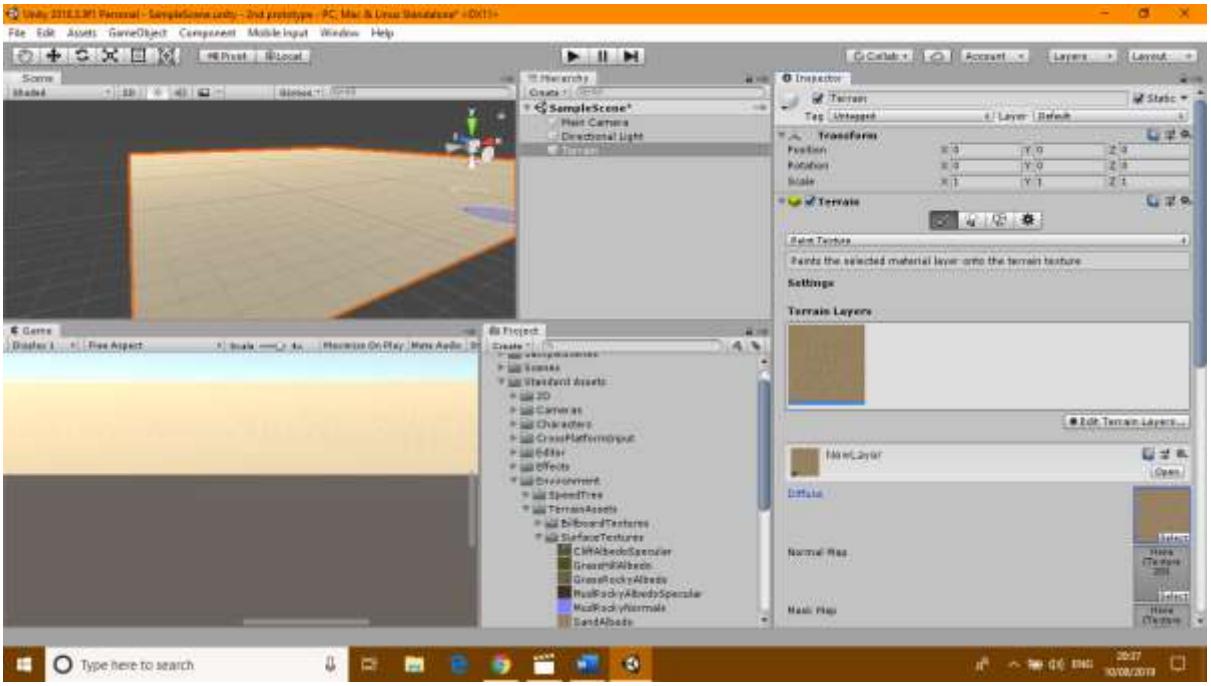


I came across this error as I went try out the movement script, after relooking over the C# code I realized I didn't capitalize the "Awake" function causing the error to occur.

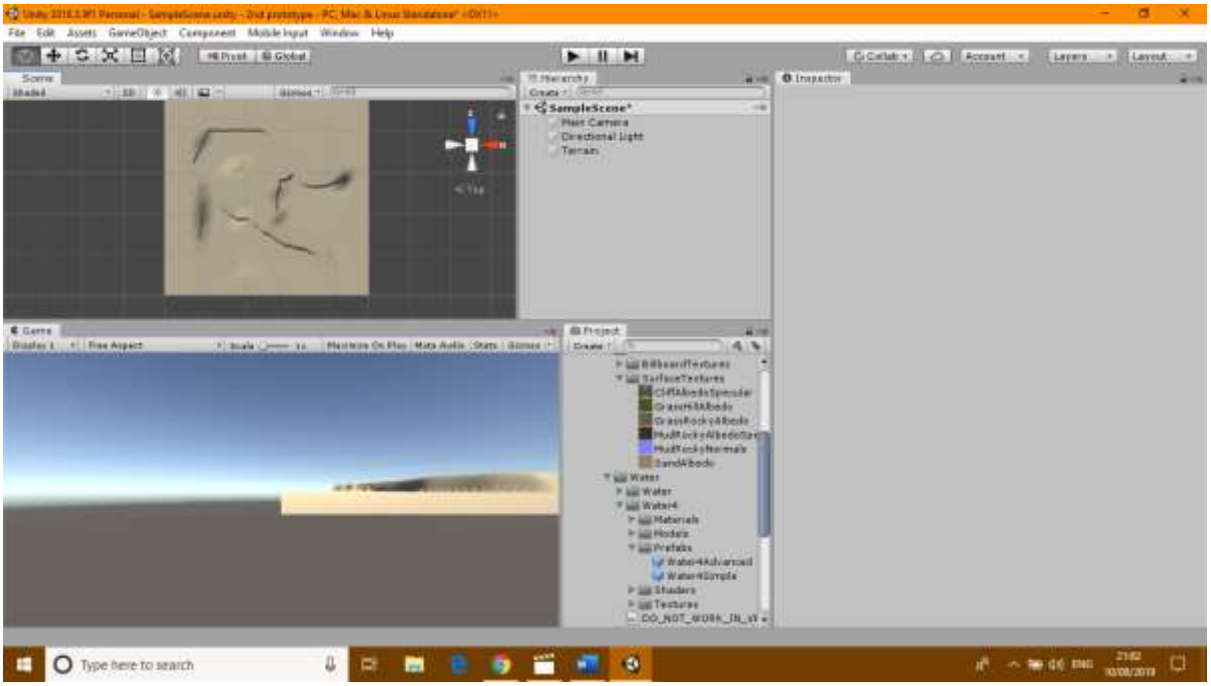
FIRST PROTOTYPE -THE MAP



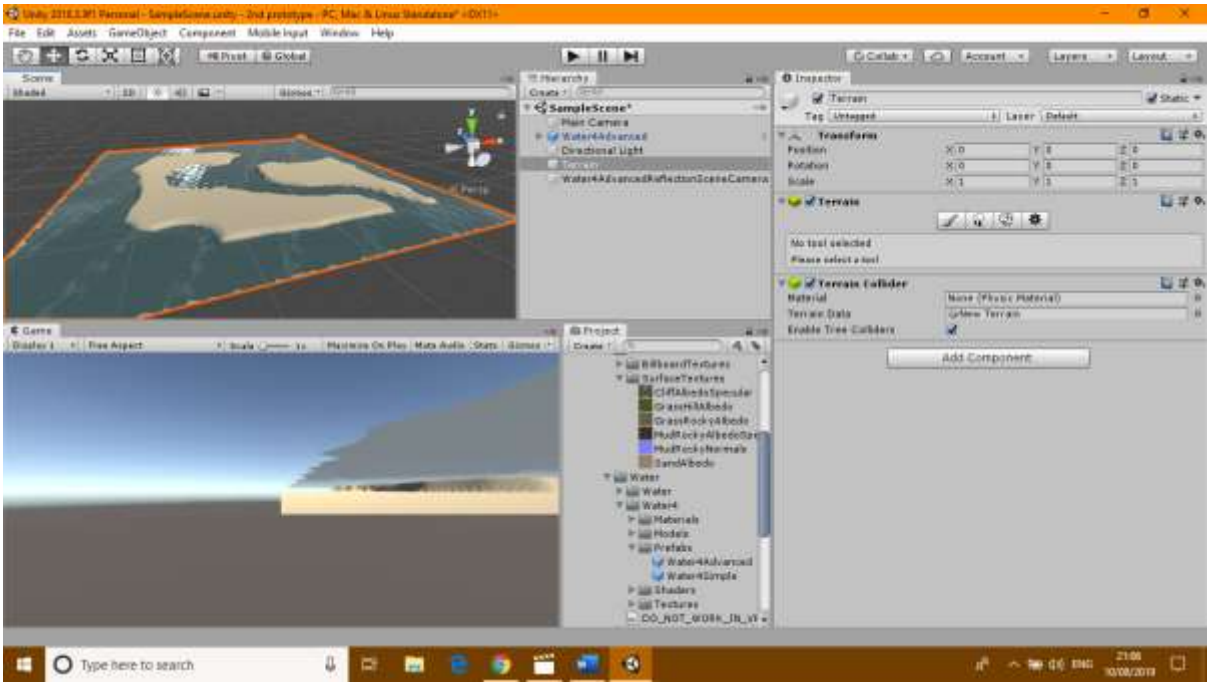
At this point I created my second prototype game but instead of importing it premade by someone else I would create it. I started off by starting a new project 3D with assets and adding some terrain.



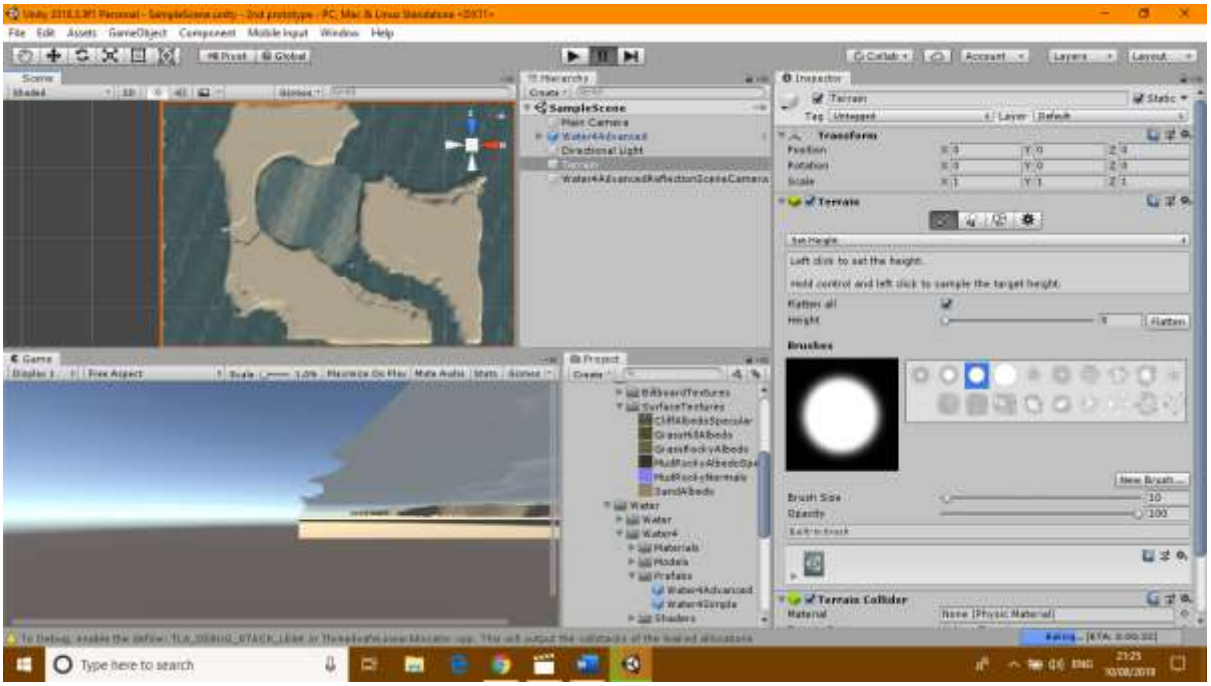
I then textured the terrain from the default assets package in"Standard Assets/Environment/Terrain Assets/SurfaceTextures/SandAlbedo" as well as adding a height of 10 to the terrain



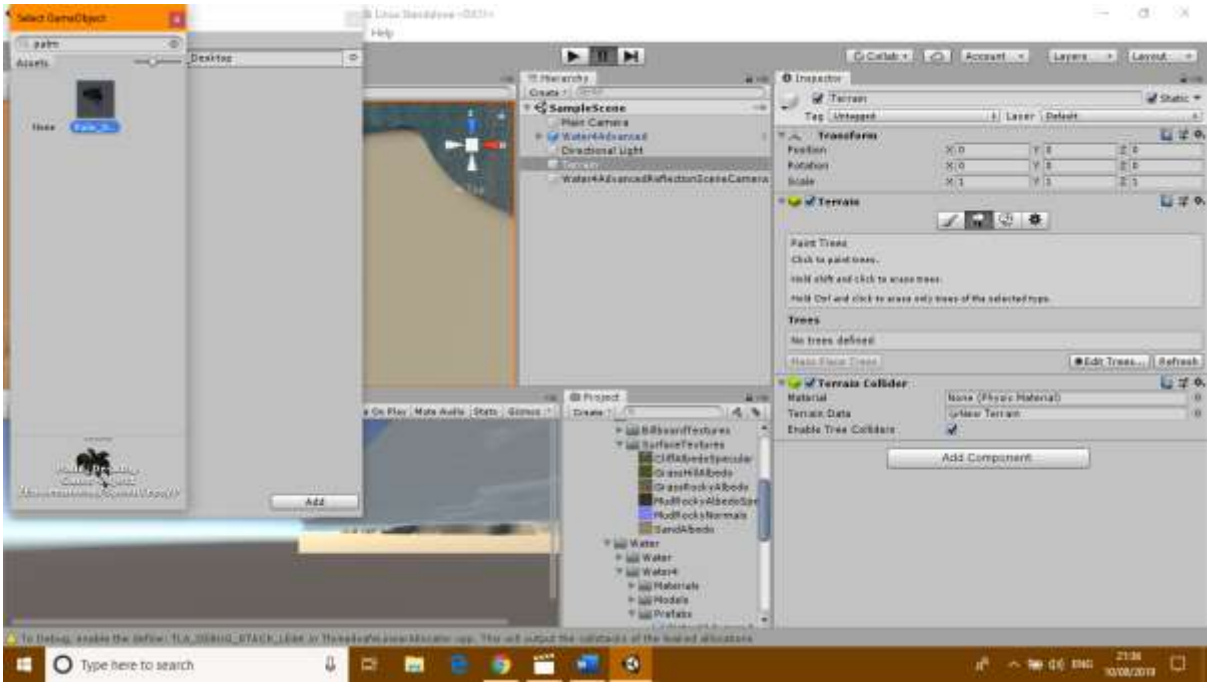
I then set a paint brush size to 100 and opacity to 100 and a height to 6 allowing me to lower the ground making the shape of my island



I added water to the game changed the size to 5,1,5 which I got from the standard assets “Standard Assets/Environment/Terrain Assets/Water/Water4/Prefabs/Wter4Advanced”and positioned it so it was just below the island outline. I proceeded to change the Y of the water as the game was played which effected the wave size and stayed on 3 as to me it showed what the sea would look like on a clear day.

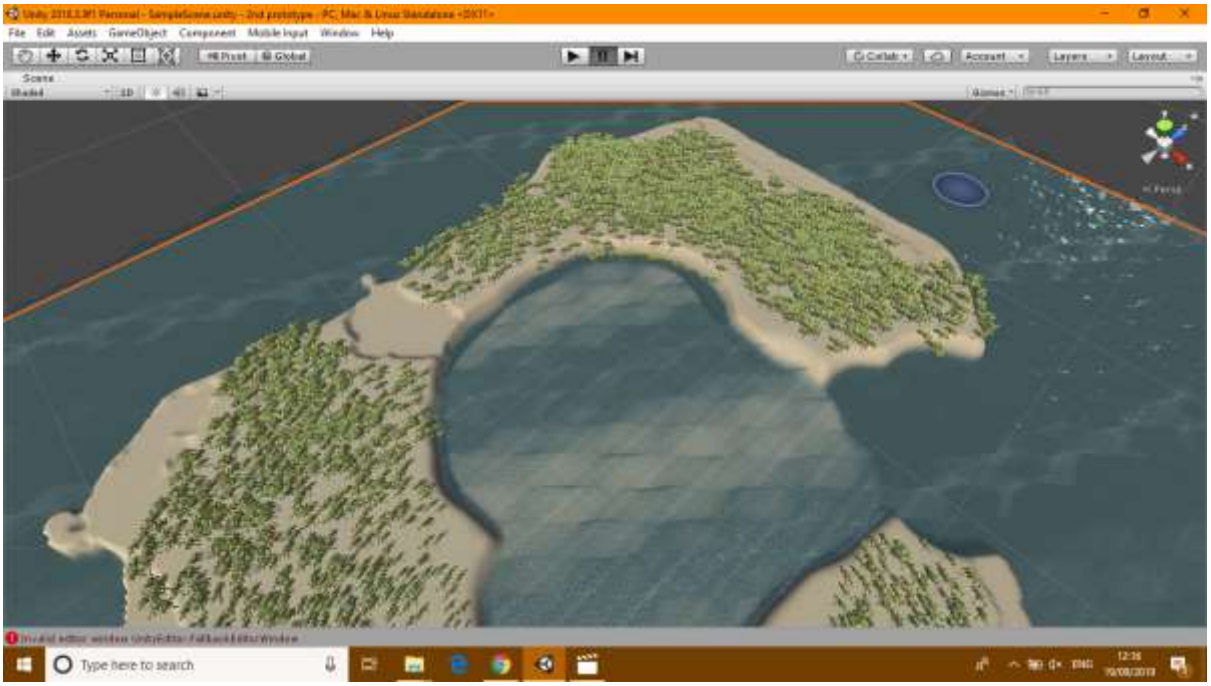


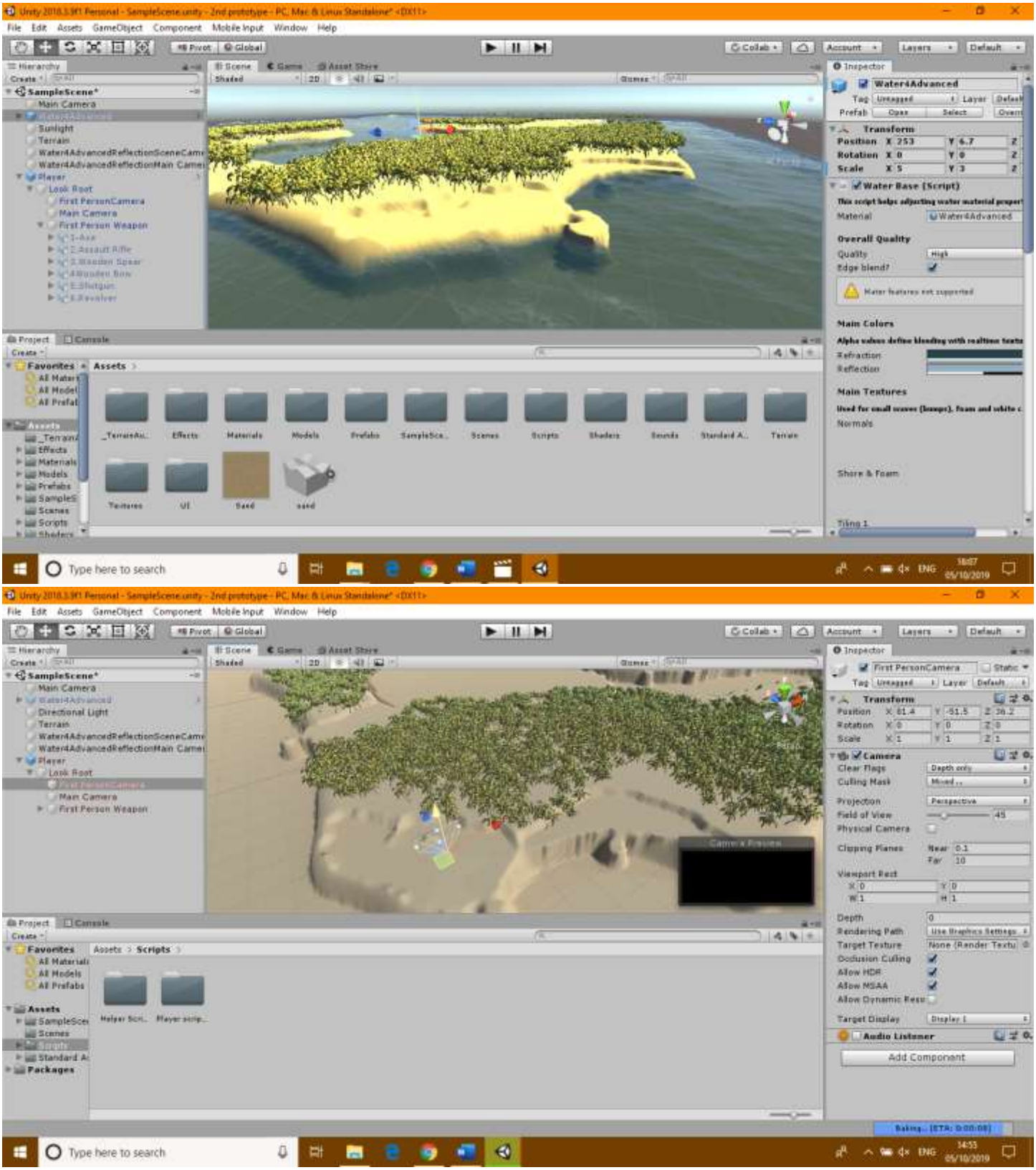
I then started to add beaches around the island by using a brush and lowering the ground. I also added a new texture for the middle of the island to show a rock muddy look



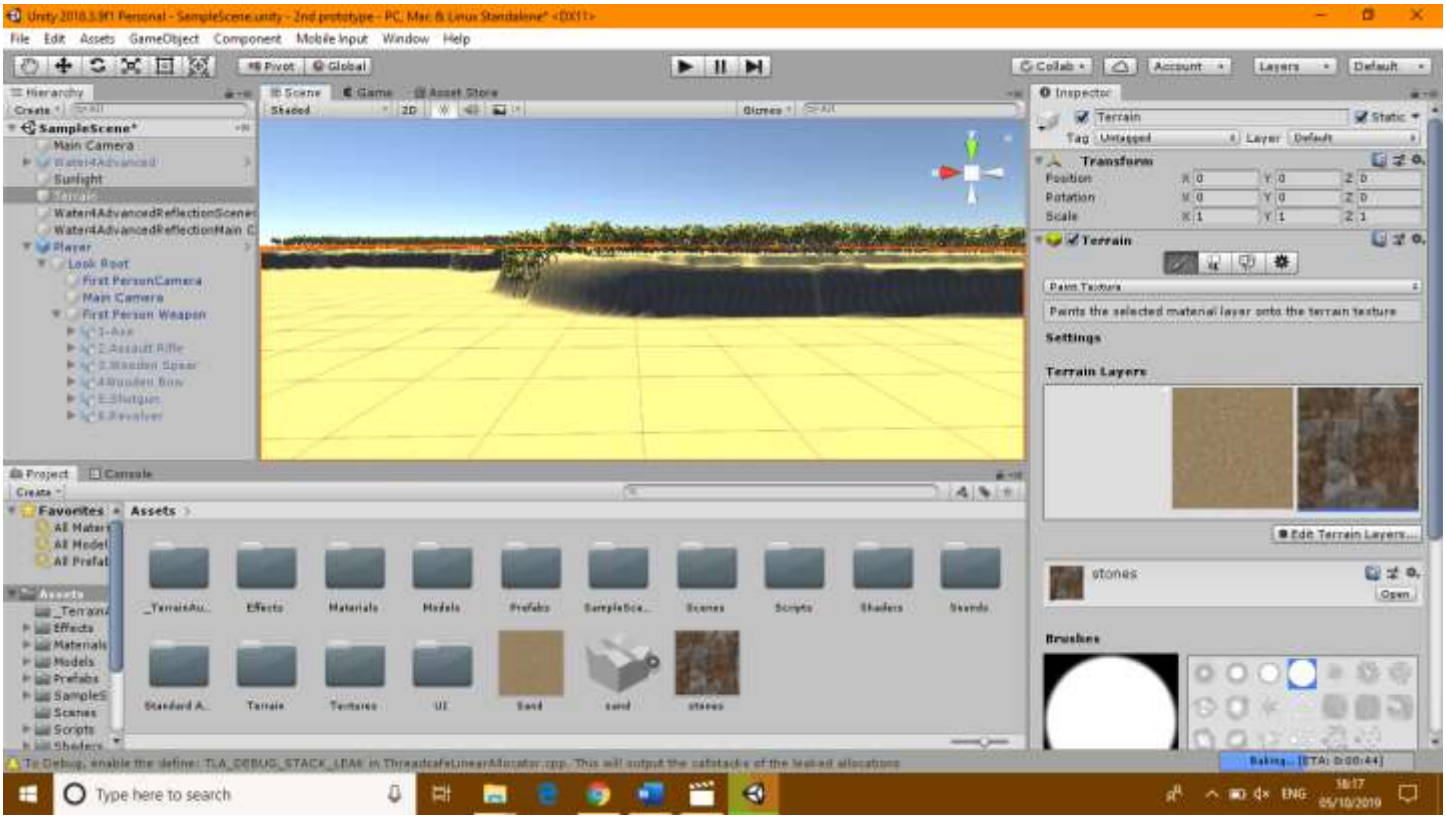
Then grass was added to the island using a brush

FIRST PROTOTYPE -MOVEMENT

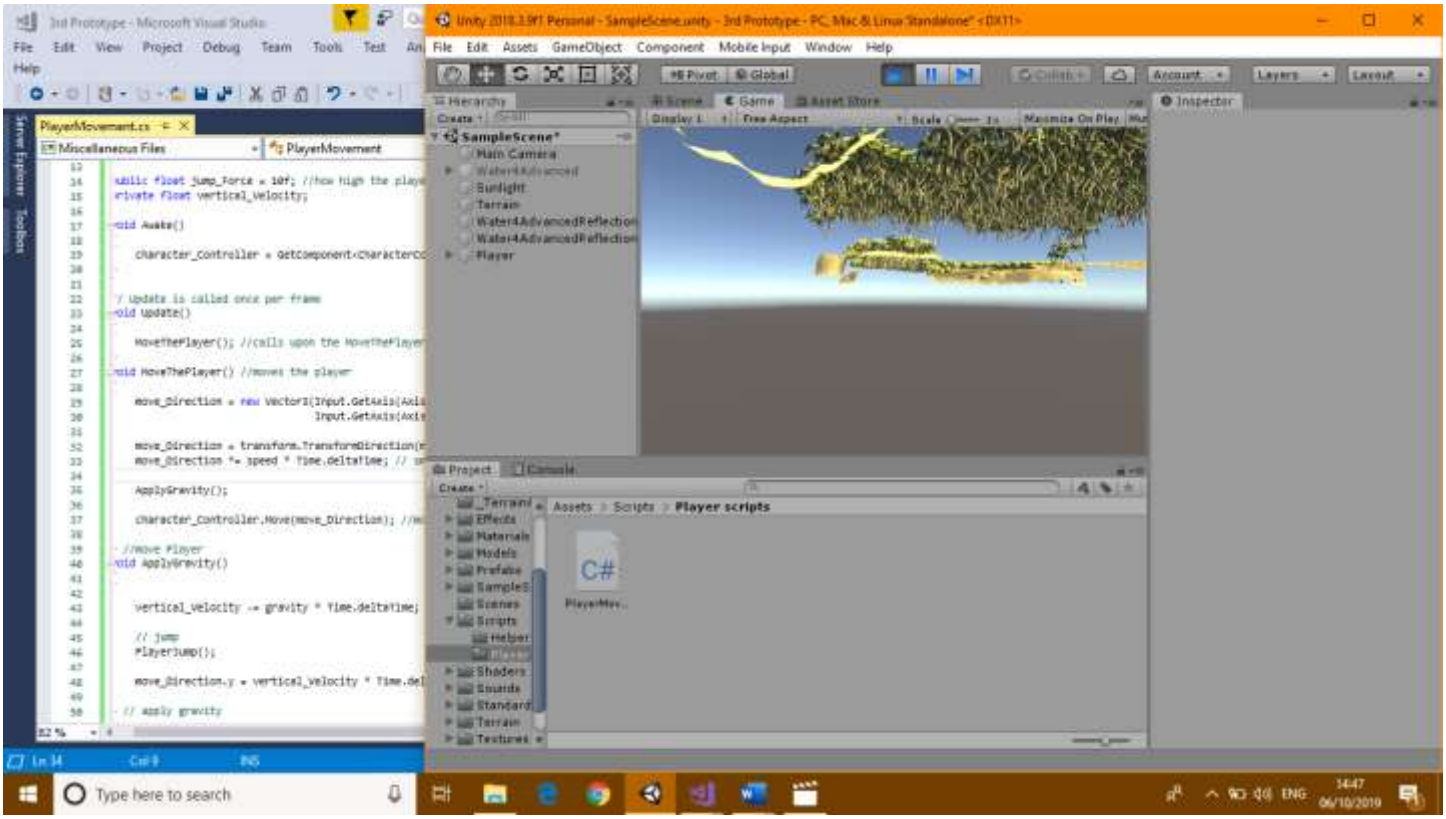

File Edit Assets GameObject Component Mobile Input Window Help
Scene
Game
Type here to search



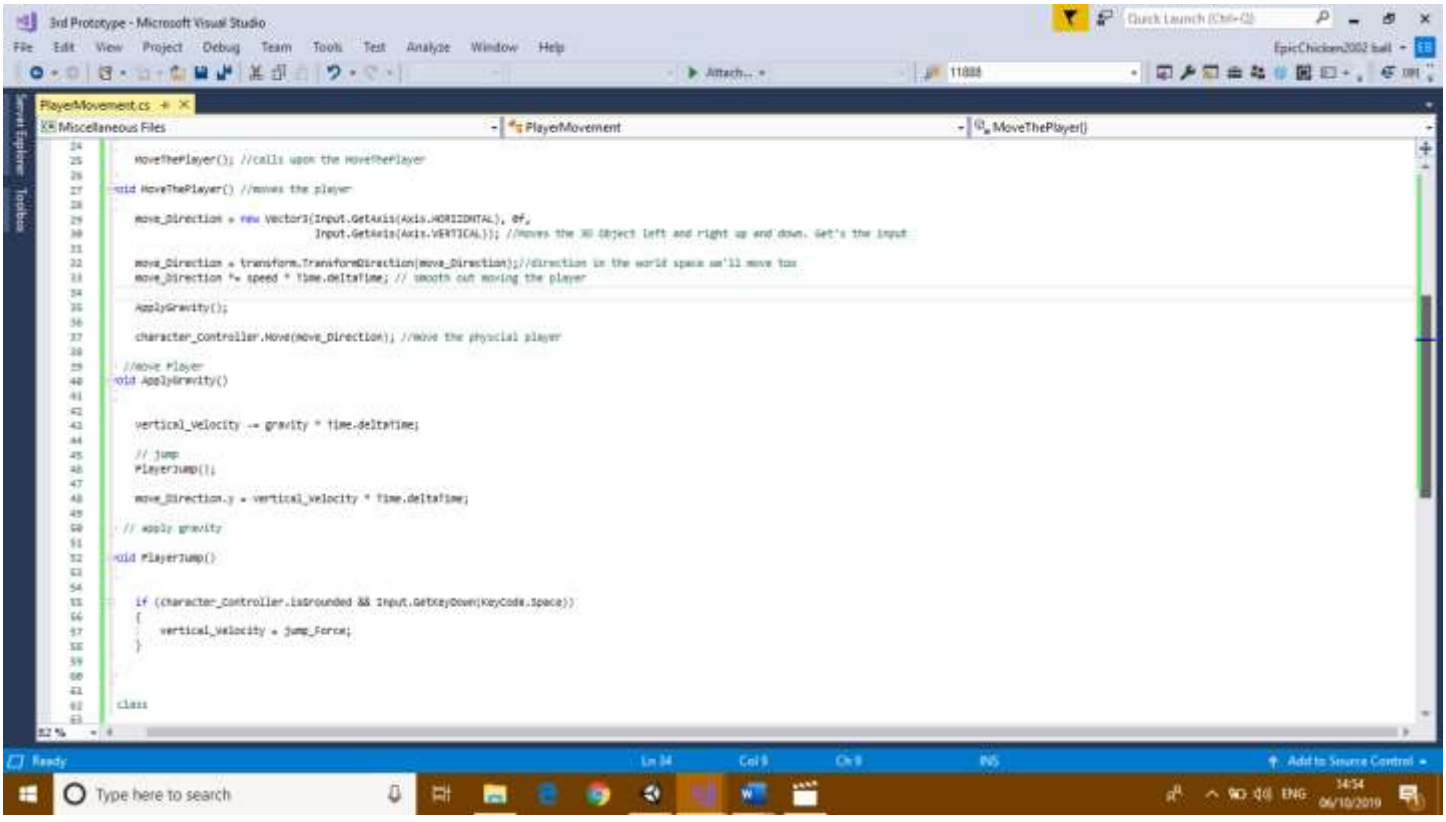
I imported the player script from my first prototype to allow the weapons and movement to be in the game. This required me to reposition the camera weapons and spawn point



This photo shows a stone texture added to the circumference of the island as well as on the floor of the water in the middle of the island. This was added with a paintbrush allowing me to place on specific areas as well as the stone texture being renamed to “stones” in the assets folder.



```
    public float jump_Force = 10f; //how high the player
    private float vertical_velocity;

    void Awake()
    {
        character_controller = GetComponent<CharacterCo
    }

    // Update is called once per frame
    void Update()
    {
        MoveThePlayer(); //calls upon the MoveThePlayer
    }
    void MoveThePlayer() //moves the player
    {
        move_Direction = new Vector3(Input.GetAxis(Axis
                                     Input.GetAxis(Axis
        move_Direction = transform.TransformDirection(m
        move_Direction *= speed * Time.deltaTime; // sm

        ApplyGravity();

        character_controller.Move(move_Direction); //m
    }
    //move Player
    void ApplyGravity()
    {

        vertical_velocity -= gravity * Time.deltaTime;

        // jump
        PlayerJump();

        move_Direction.y = vertical_velocity * Time.del

    // apply gravity
```



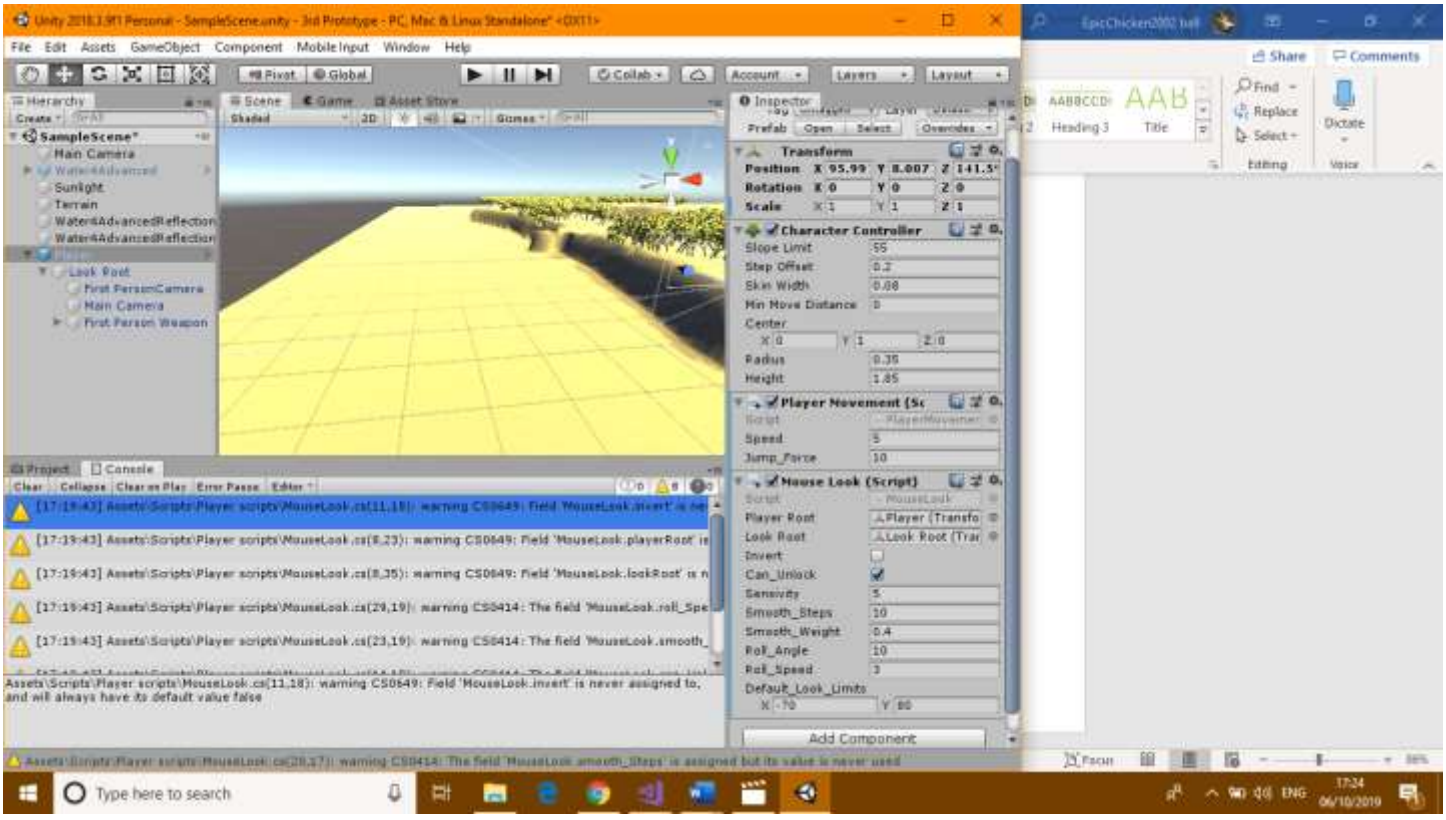
```
    MoveThePlayer(); //calls upon the MoveThePlayer

}
void MoveThePlayer() //moves the player
{
    move_Direction = new Vector3(Input.GetAxis(Axis.HORIZONTAL), 0f,
                                 Input.GetAxis(Axis.VERTICAL)); //Moves the 3D Object left and right up and down. Get's the input

    move_Direction = transform.TransformDirection(move_Direction);//direction in the world space we'll move too
    move_Direction *= speed * Time.deltaTime; // smooth out moving the player

    ApplyGravity();

    character_controller.Move(move_Direction); //move the physcial player
}
//move Player
void ApplyGravity()
{

    vertical_velocity -= gravity * Time.deltaTime;

    // jump
    PlayerJump();

    move_Direction.y = vertical_velocity * Time.deltaTime;

// apply gravity

void PlayerJump()
{

    if (character_controller.isGrounded && Input.GetKeyDown(KeyCode.Space))
    {
        vertical_velocity = jump_Force;
    }


class
```

As soon as the gravity code was added I fell through the floor however the jump function using the space bar worked. This required me to amend the movement script by applying a gravity function which is shown in the picture above. This issue was fixed by adding to the MoveThePlayer function a reference to the apply gravity function (due to it not being called so the gravity script wouldn't work properly).

