CAR RACING GAME

By James Gammon

Name: Racing Game

Contents

Scenario/Problem Definition:

For my computer science project, I will create a car controller game which allows the user to maneuver a vehicle around a track. The game will in cooperate a single vehicle moving around a track/area thanks to the user inputs. The game will include a variety of stages where there are obstacles to drift around, each with different complexities. Included a tutorial arena at the start where