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerMovement : MonoBehaviour
{
    private CharacterController character_Controller; //Component that allows the player to move in the
way its coded below

    private Vector3 move_Direction; //direction we're moving too

    public float speed = 5f; //controls how fast the player walks
    private float gravity = 20f; //Sets the gravity on the player

    public float jump_Force = 10f; //how high the player can jump
    private float vertical_Velocity;

    void Awake()
    {
        character_Controller = GetComponent<CharacterController>(); //Get's a reference to the
character controller component
    }

    // Update is called once per frame
    void Update()
    {
        MoveThePlayer(); //calls upon the MoveThePlayer
    }
    void MoveThePlayer() //moves the player
    {
        move_Direction = new Vector3(Input.GetAxis(Axis.HORIZONTAL), 0f,
                                     Input.GetAxis(Axis.VERTICAL)); //Moves the 3D Object left and
right up and down. Get's the input

        move_Direction = transform.TransformDirection(move_Direction);//direction in the world space
we'll move too
        move_Direction *= speed * Time.deltaTime; // smooth out moving the player

        ApplyGravity();

        character_Controller.Move(move_Direction); //move the physcial player

    } //move Player
    void ApplyGravity()
    {
        if (character_Controller.isGrounded)
        {
            vertical_Velocity -= gravity * Time.deltaTime;
            // jump
            PlayerJump();
        }
        else
```

```
        {
            vertical_Velocity -= gravity * Time.deltaTime;
        }


        move_Direction.y = vertical_Velocity * Time.deltaTime;
    }
     // apply gravity

    void PlayerJump()
    {

        if (character_Controller.isGrounded && Input.GetKeyDown(KeyCode.Space))
        {
            vertical_Velocity = jump_Force;
        }

    }

} // class
```

This is the final c# script for the movement of the player



I then needed to create a c# script allowing the player to look around and move in eth direction the player is looking. This script included inverted controls/look sensitivity control (which is adjustable) as well as other but they are mainly used to make the game more life like. I also added an escape button press which turns on the cursor when pressed allowing the user to interact with other tabs than just the game, this mean as well that when the game is started the cursor button is made invisible and locked allowing a

full screen prospectus. A few errors occurred when testing this but was all down to capitalised letters needing to be lower case and forgetting of the “f” after a number (0f) with certain commands.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class MouseLook : MonoBehaviour
{
    [SerializeField] //visable in the inspector panel
    private Transform playerRoot, lookRoot;

    [SerializeField] //visable in the inspector panel
    private bool invert;

    [SerializeField] //visable in the inspector panel
    private bool can_Unlock = true; //mouse cursor

    [SerializeField] //visable in the inspector panel
    private float sensivity = 5f; //sensitivity of the mouse

    [SerializeField] //visable in the inspector panel
    private int smooth_Steps = 10;

    [SerializeField] //visable in the inspector panel
    private float smooth_Weight = 0.4f;

    [SerializeField] //visable in the inspector panel
    private float roll_Angle = 10f;

    [SerializeField] //visable in the inspector panel
    private float roll_Speed = 3f;

    [SerializeField] //visable in the inspector panel
    private Vector2 default_Look_Limits = new Vector2(-70f, 80f);

    private Vector2 look_Angles;

    private Vector2 current_Mouse_Look;
    private Vector2 smooth_Move;

    private float current_Roll_Angle;

    private int last_Look_Frame;

    // Start is called before the first frame update
    void Start()
    {
        Cursor.lockState = CursorLockMode.Locked; //locks the cursor to the center of the screen
    }

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

        if (Cursor.lockState == CursorLockMode.Locked)
        {
```

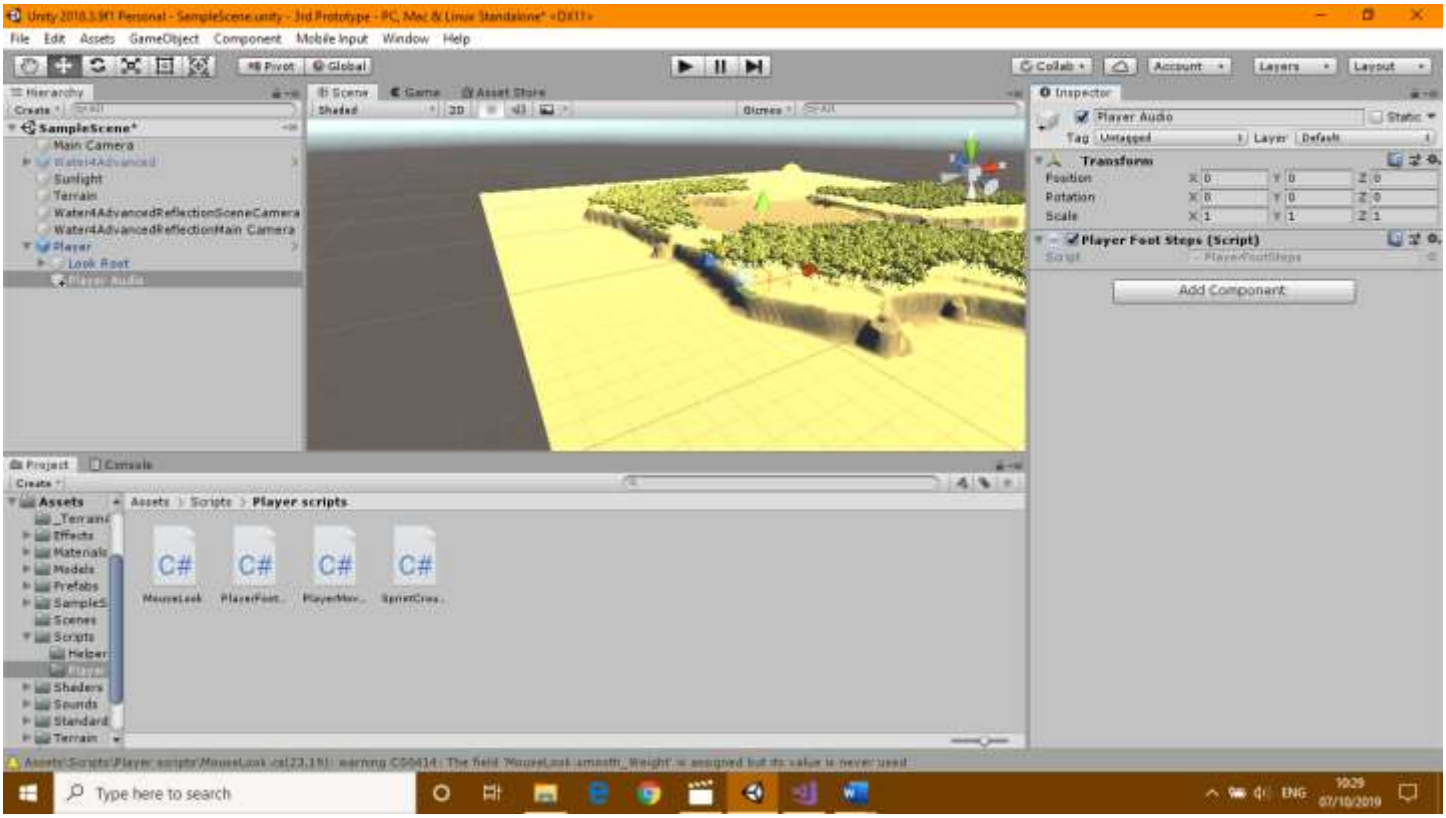
```
            LookAround();
        }
    }
    void LockAndUnlockCursor() //allows the mouse button to be used outside of the game when the scape
button is pressed
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (Cursor.lockState == CursorLockMode.Locked)
            {
                Cursor.lockState = CursorLockMode.None;
            }
            else
            {
                Cursor.lockState = CursorLockMode.Locked;
                Cursor.visible = false;
            }
        }
    }// Lock and unlock
    void LookAround()
    {
        current_Mouse_Look = new Vector2(
            Input.GetAxis(MouseAxis.MOUSE_Y), Input.GetAxis(MouseAxis.MOUSE_X)); //finds the current
postion of the x and y of the mouse

        look_Angles.x += current_Mouse_Look.x * sensivity * (invert ? 1f : -1f); //checks if the invert
is on
        //also calculates and moves the mouse
        look_Angles.y += current_Mouse_Look.y * sensivity;

        look_Angles.x = Mathf.Clamp(look_Angles.x, default_Look_Limits.x, default_Look_Limits.y);
//Limits the look value
        current_Roll_Angle =
            Mathf.Lerp(current_Roll_Angle, Input.GetAxisRaw(MouseAxis.MOUSE_X)
              * roll_Angle, Time.deltaTime * roll_Speed); //smoothes the movement out by going from one
value to the next in a set time

        lookRoot.localRotation = Quaternion.Euler(look_Angles.x, 0f, 0f);
        playerRoot.localRotation = Quaternion.Euler(0f, look_Angles.y, 0f);
    }
}
```

This is the full “MouseLook.cs“ script^

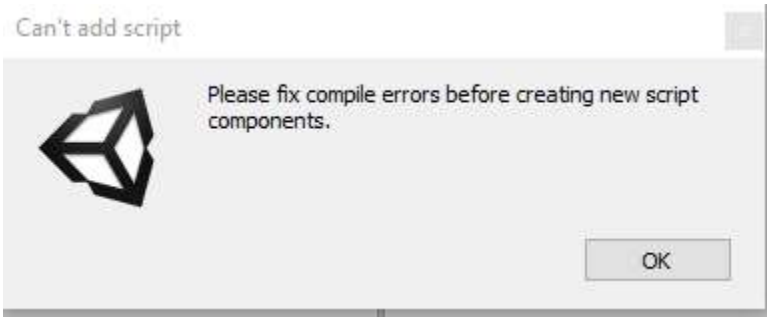
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SprintCrouch : MonoBehaviour
{
    private PlayerMovement playerMovement; //references to the previous made playermovement variable

    public float sprint_Speed = 10f; //speed when sprinting
    public float move_Speed = 5f; //speed when moving(default)
    public float Crouch_Speed = 2f; //speed when crouching
```

```
    private Transform look_Root;
    private float stand_Height = 1.6f; //height when the character is standing up
    private float Crouch_Height = 1f; //height when the character is crouching
    bool is_Crouching;

    // Start is called before the first frame update
    void Awake()
    {
        playerMovement = GetComponent<PlayerMovement>(); //open'sthe playermovement component

        look_Root = transform.GetChild(0); //get's the first child in the heirachy

    }

    // Update is called once per frame
    void Update()
    {
        Sprint(); //refrences to Sprint function
        Crouch(); //refrences to Crouch function
    }
    void Sprint()
    {
        if (Input.GetKeyDown(KeyCode.LeftShift) && !is_Crouching) //checks if the left shift button is
pressed and is not crouching
        {
            playerMovement.speed = sprint_Speed; //goes to the sprint_Speed function
        }
        if (Input.GetKeyUp(KeyCode.LeftShift) && !is_Crouching) //checks if the left shift button is
pressed and is not crouching
        {
            playerMovement.speed = move_Speed; //goes to the move_Speed function
        }
    }// sprint
    void Crouch()
    {
        if (Input.GetKeyDown(KeyCode.C))
        {
            if (is_Crouching)//if we are crouching - stand up
            {
                look_Root.localPosition = new Vector3(0f, stand_Height, 0f);//changes the y position to
the normal height
                playerMovement.speed = move_Speed;  // makes the speed eback to the normal speed

                is_Crouching = false; // changes the variable to false
            }
            else //if we are not crouching - crouch
            {
                look_Root.localPosition = new Vector3(0f, Crouch_Height, 0f); //changes the y position
to the crouching height
                playerMovement.speed = Crouch_Speed; // makes the speed equal to the crouching speed

                is_Crouching = true; // changes the variable to true
            }
        } // if we press c

    }
}
```

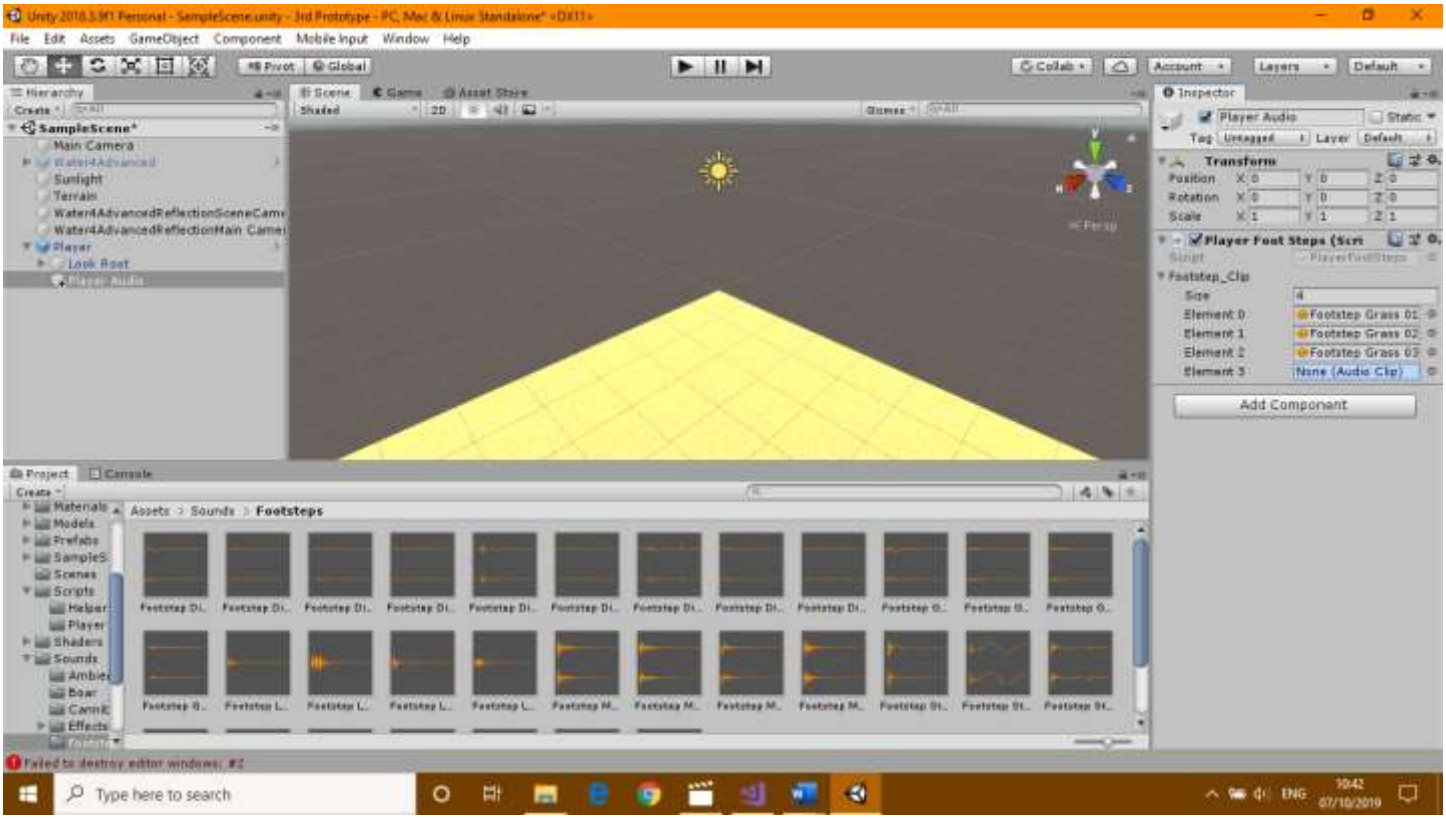
This is my “SprintCrouch.cs” script this allows me to sprint by holding shift and crouch by pressing c. I have made it so the crouch speed is different to the normal speed and that you can’t run while in the crouch position. When pressing C the Z co-ordinate changes making it seem like the character is actually crouching. Once finished crouching you can change back to normal height by pressing “c” again which goes back to the default controls



FIRST PROTOTYPE -SOUND EFFECTS



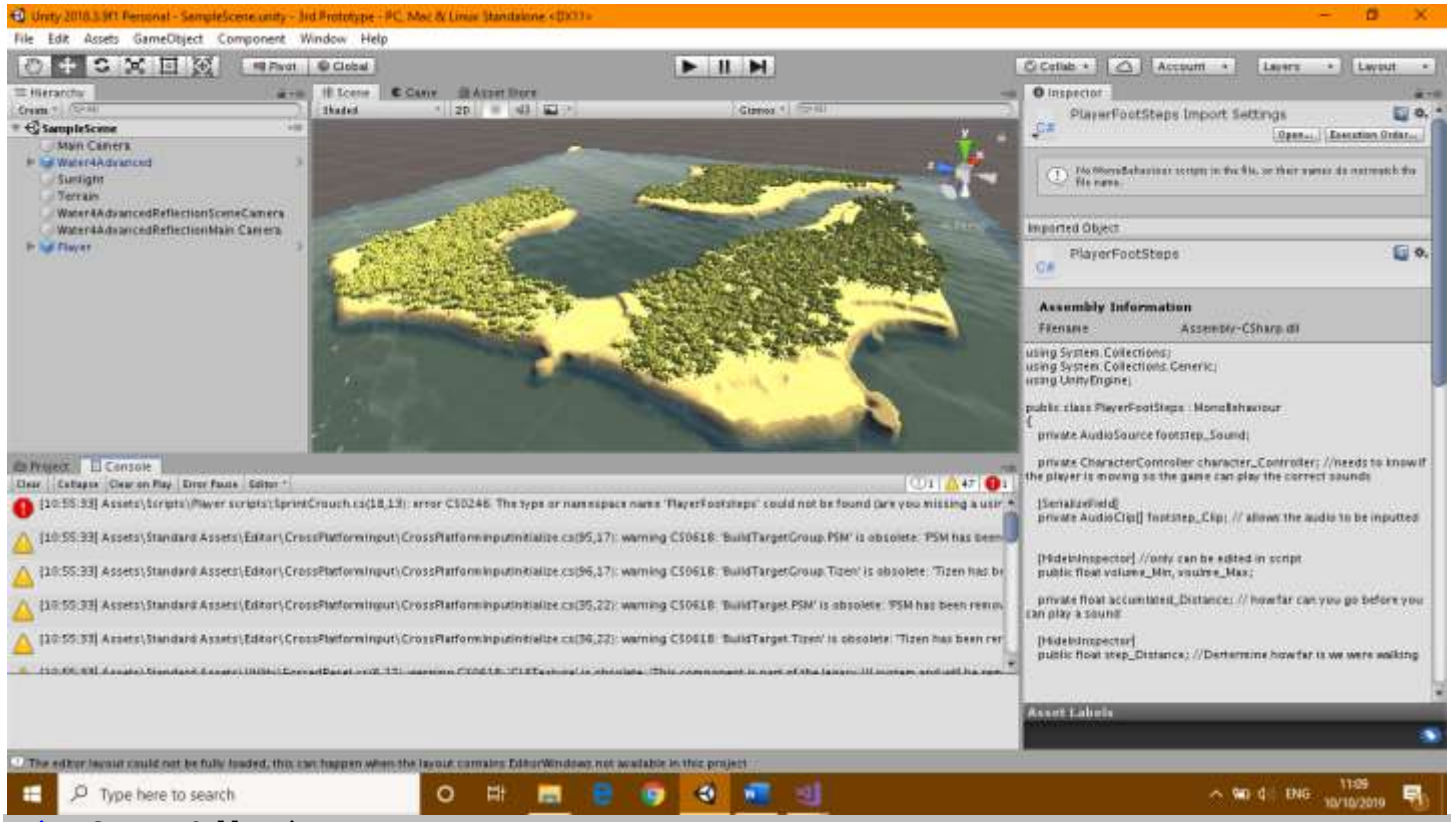
Here I am adding player footstep noises making the game more realistic. This required me to make another C# script called “PlayerFootSteps.cs” drag it into the new object I created called player audio. I initially got an error due to the name of the C# having a space in it but after deleting it and renaming it without any spaces the error disappeared.



This was first done by editing the script by putting "[SerializeField] private AudioClip[] footstep_Clip;" in the "PlayerFootSteps.cs" script this allowed me to change the "footstep clipsize" to 4 in unity giving me 4 different sounds for footsteps. I then went into the "Assets/SoundsFootsteps" folder to retrieve the grass footstep sounds and dragged 0-1 into the allocated slots.

```
private PlayerFootsteps player_Footsteps;

    private float sprint_Volume = 1f; //sprint volume
    private float crouch_Volume = 0.1f; //Crouch volume
    private float walk_Volume_Min = 0.2,walk_Volume_Max = 0.6f; //minimum and max volume

    // Start is called before the first frame update
    void Awake()
    {
         playerMovement = GetComponent<PlayerMovement>(); //open's the playermovement component

        look_Root = transform.GetChild(0); //get's the first child in the heirachy

        player_Footsteps = GetComponentInChildren<PlayerFootsteps>(); //Get's the playerfootstepshe
child component loaded up
    }
```

Above is the code I amended in the player "SprintCrouch.cs" script allowing the volume to be used when sprinting and crouching



```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerFootSteps : MonoBehaviour
{
    private AudioSource footstep_Sound;

    private CharacterController character_Controller; //needs to know if the player is moving so the
game can play the correct sounds

    [SerializeField]
    private AudioClip[] footstep_Clip; // allows the audio to be inputted


    [HideInInspector] //only can be edited in script
    public float volume_Min, voulme_Max;

    private float accumlated_Distance; // how far can you go before you can play a sound

    [HideInInspector]
    public float step_Distance; //Dertermine how far is we were walking



    // Start is called before the first frame update
    void Awake()
    {
        footstep_Sound = GetComponent<AudioSource>(); //open'sthe playermovement component
```

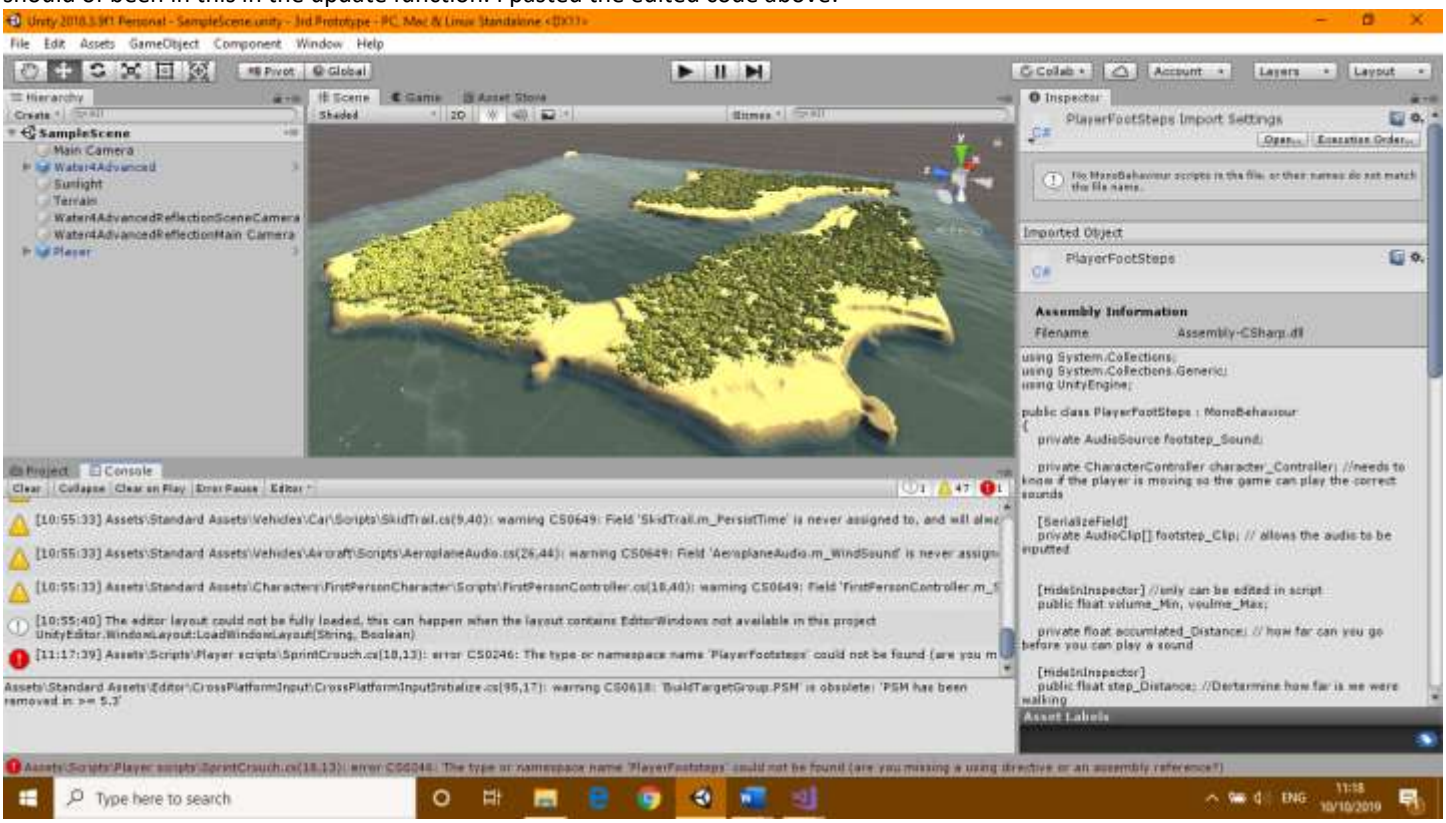
```
        character_Controller = GetComponentInParent<CharacterController>(); //get's the first child in
the heirachy
    }

    // Update is called once per frame
    void Update()
    {
        CheckToPlayFootStepSound();
        if (!character_Controller.isGrounded) //the chcaracter is not on the ground
            return; //exist's the function
        if (character_Controller.velocity.sqrMagnitude > 0) //check's if the player is moving
        {
            accumlated_Distance += Time.deltaTime; //Accumlated distance is how far the user can go
(the distnace of a step before a noise is played)
            if (accumulated_Distance > step_Distance)
            {
                footstep_Sound.volume = Random.Range(volume_Min, volume_Max); //limits the volume of
the sound effects
                footstep_Sound.clip = foot_Clip[Random.Range(0, footstep_Clip.Length)]; //gets a number
between 0 and (footstep_Clip.Length -1)
                footstep_Sound.Play(); //play's the sound in the libary

                accumulated_Distance = 0f; //makes the default "accumlated distance" equal to 0

                Random.Range(1f, 3f); //returns a value between 1 and 2
            }
        }
        else
        {
            accumlated_Distance = 0f; //makes the default "accumlated distance" equal to 0
        }

    }
}
```

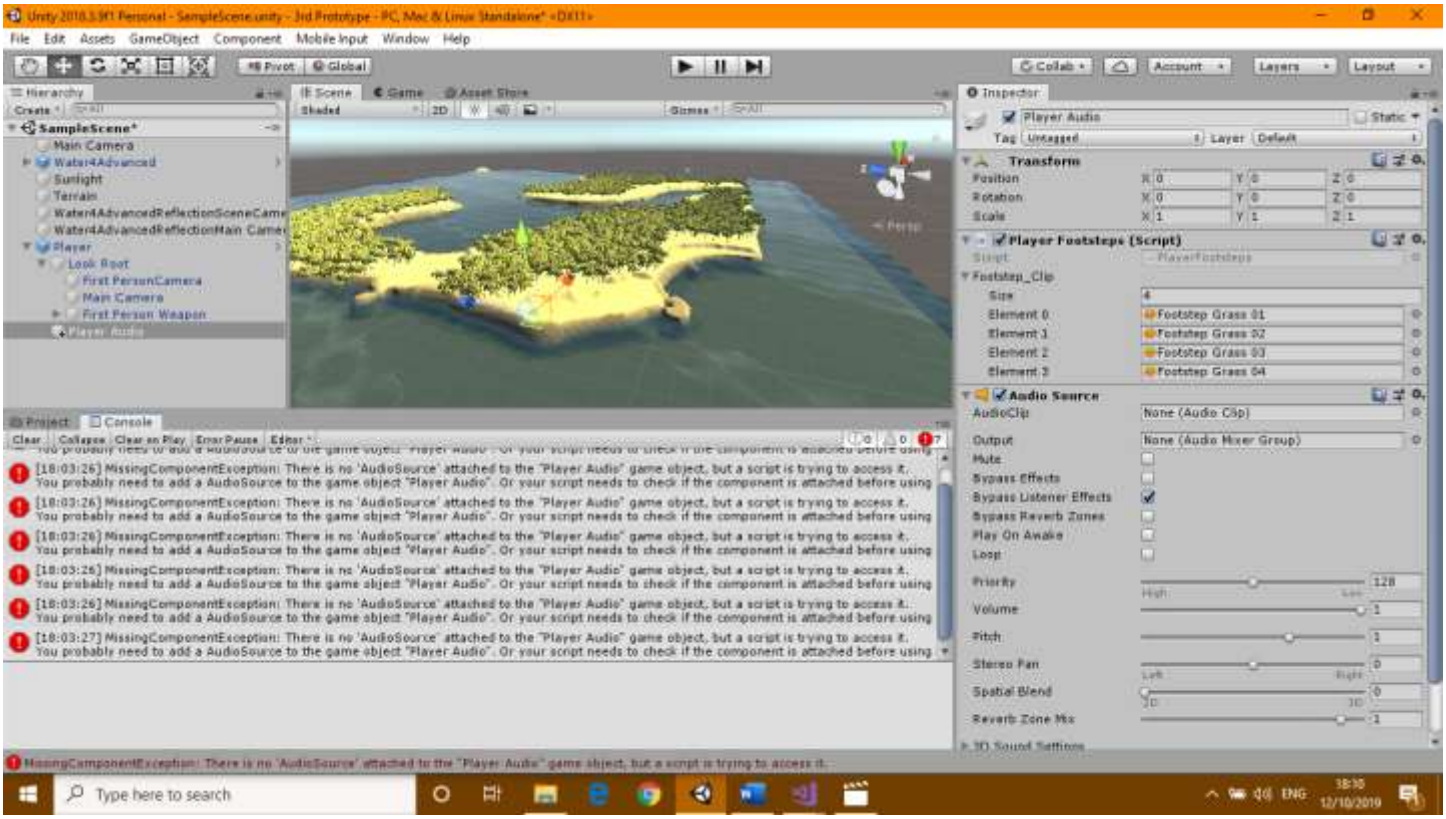
Once I created my “footsteps.cs” script an error occurred “Assets\Scripts\Player scripts\SprintCrouch.cs(18,13): error CS0246: The type or namespace name 'PlayerFootsteps' could not be found (are you missing a using directive or an assembly reference?)”

```
void Update()
    {
        CheckToPlayFootStepSound();
    }
    void CheckToPlayFootStepSound() {
        if (!character_Controller.isGrounded) //the chcaracter is not on the ground
            return; //exist's the function
        if (character_Controller.velocity.sqrMagnitude > 0) //check's if the player is moving
        {
            accumlated_Distance += Time.deltaTime; //Accumlated distance is how far the user can go
(the distnace of a step before a noise is played)
            if (accumulated_Distance > step_Distance)
            {
                footstep_Sound.volume = Random.Range(volume_Min, volume_Max); //limits the volume of
the sound effects
                footstep_Sound.clip = foot_Clip[Random.Range(0, footstep_Clip.Length)]; //gets a number
between 0 and (footstep_Clip.Length -1)
                footstep_Sound.Play(); //play's the sound in the libary

                accumulated_Distance = 0f; //makes the default "accumlated distance" equal to 0

                Random.Range(1f, 3f); //returns a value between 1 and 2
```

```
            }
        }
        else
        {
            accumlated_Distance = 0f; //makes the default "accumlated distance" equal to 0
        }

    }
```

After relooking at my code I realised I haven't made the void function for "CheckToPlayFootsepSound.cs" I put all the code that should of been in this in the update function. I pasted the edited code above.





I came across the error shown in the console in the picture above to I then realised I forgot to place the “audioscource” component in the player audio section and untick everything which is set ticked as default.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class SprintCrouch : MonoBehaviour
{
    private PlayerMovement playerMovement;

    public float sprint_Speed = 10f; //speed when sprinting
    public float move_Speed = 5f; //speed when moving(default)
    public float crouch_Speed = 2f; //speed when crouching

    private Transform look_Root;
    private float stand_Height = 1.6f; //height when the character is standing up
    private float crouch_Height = 1f; //height when the character is crouching

    private bool is_Crouching; //makes a decision if the player is crouching or not

    private PlayerFootsteps player_Footsteps;
```

```
    private float sprint_Volume = 1f; //sprint volume
    private float crouch_Volume = 0.1f; //Crouch volume
    private float walk_Volume_Min = 0.2f, walk_Volume_Max = 0.6f; //minimum and max walking volume

    private float walk_Step_Distance = 0.4f; //walk step distance
    private float sprint_Step_Distance = 0.25f; //sprint step distance
    private float crouch_Step_Distance = 0.5f; //sprint crouch distance
    // Start is called before the first frame update

    void Awake()
    {

        playerMovement = GetComponent<PlayerMovement>(); //open's the playermovement component

        look_Root = transform.GetChild(0); //get's the first child in the heirachy

        player_Footsteps = GetComponentInChildren<PlayerFootsteps>(); //Get's the playerfootstepshe
child component loaded up



    }
    void Start()
    {
        player_Footsteps.volume_Min = walk_Volume_Min;
        player_Footsteps.volume_Max = walk_Volume_Max;
        player_Footsteps.step_Distance = walk_Step_Distance;
    }

    // Update is called once per frame
    void Update()
    {
        Sprint(); //refrences to Sprint function
        Crouch(); //refrences to Crouch function
    }

    void Sprint()
    {

        // if we have stamina we can sprint
        if (Input.GetKeyDown(KeyCode.LeftShift) && !is_Crouching) //checks if the left shift button is
pressed and is not crouching
```

```
        {

            playerMovement.speed = sprint_Speed; //variable that goes to the sprint_Speed function

            player_Footsteps.step_Distance = sprint_Step_Distance; //variable that goes to the
sprint_Step_Distance function
            player_Footsteps.volume_Min = sprint_Volume;  //more realistic due to making louder noises
as the player steps forward-at it's minium
            player_Footsteps.volume_Max = sprint_Volume; //more realistic due to making louder noises
as the player steps forward - at it's maxium

        }

        if (Input.GetKeyUp(KeyCode.LeftShift) && !is_Crouching)
        {

            playerMovement.speed = sprint_Speed; //variable that goes to the sprint_Speed function

            player_Footsteps.step_Distance = walk_Step_Distance; //variable that goes to the
sprint_Step_Distance function
            player_Footsteps.volume_Min = walk_Volume_Min;  //more realistic due to making louder
noises as the player steps forward-at it's minium
            player_Footsteps.volume_Max = walk_Volume_Max; //more realistic due to making louder noises
as the player steps forward - at it's maxium
        }




    } // sprint

    void Crouch()
    {

        if (Input.GetKeyDown(KeyCode.C)) //Checks; oif the "c" key is pressed
        {
            if (is_Crouching) //if we are crouching - stand up
            {

                look_Root.localPosition = new Vector3(0f, stand_Height, 0f); //changes the y position
to the normal height
                playerMovement.speed = move_Speed; // makes the speed back to the normal speed
```

```
                player_Footsteps.step_Distance = walk_Step_Distance; //variable that goes to the
sprint_Step_Distance function
                player_Footsteps.volume_Min = walk_Volume_Min;  //more realistic due to making louder
noises as the player steps forward-at it's minium
                player_Footsteps.volume_Max = walk_Volume_Max; //more realistic due to making louder
noises as the player steps forward - at it's maxium


                is_Crouching = false; //changes the "is_Crouching" variable to false
            }
            else
            {
                // if we are not crouching - crouch


                look_Root.localPosition = new Vector3(0f, crouch_Height, 0f); //if we are not crouching
- crouch
                playerMovement.speed = crouch_Speed; // makes the speed equal to the crouching speed


                player_Footsteps.step_Distance = crouch_Step_Distance; //variable that goes to the
crouch_Step_Distance function
                player_Footsteps.volume_Min = crouch_Volume; //more realistic due to making louder
noises as the player steps forward-at it's minium
                player_Footsteps.volume_Max = crouch_Volume; //more realistic due to making louder
noises as the player steps forward - at it's maxium


                is_Crouching = true; //changes the "is_Crouching" variable to true
            }



        } // if we press c



    } // crouch


} // class
```

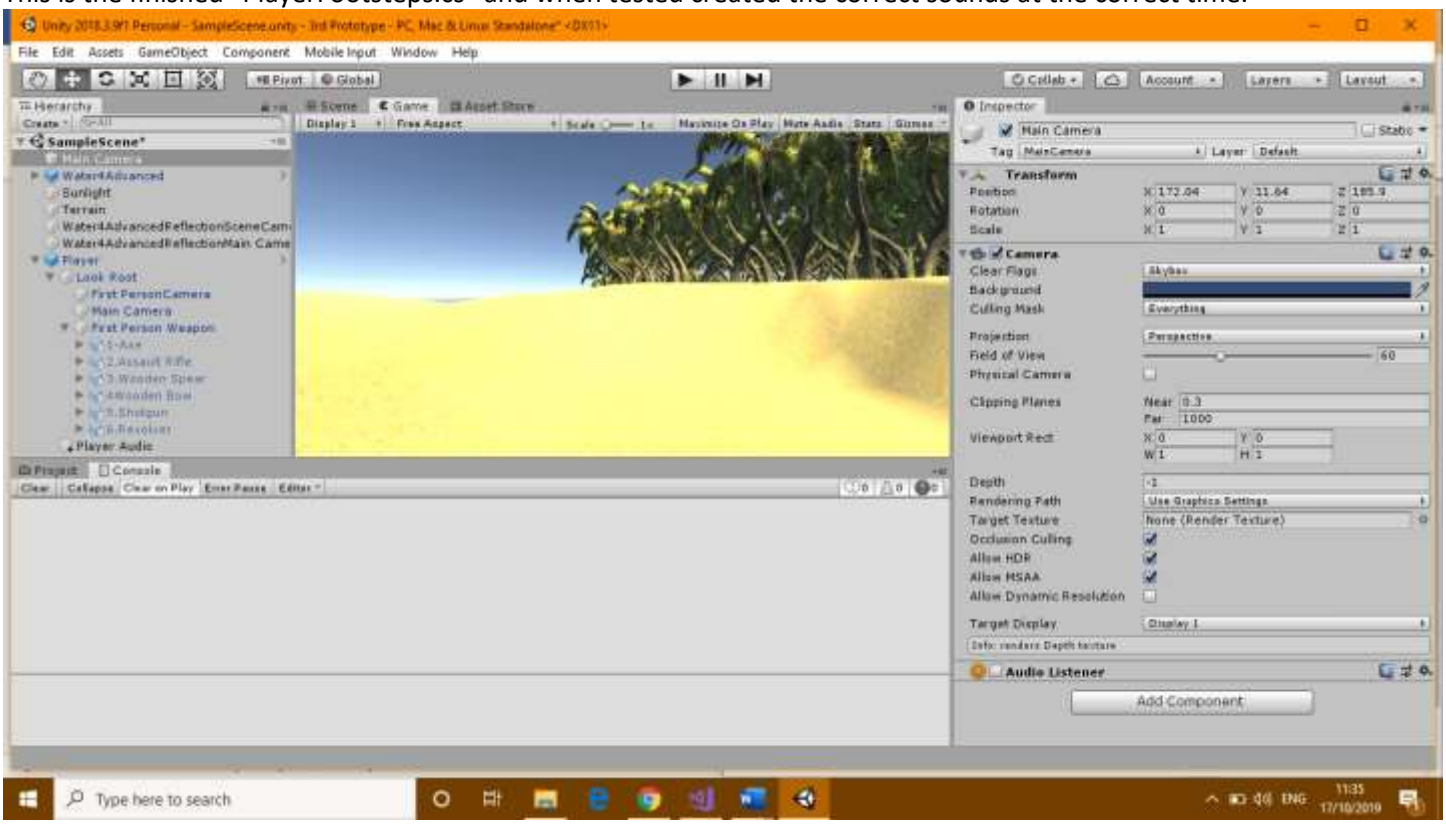
The code above is the “SprintCrouch.cs” script modified to allow sound affects to be used

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class PlayerFootsteps : MonoBehaviour
{
    private AudioSource footstep_Sound; //Creates a variable for the footstep_function

    [SerializeField] //can be edited in unity
```

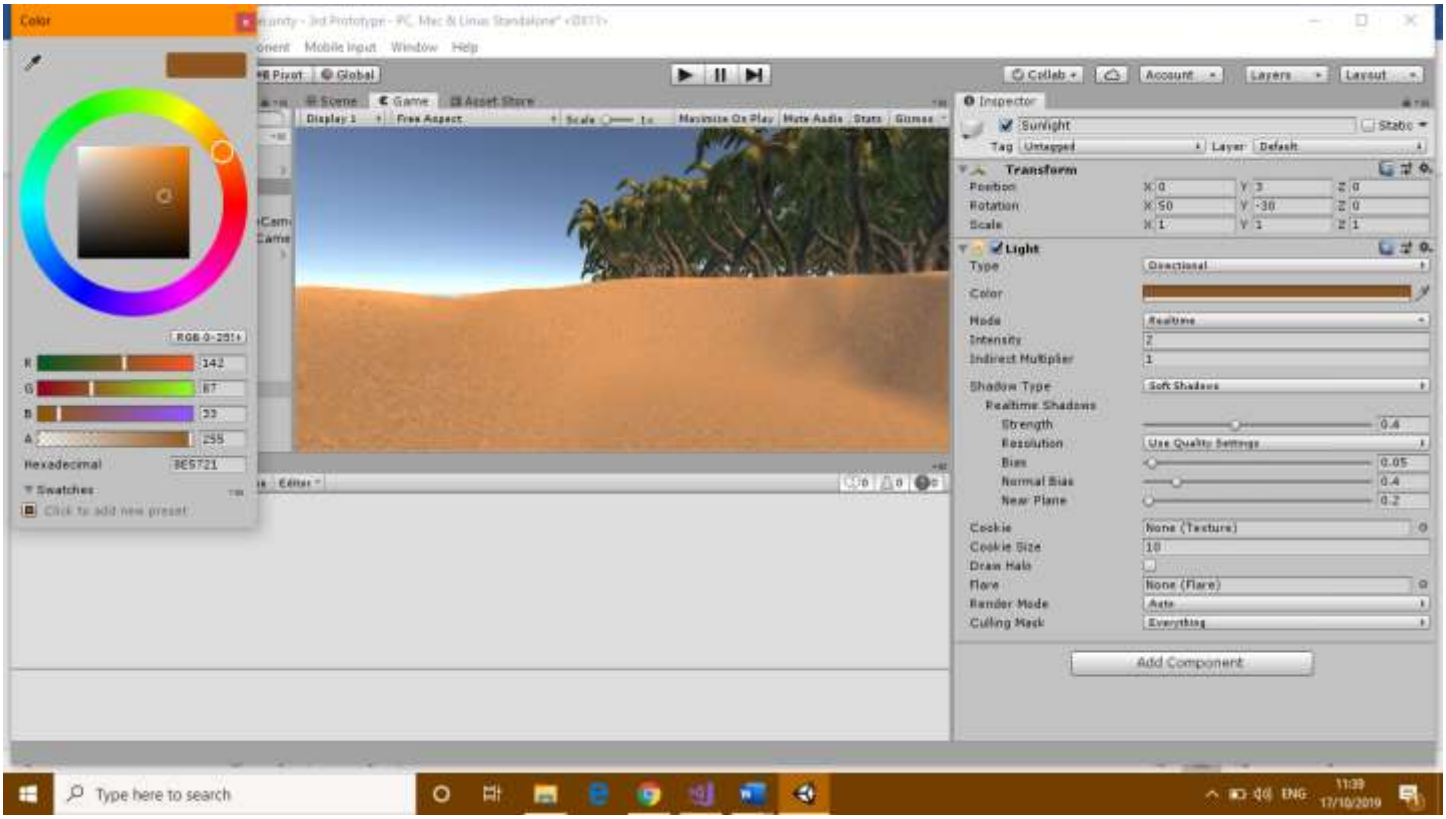
```
    private AudioClip[] footstep_Clip; // allows the audio to be inputted

    private CharacterController character_Controller; //references the character controller

    [HideInInspector] //only can be edited in script
    public float volume_Min, volume_Max; //Creates a variable for the maxium and minium volume

    private float accumulated_Distance; // how far can you go before you can play a sound

    [HideInInspector] //only can be edited in script
    public float step_Distance; //creates a variable for the step_Distance

    // Start is called before the first frame update
    void Awake()
    {
        footstep_Sound = GetComponent<AudioSource>(); //open's the footstep_Sound component

        character_Controller = GetComponentInParent<CharacterController>();//get's the first child in
the heirachy character controller component
    }

    // Update is called once per frame
    void Update()
    {
        CheckToPlayFootstepSound(); //Calls the "CheckToPlayFootStepSound"
    }

    void CheckToPlayFootstepSound() //needs to know if the player is moving so the game can play the
sound effect
    {

        // if we are NOT on the ground
        if (!character_Controller.isGrounded)
            return; //exist the function calling back no variables


        if (character_Controller.velocity.sqrMagnitude > 0) //check's if the player is moving
        {
            accumulated_Distance += Time.deltaTime; //Accumlated distance is how far the user can go
(the distance of a step before a noise is played)

            if (accumulated_Distance > step_Distance)
            {

                footstep_Sound.volume = Random.Range(volume_Min, volume_Max); //limits the volume of
the sound effects
                footstep_Sound.clip = footstep_Clip[Random.Range(0, footstep_Clip.Length)];
                footstep_Sound.Play(); //plays the sound on the computer

                accumulated_Distance = 0f; //makes the "accumulated_Distance" back to the default value
of 0

            }

        }
        else
        {
```

```
            accumulated_Distance = 0f; //makes the "accumulated_Distance" back to the default value of
0
        }


    }

} // class
```

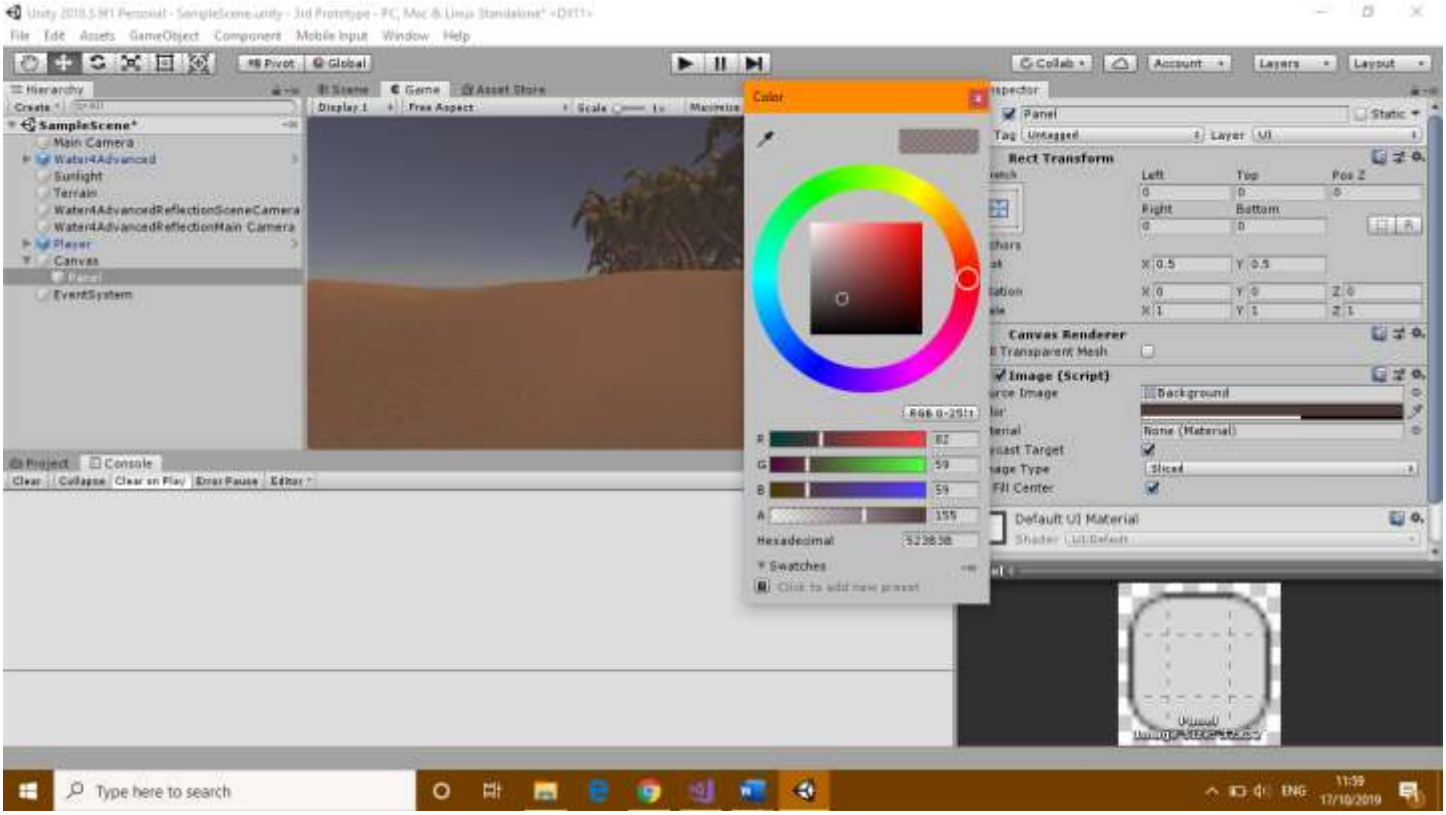
This is the finished “PlayerFootsteps.cs” and when tested created the correct sounds at the correct time.



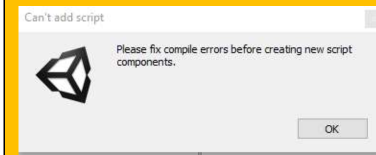
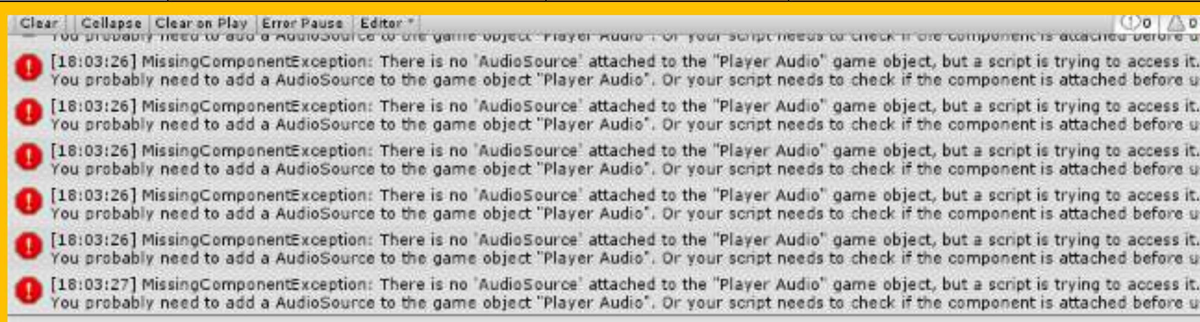
I then saw in the console that 2 “Audio listeners” were active when only one is needed. I then found out I hadn’t unticked the main camera one which then removed the error.



I didn’t like the sand being a very vibrant yellow so after relooking at the colour schemes for the sunlight I found a more realistic colour scheme the hex value for it is “8E5721”.



First Protoype Testing Table

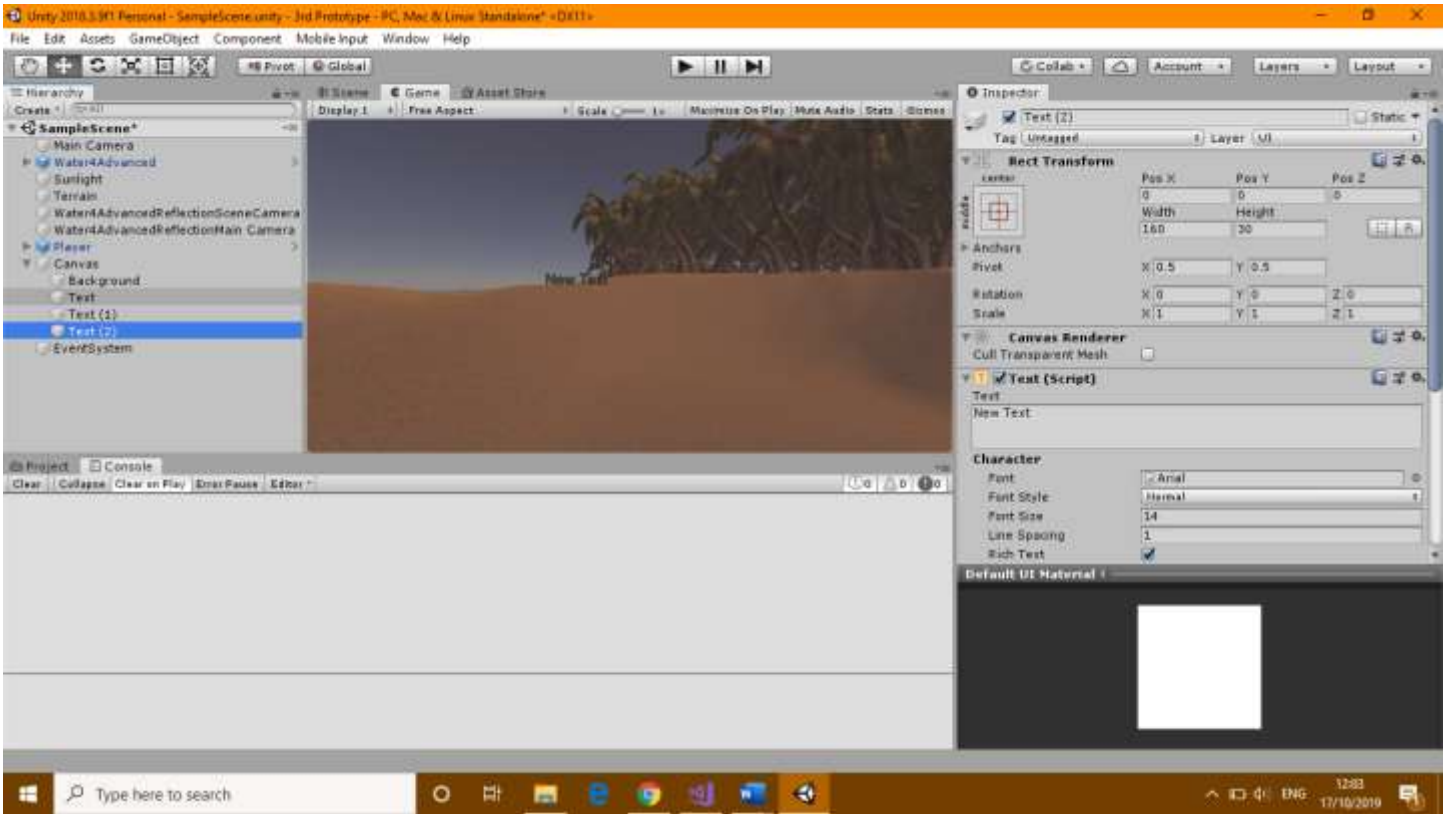
11b.		Player Foot Steps.cs()- Sound effects for the player moevment	1-Can't add script error 2 Remaining error 	Remove the spacing in the text of the files name	Yes- due to the name of the C# havin space in it but after deleting it and renaming it without any spaces the e disappeared. 12b for the next test
12b.		Player Foot Steps.cs()- Sound effects for the player moevment	1-"Assets\Scripts\Player scripts\SprintCrouch.cs(18,13): error CS0246: The type or namespace name 'PlayerFootsteps' could not be found (are you missing a using directive or an assembly reference?)" 1 remaining error	Need to relook over my full code step by step due not having an idea what this could be	Yes-After relooking at my code I reali haven't made the void function for "CheckToPlayFootsepSound.cs" I put the code that should of been in this i update function. This therefore fixed error. 13b for the next test
13b.		Audio Source	1-"There is no audiosource attached to the player Audio" See below:	I needed to place the "audioscource" component in the player audio section and untick everything which is set ticked as default.	Yes-I replaced the component which set to a default as empty with the co audio source components and the er was removed.14b for the next test
					
14b.		Audio Source	Yes	N/A	N/A
15b.		Check for any warning error's	1-in the console that two "Audio listeners" were active when only one is needed	Find both ticked boxes and de-tick the necessary one	Yes- I found out I hadn't unticked the camera one which then removed the error. 16b for the next test
16b.		Audio Source	Yes	N/A	N/A
22a.-Map		Water physics	Partially	-next test	
23a.-Sprint		Decrease the speed of the player. After	Yes	No	

		1A user's feedback			
24a.-Yield Of trees		Created paths in the island and made the yield of trees a lot smaller therefore not as dense. After 1A user's feedback	Yes	No	

USER'S FEEDBACK

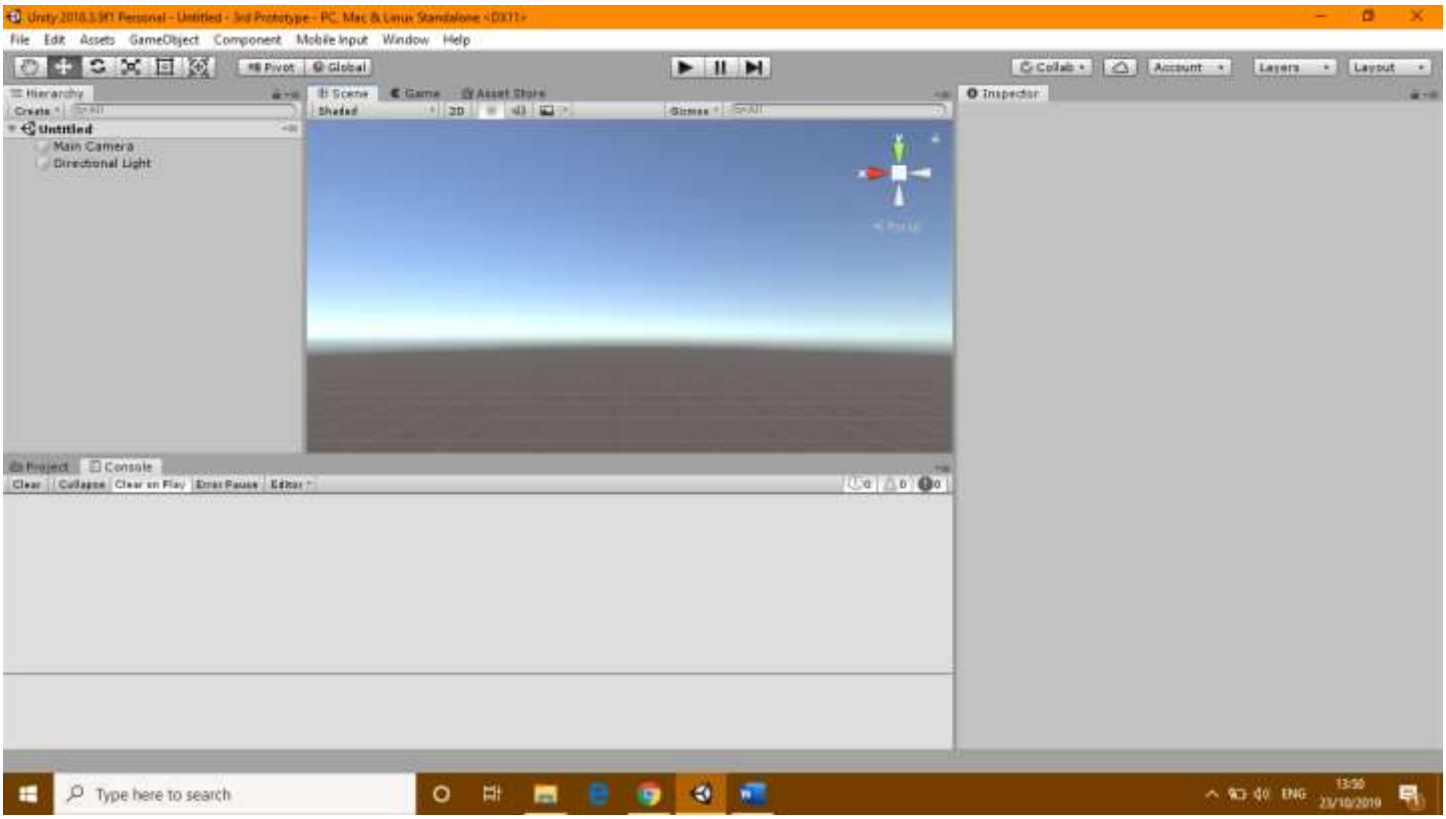
Luke Callison-Bailey- After my play test of the game, I have found some things that could do with improvements and some things that I like. Firstly, I think there are too many trees and it is causing lag and causing the player to get stuck in between the trees. Also, the player can jump up on the trees, which I don't think is intended. I think the best way of fixing this is to increase the size of trees but reduce the amount and space them out more. Also, when I walked into the water, there was no ability to swim, but I believe this will be added later. I do like the fact that there is sprinting, crouching and walking, and that it makes a sound when you walk on the sand, but I do think that the sprint speed needs to be reduced since currently it feels a bit too fast.

SECOND PROTOTYPE -LOGIN SYSTEM

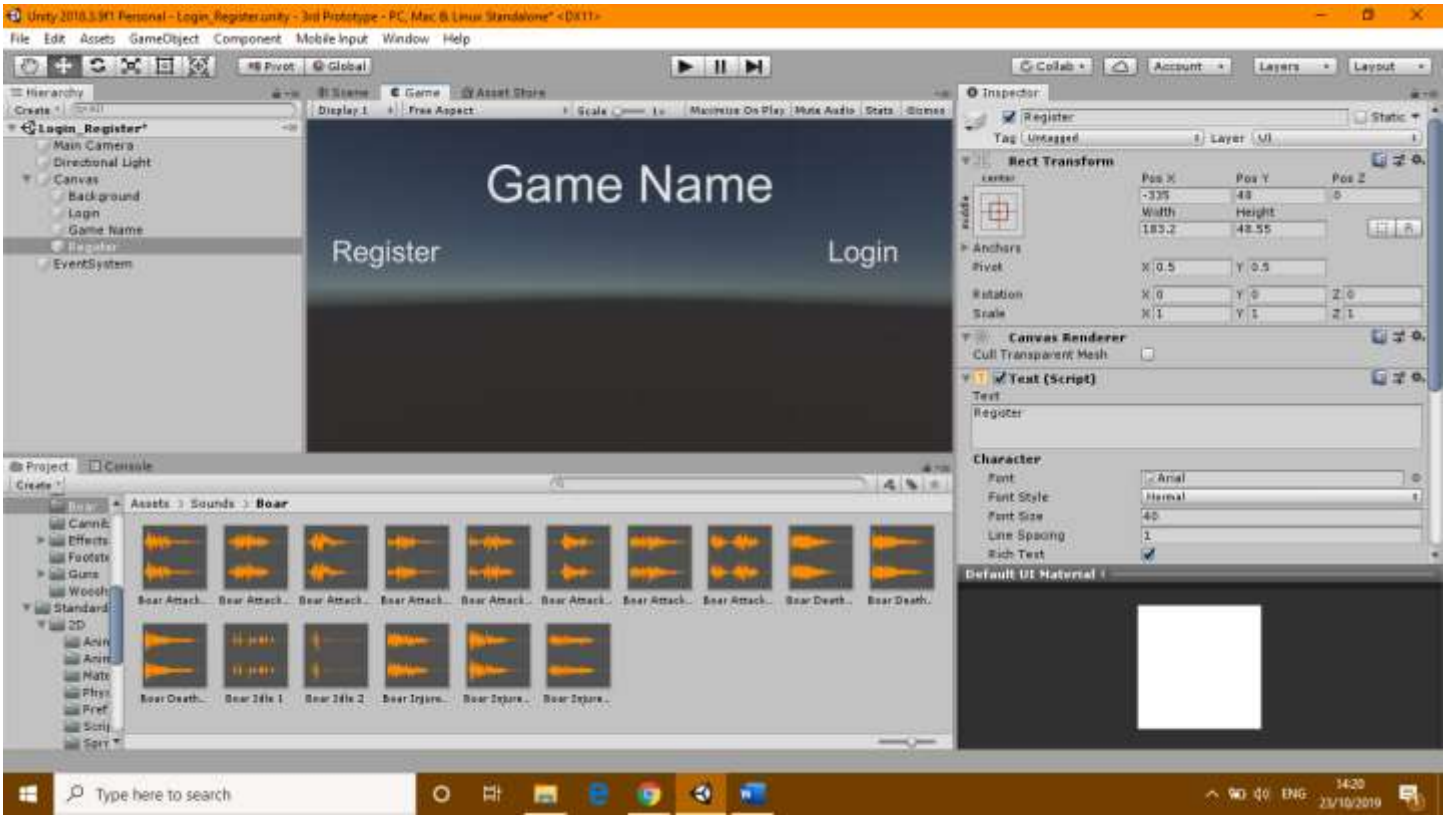


I then moved on to create a log in screen this required me to make a "UI panel" and change the colour to black and translucent. I then renamed the panel to "Background" and added 3 "UI text" to the canvas renaming them to "Login", "Register", "Game Name" as well as changing the text script for each UI text. I proceeded to adjust each text to the correct colour, alignment and size.

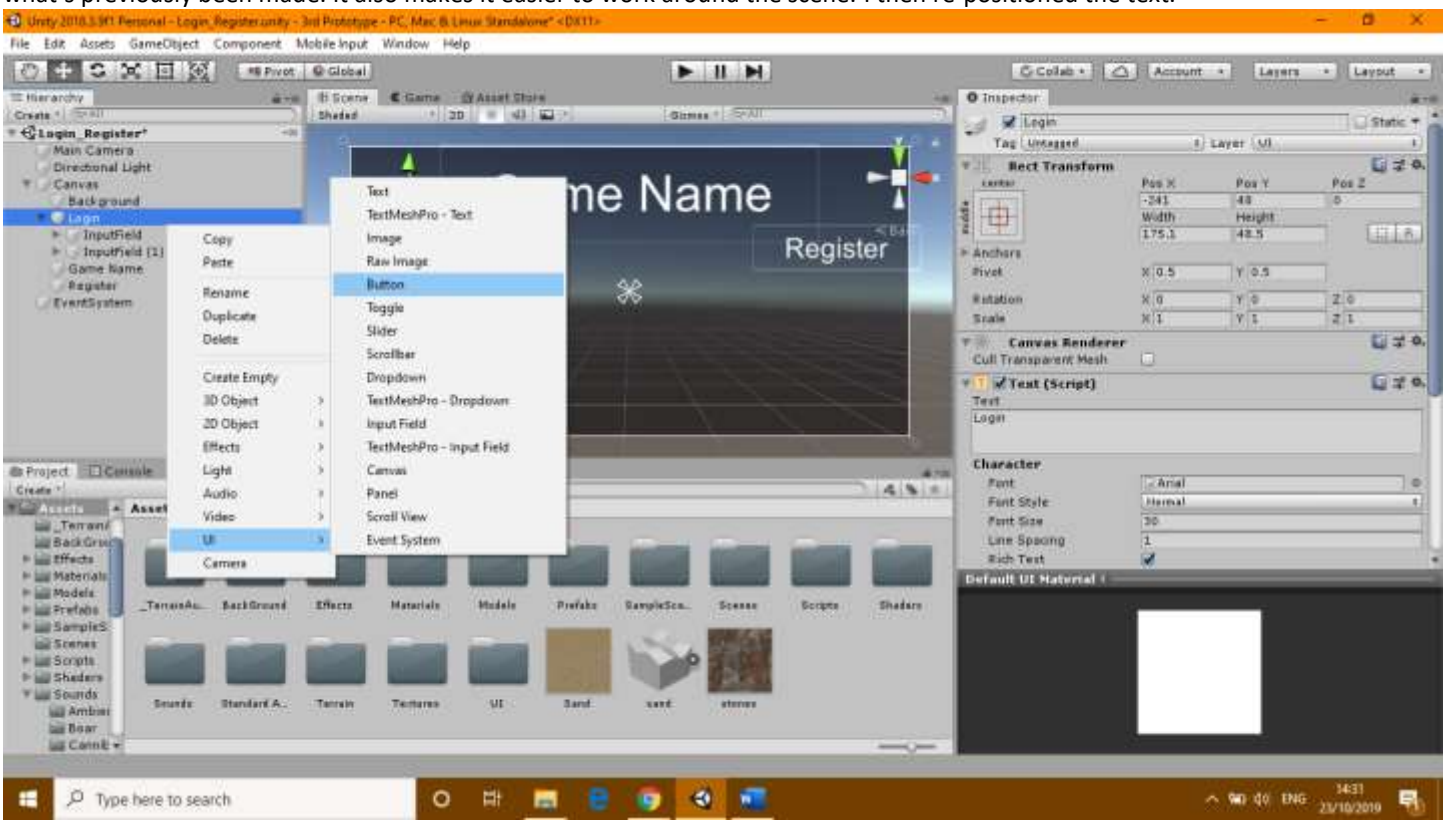
SECOND PROTOTYPE -LOGIN SYSTEM-GUI/NEW SCENE



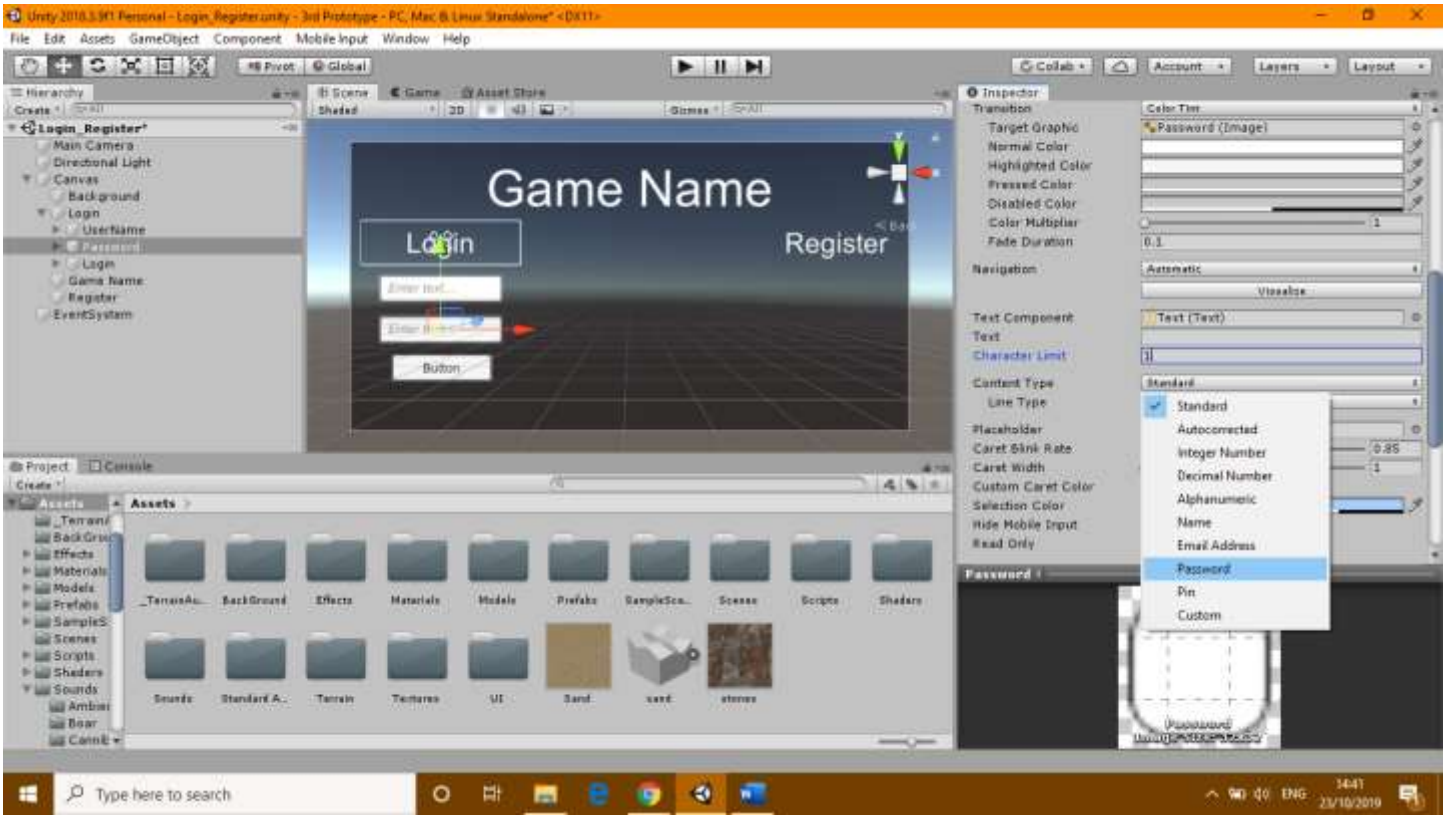
This screenshot shows a new scene has been created for the login system.



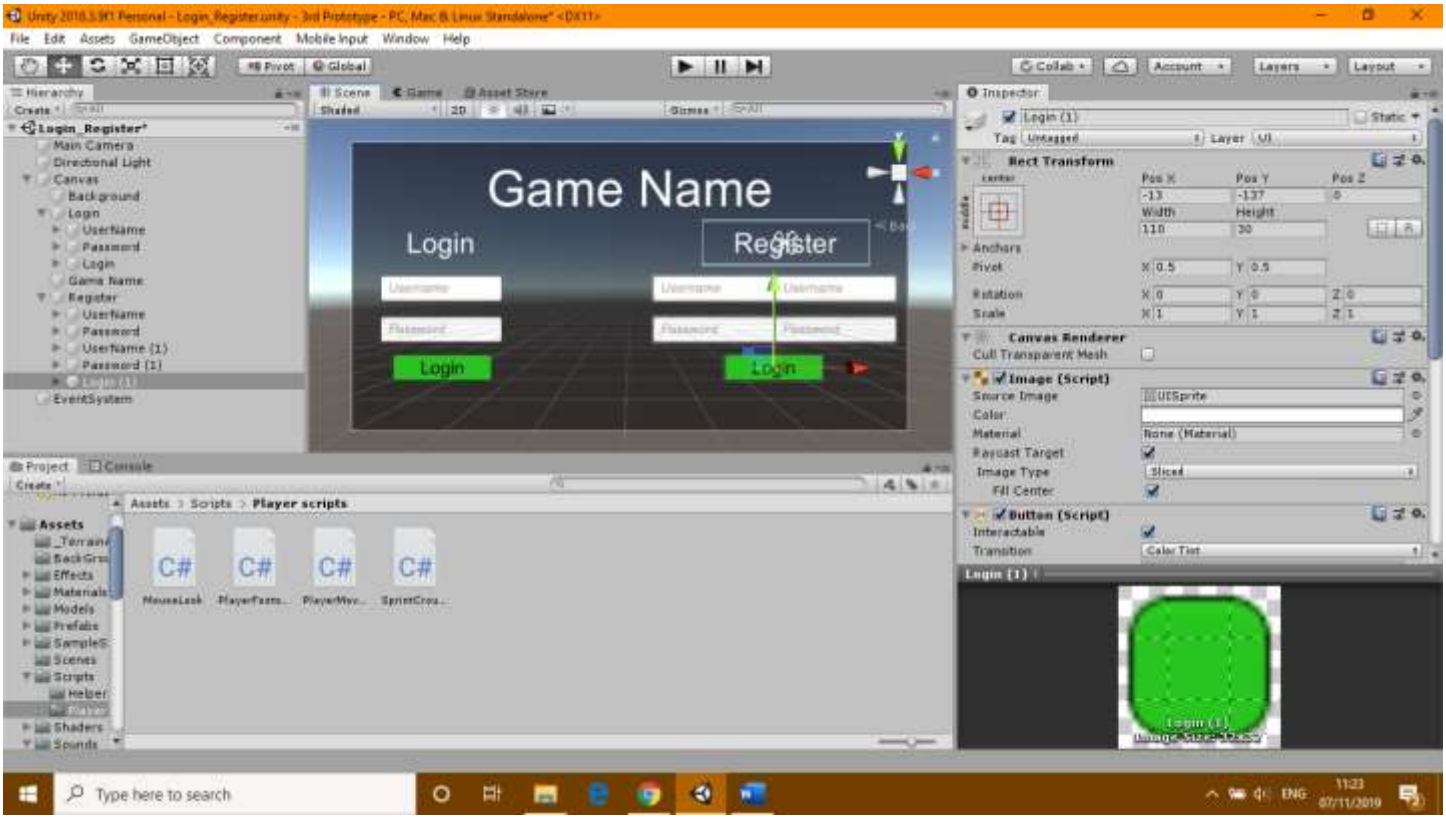
Stakeholder advice-I decided to create a new scene and copy what I previously made into the new scene. I chose to do this to make the game less complicated when developing the code and if any adjustments are made that corrupt the game that it won't affect what's previously been made. It also makes it easier to work around the scene. I then re-positioned the text.



I went onto adding an "InputField" in the Login section to then duplicate it to make 2 as well as adding a button. This was then resized and positioned to how I wanted it, with the correct names of each of their functions.

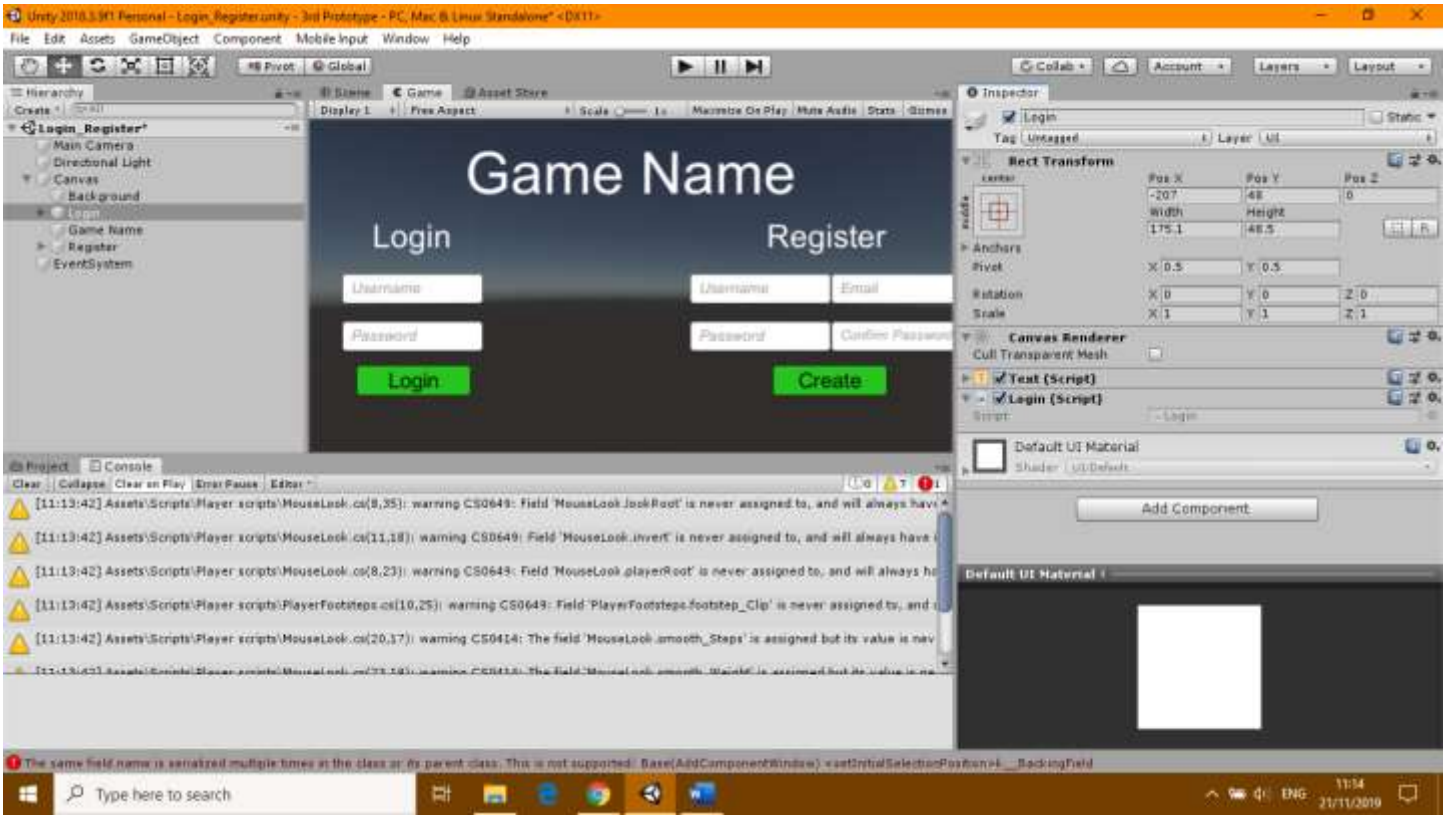


The password input field needed to be changed to ”Password” under content type, allowing it to be more secure by changing the character’s to an Asterix when displayed on the screen. I also changed the character limit to 17 so not too much memory is taken up when holding a password. Alongside this I changed the “placeholder” for both the username and password to “Password” and “Username” (both sized and aligned central) allowing the user to know what to type in the box. To make the layout easier on the user’s eyes I changed the login button to a pale green (once all the basic login functions are done, I will be altering the design making it more tailored to my game).



I copied and pasted the “Username”, “Register” and Login button from the previous made button and text box’s, to then copy them again remove the extra login button and drag and drop it into the register section. Proceeding to change the names to the correct register names as well as changing the text size and button colour. Top right input text box was changed to “Email” and the bottom left right text box was changed to confirm password. I re-arranged the order of box making the layout more user friendly. I finally changed the text of the second login button to create.

-User Feedback Point

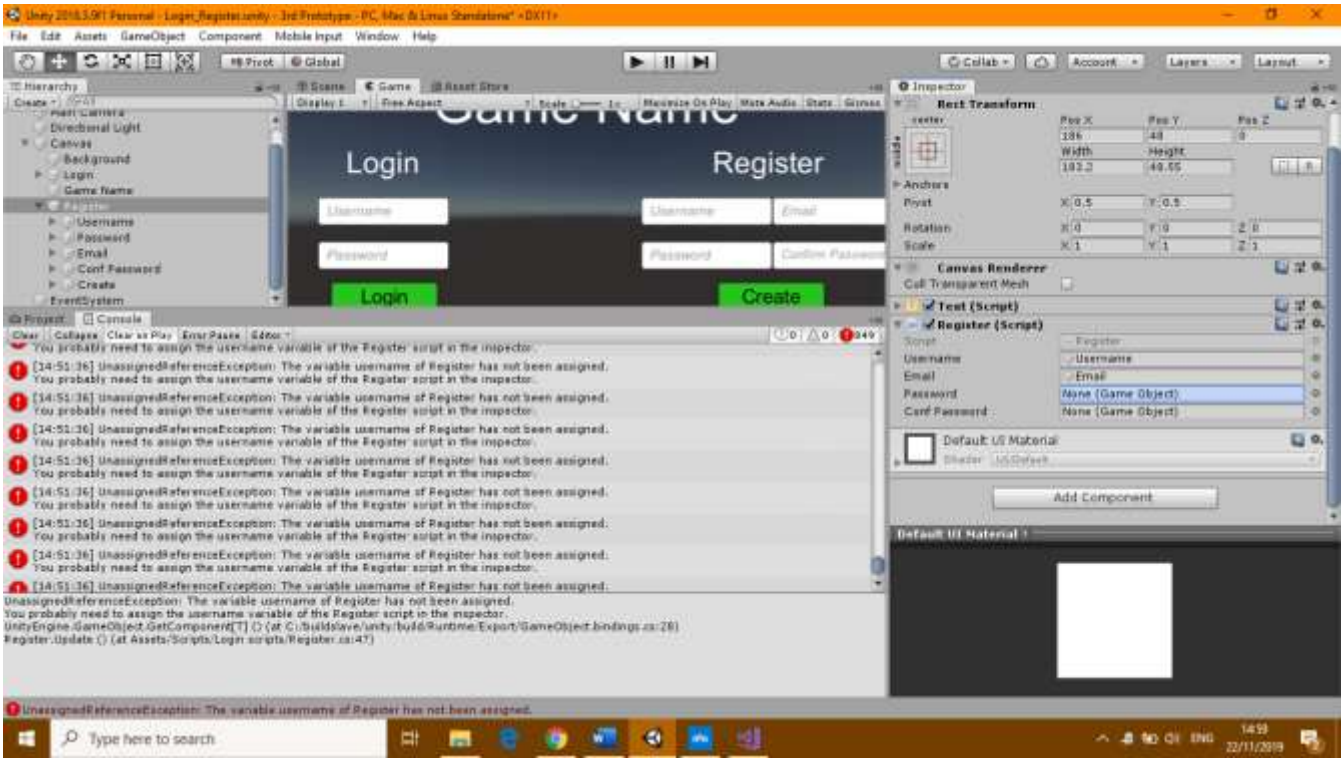


I created 2 new components one in the login section called “login.cs” and one in the register section called “register.cs” I then created a new file in “assets/Scripts” called “Login scripts” and placed these new c# scripts into it.

```
if (Input.GetKeyDown(KeyCode.Tab)) //checks if the Tab button is pressed down
        {
            if (Username.GetComponent<InputField>().isFocused) //checks if the username component is
selected
            {
                email.GetComponent<InputField>().Select(); //Select's the email field
            }
            if (email.GetComponent<InputField>().isFocused)//checks if the email component is selected
            {
                password.GetComponent<InputField>().Select();//Select's the password field
            }
            if (password.GetComponent<InputField>().isFocused)//checks if the password component is
selected
            {
                confPassword.GetComponent<InputField>().Select(); //Select's the confirmed password
field
            }
        }
```

-Additional pieces of code to make it more affiant

This was added into the login script which allows the user to press tab in one of the fields and it automatically go into the next appropriate field

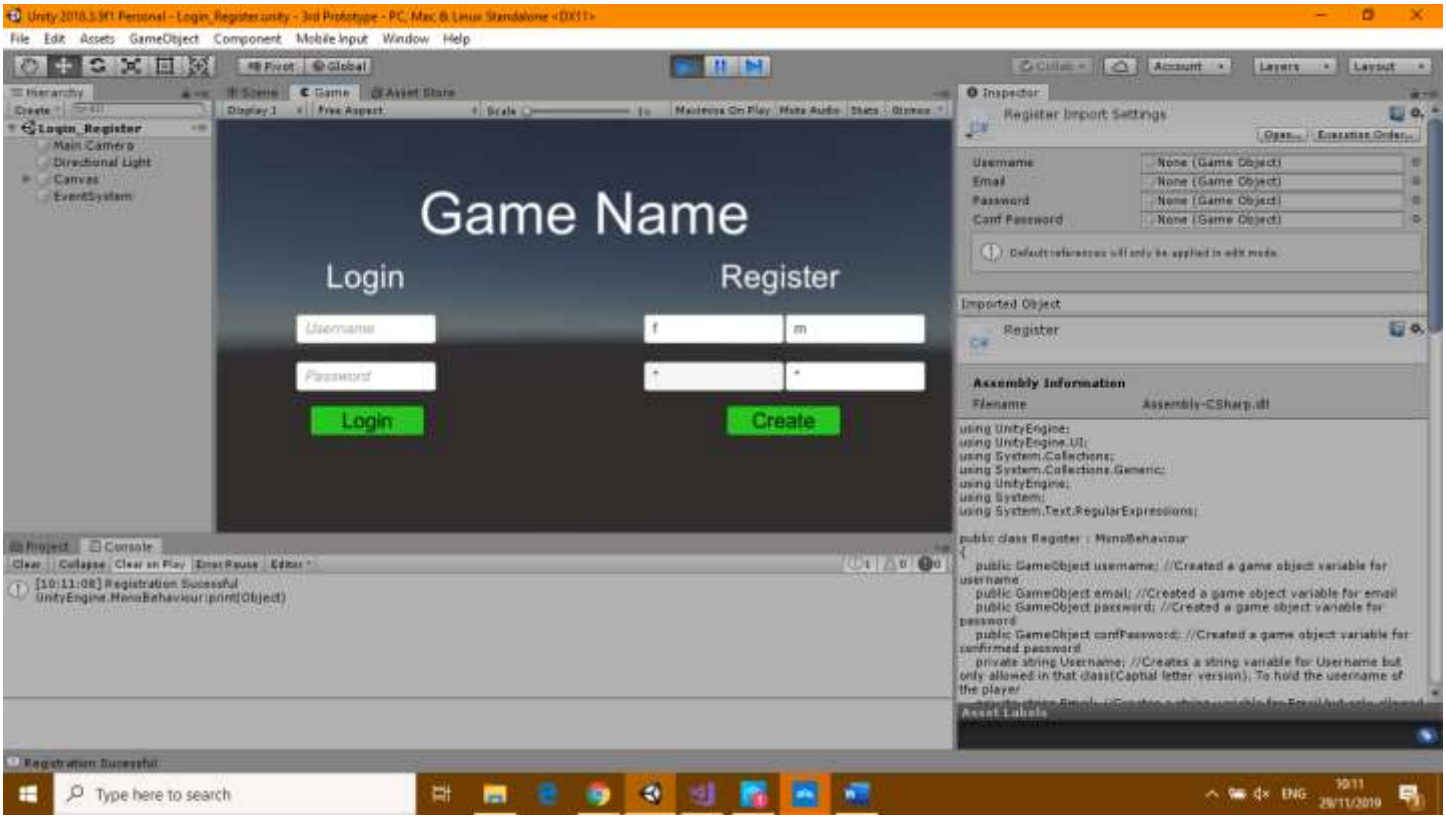


After running the script I gained this error “UnassignedReferenceException: The variable username of Register has not been assigned.” to then realise that I haven't assigned my variables to my components therefore I dragged the correct one to each variable and this fixed the error and my tabbed function worked.

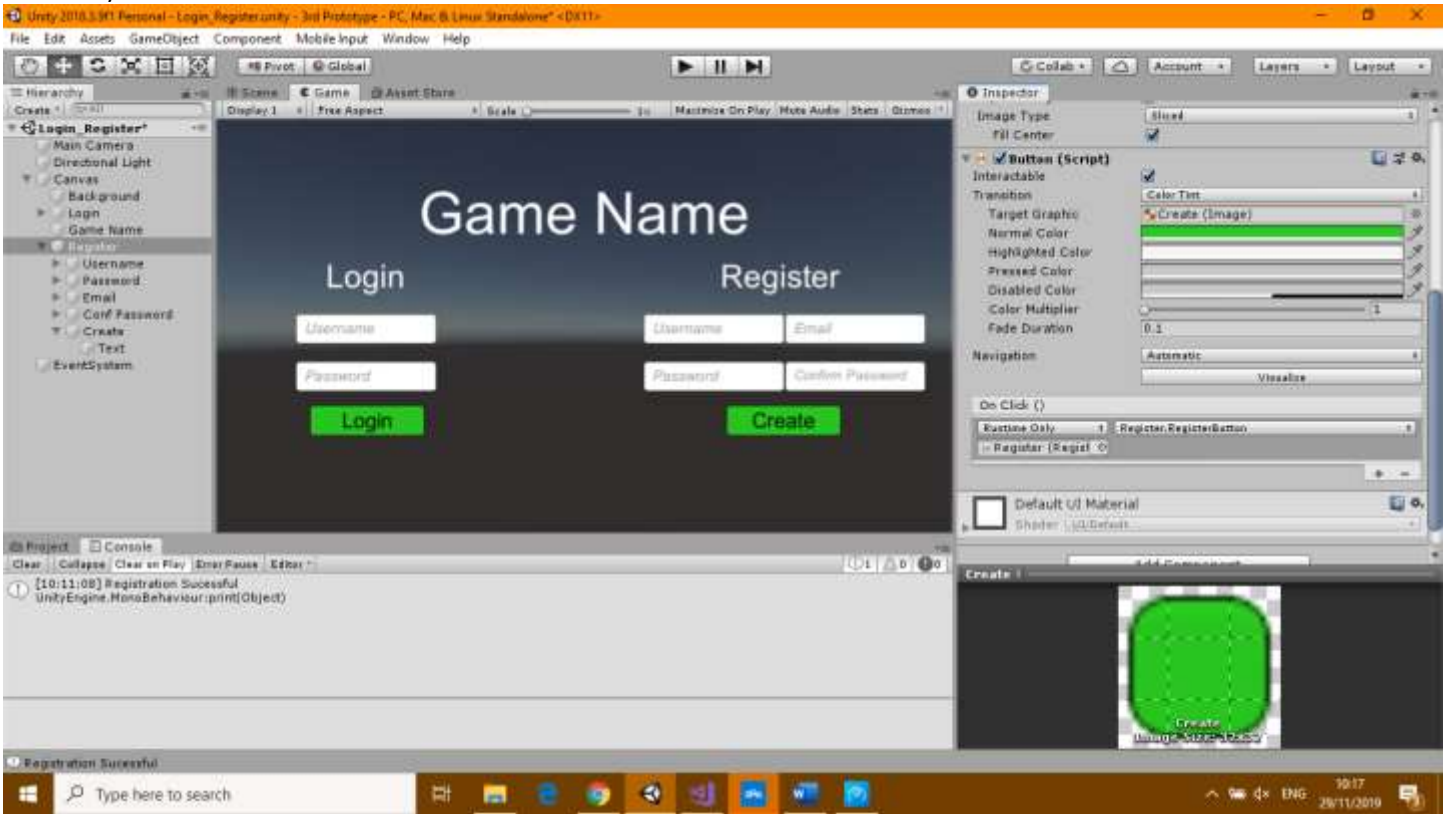
```
public void RegisterButton()//A public function that's called upon when the register button is pressed
    {
        print("Registration Sucessful"); //tests if the function has been called
    }

if (Input.GetKeyDown(KeyCode.Return))//Checks if the enter button is pressed
        {
            if (Password !="" &&Email != ""&&ConfPassword !=""&& Username != "") //checks that all the
correct fields aren't empty to allow the user to continue
            {
                RegisterButton(); //calls the register button function
            }
        }
```

-Additional pieces of code to make it more affiant

I then added to the code to check if all the correct fields for register aren’t blank and if so it wouldn’t go a new function called “Register Button” and output “Registration Successful” to test if there are no bugs. This only worked when the enter button was pressed I still needed to add it onto the create button.



This proved to be successful and printed out ""Registration successful" when no register fields were blank and didn't print anything when they at least one was.



-Additional pieces of code to make it more affiant

SECOND PROTOTYPE -LOGIN SYSTEM-RULES

I then assigned the “create” button to the register function by dragging the register variable into the object box and attaching it to the “RegisterButton” function.

```
if (System.IO.File.Exists(@"C: \Users\Owen Ball\Documents\unity projects\Game data" + Username + ".txt"))//checks if that file existsand so that duplicate usernames don't occur
```

This piece of code stop’s any duplicate usernames being created by checking if it already exists in that file path.

```
public void RegisterButton()//A public function that's called upon when the register button is pressed
    {
        bool UN = false; //creates a decision variable UN for username. Checks if it passes all the correct rules
        bool EM = false; //creates a decision variable EM for email. Checks if it passes all the correct rules
        bool PW = false; //creates a decision variable PW for password. Checks if it passes all the correct rules
        bool CPW = false; //creates a decision variable CPW for confirm password. Checks if it passes all the correct rules

        if (Username != "") //Checks if the Username field is blank
        {
            if (System.IO.File.Exists(@"C: \Users\Owen Ball\Documents\unity projects\Game data" + Username + ".txt"))//checks if that file existsand so that duplicate usernames don't occur
            {
                UN = true; //makes the variable UN true, therefore stating all the rules are correct and the username is valid
            }
            else
            {
                debug.LogWarning("Username Taken"); //Tells the user that the username has already been taken
            }
        }
        else
        {
            Debug.LogWarning("User Field empty"); //Tells the user that the username field is empty
        }
        if (Email != "") //Checks if the Email field is blank
        {
            if (EmailValid) //Checks if the first character and last character is a letter not a symbol
            {
                if (Email.Contains("@")) //Checks if the Email variable contains a "@". Makes sure its a valid email address
                {
                    if (Email.Contains(".")) //Checks if the Email variable contains a ".". Makes sure its a valid email address
                    {
                        EM = true; //makes the variable EM true, therefore stating all the rules are correct and the email is valid
                    }
                    else
                    {
                        Debug.LogWarning("Email Is Incorrect"); //Tells the user that the Email is incorrect
                    }
                }
                else
                {
```

```
                    Debug.LogWarning("Email Is Incorrect"); //Tells the user that the Email is
incorrect
                }
            }
            else
            {
                Debug.LogWarning("Email Is Incorrect"); //Tells the user that the Email is incorrect
            }
        }
        else
        {
            Debug.LogWarning("Email Field Empty"); //Tells the user that the email field is empty
        }
        if (Password != "") //Checks if the Email field is blank
        {
            if (Password.Length > 5) //Checks if the password length is greater than 5
            {
                PW = true; //makes the variable PW true, therefore stating all the rules are correct
and the password is valid
            }
            else
            {
                Debug.LogWarning("Password Must Be At Least 6 Characters long"); //Tells the user that
the password needs to be at least 6 characters long
            }
        }
        else
        {
            Debug.LogWarning("Password Field Empty"); //Tells the user that the password field is empty
        }
        if (ConfPassword != "")  //Checks if the Confirmed password field is blank
        {
            if (ConfPassword == Password) //Checks if both the password variable and confirm password
are the same
            {
                CPW = true; //makes the variable CPW true, therefore stating all the rules are correct
and the passwords are the same
            }
        }
        else
        {
            Debug.LogWarning("Passwords Don't Match"); //Tells the user that the passwords don't match
        }
    }
```

This makes piece of code in the register.cs() makes sure the password's match, that no fields are blank, the password contains more than 6 characters and the email is valid by seeing if it has a letter in the front of the email and end, as well as an "@" and ".".

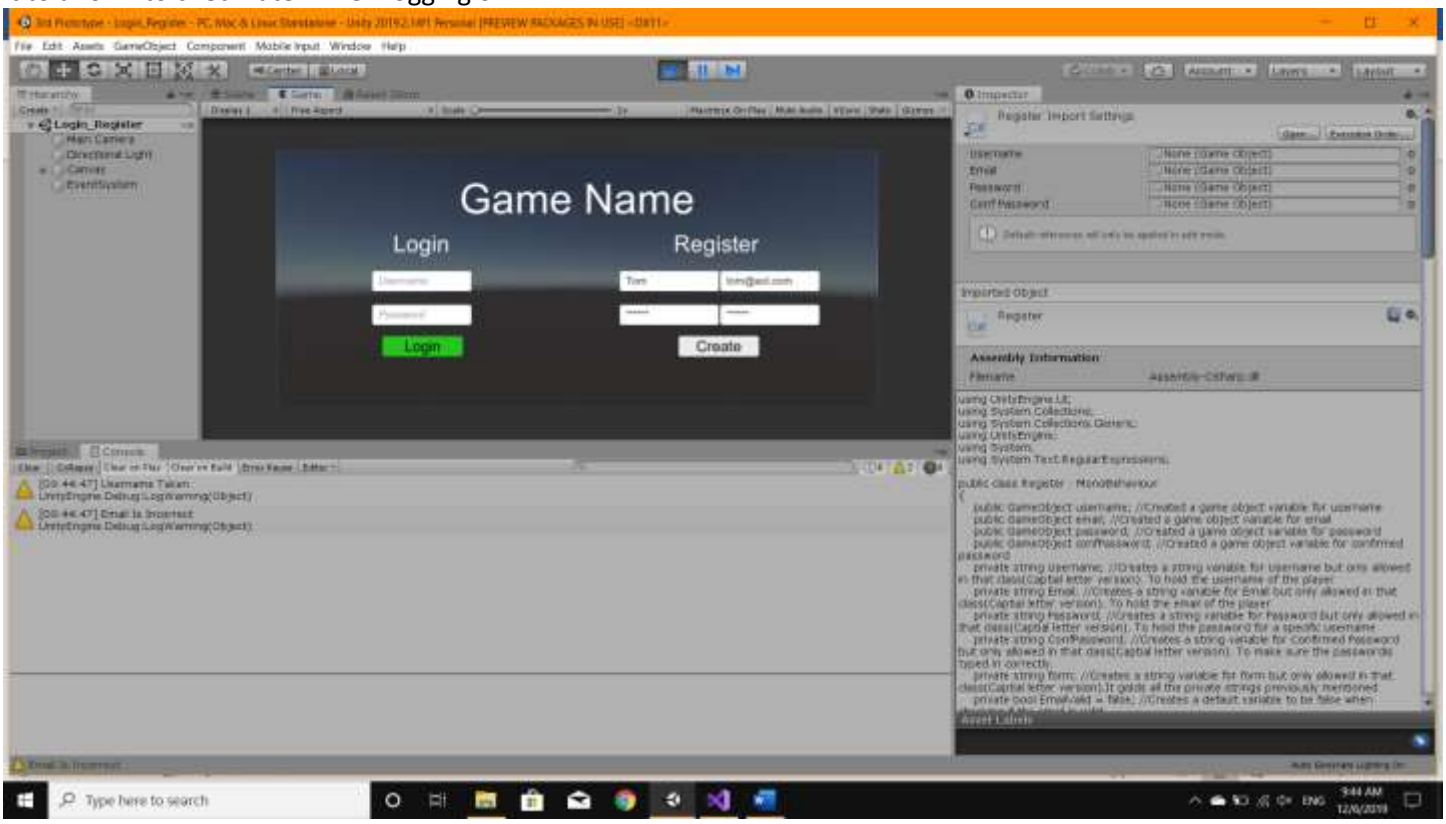
```
if (UN == true &&EM ==true && PW == true && CPW == true) //checks if all the rules have been checked
and are passed off as okay
        {
            bool Clear = true; //creates a variable that states that every variable is correct
            int i = 1;
            foreach(char c in Password) //for each character in password
            {
                if (Clear) //doesn't clear password everytime it loops through the function
                {
                    Password = "";
                    Clear = false;
```

```
                }
                i++; //iterates i by 1
                char Encrypted = (char)(c * i); //encryptes each letter of the password
                Password += Encrypted.ToString(); //re-puts it as a string
            }
```

This part of the code for register.cs() encrypts the code by iterating I by one every loop searching through each character's individually in the password and timesing the character by I.

```
form = (Username + "/n" + Email + "/n" + Password); //concatinates all the data for the new created user
            System.IO.File.WriteAllText(@"C: \Users\Owen Ball\Documents\unity projects\Game data" + Username + ".txt", form); //writes the data into a form for the new created user
            username.GetComponent<InputField>().text =""; //Set's the username field to blank
            email.GetComponent<InputField>().text = ""; //Set's the email field to blank
            password.GetComponent<InputField>().text = ""; //Set's the password field to blank
            confPassword.GetComponent<InputField>().text = ""; //Set's the confirm password field to blank
            print("Registration Complete");
```

This piece of code also from the register.cs() file concatenates all the data that has been created and checked and write it to a form to check later when logging on.



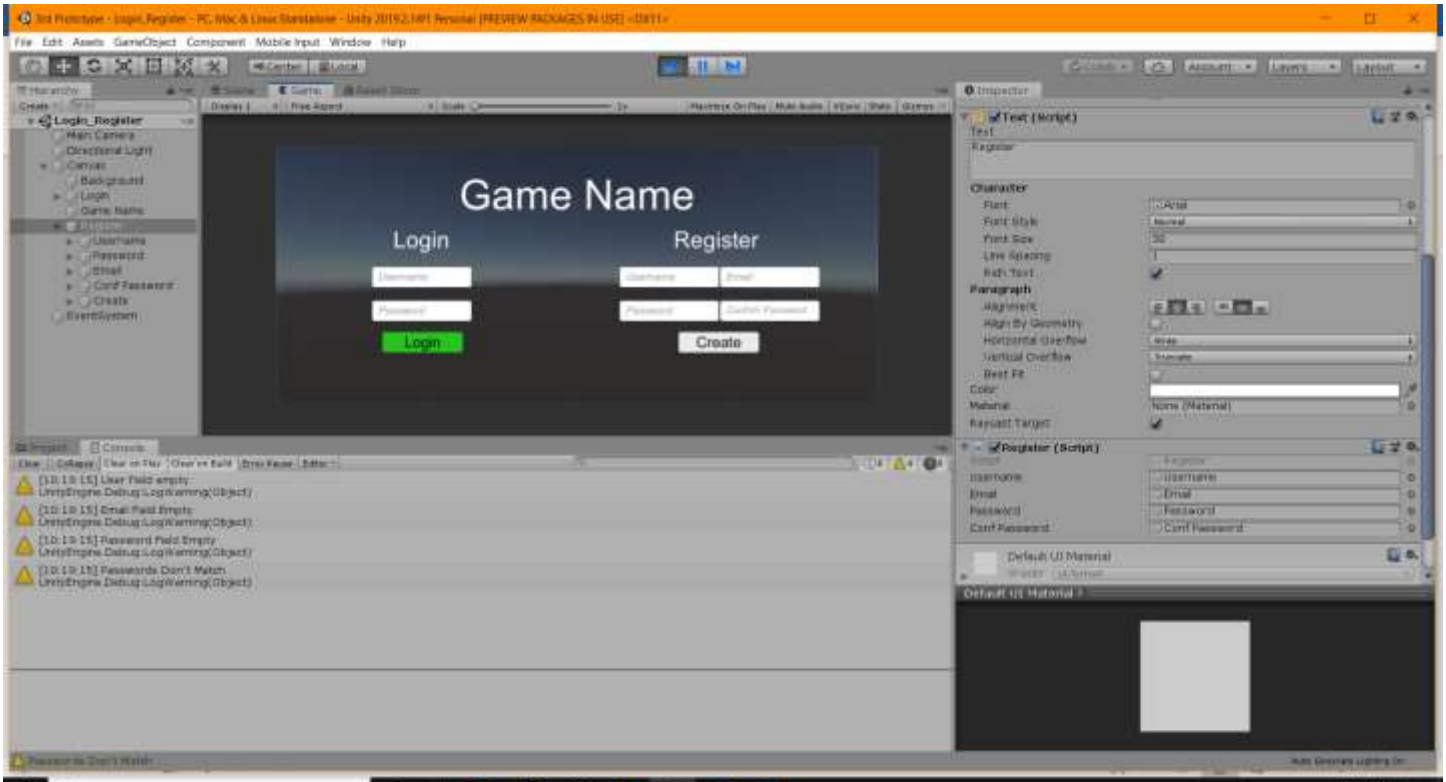
I then tested the script to register a user with the correct details, therefore no error's should occur. However the console appeared with "UserName Taken" and "Email Incorrect" so I needed to relook at my code

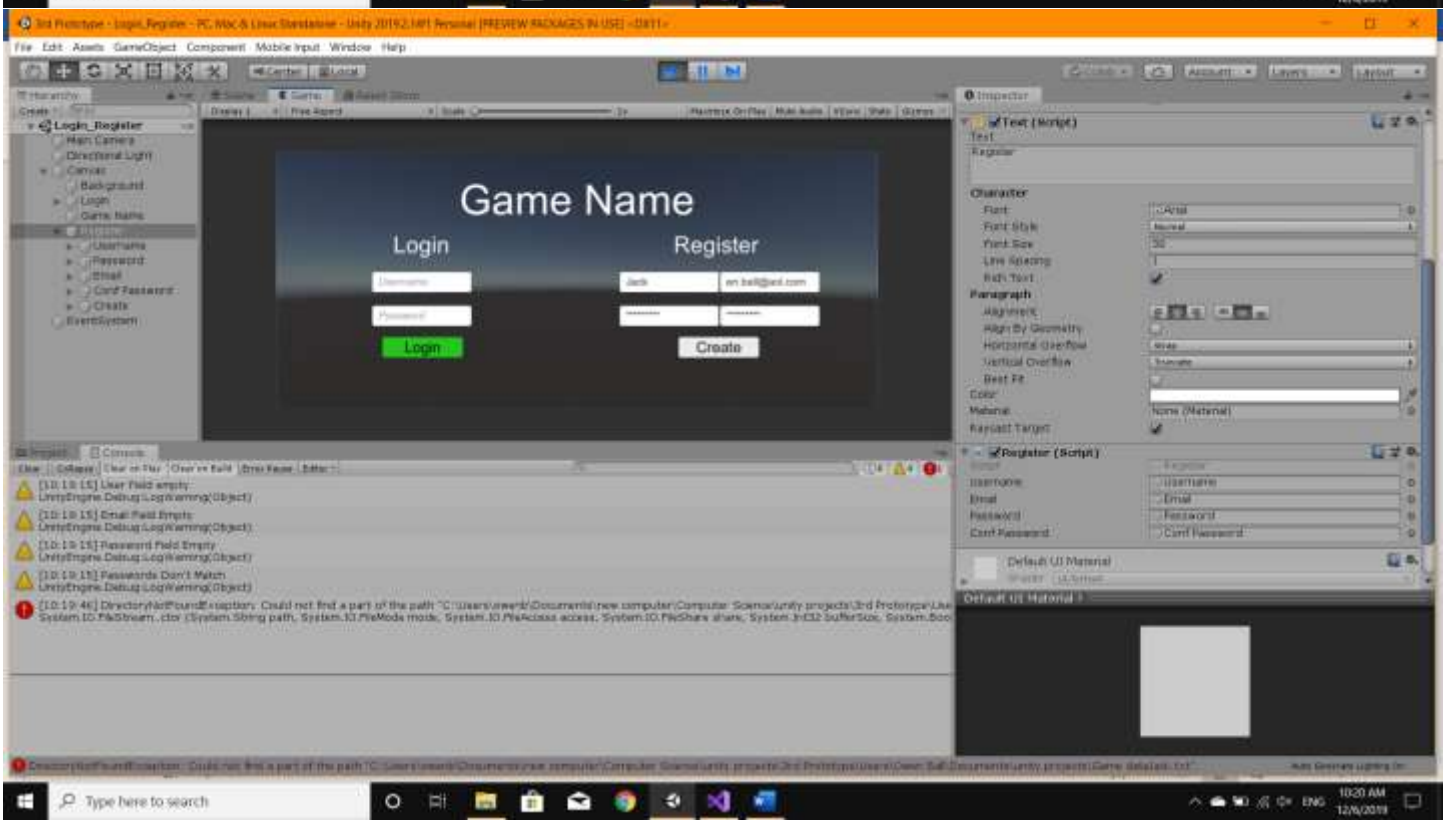
```
if (!System.IO.File.Exists(@"C: \Users\Owen Ball\Documents\unity projects\Game data" + Username + ".txt"))//checks if that file existsand so that duplicate usernames don't occur
```

The first error to do with username already taken was due to their being no "!" in front of the System.IO.FileExists causing this error to occur. I corrected this as seen above

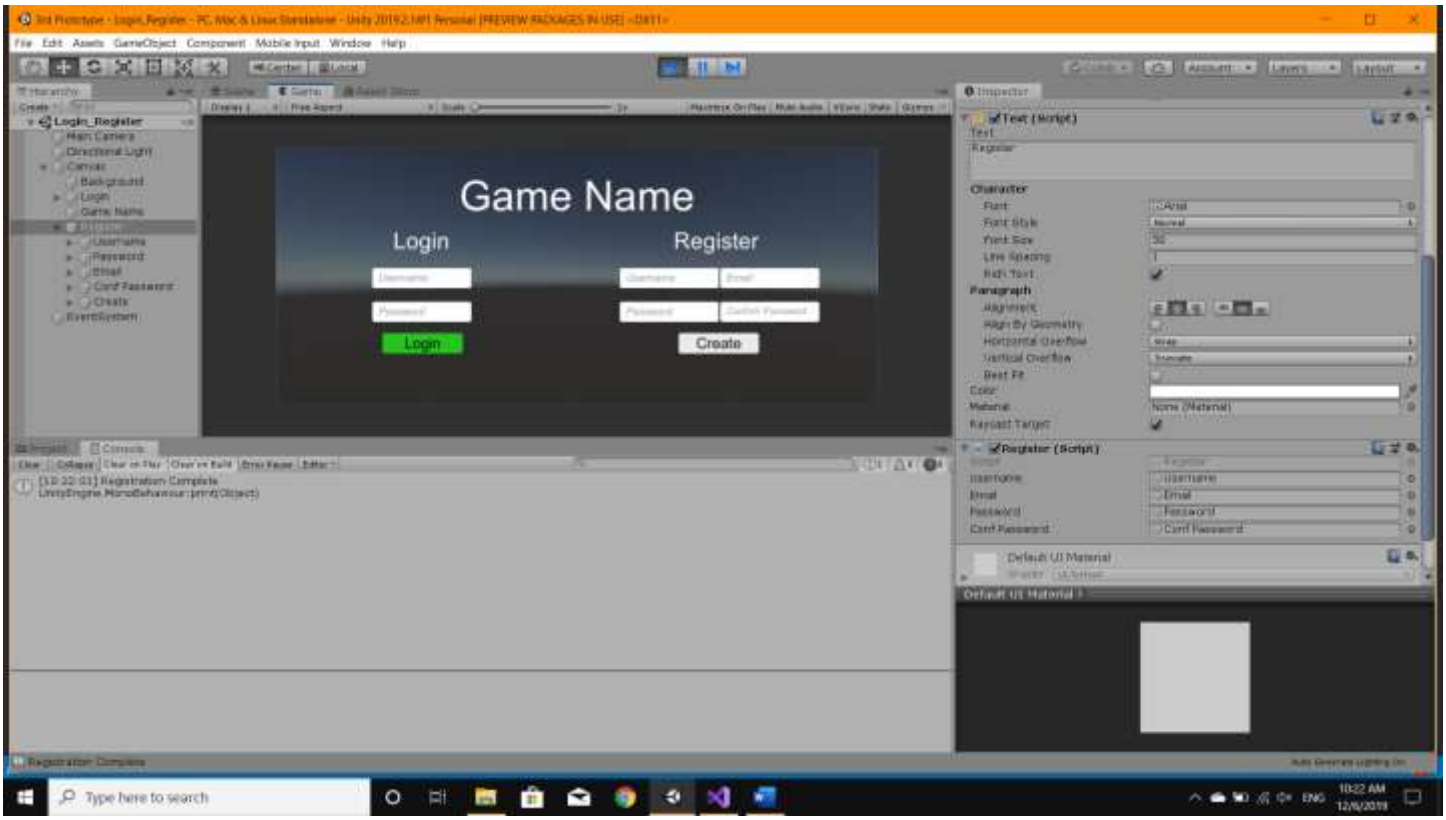
```
private string[] Characters = {"a", "b", "c", "d", "e", "f", "g", "h", "i", "j", "k", "l", "m", "n",
"o", "p", "q", "r", "s", "t", "u", "v", "w", "x", "y", "z",
    "A", "B", "C", "D", "E", "F", "G", "H", "I", "J", "K", "L", "M", "N", "O", "P", "Q", "R", "S", "T",
"U", "V", "W", "X", "Y", "Z", "1", "2", "3", "4", "5", "6", "7", "8","9", "0","_","-",};
    //Allows the encrypting to be able to know what letter is equivilent to the number by using a an
array called Characters
void EmailValidation() //a function to validate the email
    {
        bool SW = false;  //creates a decision variable SW for starts with. To check if it passes all
the correct rules for emails
        bool EW = false; //creates a decision variable EW for ends with. To check if it passes all the
correct rules for emails
        for (int i = 0; i < Characters.Length; i++) //Loops through all the characters in the email
        {
            if (Email.StartsWith(Characters[i]))//Checks if a character is present at the start
            {
                SW = true;  //sets SW to true therfore passing the correct rule
            }
        }
        for (int i = 0; i < Characters.Length; i++) //Loops through all the characters in the email
        {
            if (Email.EndsWith(Characters[i])) //Checks if a character is present at the end
            {
                EW = true; //sets EW to true therfore passing the correct rule
            }
        }
        if(SW == true &&EW == true) //Checks both rules have been set to true
        {
            EmailValid = true; //Set's the validation to true so all rules are met for the email
        }
    }
```

After re-looking over my code I realised forgot to put in the array to find which character represents each number and put the full validation for the email

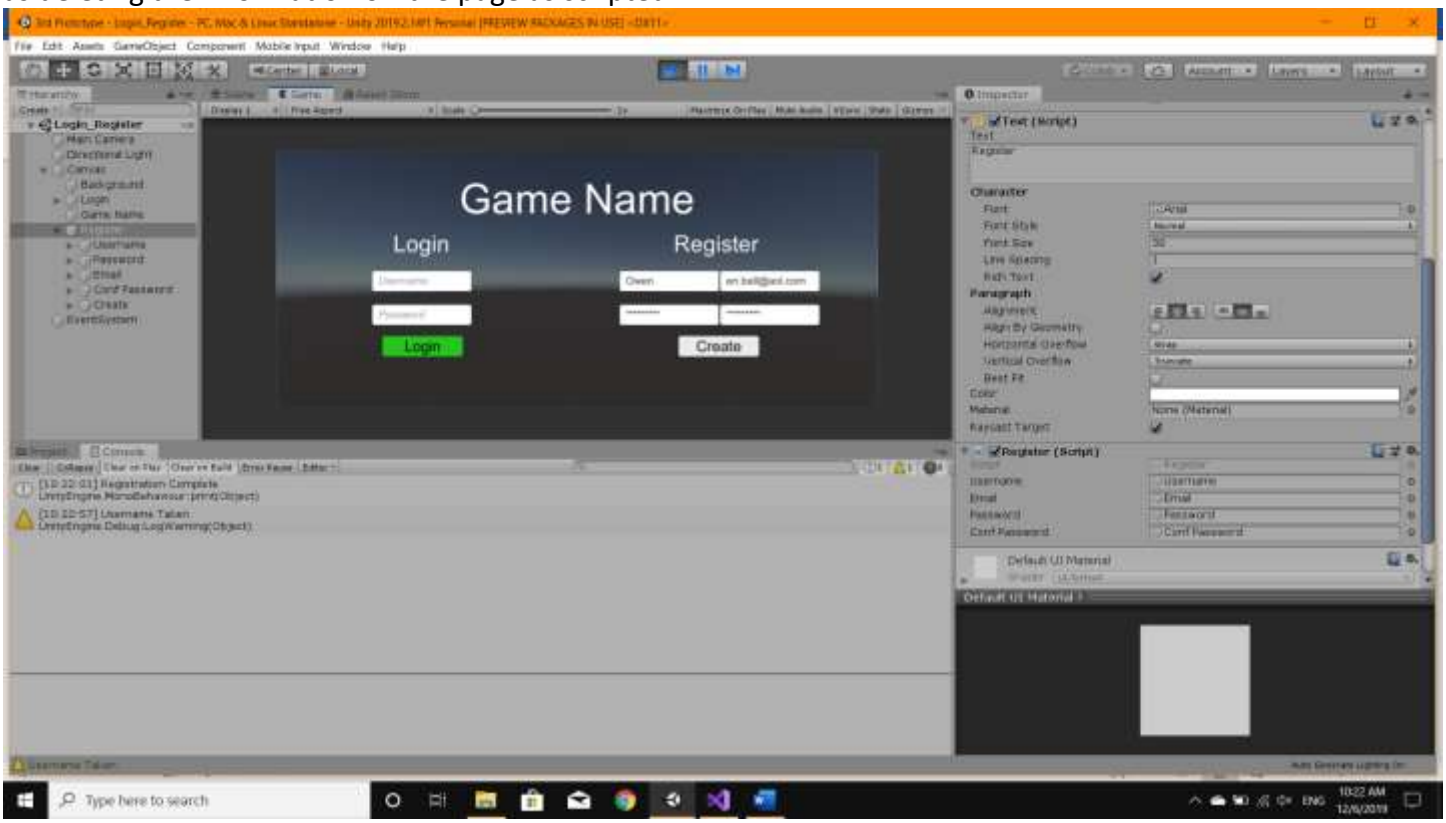

Game Name
Login
Register
Login
Create


Game Name
Login
Register
Login
Create

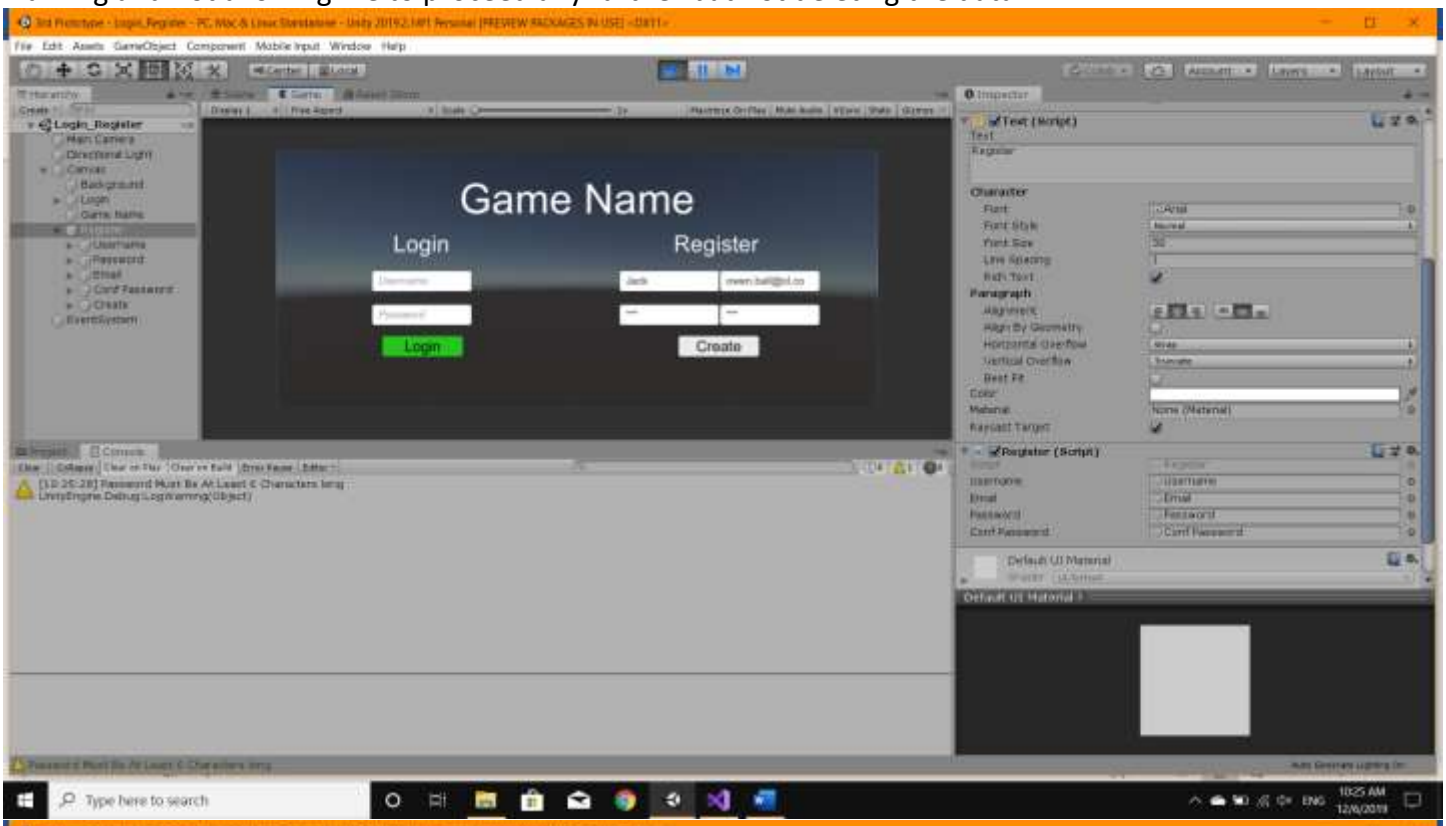
I the tested the field being blank and the correct warning showed up I then put the correct details in with no warning occurring except the wrong file path. This was due to me having a new laptop so a new file path was needed

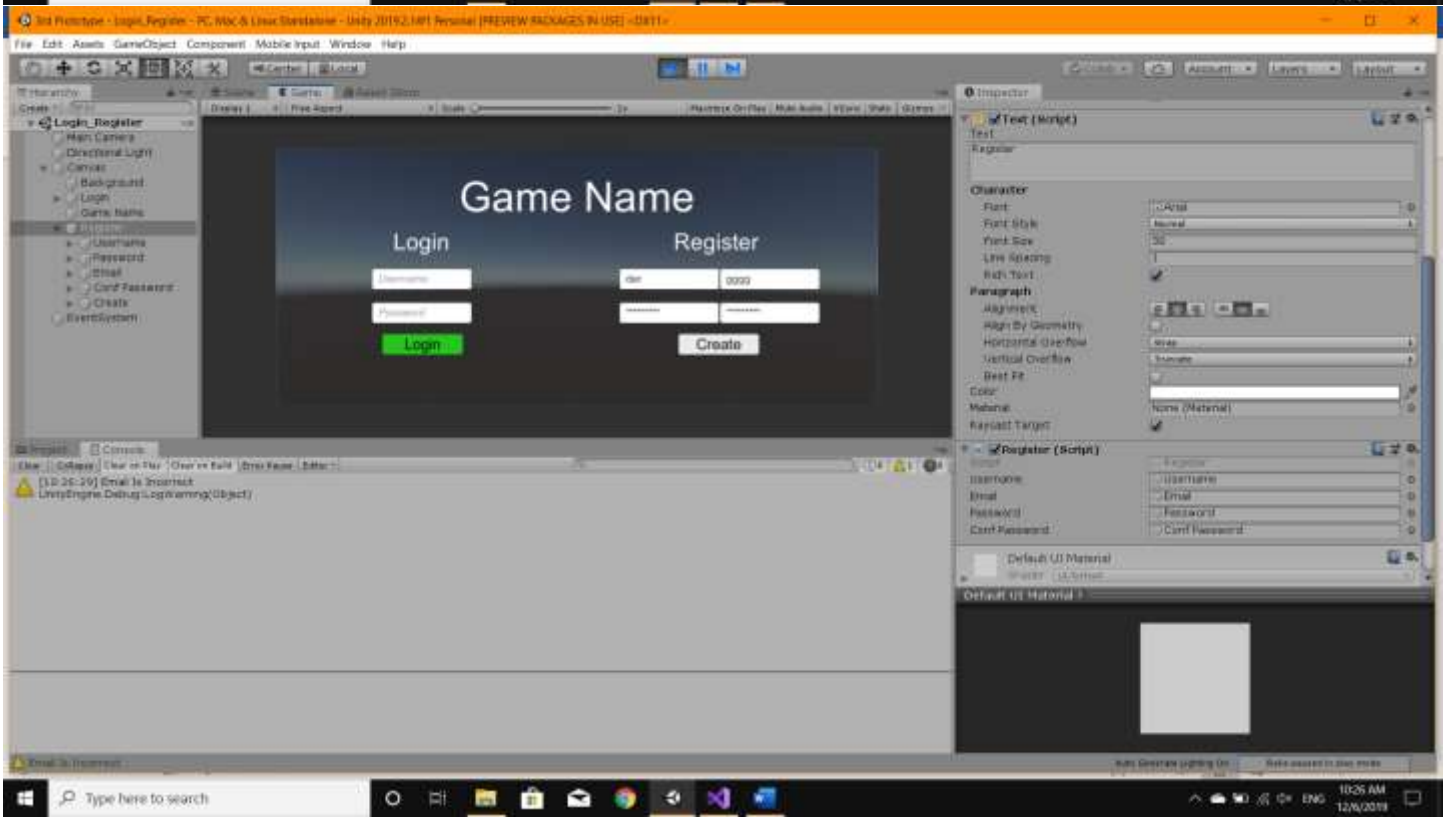


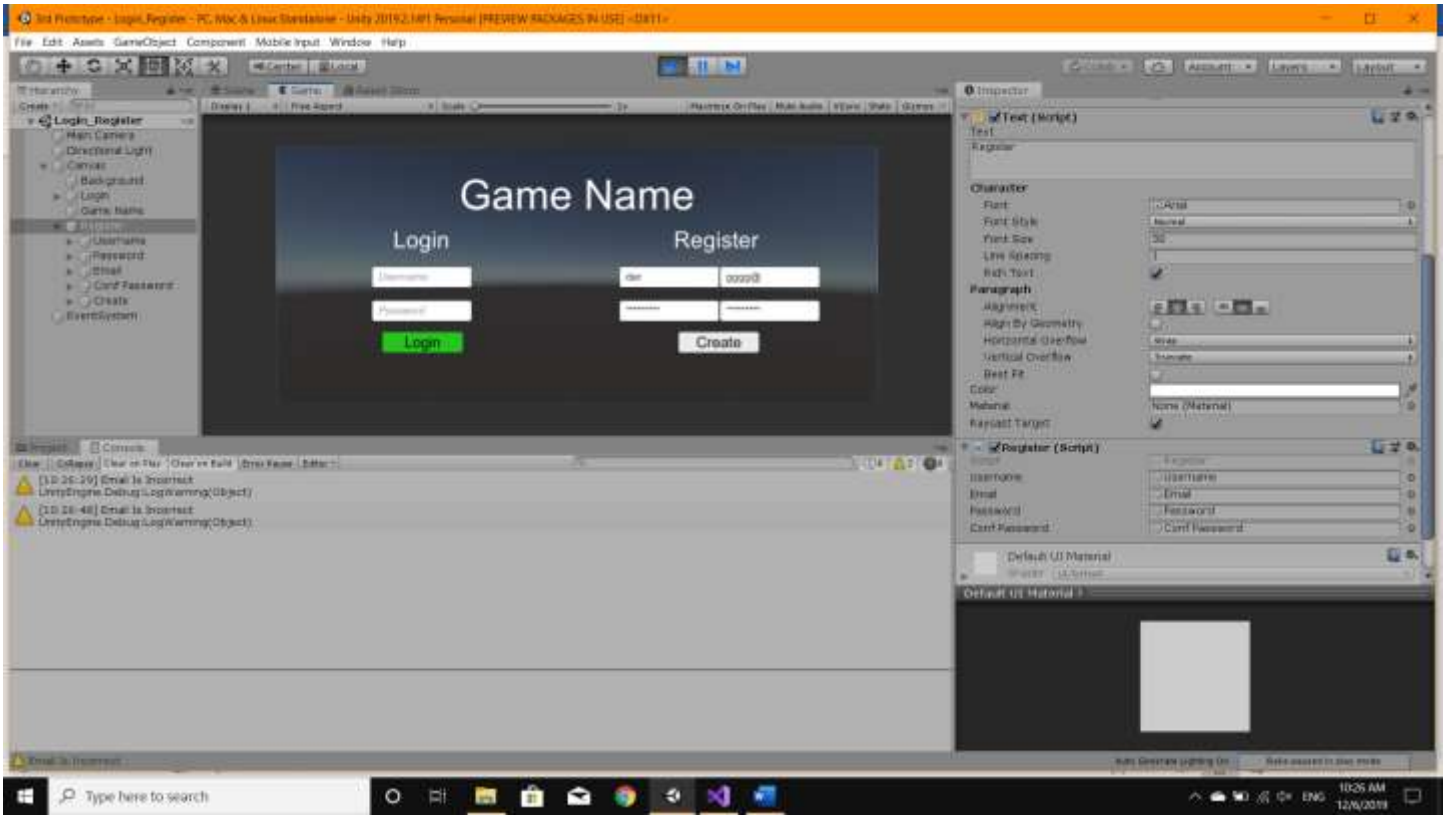
I replaced both file paths with the new one and in the correct data and my registration was completed as well as deleting the information off the page as scripted.



I the re-typed the same information to test the username taken feature and it worked giving the correct warning and not allowing me to proceed any further but not deleting the data

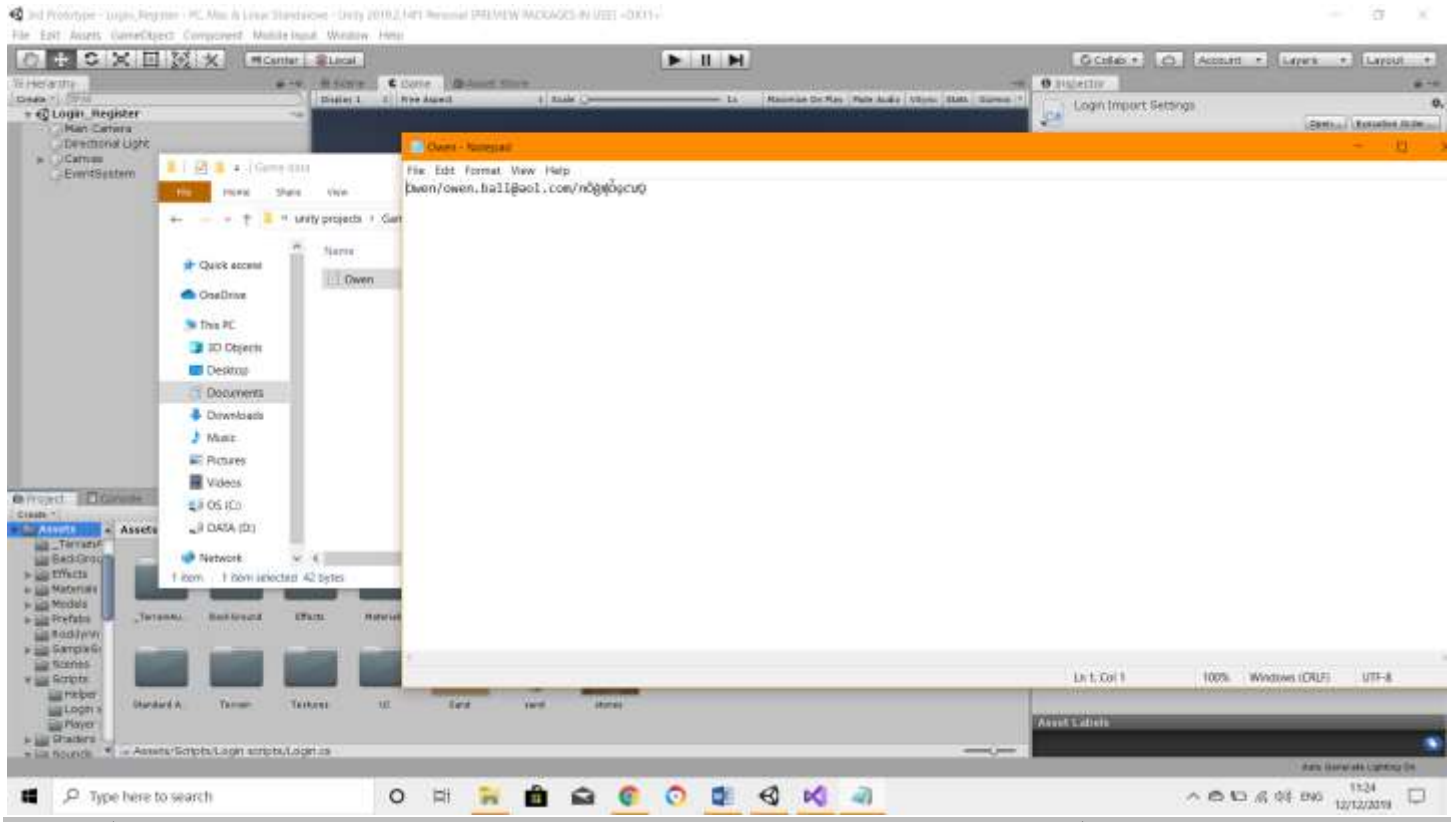






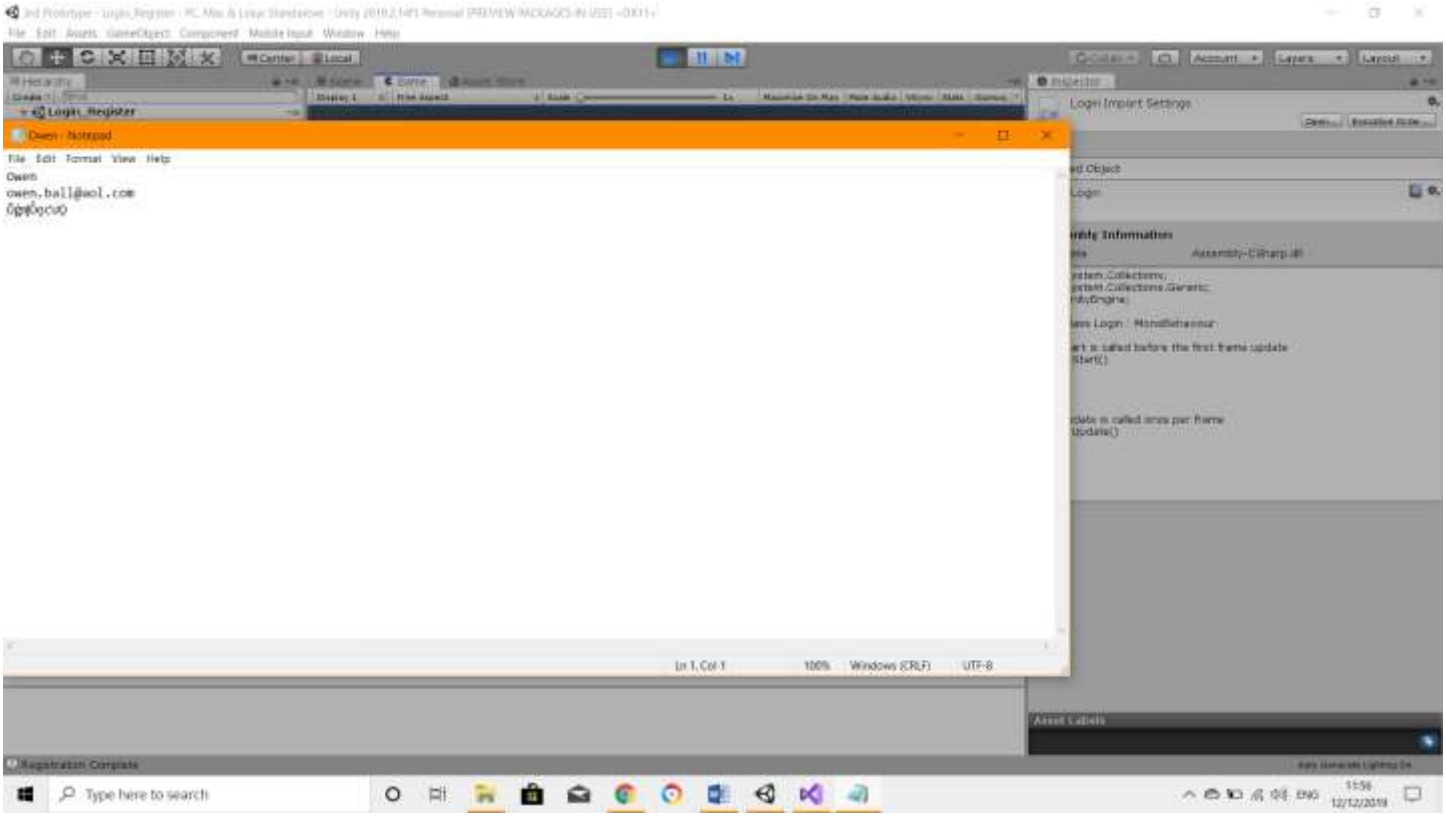
All the other features of warning ect work as tested however the password equaling the confirmed password didn't let the registration go through but no warning occurred I then added this as it as missing in the register().cs

```
else
            {
                Debug.LogWarning("Password Not being the same"); //Tells the user that the password needs to be the same
            }
```

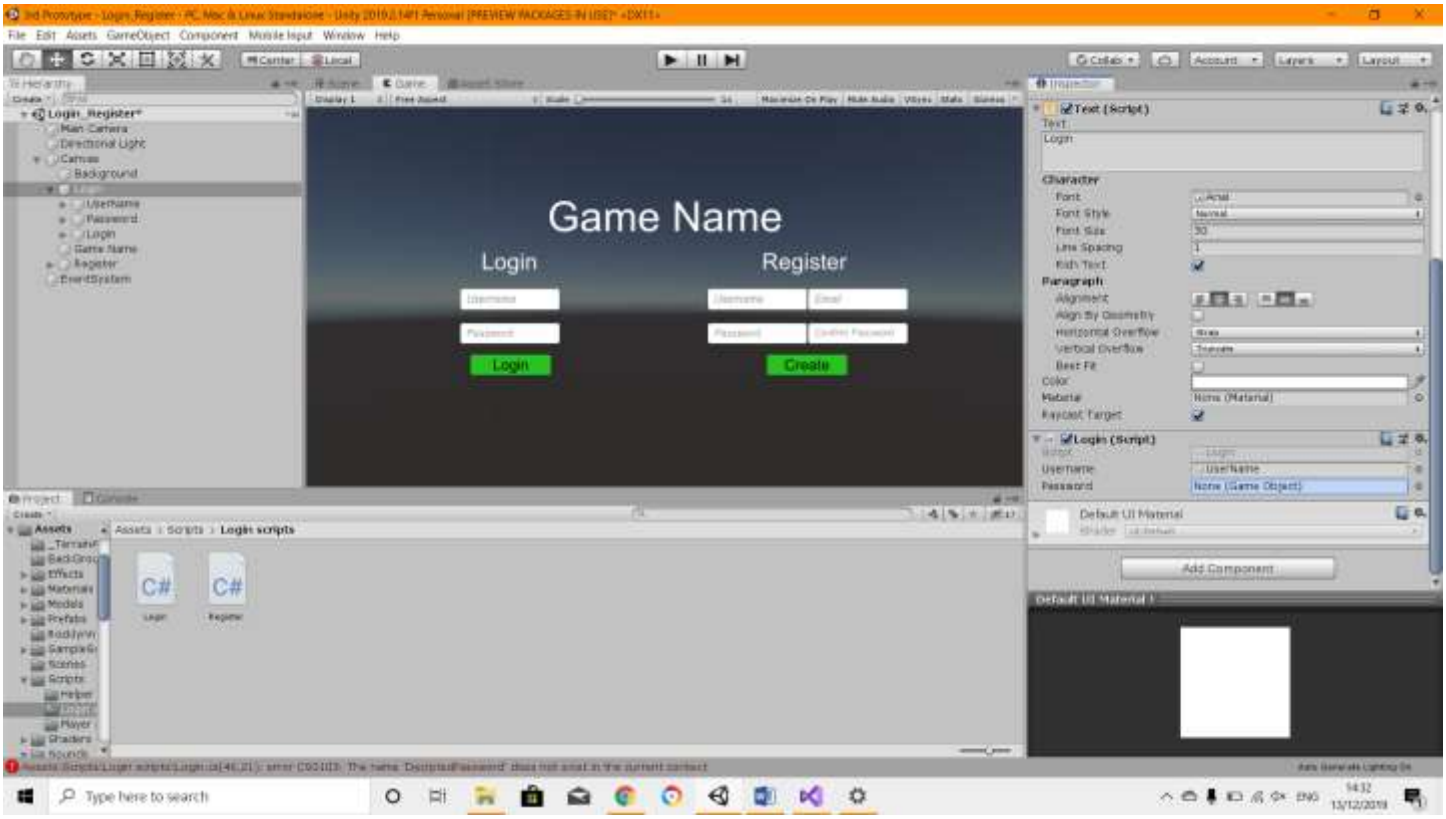



```
form = (Username +Environment.NewLine+ Email + Environment.NewLine + Password); //concatinates all the data for the new created user
```

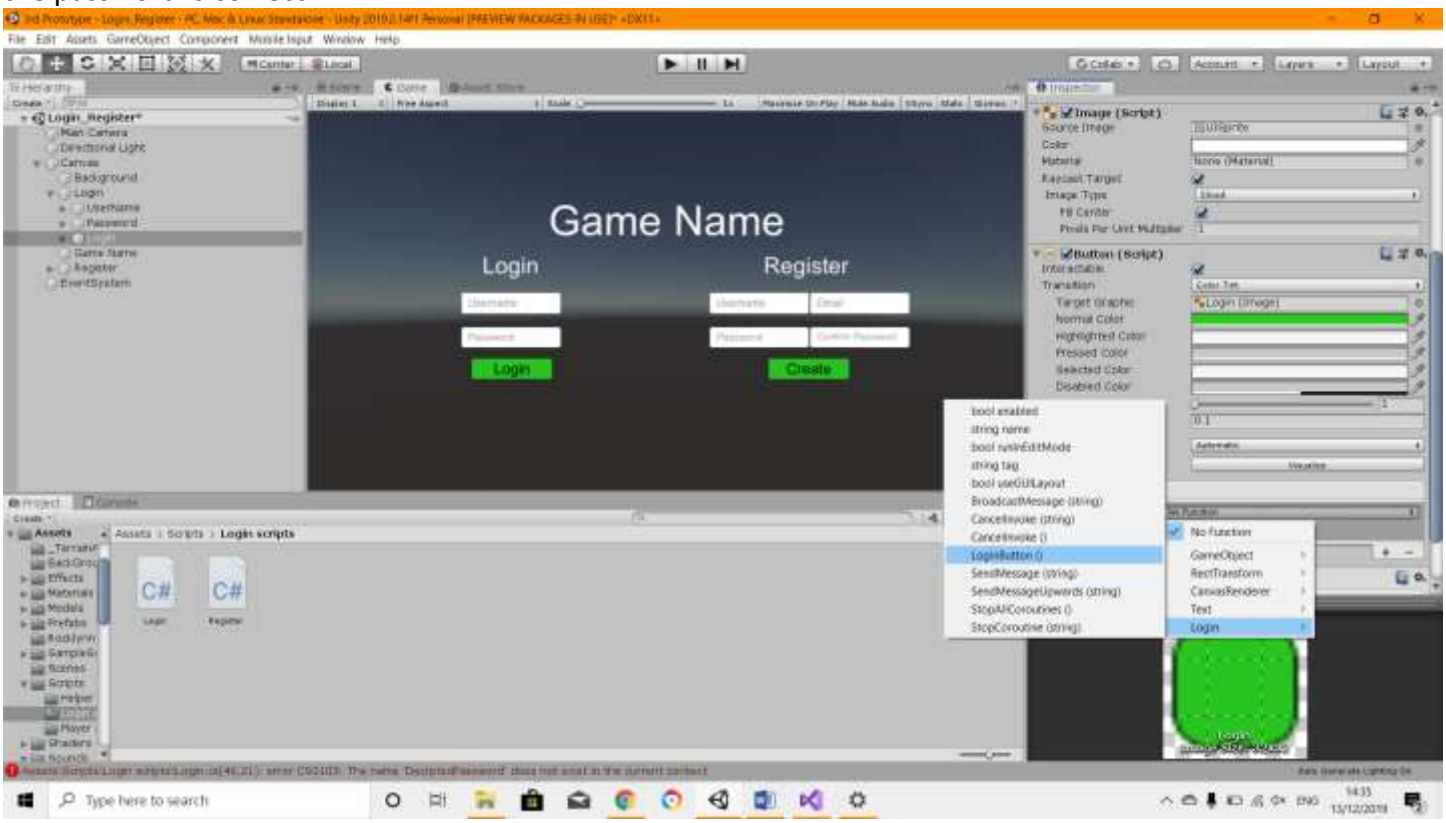
When opening the ".txt" file the new user's information was wrote on one line when it's supposed separated line by line. I initially added "/n" to the code to do this but it hadn't been successful therefore I changed it to the code above. Which fixed the error.



This showed the new spacing script worked correctly



Here I assigned the Username and Password to the components so they know what field to look in to check if the password is correct

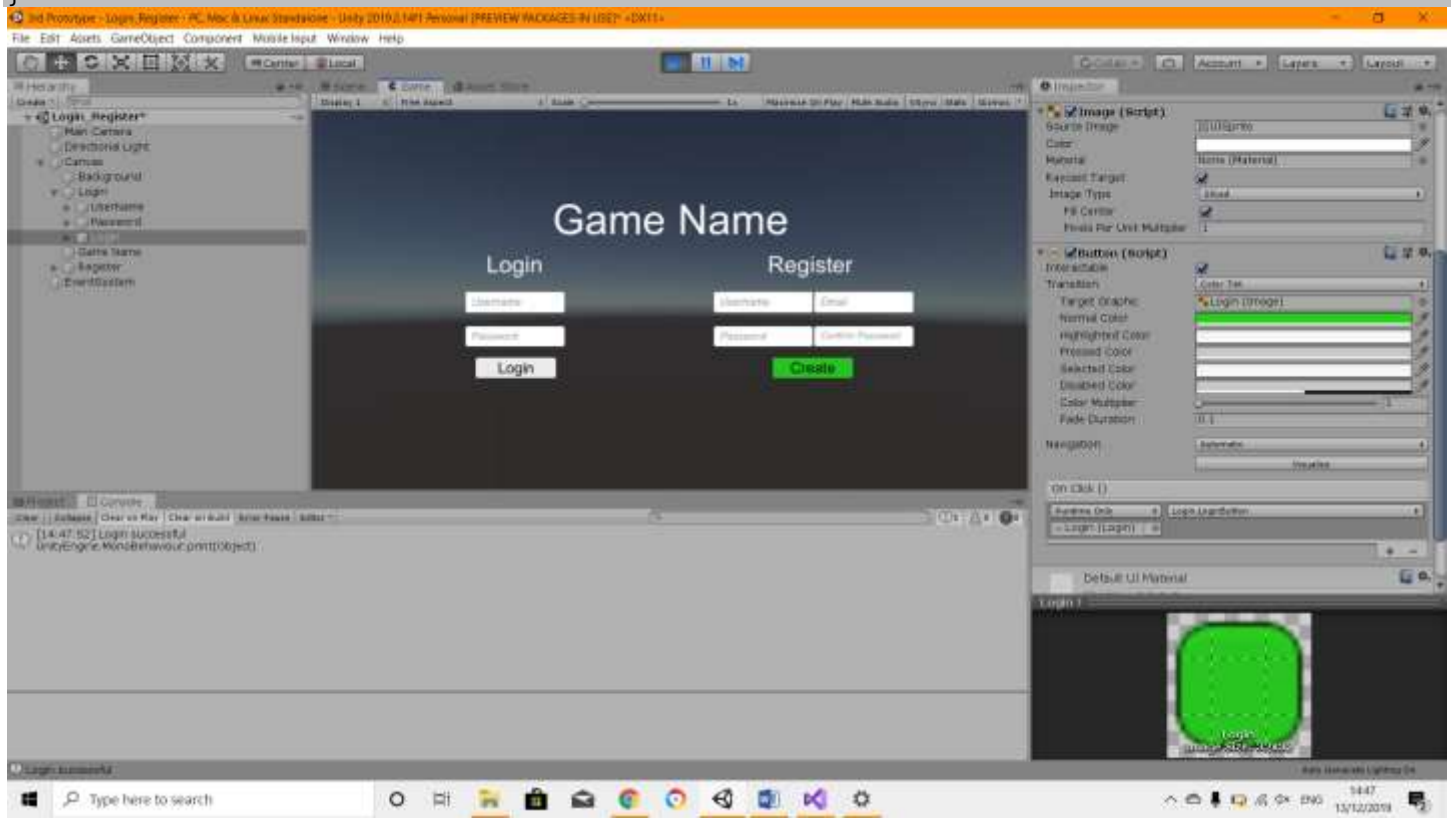


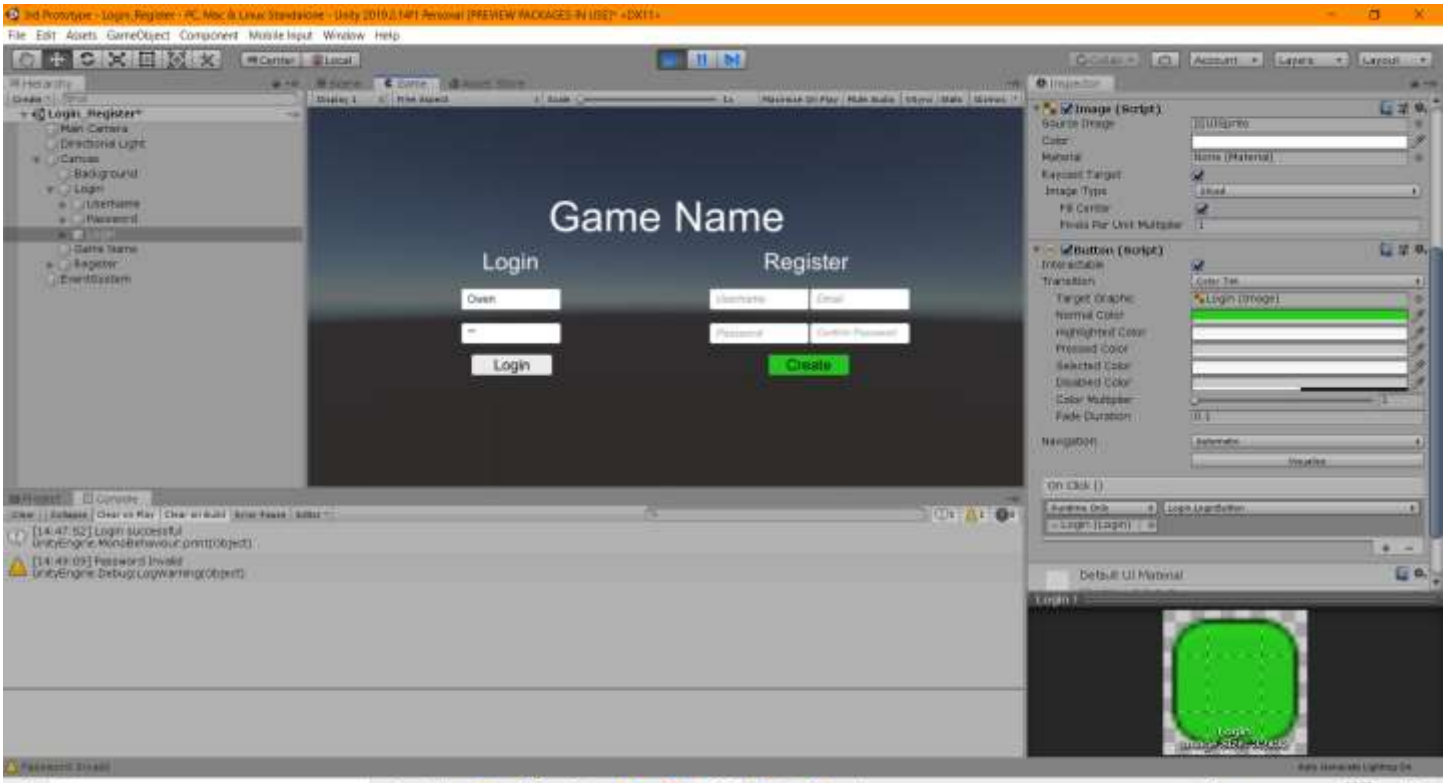
Here I assigned the button to the loginfunction() so when pressed that function would load

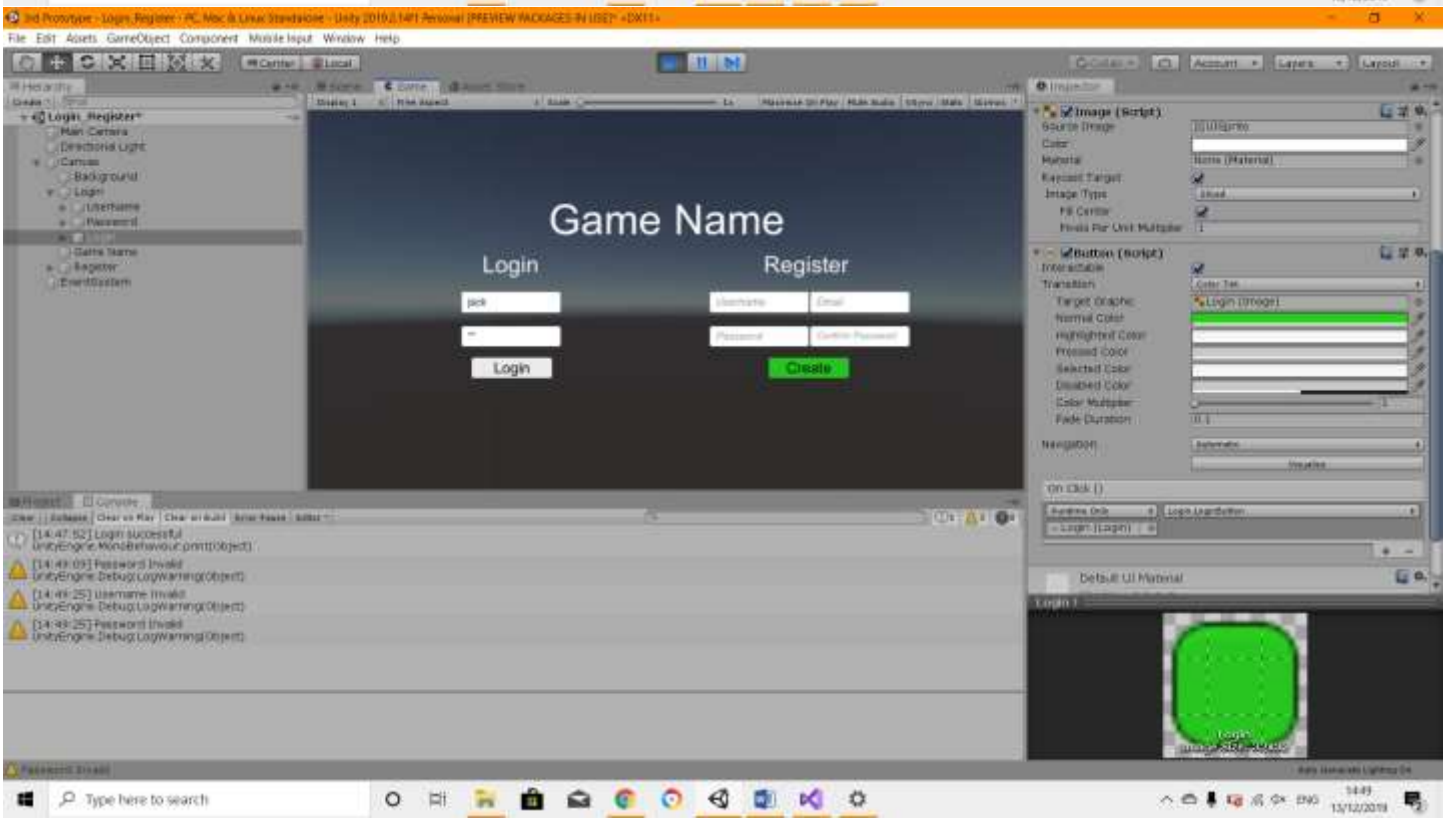
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;
using System;
using System.Text.RegularExpressions;

public class Login : MonoBehaviour
{
    public GameObject username; //Created a game object variable for username
    public GameObject password; //Created a game object variable for password
    private string Username; //Creates a string variable for Username but only allowed in that
class(Captial letter version). To hold the username of the player
    private string Password; //Creates a string variable for Password but only allowed in that
class(Captial letter version). To hold the password for a specific username
    private string[] Lines; //Creates an array that sores each line of data
    private string DecryptedPass; //Creates a string variable for the decrpted password

    public void LoginButton()
    {
        bool UN = false; //creates a decision variable UN for username. Checks if it passes all the
correct rules
        bool PW = false; //creates a decision variable PW for password. Checks if it passes all the
correct rules
        if (Username != "") //checks if the username filed is blank
        {
```

```
            if(System.IO.File.Exists(@"C:\Users\owenb\Documents\new computer\Computer Science\unity
projects\Game data\" + Username + ".txt")) //checks the file path to see if the username exists
            {
                UN = true; //sets the username "UN" to true meaning a username is found
                Lines = System.IO.File.ReadAllLines(@"C:\Users\owenb\Documents\new computer\Computer
Science\unity projects\Game data\" + Username + ".txt"); //Read's the lines of the speicified .txt file
line by line
            }
            else
            {
                Debug.LogWarning("Username Invalid"); //Tells the user that the username is incorrect
            }
        }
        else
        {
            Debug.LogWarning("Username Field Empty"); //Tells the user that the username field is
missing
        }
        if (Password != "")
        {
            if (System.IO.File.Exists(@"C:\Users\owenb\Documents\new computer\Computer Science\unity
projects\Game data\" + Username + ".txt")) //checks the file path to see if the username exists
            {
                int i = 1;
                foreach (char c in Lines[2]) //for each character in password
                {
                    i++; //iterates i by 1
                    char Decrypted = (char)(c / i); //Decryptes each letter of the password
                    DecryptedPass += Decrypted.ToString(); //re-puts it as a string
                }
                if (Password == DecryptedPass) //check's if the password entered is the correct one
                {
                    PW = true; //changes the variable Pw for password to true meaning the correct
password is true
                }
                else
                {
                    Debug.LogWarning("Password Invalid"); //Tells the user that the password is invalid
                }
            }
            else
            {
                Debug.LogWarning("Password Invalid"); //Tells the user that the password is invalid
            }
        }
        else
        {
            Debug.LogWarning("Password Field Empty"); //Tells the user that the password field is blank
        }
        if (UN == true && PW == true) //checks if the password and username variables were both set to
true
        {
            print("Login successful"); //prints to the screen that the login was successful
            username.GetComponent<InputField>().text = ""; //Get's the username entered in the username
component assighning it to the Username variable
            password.GetComponent<InputField>().text = ""; //Get's the password entered in the username
component assighning it to the Password variable
        }
```

```
    }
    // Start is called before the first frame update
    // Update is called once per frame
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Tab)) //checks if the Tab button is pressed down
        {
            if (username.GetComponent<InputField>().isFocused) //checks if the username component is
selected
            {
                password.GetComponent<InputField>().Select(); //Select's the email field
            }
        }
        if (Input.GetKeyDown(KeyCode.Return)) //checks if the Return button is pressed down
        {
            if (Password != "") //Checks if the password field is blank
            {
                LoginButton(); //goes to the LoginButton function
            }
        }
        Username = username.GetComponent<InputField>().text; //Get's the username entered in the
username component assighning it to the Username variable
        Password = password.GetComponent<InputField>().text; //Get's the password entered in the
username component assighning it to the Password variable
    }
}
```

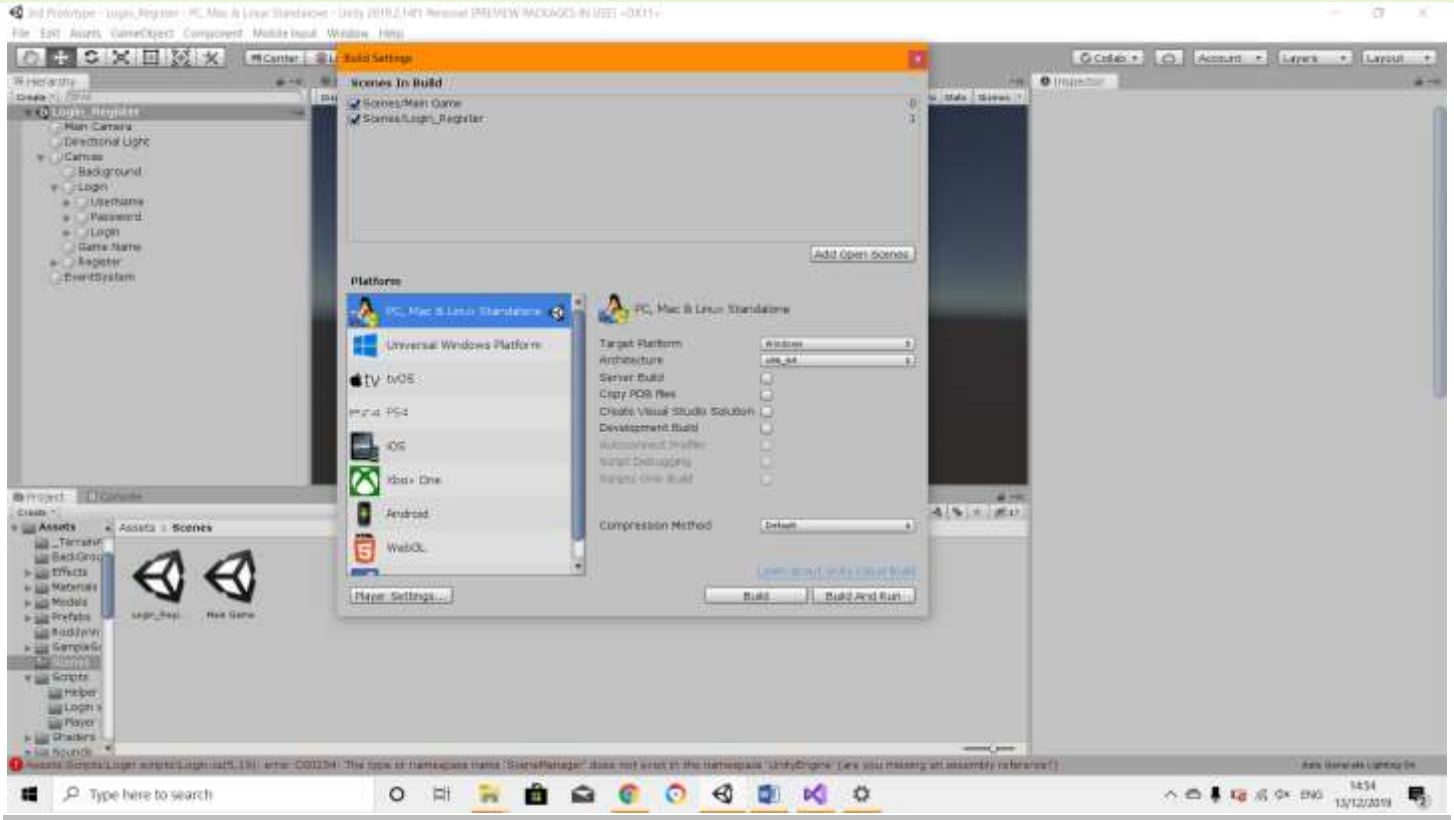



Game Name
Login
Register
Owen
Login
Create
[14:47:52] Login Successful
[14:49:09] Password Invalid


Game Name
Login
Register
jack
Login
Create
[14:47:52] Login Successful
[14:49:09] Password Invalid
[14:49:25] Username Invalid
[14:49:25] Password Invalid

This is the script I created for the login.cs() and the screen shots of me testing if any errors occurred which did not happen. It would give me a warning if the username is not valid or the password typed in being incorrect however if both were correct a print message would appear showing a successful
login and the username and password field to be deleted. To do this I had to decrypt the password as well as read line by line specified .txt files to make this code work

SECOND PROTOTYPE -LOGIN SYSTEM-LOAD SCENE



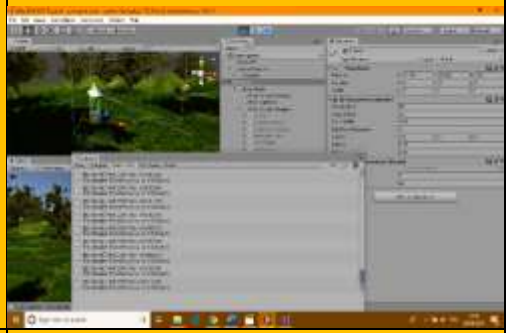

```
using UnityEngine.SceneManagement;
SceneManager.LoadScene(0);
```

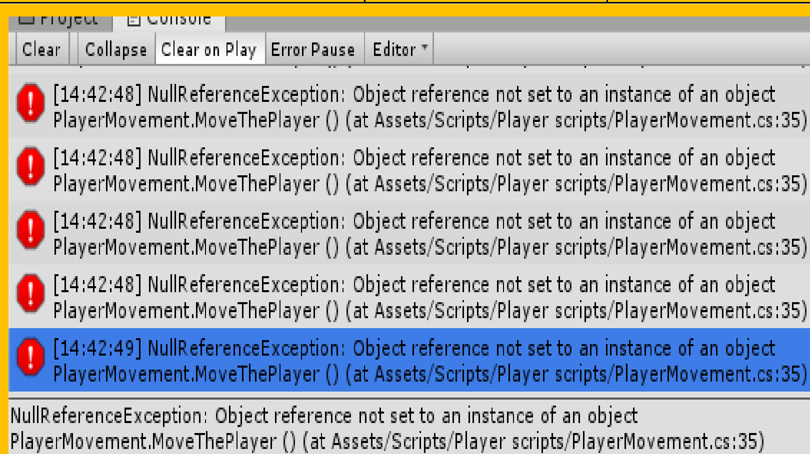
I added this piece of code, so that when the login is successful, or a new account is made the actual game scene is loaded up straight away.

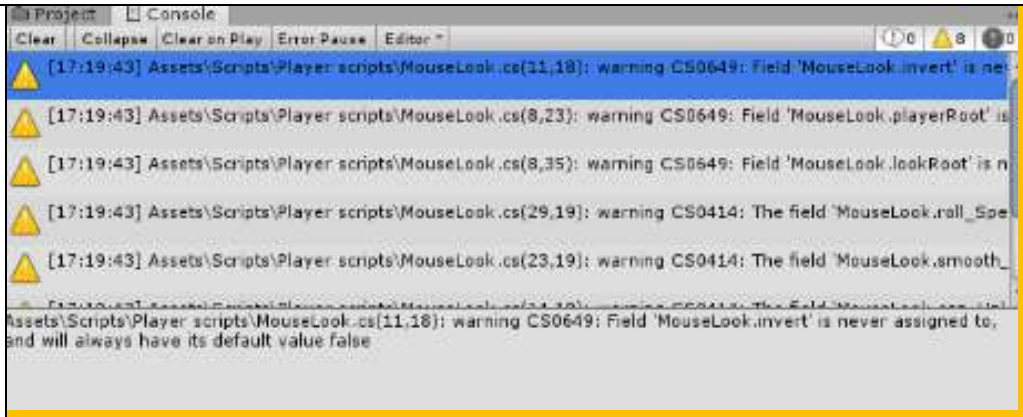
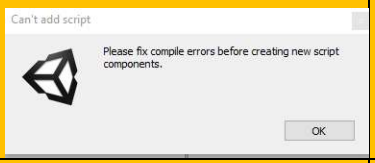
Second Prototype -Testing table

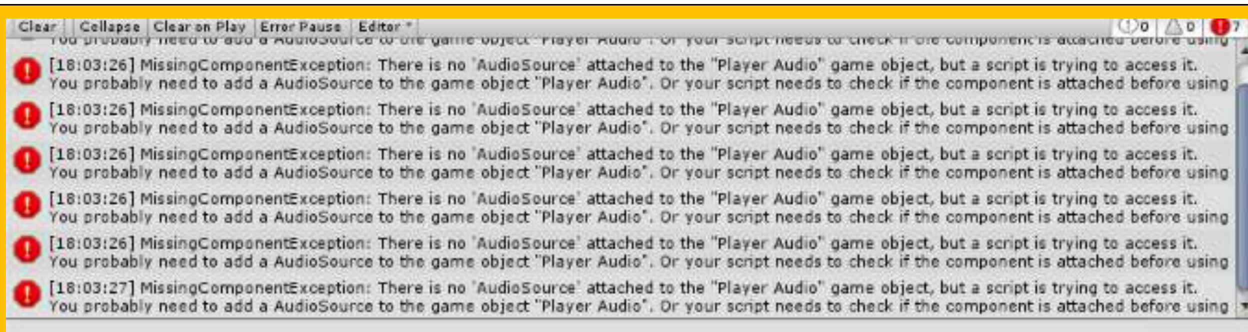
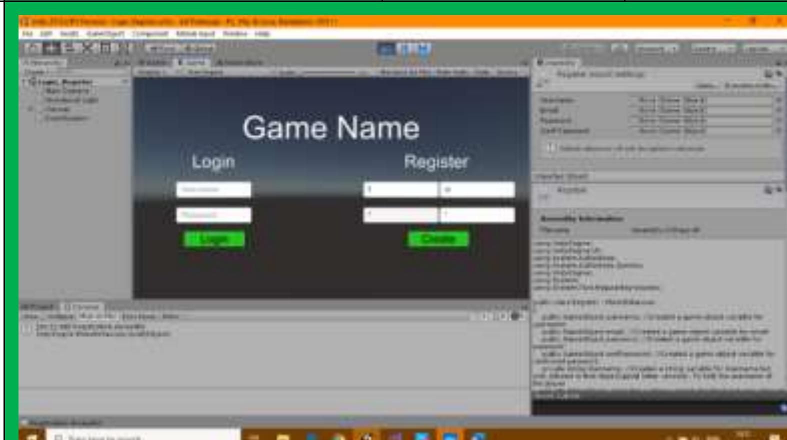
11a.-Login	Password replaces characters with “*”	Yes	No
12a.-Register	Password replaces characters with “*”	Yes	No
13a.-Login	Username has 17 character max limit	Yes	No
14a.-Register	Username has 17 character max limit	Yes	No
15a.-Login	Email has 17 character max limit	Yes	No
16a.-Register	Email has 17 character max limit	Yes	No

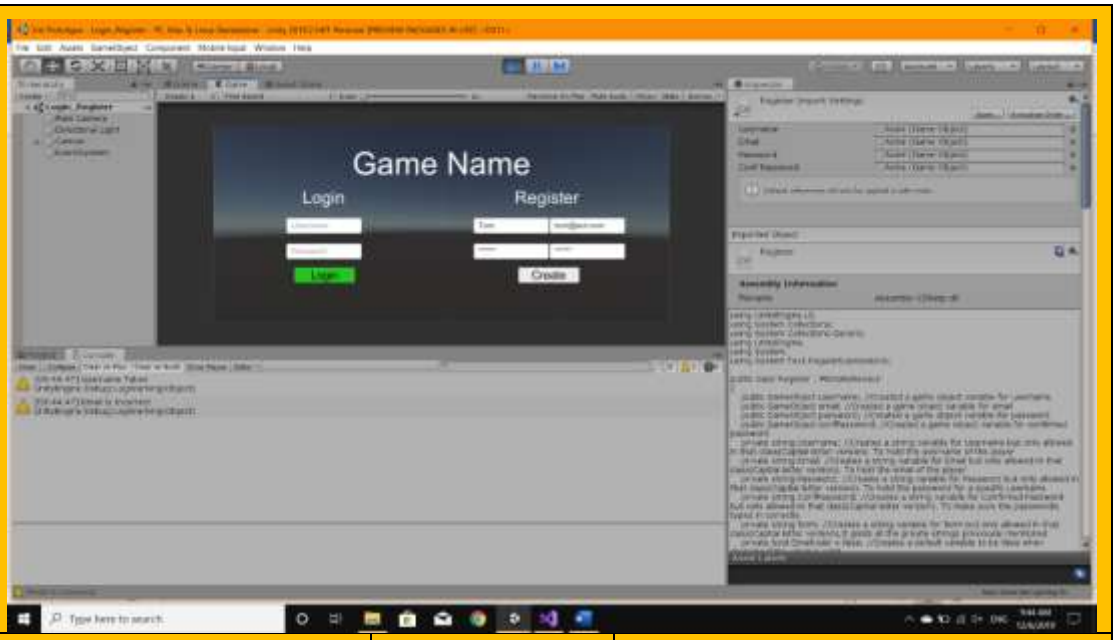
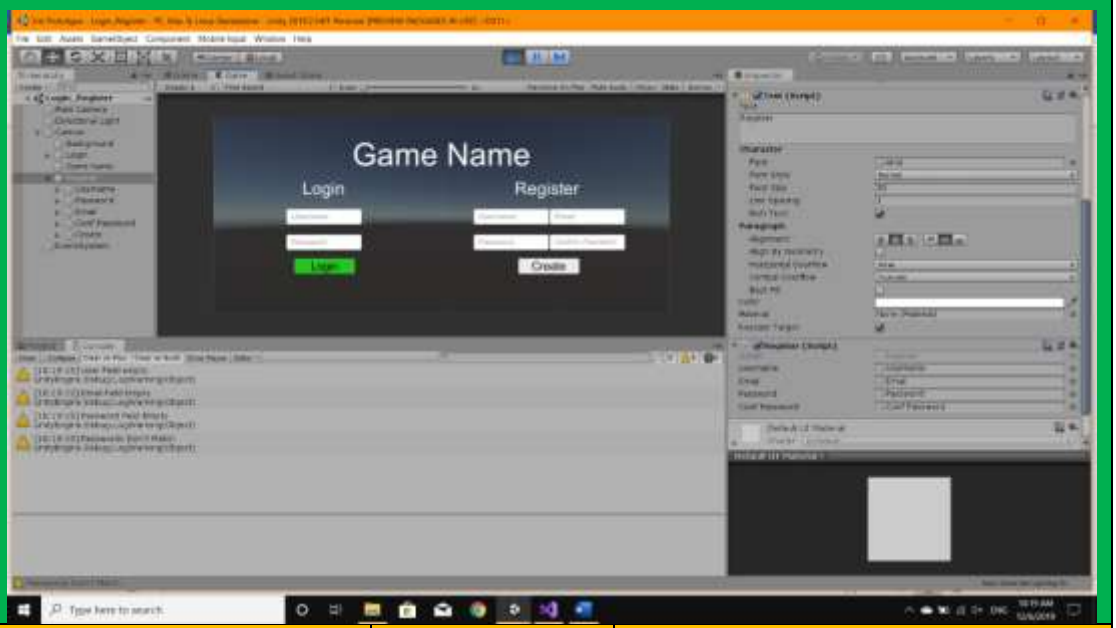
17a.-Password confirmed	The password entered in is the same as the confirmed password	Yes	No
18a.-Password length	The password entered in has a max length of 6 characters	Yes	No
19a.-Username Unique	Username hasn't already been used	Yes	No
20a.-Correct Email	Email contains "@" and "."	Yes	No
21a.-Login	Login Button loads scene	Yes	No
22a.-Login	Create button creates and writes new username details and loads into scene	Yes	No
22a.-Map	Water physics	Partially	-next test
23a.-Sprint	Decrease the speed of the player. After 1A user's feedback	Yes	No
24a.-Yield Of trees	Created paths in the island and made the yield of trees a lot smaller therefore not as dense. After 1A user's feedback	Yes	No

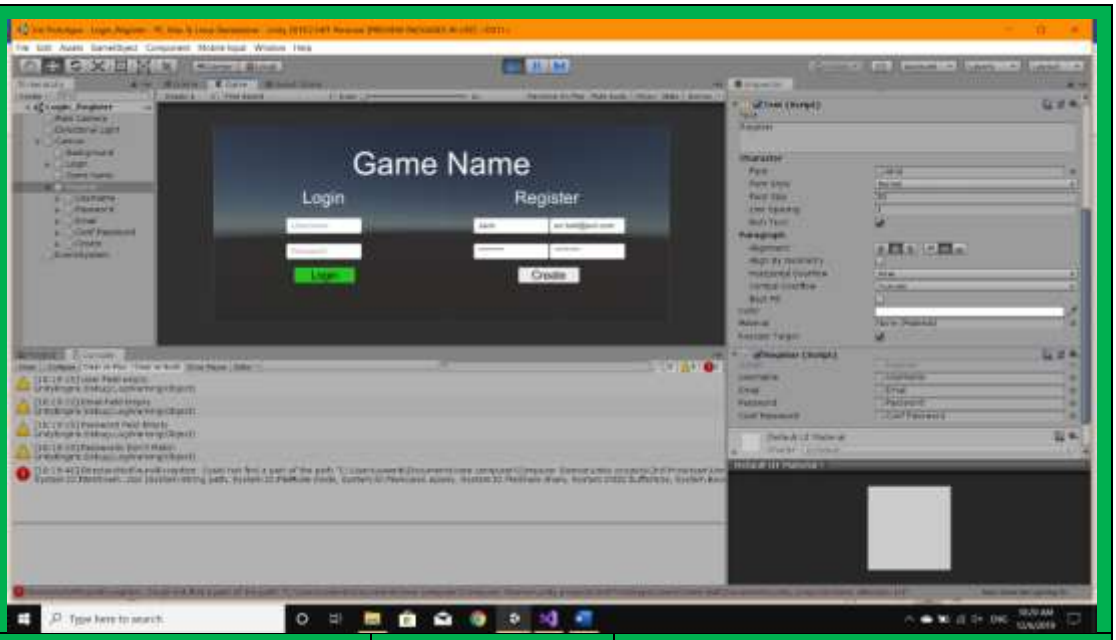
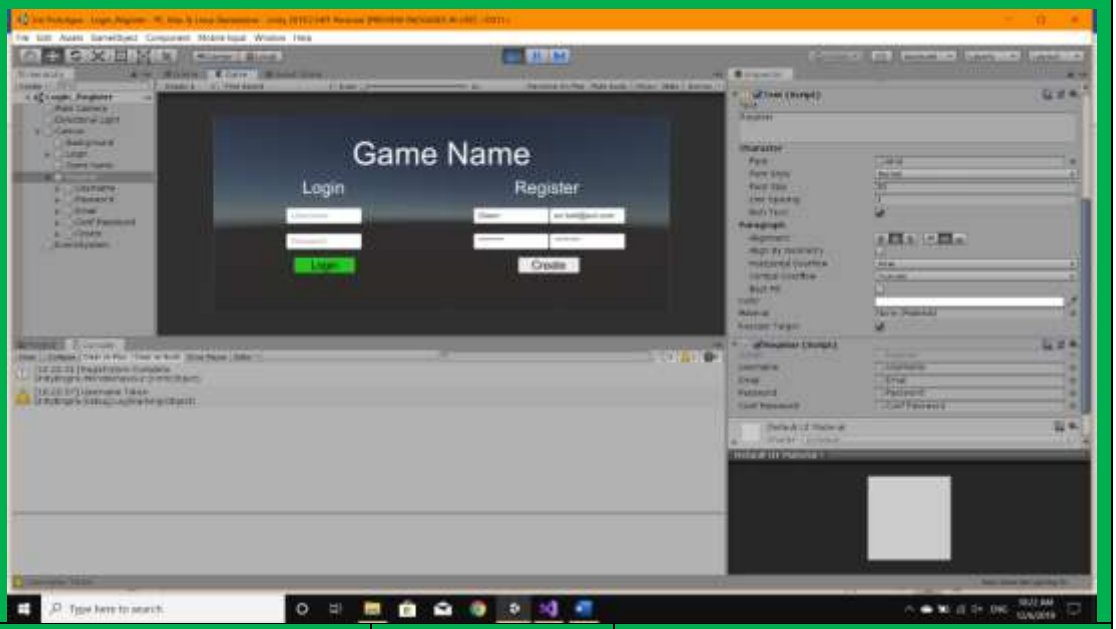
1b.	Player Movement.cs ()-general movement	1-error occurring due to an unneeded bracket 2 errors remaining See Below:	Remove the unneeded bracket	Yes 2b for the next test
				
2b.	Player Movement.cs ()-general movement	1-unlinked script so action was occurring 1 error remaining	Drag and drop the script into the player folder attached on the game	Yes, 3b for the next test 
3b.	Player Movement.cs	1-error occurring due to "Object not referenced"	Check Grammar on variables	Yes- after relooking over the C# code I realised I didn't capitalize the "Awake"

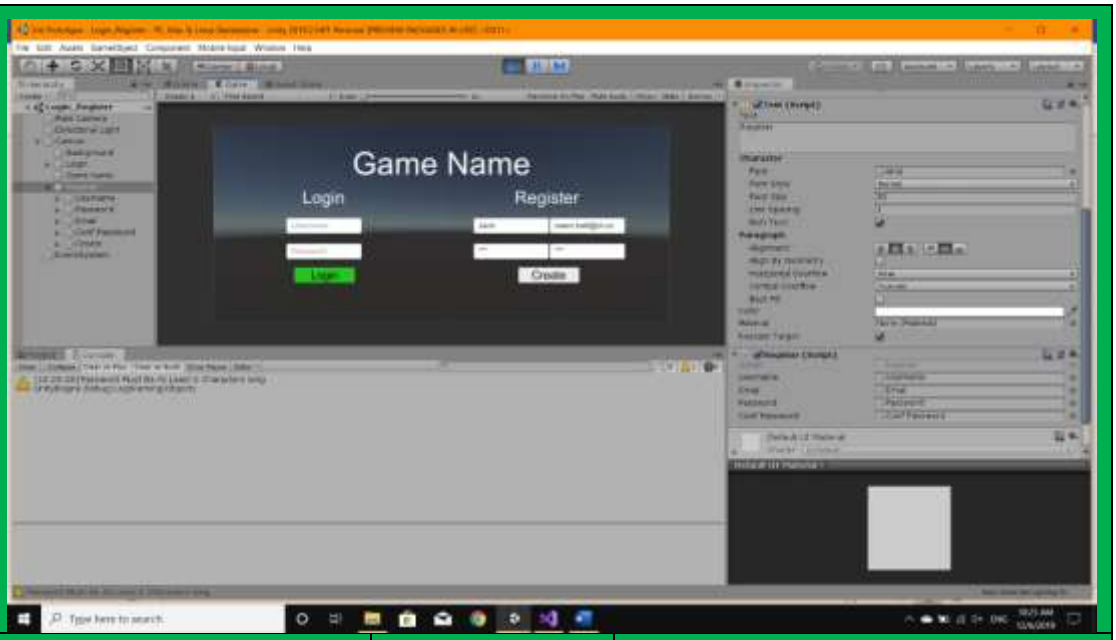
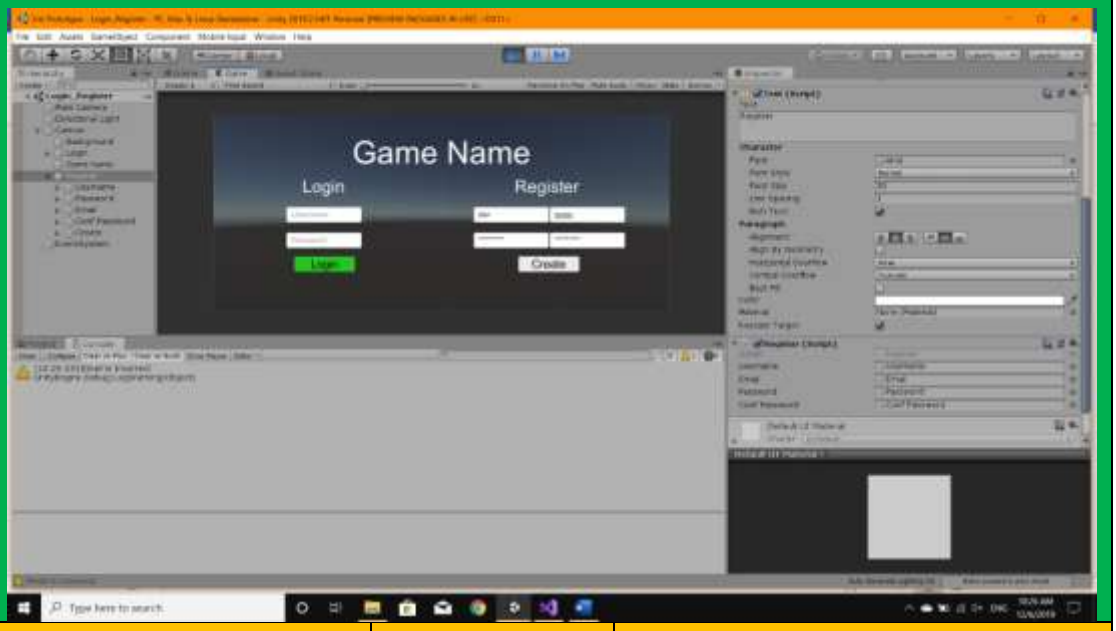
	()-general movement	See Below:		function causing the error to occur. 4b for the next test
				
4b.	Player Movement.cs ()-general movement	Yes	N/A	N/A
5b.	Player Movement.cs ()-Gravity	1-Falling through the floor of the map 	Re-look over the code make sure gravity is assigned and all variables are spelt correctly	Yes-This issue was fixed by adding a reference to the gravity function (due to it not being called so the gravity script wouldn't work properly). 6b for the next test 
6b.	Player Movement.cs ()-Gravity	Yes	N/A	N/A
7b.	Mouse Look.cs()-General Movement	Multiple-Variables assigned but never used error See Below:	Check all spelling on variables	Yes- all down to capitalised letters needing to be lower case and forgetting the "f" after a number (0f) with certain commands. 8b for the next test

				
8b.	Mouse Look.cs()- General Movement	Yes	N/A	N/A
9b.	Mouse Look.cs()- Developer “esc button”	Yes	N/A	N/A
10b.	Sprint Crouch.cs()- Crouch “c”	Yes	N/A	N/A
11b.	Player Foot Steps.cs()- Sound effects for the player moevment	1-Can’t add script error 2 Remaining error 	Remove the spacing in the text of the files name	Yes- due to the name of the C# having a space in it but after deleting it and renaming it without any spaces the error disappeared. 12b for the next test
12b.	Player Foot Steps.cs()- Sound effects for the player moevment	1-“Assets\Scripts\Player scripts\SprintCrouch.cs(18,13): error CS0246: The type or namespace name 'PlayerFootsteps' could not be found (are you missing a using directive or an assembly reference?)” 1 remaining error	Need to relook over my full code step by step due not having an idea what this could be	Yes-After relooking at my code I realised I haven’t made the void function for “CheckToPlayFootsepSound.cs” I put all the code that should of been in this in the update function. This therefore fixed this error. 13b for the next test
13b.	Audio Source	1-“There is no audiosource attached to the player Audio” See below:	I needed to place the “audioscource” component in the player audio section and untick everything which is set ticked as default.	Yes-I replaced the component which was set to a default as empty with the correct audio source components and the error was removed.14b for the next test

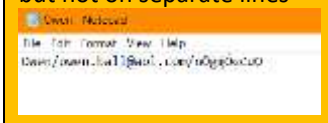
				
14b.	Audio Source	Yes	N/A	N/A
15b.	Check for any warning error's	1-in the console that two "Audio listeners" were active when only one is needed	Find both ticked boxes and de-tick the necessary one	Yes- I found out I hadn't unticked the main camera one which then removed the error. 16b for the next test
16b.	Audio Source	Yes	N/A	N/A
17b.	Register.cs()-Login	1-I gained this error "UnassighnedReferenceException: The variable username of Register has not been assighned"	I need to check they I have assigned all the correct components	Realise that I haven't assigned my variables to my components therefore I dragged the correct one to each variable and this fixed the error and my tabbed function worked.18b for next test
18b.	Register.cs()-Login	Yes	N/A	N/A
19b.	Register.cs()-Check for any blank fields	Yes See Below:	N/A	N/A
				
20b.	Register.cs()-Correct details	1-"UserName Taken" 1-Remaining error "Email Incorrect" :See Below	Re-look at the code	Yes-The first error to do with username already taken was due to their being no "!" in front of the System.IO.FileExists causing this error to occur. I corrected this as seen above.21b for the next test.

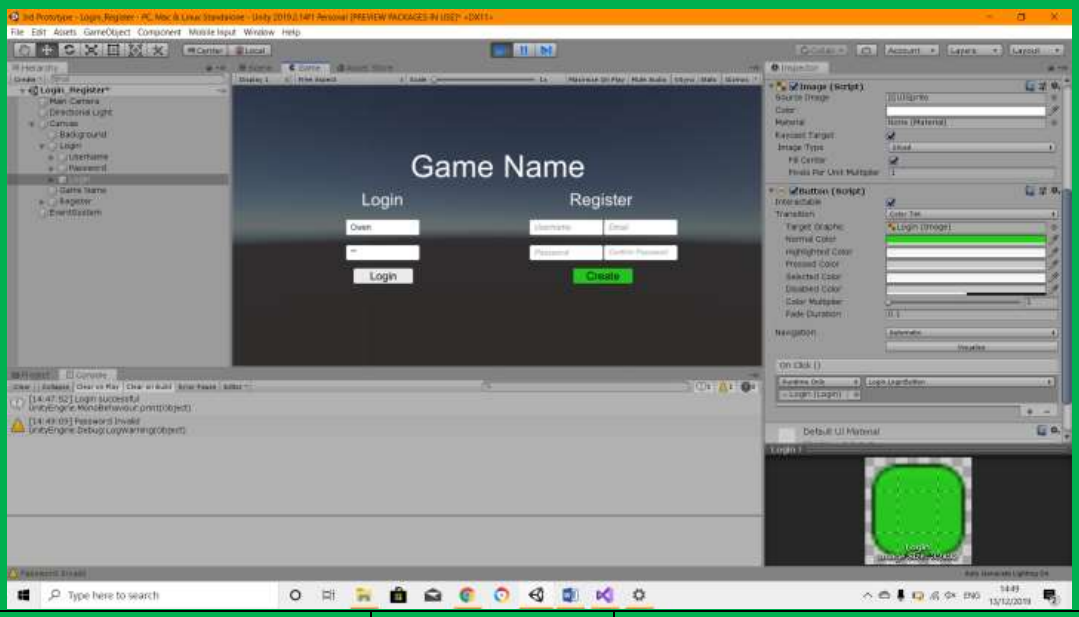
				
21b.	Register.cs()-Correct Details	1-"Email incorrect"	Re-look at the code	Yes-After re-looking over my code I realised forgot to put in the array to find which character represents each number and put the full validation for the email. 22b for the next test
22b.	Register.cs()-Check Fields are blank warning	Yes See Below:	N/A	N/A
				
23b.	Register.cs()-Correct Details	1-Could not fin the correct file path	Need to re-look at file paths	Yes- This was due to me having a new laptop so a new file path was needed

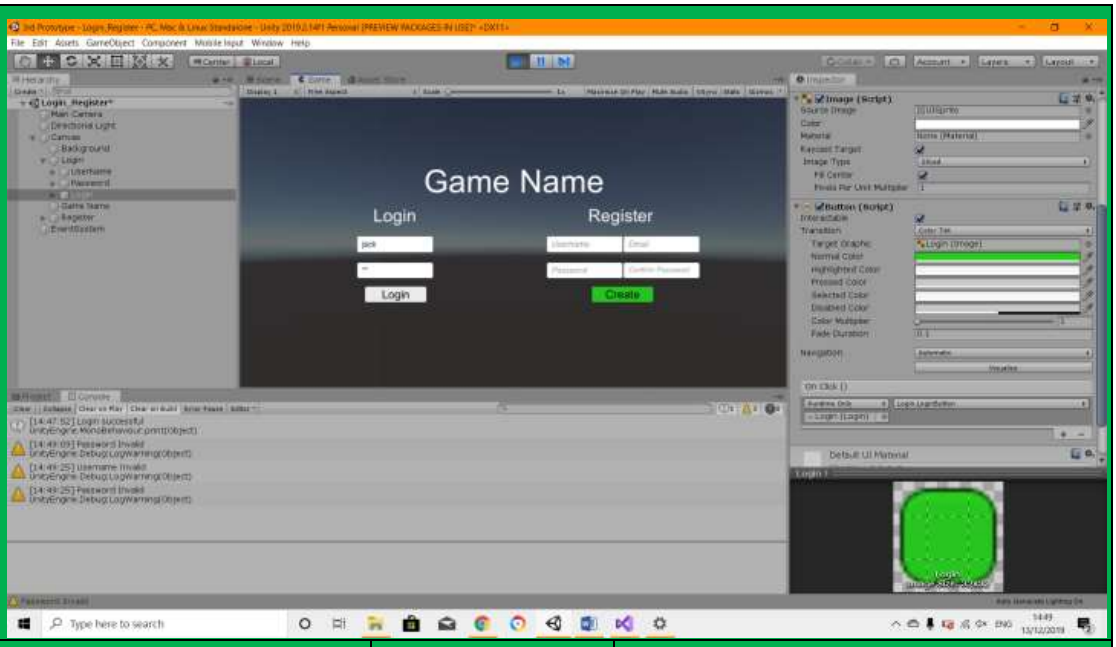
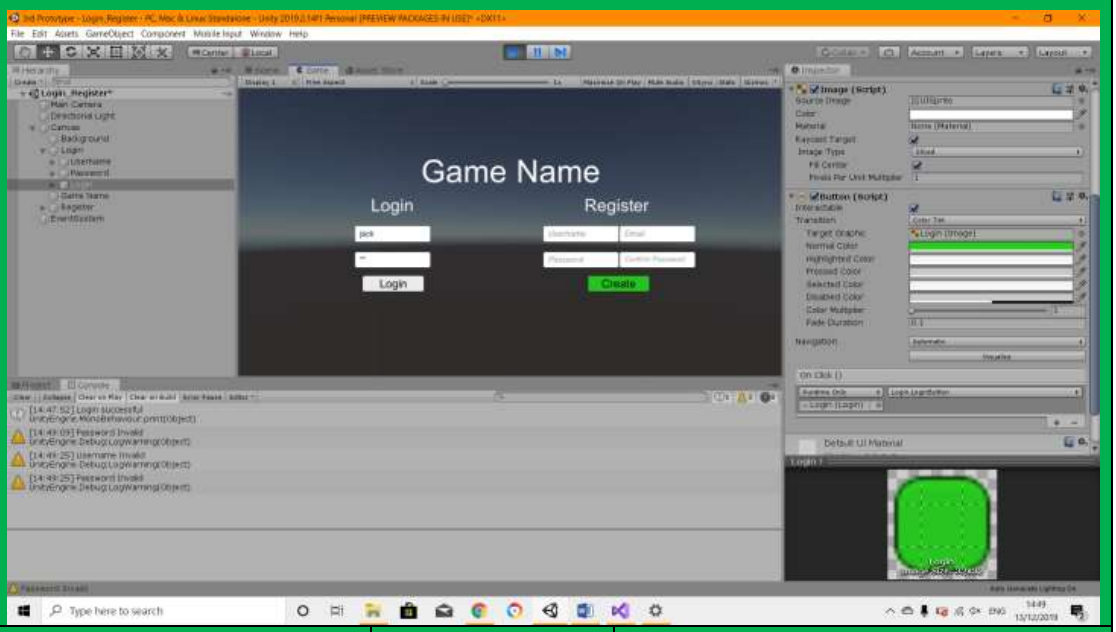
				
24b.	Register.cs()-Username warning check	Yes See Below:	N/A	N/A
25b.				
26b.	Register.cs()-Character length warning check	Yes See Below:	N/A	N/A

27b.				
28b.	Register.cs()- Character length warning check	Yes See Below:	N/A	N/A
29b.				
30b.	Register.cs()- Password equalling the same warning check	1-Won't login but no error occurs	Check if everything's spelt correctly when giving the warning error	Yes- it was missing from the code. Test for 31b
31b.	Register.cs()- Password equalling the	Yes See Below:	N/A	N/A

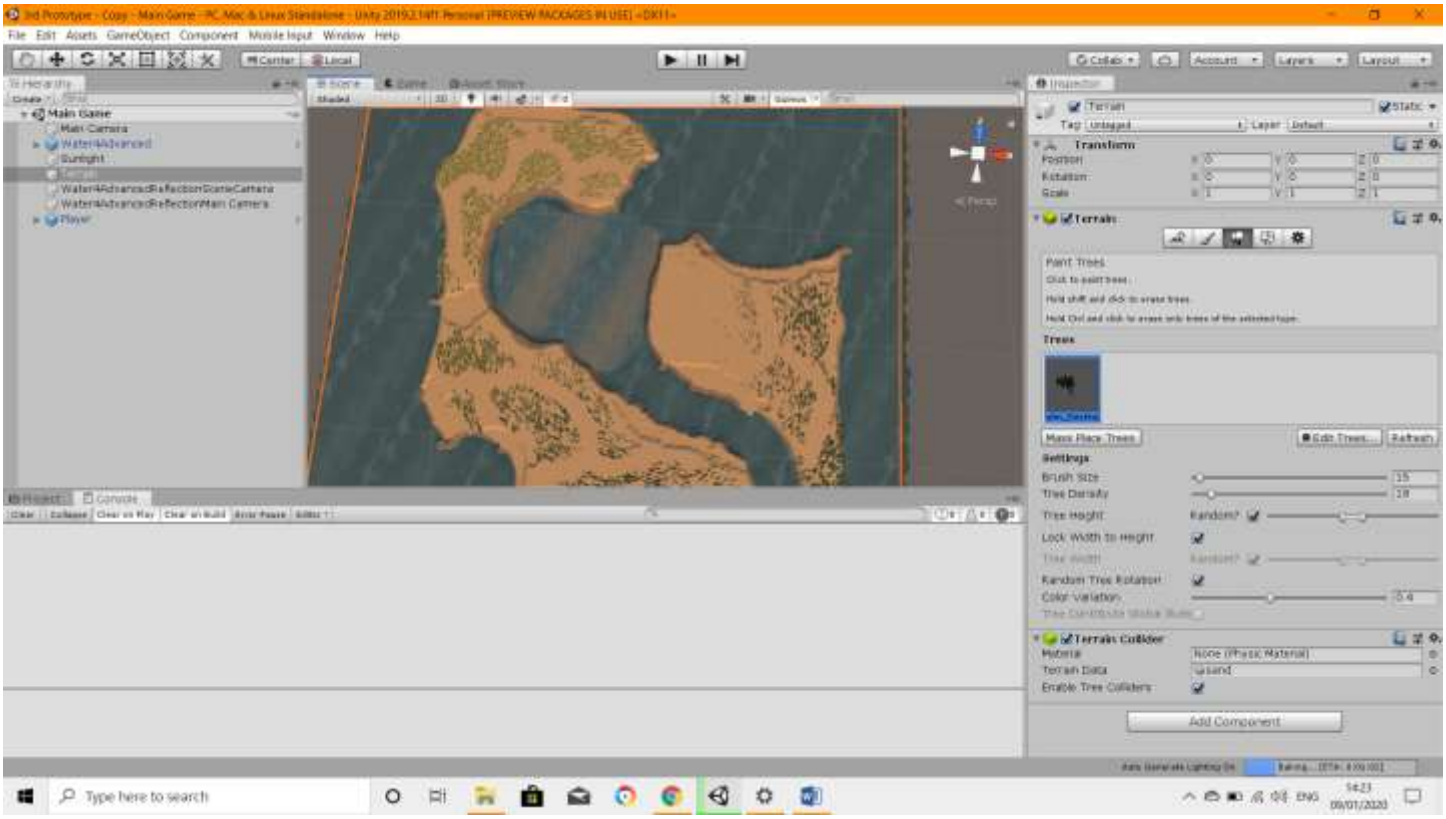
	same warning check			

32b.	User's details-.txt form separate lines	1-Writes in the correct details but not on separate lines 	Amend the piece of code to write each piece of information onto a separate line in the login.cs() script	Yes-Due to an update on unity my old method of writing on separate lines no longer worked. Therefore I amended "form = (Username +Environment.NewLine+ Email + Environment.NewLine + Password); //concatinates all the data for the new created user
				"to the login.cs() script. Test for 33b

33b.	User's details-.txt form separate lines	Yes 	N/A	N/A
34b.	Login.cs()-Password incorrect	Yes See Below:	N/A	N/A
				
35b.	Login.cs()-Usernames incorrect	Yes See Below:	N/A	N/A

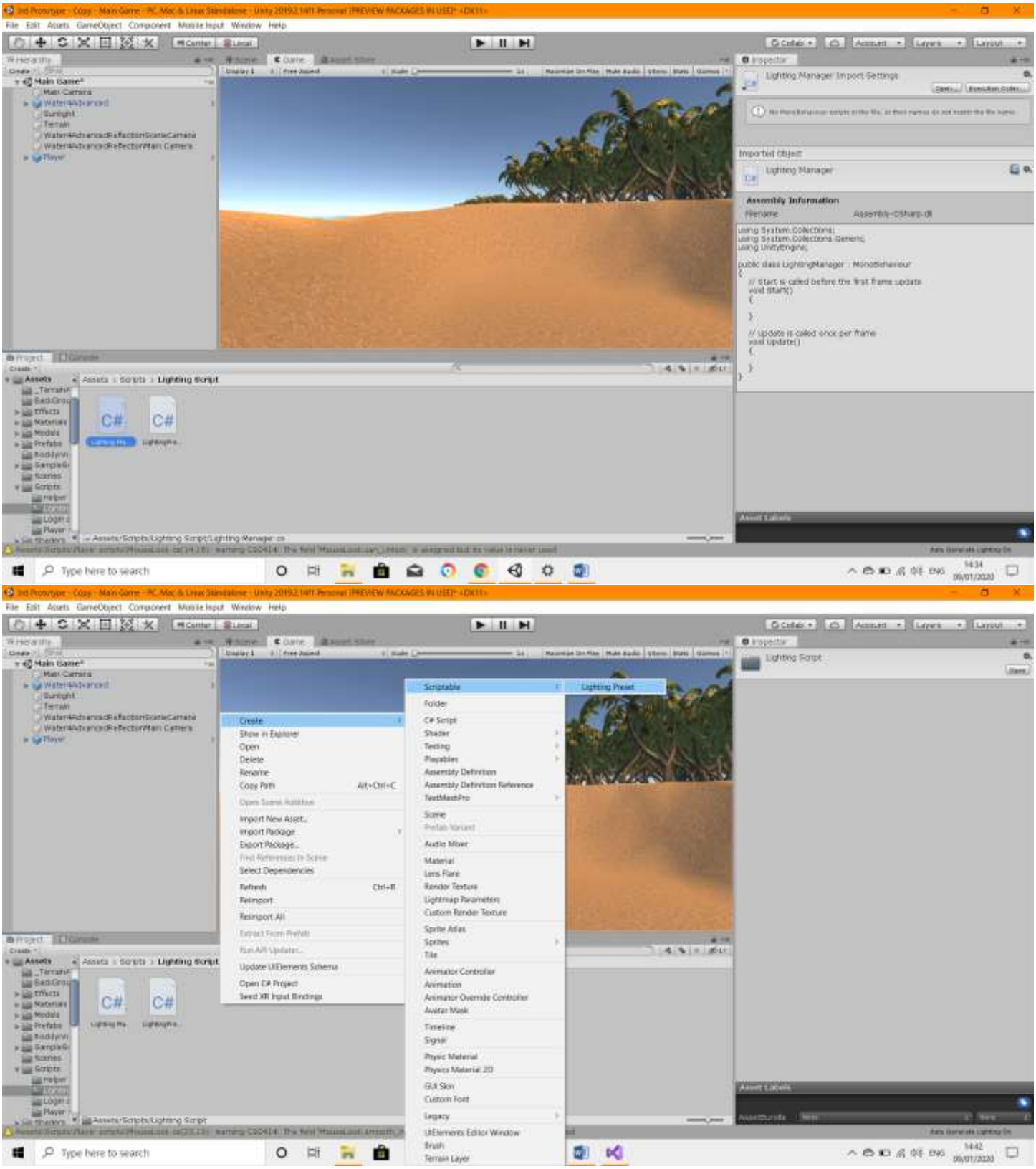
				
35b.	Login.cs()- Usernames and password correct	Yes See Below:	N/A	N/A
				

THIRD PROTOTYPE- TOO MANY TREE'S-STAKEHOLDER ISSUE



After addressing my issue to do with too much tree's in the game I decided to create paths making it easier to move around. This was highlighted by one of my stakeholders Luke Callison Bailey

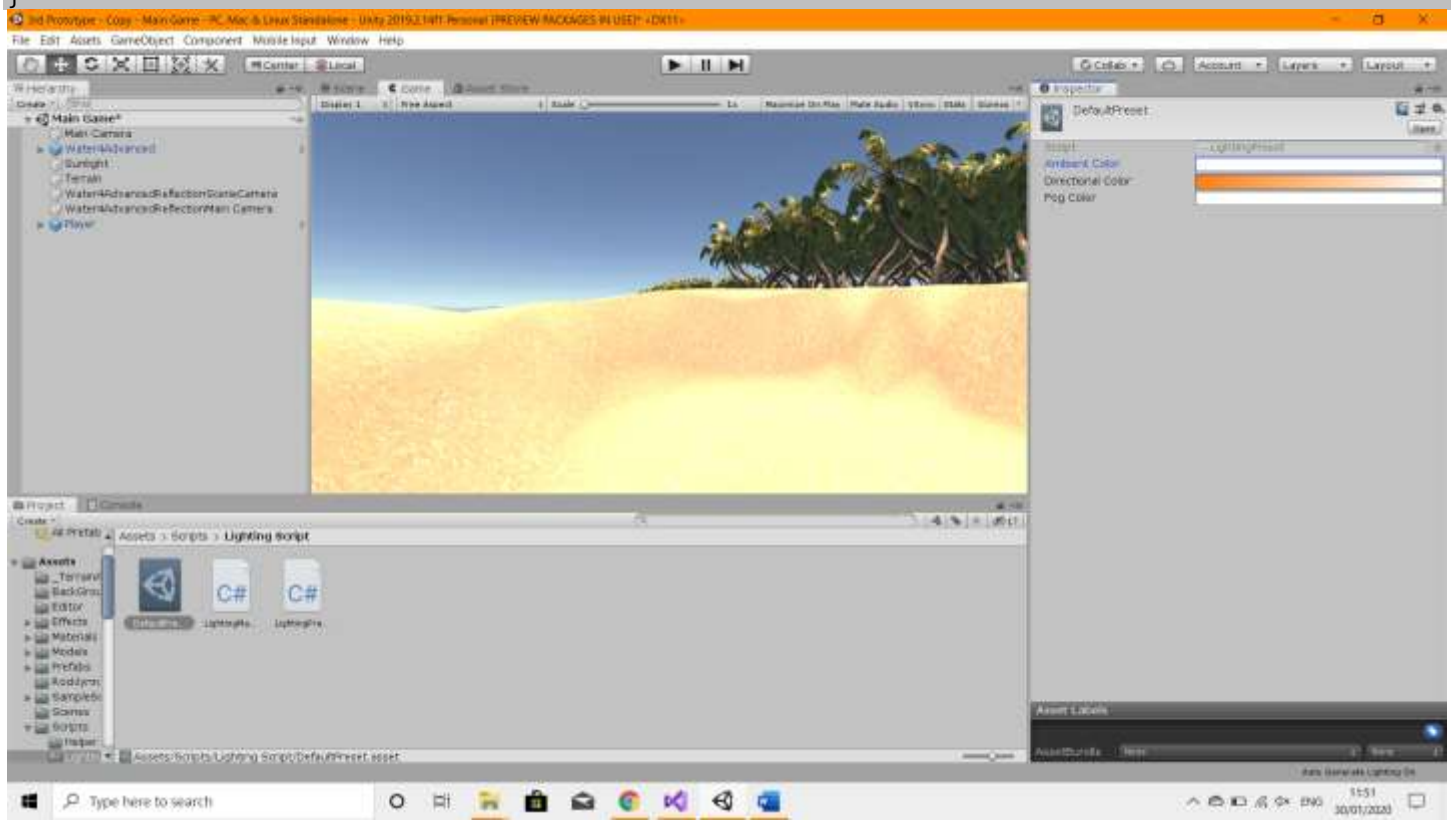
THIRD PROTOTYPE -DAY AND NIGHT CYCLE



I wanted to make a day and night cycle. This was created with a new folder in the Assets folder called “Lighting

Script” as well as 2 new 2 c# scripts called “Lighting Manager” and “Lighting Preset”. I also added the lighting preset to the same folder calling it “Default Preset”.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[System.Serializable]
[CreateAssetMenu(fileName = "Lighting Preset", menuName = "Scriptable/Lighting Preset", order = 1)]
//creates an instance of the class
public class LightingPreset : ScriptableObject
{
    public Gradient AmbientColor; //allows the ambient colour to be edited in the unity editor mode
    public Gradient DirectionalColor; //allows the directional colour to be edited in the unity editor
mode
    public Gradient FogColor; //allows the fog colour to be edited in the unity editor mode

}
```



This is the code I used for my preset default lighting

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

[ExecuteAlways] //executes certian methods methods whilst we aren't in the game
public class LightingManager : MonoBehaviour
{
    [SerializeField] private Light DirectionalLight; //a reference to directional light
    [SerializeField] private LightingPreset Preset; //a reference to the preset
    [SerializeField, Range(0, 24)] private float TimeOfDay; //Creates a variable for the TimeOfDay
ranging between 0 and 24
```

```
    private void Update()
    {
        if (Preset == null) //checks if preset has been assighned
            return;
        if (Application.isPlaying)
        {
            TimeOfDay += Time.deltaTime;
            TimeOfDay %= 24; //clamp between 0-24
            UpdateLighting(TimeOfDay / 24f);
        }
        else
        {
            UpdateLighting(TimeOfDay / 24f);
        }
    }
    private void UpdateLighting(float timePercent) //Evaluates all the different gradients in the
preset to the correct values
    {
        RenderSettings.ambientLight = Preset.AmbientColor.Evaluate(timePercent);
        RenderSettings.fogColor = Preset.FogColor.Evaluate(timePercent);
        if (DirectionalLight != null)
        {
            DirectionalLight.color = Preset.DirectionalColor.Evaluate(timePercent);
            DirectionalLight.transform.localRotation = Quaternion.Euler(new Vector3((timePercent *
360f) - 90f, 170f, 0));
        }
    }
    private void OnValidate()
    {
        if (DirectionalLight != null) //checks if their is any value in Directional light
            return;
        if (RenderSettings.sun != null) //checks if their is any value in Render settings
        {
            DirectionalLight = RenderSettings.sun; //Changes the directional light to the settings
displayed in the sun
        }
        else
        {
            Light[] lights = GameObject.FindObjectsOfType<Light>(); //gets loads the gameobject light
            foreach (Light light in lights) //Goes throigh every light in lights
            {
                if (light.type == LightType.Directional) //Checks if the light type is directional
                {
                    DirectionalLight = light; //chnages the variable DirectionalLight to light
                    return;
                }
            }
        }
    }
}
```

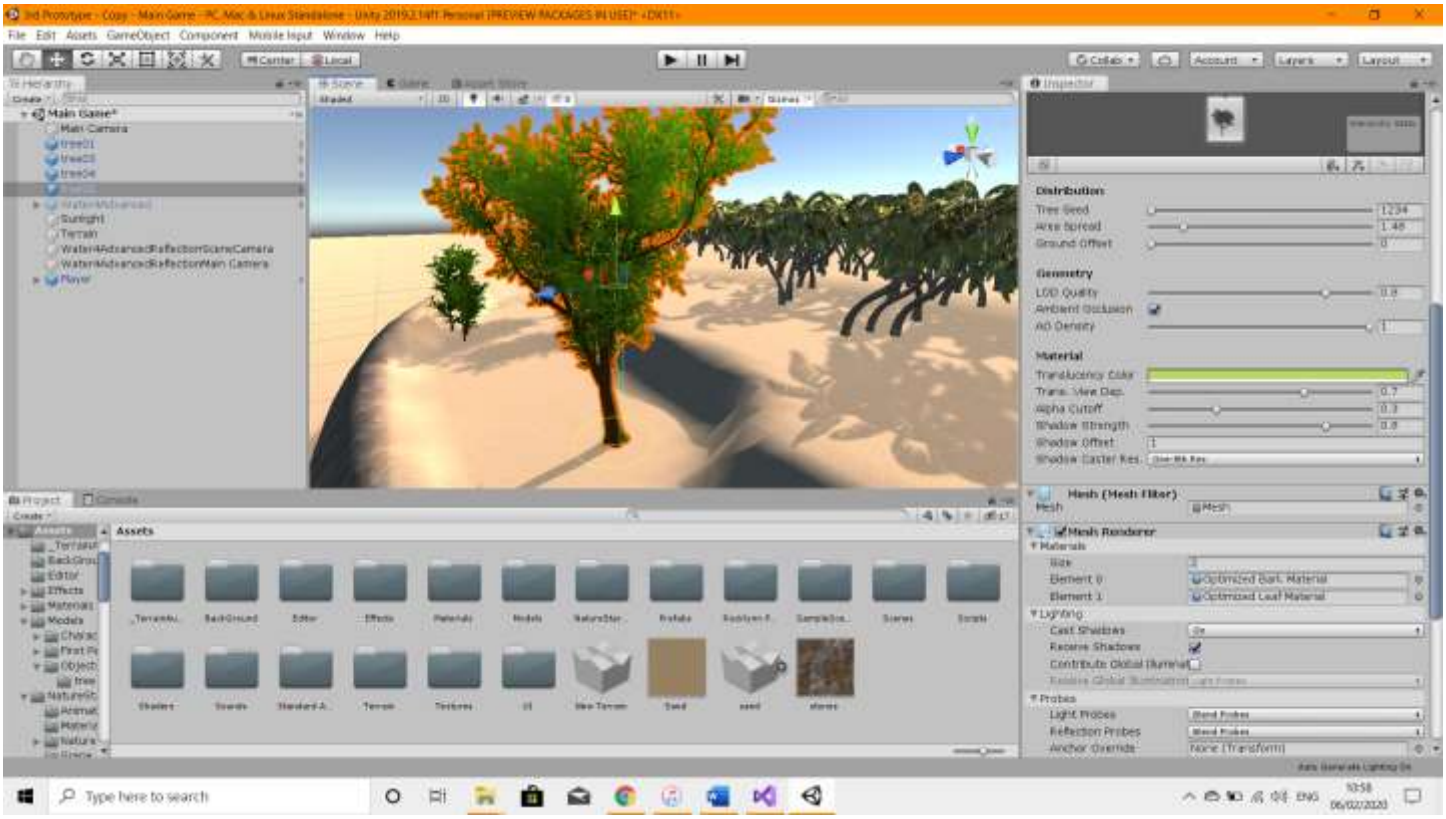
This is the c# for the lighting management

```
TimeOfDay += (Time.deltaTime/25); //updates and interates TimeOfDay to allow the cycle to update
```

The day night cycle was going too fast to I asked my stakeholder how long a day should last in real time and edited it so that 24 seconds divided by 25 will make a day last 10 mins.

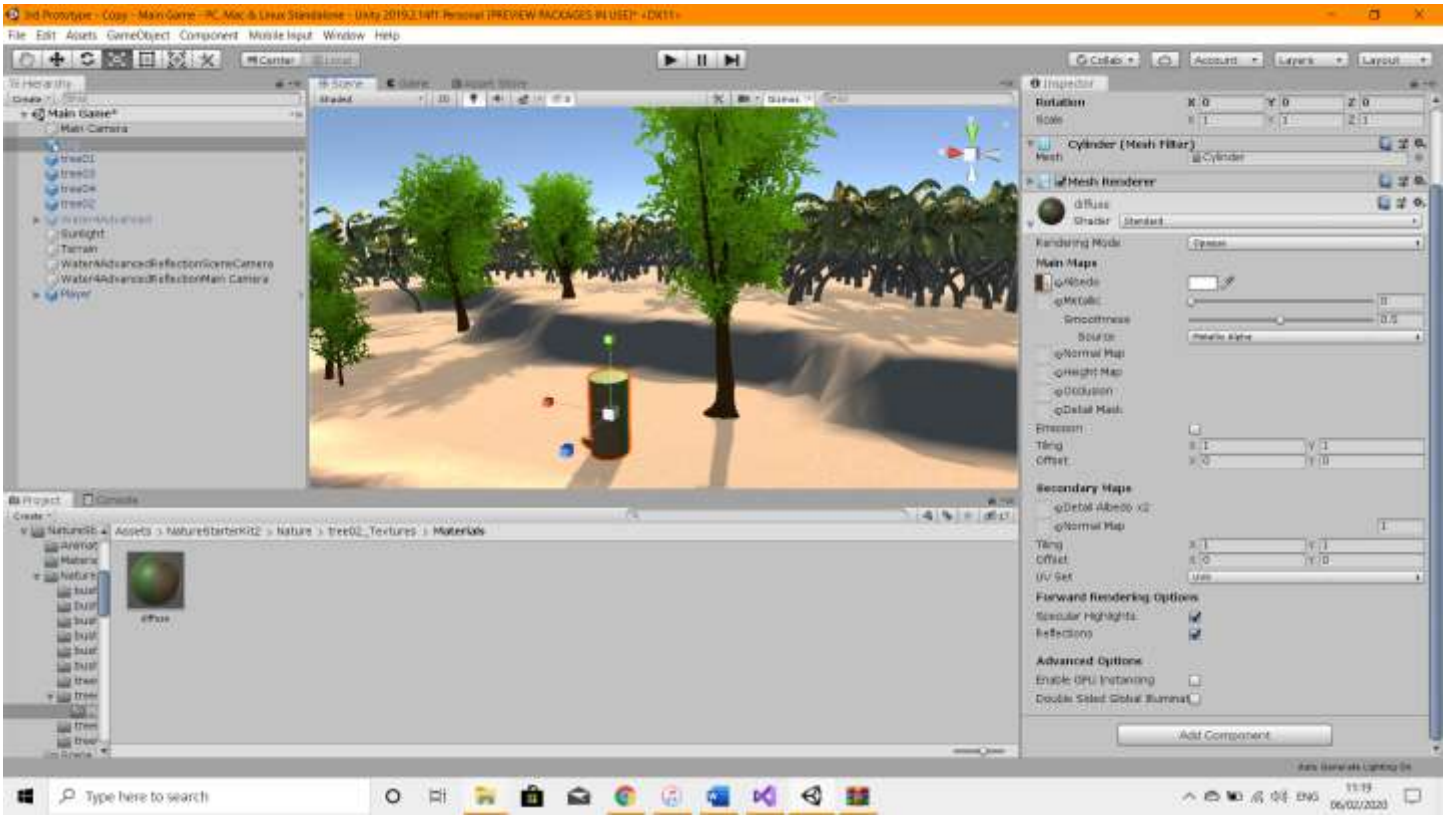
THIRD PROTOTYPE -TREE PHYSICS

https://assetstore.unity.com/packages/3d/environments/nature-starter-kit-2-52977



I found a tree mesh from “natureStarterKit” and used this as I needed a mesh instead of the default tree painting which I used before. To allow them to be selected and scripts to be added therefore it can fall and drop logs for crafting. Where as the painted tree’s I was unable to do this. This assets pack also has different types of tree’s I could use. I first added a rigid body, made sure “use kinematic” and “use gravity” was ticked allowing it to fall over.

Here I added a box collider and resized to the trunk therefor when the tree falls over it knows where the dimensions of what can collide and what can't.



I then added logs so when the tree is chopped down you can pick up the logs making the game more realistic. I added a box collider and rigid body so they follow the physics for falling with gravity and not go through the floor.

```
var treeHealth : int = 5; #creates a variable treeHealth and sets its value to 5 stating how many times it needs to be hit to be broken down

var logs : Transform; #Creates a variable for logs
var coconut : Transform; #Creates a variable for coconut
var tree : Gameobject; #Creates a variable for the tree stating also, that it is a gameobject

var speed : int = 8; #causes the force for the tree to fall over

function Start()
{
	tree = this.gameObject; #adds the tree object to the inspector automatically
	rigidbody.isKinematic = true; #Makes sure the rigidbody is set to true
}
function Update()
{
	if(treeHealth <= 0)
	{
		rigidbody.isKinemtic = false; #makes sure that the tree is able to fall over when the tree's health is 0
		rigidbody.AddForce(transform.forward * speed); #able to push the tree over
		DestroyTree(); #calls the funmction to destory the tree
	}

}
function DestroyTree()
```

```
{
	yield WaitForSeconds(7); #Holts the program 7 seconds before destroying the tree entity allowing
it have fallen over fully
	Destroy(tree);

	var position : Vector3 = Vector3(Random.Range(-1.0, 1.0), 0, Random.Range(-1.0,1.0)); #will
respawn the tree in a random range between the 2 stated values
	Instantiate(logs, tree.transform.position + Vector3(0, 0, 0) + position, Quaternion.identity);
#spawns the logs
	Instantiate(logs, tree.transform.position + Vector3(2, 2, 0) + position, Quaternion.identity);
#spawns the logs
	Instantiate(logs, tree.transform.position + Vector3(5, 5, 0) + position, Quaternion.identity);
#spawns the logs

	Instantiate(coconut, tree.transform.position + Vector3(0, 0, 0) + position,
Quaternion.identity); #spawns the coconut
	Instantiate(coconut, tree.transform.position + Vector3(2, 2, 0) + position,
Quaternion.identity); #spawns the coconut
	Instantiate(coconut, tree.transform.position + Vector3(5, 5, 0) + position,
Quaternion.identity); #spawns the coconut
}
```

I then created this script “TreeController.js”in JavaScript for the tree, which held the health of the tree the physics when it was chopped down and it actually falling to spawning logs on the ground.

```
var rayLength : int =10;

private var treeScript : TreeController; #Creates variable tree script
private var playerAnim : PlayerControl;

function Update()
{
	var hit : RaycastHit;
	var fwd = transform.TransformDirection(Vector3.forward);

	if(Physics.Raycast(transform.position, fwd, hit, rayLength))
	{
		if(hit.collider.gameObject.tag == "Tree") #if the player has hit the collider
		{
			treeScript
=GameObject.Find(hit.collider.gameObject.name).GetComponent(TreeController); #doesn't knock down all
the tree's

		}
	}
}
```

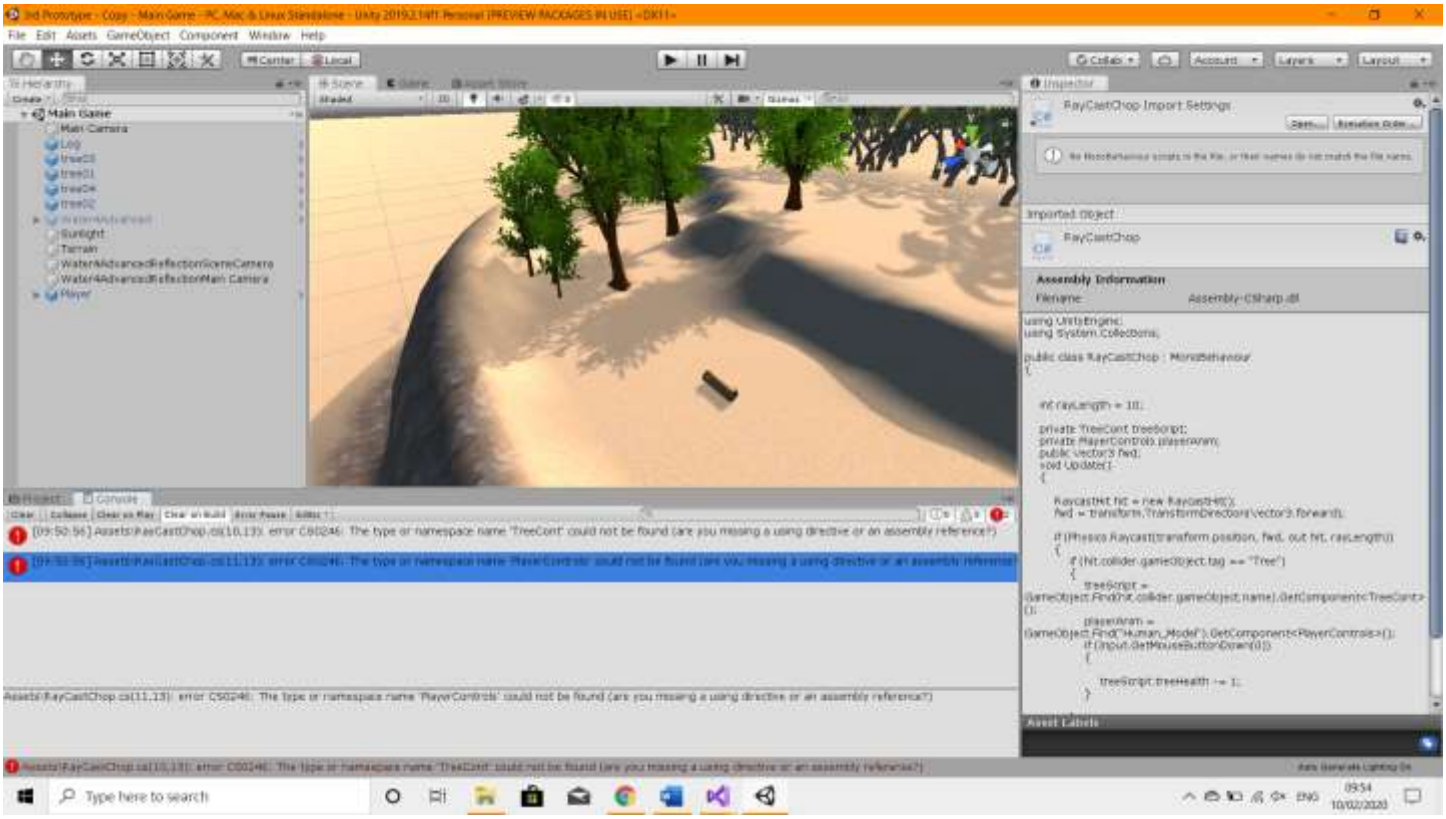
This created another JavaScript file called “raycast.js”that states if the tree is hit with an object it will reduce the health and the tree will fall down.

```
using UnityEngine;
using System.Collections;

public class RayCastChop : MonoBehaviour
{
```

```
    int rayLength = 10;

    private TreeCont treeScript; //Creates a private variable for the treeCont
    private PlayerControls playerAnim; //Creates a private variable for the PlayerControls
    public Vector3 fwd; //Creates a private variable for the Vector3
    void Update() //function fr update
    {

        RaycastHit hit = new RaycastHit();
        fwd = transform.TransformDirection(Vector3.forward);

        if (Physics.Raycast(transform.position, fwd, out hit, rayLength)) //if the player has hit the
collider
        {
            if (hit.collider.gameObject.tag == "Tree")
            {
                treeScript = GameObject.Find(hit.collider.gameObject.name).GetComponent<TreeCont>();
                playerAnim = GameObject.Find("Human_Model").GetComponent<PlayerControls>();
                if (Input.GetMouseButtonDown(0))
                {

                    treeScript.treeHealth -= 1;
                }

            }
        }
    }
}



using UnityEngine;
using System.Collections;

public class TreeController : MonoBehaviour
{

    //Tree Health Value
    public int TreeHealth = 5;
    //Rate at which the speed hits the ground (Force applied)
    public int speed = 4;


    public Transform Log; //Creates a variable for logs

    //GameObject
    public GameObject Tree; //Creates a gameobject for tree



    // Use this for initialization
    void Start()
    {
        Tree = this.gameObject; //adds the tree object to the inspector automatically
        Tree.GetComponent<Rigidbody>().isKinematic = true; //Makes sure the rigidbody is set to true
```

```
    }

    // Update is called once per frame
    void Update()
    {
        if (TreeHealth <= 0)
        {
            Tree.GetComponent<Rigidbody>().isKinematic = false; //makes sure that the tree is able to
fall over when the tree's health is 0
            Tree.GetComponent<Rigidbody>().AddForce(transform.forward * speed); //able to push the tree
over
            DestroyTree(); //calls the funmction to destory the tree
        }
    }

    public void DestroyTree() //Destorys the tree mesh
    {
        StartCoroutine(TreeToGroundWait());
    }




    IEnumerator TreeToGroundWait()
    {
        yield return new WaitForSeconds(7); //Waits 7 seconds to allow the tree to fall down
        Destroy(Tree); //destroys the tree mesh

        Vector3 position = new Vector3(Random.Range(-1.0f, 1.0f), 0, Random.Range(-1.0f, 1.0f)); //will
respawn the tree in a random range between the 2 stated values
        Vector3 offset = new Vector3(0, 0, 0);
        Vector3 offset1 = new Vector3(2, 2, 0);
        Vector3 offset2 = new Vector3(5, 5, 0);

        Instantiate(Log, Tree.transform.position + offset + position, Quaternion.identity); //spawns a
log
        Instantiate(Log, Tree.transform.position + offset1 + position, Quaternion.identity); //spawns a
log
        Instantiate(Log, Tree.transform.position + offset2 + position, Quaternion.identity); //spawns a
log
    }
}
```

After finding out that only the old version of unity runs on JavaScript and therefor the script wouldn't able to be a component of the tree I had to convert the JavaScript into code. For both the "TreeController.js" and "RayCastChop.js". Above is the examples.



Due to me not creating any animation yet to attack the tree these errors occurred so I decided to import some animation for attacking the tree.

-I decided to remove the script as the script was heavily based around JavaScript therefore constantly converting to c# which led to constant errors.

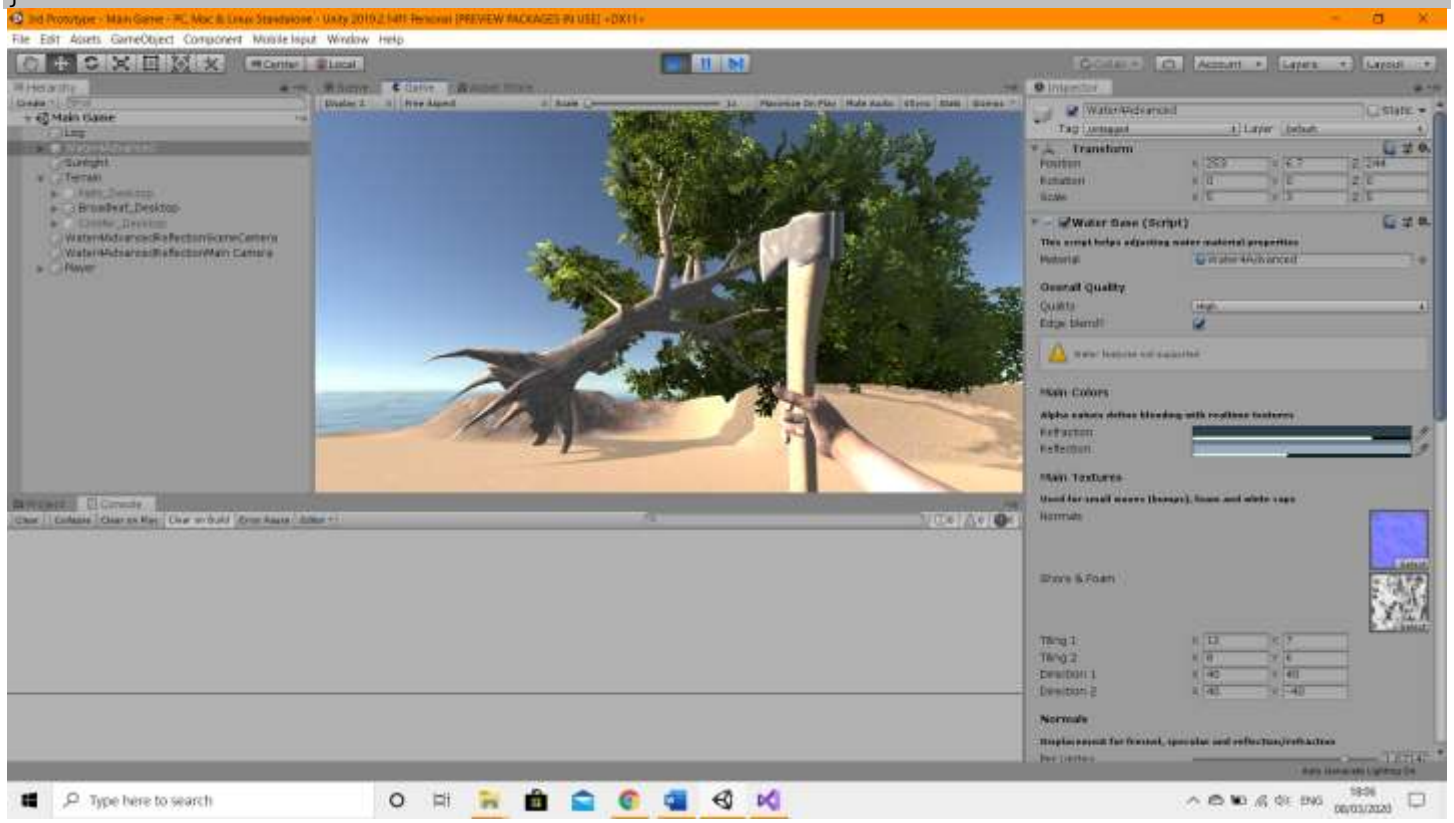
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class AxeRaycast : MonoBehaviour
{
    //Variables
    public GameObject axe;
    private bool isEquiped = false;

    private void Update()
    {
        if(!axe.activeSelf && Input.GetKeyDown(KeyCode.Alpha1))
        {
            isEquiped = true;
            axe.SetActive(true);
        }
        else if(Input.GetKeyDown(KeyCode.Alpha1))
        {
            isEquiped = false;
```

```
            axe.SetActive(false);
        }

        //Raycast
        Vector3 fwd = transform.TransformDirection(Vector3.forward);
        RaycastHit hit;

        //Origin, Direction, RaycastHit, Length
        if(Physics.Raycast(transform.position, fwd, out hit, 10))
        {
            if(hit.collider.tag == "tree" && Input.GetMouseButtonDown(0) && isEquiped == true)
            {
                Tree treeScript = hit.collider.gameObject.GetComponent<Tree>();
                treeScript.treeHealth--;
            }
        }
    }
}
```

The new c# raycast script is placed in the camera.

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Tree : MonoBehaviour
{
    //Variables
    GameObject thisTree;
    public int treeHealth = 5;
    private bool isFallen = false;



    private void Start()
    {
        thisTree = transform.parent.gameObject;
    }

    private void Update()
    {
        if(treeHealth <= 0 && isFallen == false)
        {
            Rigidbody rb = thisTree.AddComponent<Rigidbody>();
            rb.isKinematic = false;
            rb.useGravity = true;
            rb.AddForce(Vector3.forward, ForceMode.Impulse);
            StartCoroutine(destroyTree());
            isFallen = true;
        }
    }

    private IEnumerator destroyTree()
    {
        yield return new WaitForSeconds(10);
        Destroy(thisTree);
```

```
    }

}
```



This c# is used for my new tree script which is tested and works. Therefor when the axe is in the toolbar and is clicked in the tree 3 times the tree falls over and after 7 seconds the tree is destroyed

TESTING-CHECKLIST

Testing Checklist-As game is being created

Green Box- Completed no further testing is needed currently

Amber Box- Partially completed needs fixing/amending and retesting

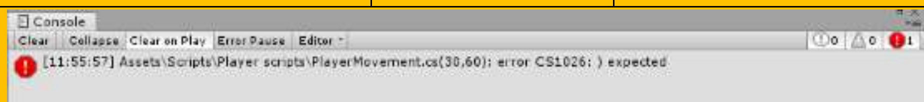
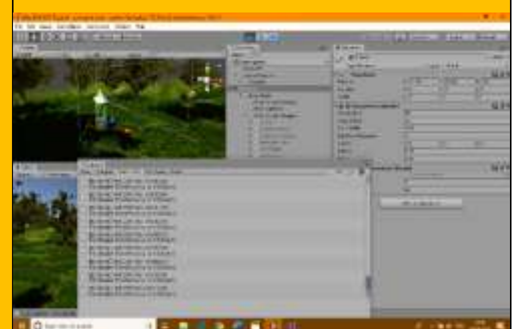
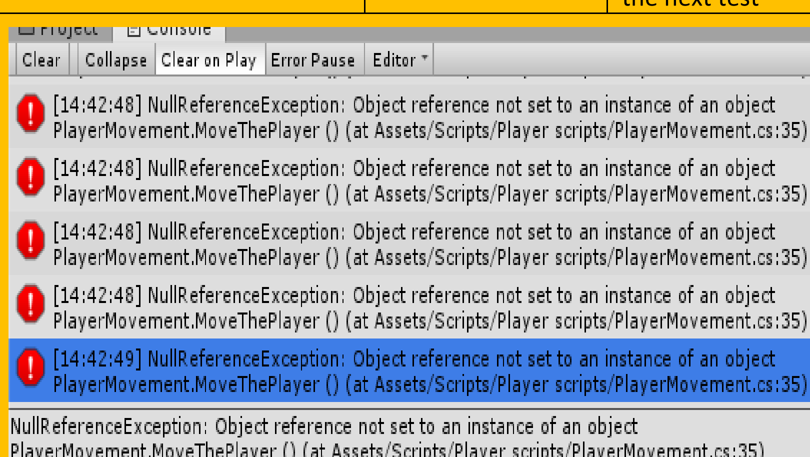
Red Box- Not completed needs doing and retesting

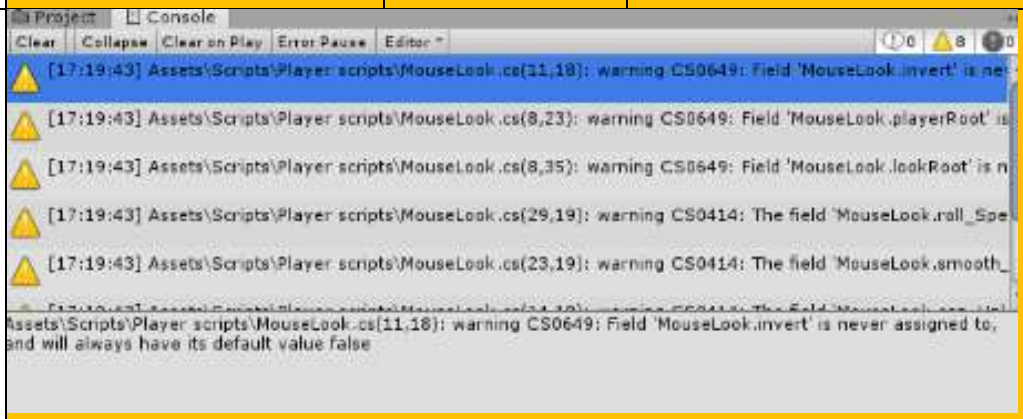
Numbered test	Action To Test	Working	Improvement
1a.-Control	Mouse look	Yes	No

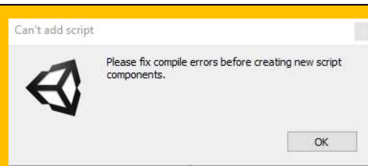
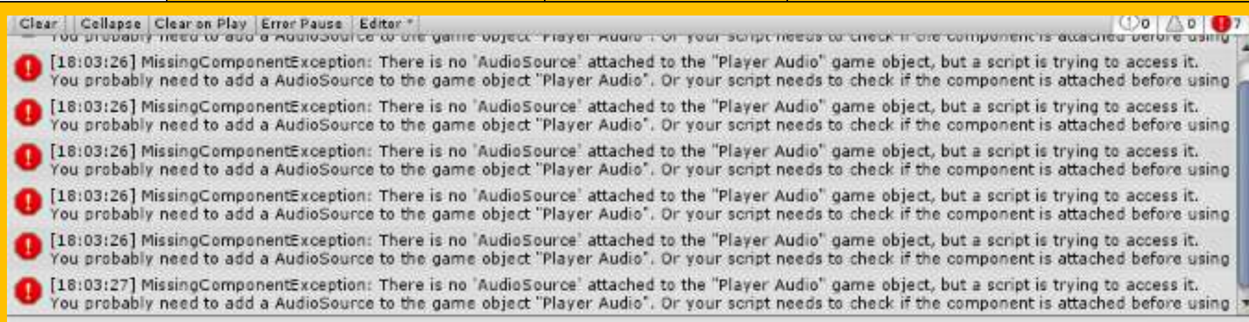
2a.-Control	ASWD-movement working	Yes	No
3a.-Control	Sprint	Yes	Yes-test 23a
4a.-Control	Crouch	Yes	No
5a.-Control	Jumping	Yes	No
6a.-Map	Collision/Gravity	Yes	No
7a.-Control	Crouch while sprinting	Yes	No
8a.-Map	Water physics	Partially	22a-next test
9a.-Developer	"esc" button to escape from the game and allow the developer to come in and out of a live game showing the cursor or not	Yes	No
10a.-Yield of Trees	Yield Of tree's	No	24a-next test
1A-Userfeeback Point-Luke Callison Bailey			
11a.-Login	Password replaces characters with "*"	Yes	No
12a.-Register	Password replaces characters with "*"	Yes	No
13a.-Login	Username has 17 character max limit	Yes	No
14a.-Register	Username has 17 character max limit	Yes	No
15a.-Login	Email has 17 character max limit	Yes	No
16a.-Register	Email has 17 character max limit	Yes	No
17a.-Password confirmed	The password entered in is the same as the confirmed password	Yes	No
18a.-Password length	The password entered in has a max length of 6 characters	Yes	No
19a.-Username Unique	Username hasn't already been used	Yes	No
20a.-Correct Email	Email contains "@" and "."	Yes	No
21a.-Login	Login Button loads scene	Yes	No
22a.-Login	Create button creates and writes new username details and loads into scene	Yes	No
22a.-Map	Water physics	Partially	-next test
23a.-Sprint	Decrease the speed of the player. After 1A user's feedback	Yes	No
24a.-Yield Of trees	Created paths in the island and made the yield of trees a lot smaller therefore not as dense. After 1A user's feedback	Yes	No

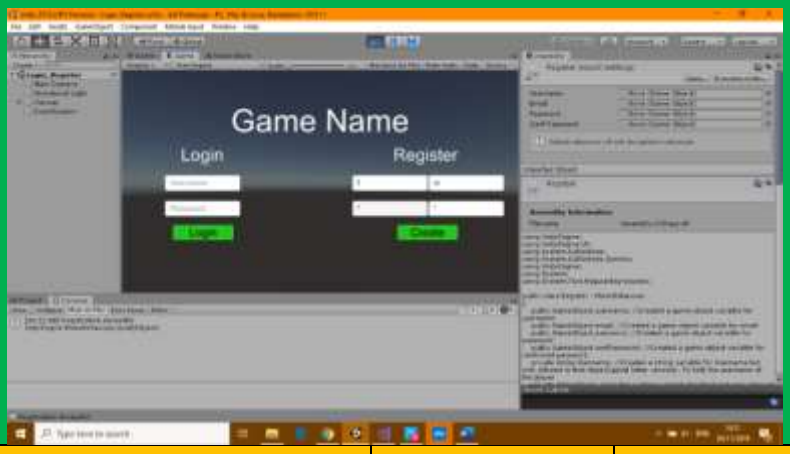
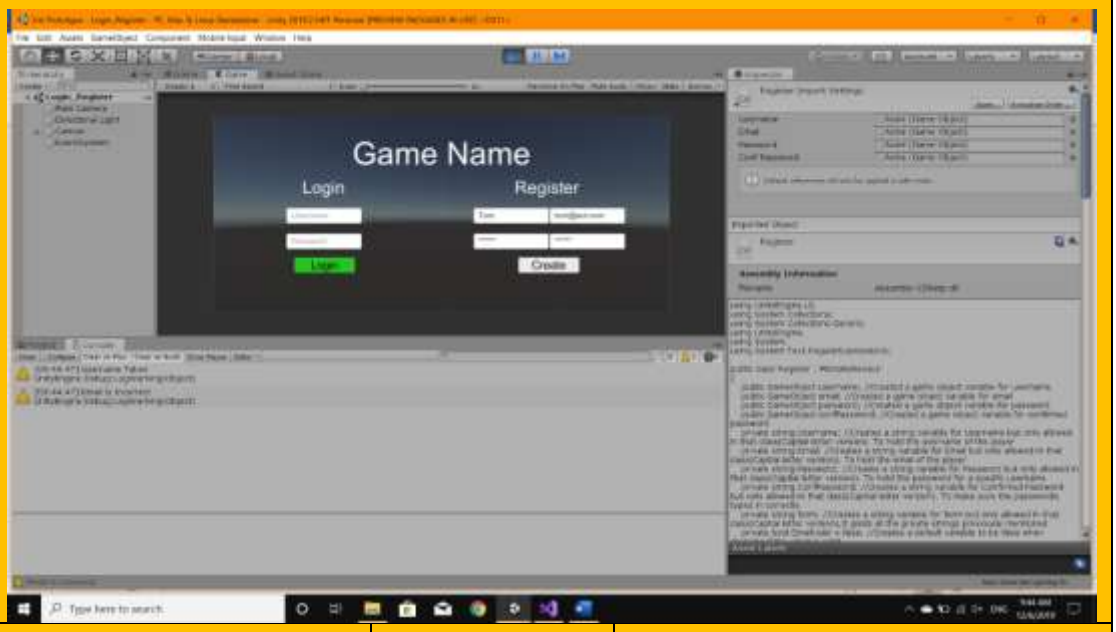
TESTING-DEVELOPMENT TABLE

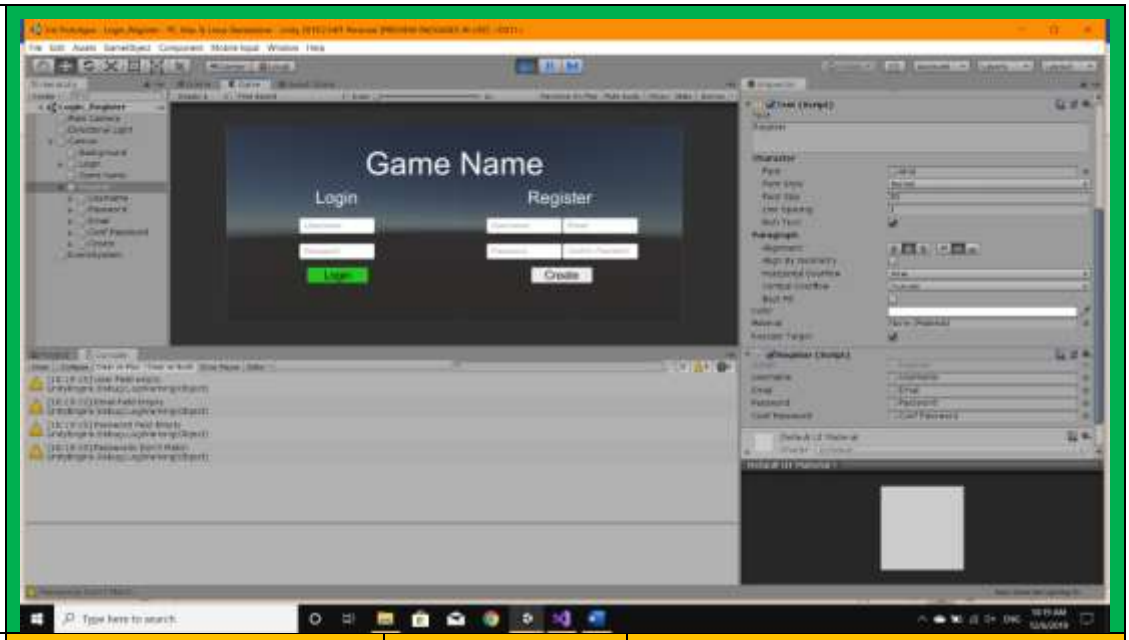
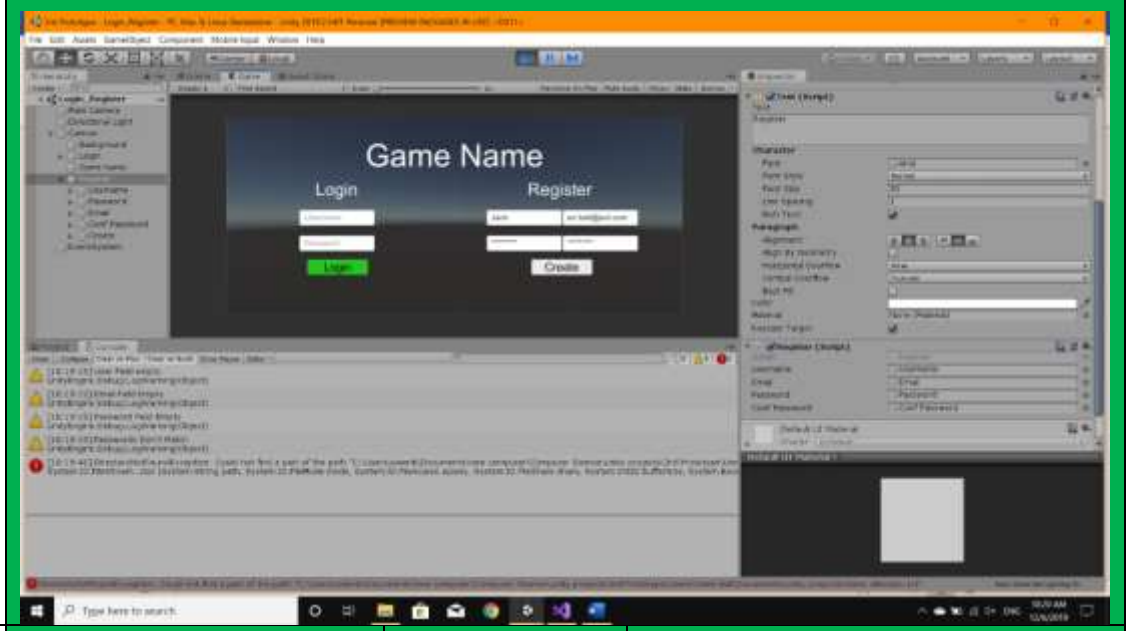
Development Table-The tests that were run after every implication of a new feature

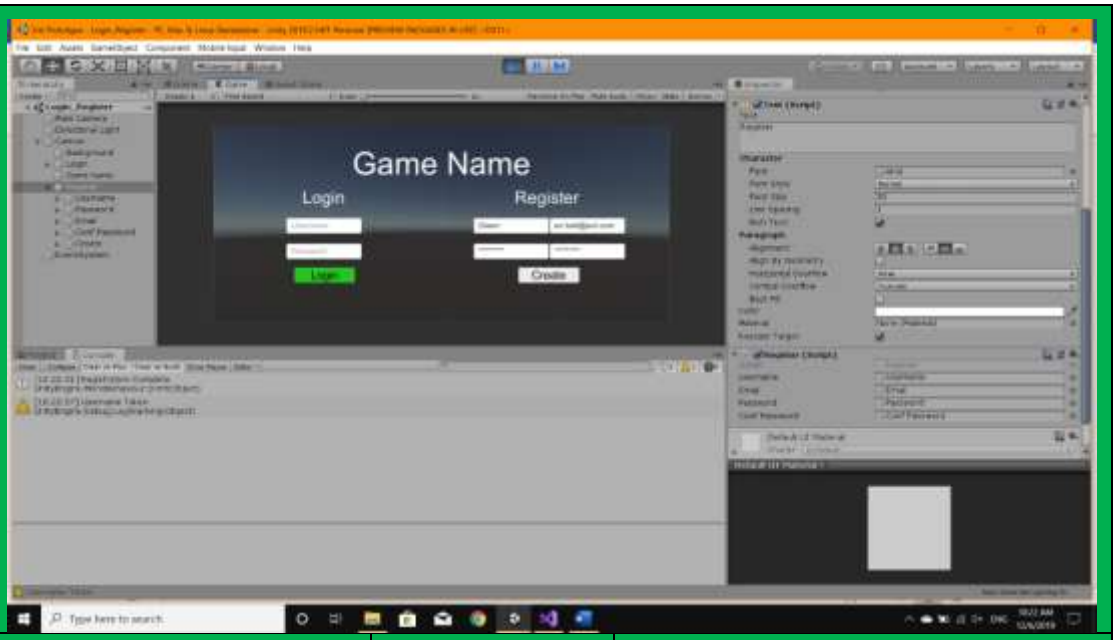
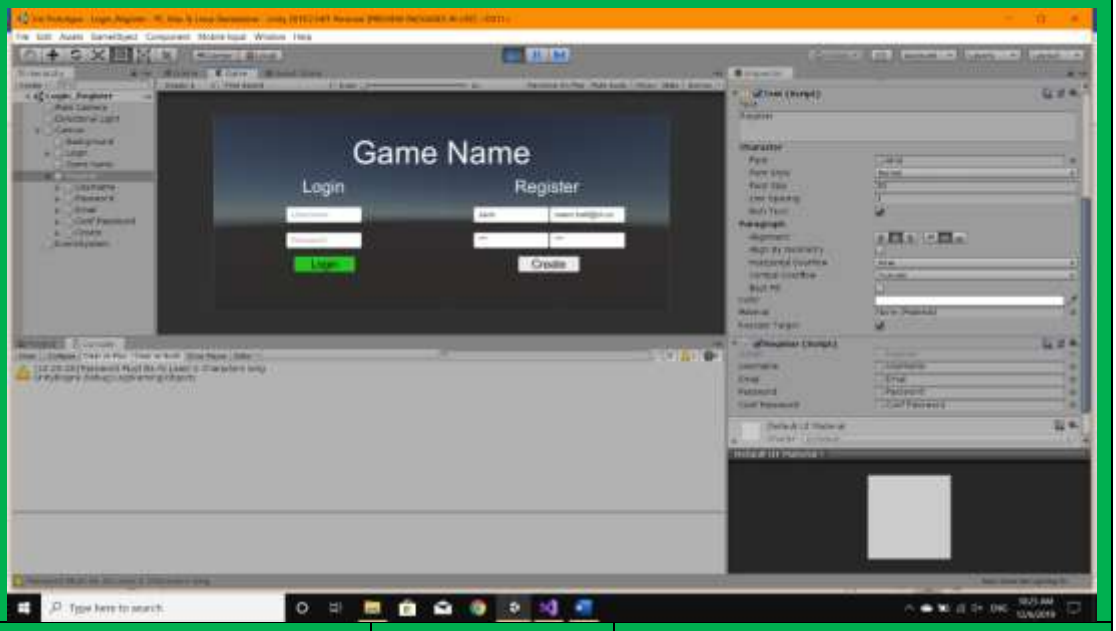
Numbered test	Action To Test	Working	What needs changing if not?	Did that change work?
1b.	Player Movement.cs ()-general movement	• 1-error occurring due to an unneeded bracket • 2 errors remaining See Below:	Remove the unneeded bracket	Yes 2b for the next test
				
2b.	Player Movement.cs ()-general movement	• 1-unlinked script so action was occurring • 1 error remaining	Drag and drop the script into the player folder attached on the game	Yes, 3b for the next test 
3b.	Player Movement.cs ()-general movement	1-error occurring due to "Object not referenced" See Below:	Check Grammar on variables	Yes- after relooking over the C# code I realised I didn't capitalize the "Awake" function causing the error to occur. 4b for the next test
				
4b.	Player Movement.cs ()-general movement	Yes	N/A	N/A

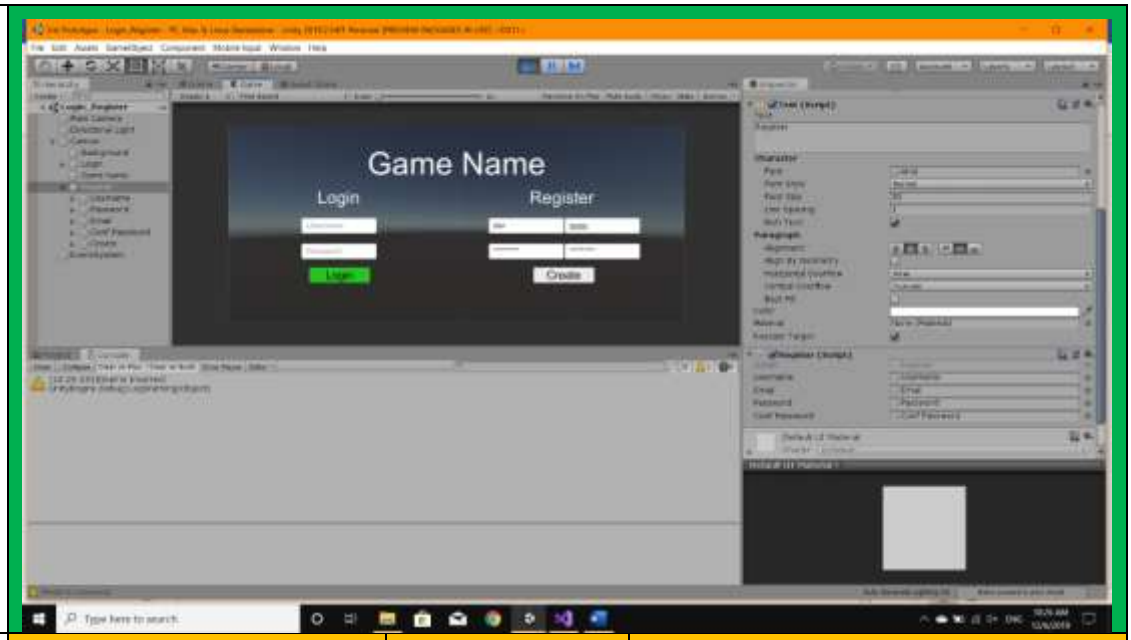
5b.	Player Movement.cs ()-Gravity	1-Falling through the floor of the map 	Re-look over the code make sure gravity is assigned and all variables are spelt correctly	Yes-This issue was fixed by adding a reference to the gravity function (due to it not being called so the gravity script wouldn't work properly). 6b for the next test 
6b.	Player Movement.cs ()-Gravity	Yes	N/A	N/A
7b.	Mouse Look.cs()-General Movement	Multiple-Variables assigned but never used error See Below:	Check all spelling on variables	Yes- all down to capitalised letters needing to be lower case and forgetting the "f" after a number (0f) with certain commands. 8b for the next test
				
8b.	Mouse Look.cs()-General Movement	Yes	N/A	N/A
9b.	Mouse Look.cs()-Developer "esc button"	Yes	N/A	N/A
10b.	Sprint Crouch.cs()-Crouch "c"	Yes	N/A	N/A
11b.	Player Foot Steps.cs()-Sound effects for the player moevment	• 1-Can't add script error • 2 Remaining error	Remove the spacing in the text of the files name	Yes- due to the name of the C# having a space in it but after deleting it and renaming it without any spaces the error disappeared. 12b for the next test

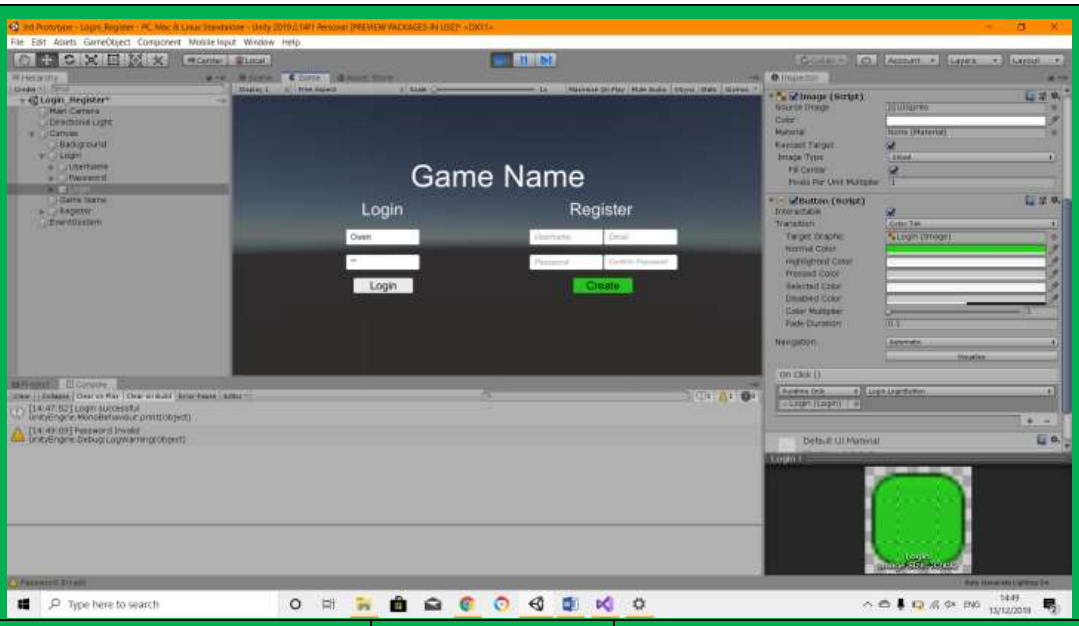
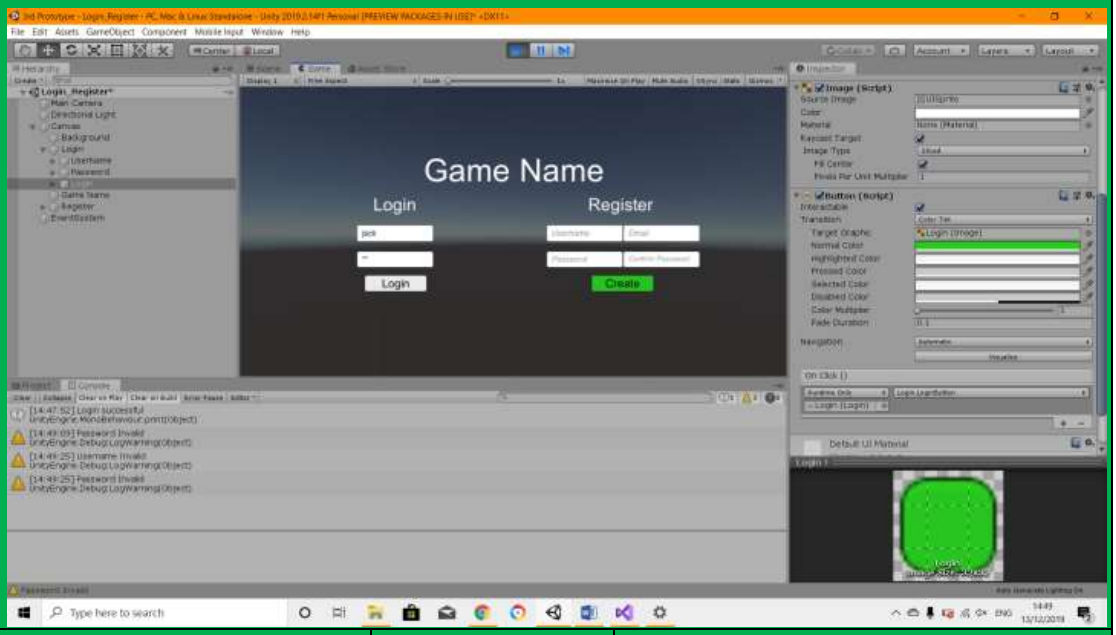
				
12b.	Player Foot Steps.cs()-Sound effects for the player moevment	• 1-“Assets\Scripts\Player scripts\SprintCrouch.cs (18,13): error CS0246: The type or namespace name 'PlayerFootsteps' could not be found (are you missing a using directive or an assembly reference?)” • 1 remaining error	Need to relook over my full code step by step due not having an idea what this could be	Yes-After relooking at my code I realised I haven’t made the void function for “CheckToPlayFootsepSound.cs” I put all the code that should of been in this in the update function. This therefore fixed this error. 13b for the next test
13b.	Audio Source	1-“There is no audiosource attached to the player Audio” See below:	I needed to place the “audioscource” component in the player audio section and untick everything which is set ticked as default.	Yes-I replaced the component which was set to a default as empty with the correct audio source components and the error was removed.14b for the next test
				
14b.	Audio Source	Yes	N/A	N/A
15b.	Check for any warning error’s	• 1-in the console that two “Audio listeners” were active when only one is needed	Find both ticked boxes and de-tick the necessary one	Yes- I found out I hadn’t unticked the main camera one which then removed the error. 16b for the next test
16b.	Audio Source	Yes	N/A	N/A
17b.	Register.cs()-Login	1-I gained this error “UnassighnedReferenceException: The variable username of Register has not been assighned”	I need to check they I have assigned all the correct components	Realise that I haven't assigned my variables to my components therefore I dragged the correct one to each variable and this fixed the error and my tabbed function worked.18b for next test
18b.	Register.cs()-Login	Yes	N/A	N/A

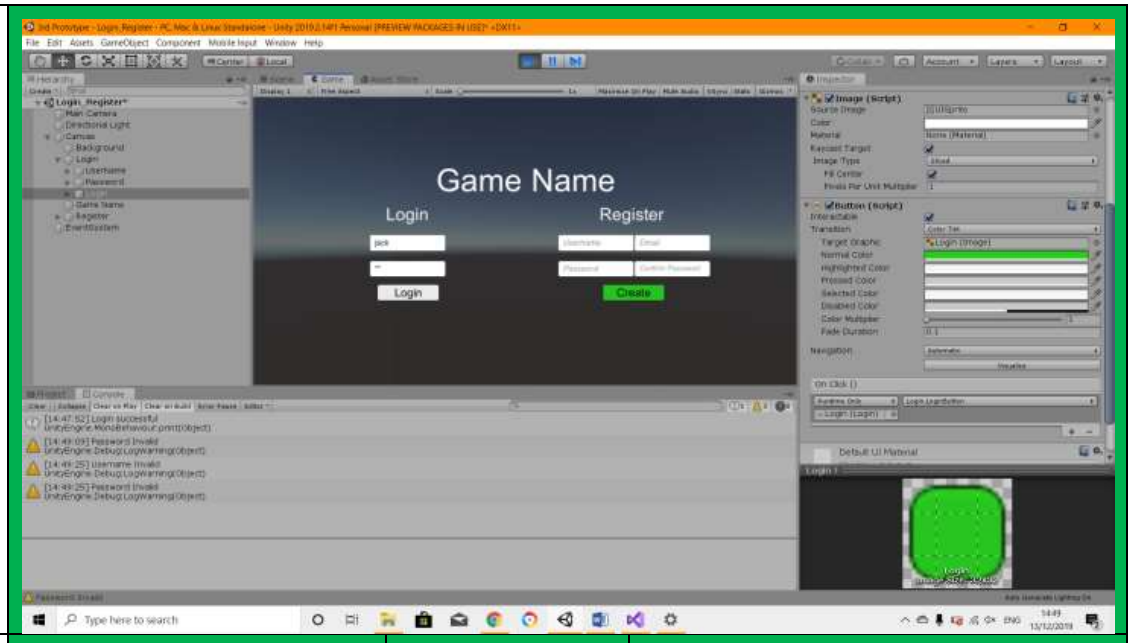
19b.	Register.cs()-Check for any blank fields	Yes See Below:	N/A	N/A
				
20b.	Register.cs()-Correct details	1-"UserName Taken" 1-Remaining error "Email Incorrect" :See Below	Re-look at the code	Yes-The first error to do with username already taken was due to their being no "!" in front of the System.IO.FileExists causing this error to occur. I corrected this as seen above.21b for the next test.
				
21b.	Register.cs()-Correct Details	1-"Email incorrect"	Re-look at the code	Yes-After re-looking over my code I realised forgot to put in the array to find which character represents each number and put the full validation for the email. 22b for the next test
22b.	Register.cs()-Check Fields are blank warning	Yes See Below:	N/A	N/A

				
23b.	Register.cs()- Correct Details	1-Could not fin the correct file path	Need to re-look at file paths	Yes- This was due to me having a new laptop so a new file path was needed
				
24b.	Register.cs()- Username warning check	Yes See Below:	N/A	N/A

25b.				
26b.	Register.cs()- Character length warning check	Yes See Below:	N/A	N/A
27b.				
28b.	Register.cs()- Character length warning check	Yes See Below:	N/A	N/A

29b.				
30b.	Register.cs()-Password equalling the same warning check	1-Won't login but no error occurs	Check if everything's spelt correctly when giving the warning error	Yes- it was missing from the code. Test for 31b
31b.	Register.cs()-Password equalling the same warning check	Yes See Below:	N/A	N/A
32b.	User's details-.txt form separate lines	1-Writes in the correct details but not on separate lines 	Amend the piece of code to write each piece of information onto a separate line in the login.cs() script	Yes-Due to an update on unity my old method of writing on separate lines no longer worked. Therefore I amended "`form = (Username +Environment.NewLine+ Email + Environment.NewLine + Password); //concatinates all the data for the new created user`"to the login.cs() script. Test for 33b
33b.	User's details-.txt form separate lines	Yes 	N/A	N/A
34b.	Login.cs()-Password incorrect	Yes See Below:	N/A	N/A

				
35b.	Login.cs()- Usernames incorrect	Yes See Below:	N/A	N/A
				
35b.	Login.cs()- Usernames and password correct	Yes See Below:	N/A	N/A

FINAL TESTING AND EVALUATION

EVALUATION OF SOLUTION

Below are they key points of the Usability features, I will be testing each one and adding it the testing table below.

- A-move left
- S-move backwards
- D-move right
- W-move forwards
- (left click)-break item in front of the player
- (right click)-use item in hand
- Built in Speakers-project the sound effects
- Scroll Wheel- Allows the player to select an item specifically in the tool bar with rearranging with the mouse
- (space bar)-Jump

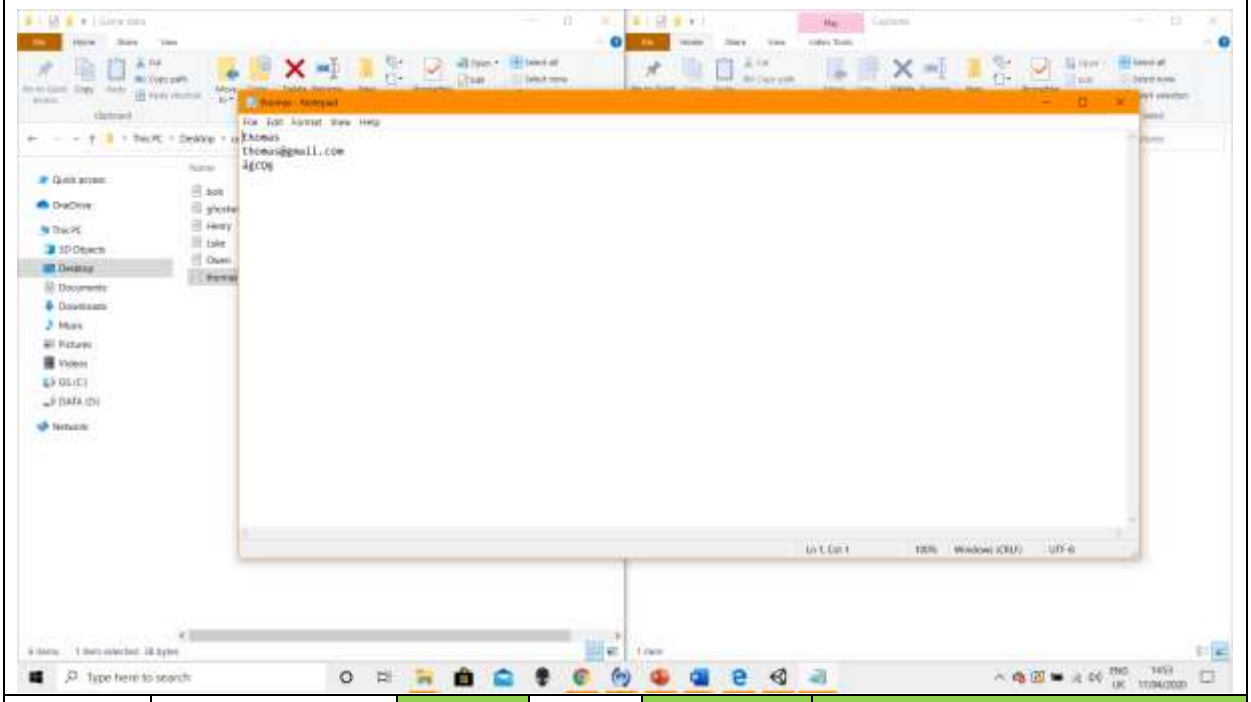
Test	Description of the tests	Justification
Movement	The movement of the player	To check if the player can move around using the correct keys.
Sound Effects	The Sound effects of the player	To see if when the player moves a sound effect is followed as well. This will be the same when the player swims
Weapons	The weapons can be mounted and used	To check if the weapons work when killing enemies or collecting materials
Enemies	The enemies attack the player and spawn at random or near specific area's	Make sure the enemies attack the player and the players health is decreased when done so

Login System	The login system works, and all the functions of the buttons works	To allow specific users to have their own account
Tree's	Tree's can be chopped down	To allow the player to gain materials
Environment conditions	Certain environments can hurt and damage the player	So, the player has something to survive too and has to use science to survive (main purpose of the game)
Players health	That when a player is hurt the health decreases and when a player eats food the health is gained back	To test of when the player is drowning or dyeing in some way the health is decreased or when the player is eating the health is increased like in real life
Day and Night	Testing if the day and night cycle work	To make sure the days are realistic and work making the game more believable
Swim	If the player can swim and if the player can drown	So the player doesn't drowned and can actual swim to different parts of the island
The Map	Can't fall through the floor no glitches	Allows the player to actually play the game instead of a glitch which the map has stopped them from continuing the game

This table above is taken from the post development table but adjusted with an extra column stating if the test had been successful or not. The results are shown in the video labelled "My Testing" with added commentary stating which test is being used and if it was successful.

Numbered test	Action To Test And Expected Outcome	Working	Time in video	Could this be improved	Justification
Movement					
0c.	(space bar)-Jump	Yes	0.04	No	N/A
1c.	A-move left	Yes	0.16	No	N/A
2c.	S-move backwards	Yes	0.22	No	N/A
3c.	D-move right	Yes	0.26	No	N/A
4c.	W-move forwards	Yes	0.3	No	N/A
Sound Effects					
5c.	Walking sound effect	Yes	0.35	Yes	This could be changed so they when you walk on different materials different sounds are heard
Added Extra's that wasn't implemented					
6c.	(left click)-break item in front of the player	No		Yes	Needs to be added as this was an additional extra what wouldn't as necessary for the game to run
7c.	(right click)-use item in hand	No		Yes	Needs to be added as this was an additional extra what wouldn't as necessary for the game to run

8c.	Scroll Wheel- Allows the player to select an item specifically	No		Yes	Needs to be added as this was an additional extra what wouldn't as necessary for the game to run
Login System					
9c.	Username Already Exists when creating an account-output error	Yes	1:00	No	N/A
10c.	Password matches with the confirmation password when creating an account-output error	Yes	1:30	No	N/A
11c.	Password has a character limit of 17 when creating an account-output error	Yes	1:40	No	N/A
12c.	Password replaces text with asterixis when typing in either login or creating an account	Yes	2:00	No	N/A
13c.	Password must be at least 6 characters long when creating an account-output error	Yes	2:10	No	N/A
14c.	Email address must contain a "." When registering -output error	Yes	2:55	Yes	There could be an email sent out that could verify that it is the correct email and when a link clicked allows the process to continue and login
15c.	Email address must contain a "@" When registering -output error	Yes	2:28	Yes	There could be an email sent out that could verify that it is the correct email and when a link clicked allows the process to continue and login
16c.	When creating an account, no field should be blank-output error	Yes	3:10	No	N/A
17c.	When creating an account, a file	Yes	Picture Below	No	N/A

	should be created (in the correct location)if successful labelled Yes with the title as the username created				
18c.	Inside the created file (when creating an account) should be the username, email and password encrypted	Yes	Picture Below	No	N/A
19c.	The password should be encrypted when saving to the file when creating an account	Yes	Picture Below	Yes	A more complex encryption method to be used the text files to be locked
					
20c.	The “Create” button should check if the correct requirements are met if not select the correct error message and if all correct create the file and load the game	Yes	3:48	No	N/A

21c.	The “Login” button should check there is no empty fields, de-crypt the password and check username if it exists. If so login and if not output the correct message	Yes		No	N/A
Tree’s					
22c.	Can’t walk through the tree’s	Yes	4:15	No	N/A
23c.	When holding an axe the tree can be chopped down after 3 swings	Yes	4:40	Yes	The tree can be chopped down using other items and depending the item depends how many swings
24c.	The tree can’t be chopped if not holding an axe	Yes	5:00	No	N/A
25c.	The tree is destroyed after 10 seconds	Yes	4:55	No	N/A
Map					
26c.	The water moves and looks realistic	Yes	5:48	No	N/A
27c.	There’s sand	Yes	5:52	No	N/A
28c.	Rocks on the edge of the cliff	Yes	5:57	No	N/A
29c.	Tree’s scattered across the island	Yes	6:00	No	N/A
30c.	A faint path in-between the tree’s	Yes	6:10	No	N/A
Day/Night Cycle					
31c.	The sun moves around the island creating shadows	Yes	6:24	No	N/A
32c.	The time in the game is realistic	Yes	6:35	No	N/A

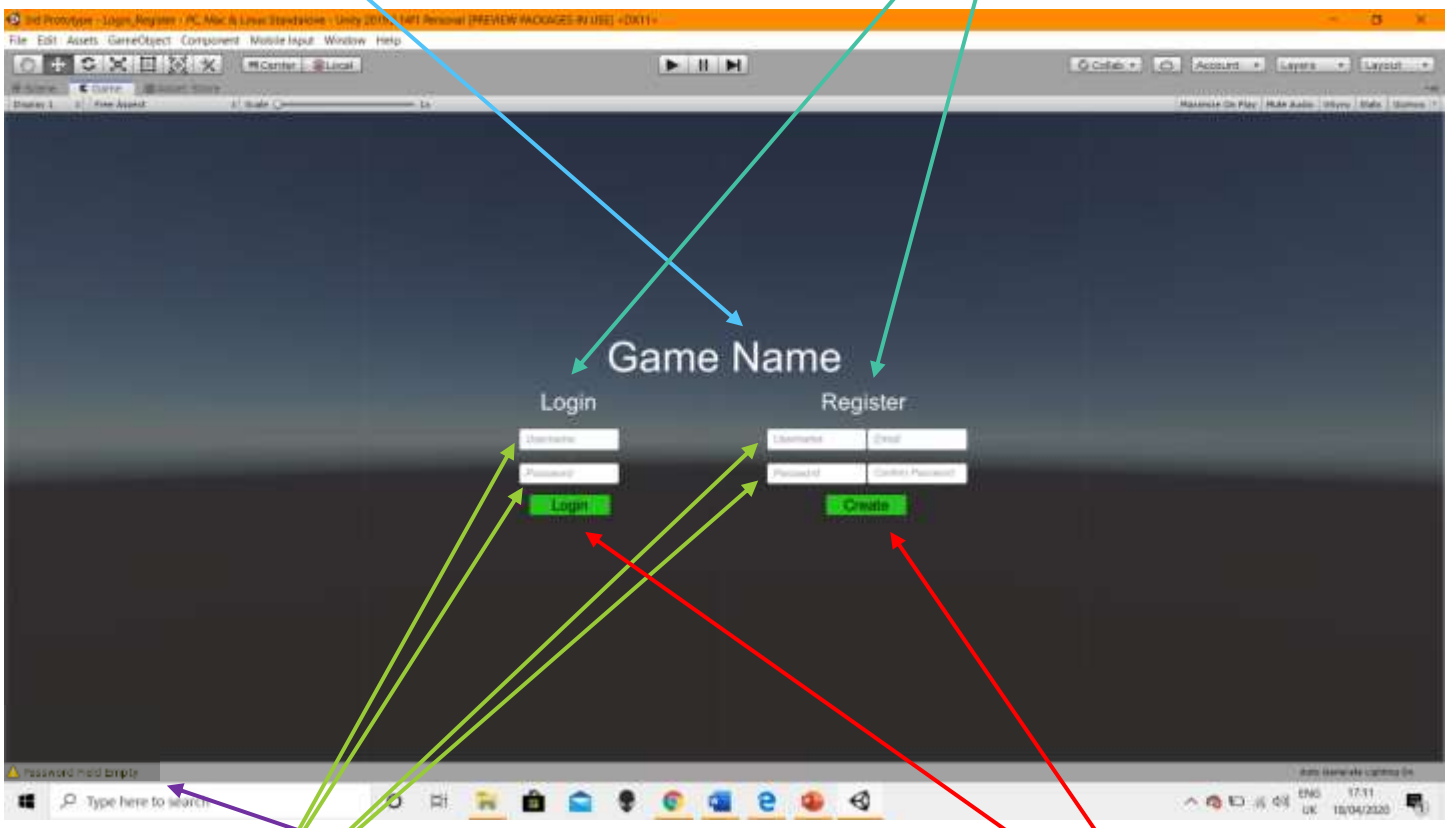
This is the final testing table above considers both the usability features and the post development testing table as well as additional tests created when the game was being developed. It’s presented with 6 columns first states which test is being used as it’s the third revulsion of types of test it has “c” on the end of each one. The second column states what’s being tested the third tells the reader if

this feature is working. The fourth indicates at which time this feature is being checked in the video labeled “My testing”, the fifth see’s if this feature could be improved and if so the fin al column (6th) justifies thus change.

USABILITY FEATURES

This text is in the greatest size and placed in the centre allowing the user to know that the name of the game is.

This font size isn’t as great as the title of the game but us in-between the title size and the button/textbox size allowing the user to know that these are sub sections of the login screen.



A light grey text is placed in the box to indicate what data should be typed in. As soon as the user clicks the box this text disappears.

If an error occurs this is where it’ll be shown with a yellow triangle which indicates an error (known as an everyday symbol so users are familiar with it)

These buttons are in a different colour (green) with a black text showing them to be buttons as other games use this as a universal way to show this.

The above diagram shows the usability features of the login screen. Each part has been identified with an arrow and a commented statement on what this shows as a usability feature. The game itself is listed in bullet points as shown for the usability of the game

1. Familiar layout controls: cursor keys, WSAD, or keyboard mapping options for the player.
2. When Creating an account, it automatically log's you into the game instead of having to retype your information in-affiance
3. Toolbar slot accessed by numbered keys assigned to each toolbar slot
4. Different functions such as crouch is the letter "c" on the keyboard as crouch begins with c
5. Sprint is the tab button as that is the universal button for the majority of PC video games
6. The right click button used to destroy items as it would for other video games

SUCCESS CRITERIA

Below is the success criteria created in the analysis section (in grey) with the additional two column stating if each feature has been fully met partially met or not met. The final column justifies the reason why it has only been partially met or not met all colour coordinated.

Number	Features	Justification	Reference	Met	Justification
1.	Startup screen	Allowing the player to adjust the settings load into a game a previous game or start new one	Previous Gaming knowledge- Similar layout to Minecraft	Fully	N/A
2.	Settings-Startup Screen	This allows the player to adjust different layouts of the game making it easier to play	Previous Gaming knowledge- Similar layout to Minecraft	Not Met	As this what not an essential part of the game I did not add it. However with more time it would have been one of the next things to be added
3.	Load into games-Startup Screen	So you don't lose progress of your previous game and continue from where you left off	Previous Gaming knowledge- Similar layout to Minecraft	Fully	N/A
4.	Start new games-Startup Screen	To start a fresh game	Previous Gaming knowledge- Similar layout to Minecraft	Partially Met	The game doesn't have saved islands as it would take too much time to implement. As you login a new or create an account a new game is started
5.	Map Layout-Island	Without this feature the player wouldn't be able to move around the world due to their being no world. This feature makes the game unique to different games (starts to separate them into different categories)	Previous Gaming knowledge as well as more in-depth research- Similar Map layout to Ark survival evolved	Fully	N/A

6.	The sand-Map Layout	This allows me to create the atmosphere of an abandoned island. Using the sand as a beech of the island.	Previous Gaming knowledge as well as more in-depth research- Similar Map layout to Ark survival evolved	Fully	N/A
7.	The Sea-Map Layout	This allows me to create the atmosphere of an abandoned island. Using the sea to complete the beach image.	Previous Gaming knowledge as well as more in-depth research- Similar Map layout to Ark survival evolved	Fully	N/A
8.	Rocks-Map Layout	To create the atmosphere of the island. To possibly use this as raw materials to create machines/tools	Previous Gaming knowledge as well as more in-depth research- Similar Map layout to Ark survival evolved	Partially Met	There is rocks in the game but as a texture (around the edge of the island) representing drop off points just no physical objects
9.	Tree's-Map Layout	To create the atmosphere of the island. To show this has a jungle layout. To possibly use this as raw materials to create machines/tools	Previous Gaming knowledge as well as more in-depth research- Similar Map layout to Ark survival evolved	Fully	N/A
10.	The character's clothing	This will emphasis the fact the character has been standard and left to rot	In depth research from the internet and Gaming magazines. Similar character looks to the character in rust	Not Met	Not an essential part, more of an added extra if any spare time were available. This feature would also take a long time to make and be pointless unless a third person camera was added what would destroy the aim of the game to be as realistic as possible
11.	Pickaxe-Tools	Allowing the player to gain raw items by mining them as they would do in the real would	Previous Gaming knowledge- Similar layout to Minecraft	Partially Met	A pickaxe itself wasn't added to te game but an axe was used to cut down trees. I would of added extra tools next and a crafting interface to make them.
12.	Part Of Books	This goes back to the storyline where a book is left from how the player got there. This book helps the player know how to survive by reading passages from this said book.	Reading of the Book The Knowledge	Not Met	As there was nit enough time to create a back story or add experiments/ways to survive I felt like adding the book pages became a waste if time so focused on more crucial aspects of the game.
13.	The Character Itself	Allows the player to move around with an	N/A	Fully	N/A

		entity instead of being invisible what would defeat the whole purpose of the game.			
14.	Monkeys- Entities	Monkeys will give the game character, by adding a combat aspect into the game as well as bringing back the idea of science to kill and eat the monkey to survive.	Research into this type of game genre	Not Met	This feature was suggested with the questions from the users. As entities took a long time to code needing the health bar a weapon to attack them and extra code knowing where and when to spawn them this led to, too much time being taken up so was decided to be dropped towards the end.
15.	How many day's survived on the island- Layout	Using the day's survived as the scoreboard it will make the game more interactive and not just a play once kind of game.	Unique- Survey Results	Partially Met	Days where added into the game and a display was to be added next showing how many of these days where survived but I ran out of time
16.	Heath Bar	Without a health bar the player wouldn't know how much health he has left before he starves/dehydrates/etc to death.	Unique- Survey Results	Not Met	As this is a key part of the game I was going to implement it first but due to deciding nit to add entities and condition to lower the player health it no longer was needed but would be one if the next things to be added if more time were available
17.	Sound effects	This gives the game some more character allowing the player to be more immersed into the scenario the character has been put in.	N/A	Fully	N/A

MAINTENANCE AND LIMITATIONS

The largest limitation of the game is that there's no objective as there are no entities or environmental condition lowering the health, so the game is limited to exploring the game/map and chopping down trees. As drowning or hyperthermia wasn't added to the game the player can also reach the end of the map in the water and fall off therefore the map is also limited. As only one noise effect was added only that noise effect will be heard therefore walking on different terrains such as sand, bottom of the sea, on rocks and swimming the same terrain sound will be heard (in this case the walking noise) limiting the sound effects.

Future maintenance could involve adding different sound effects, such as walking on different terrains swinging an item or destroying/picking up blocks. I could have also added the health bar and have different entities which attack and decrease the health bar with fruit found on trees which have been chopped down to increase the health bar back up. Finally I could of added experiments to gain drinking water so the player didn't dehydrate, added different condition such as heat stroke with needing mud to block the sun and creating a cream from different items found in the map to cure the heat stroke and increase the health what was taken off and many other environmental conditions.

If a third party such as my stakeholders wanted to add more features that could be easily done as my code has been commented throughout the process, to explain different parts of the code. Each section is created into either a function or procedure making it easier to understand and each script is in a folder structed in classes making it easy to find and modify. A third party only needs to know how to code in c# as well as that is the only coded language, I have used in this game to create it as well as my game being fully open source due to the game itself and build being released.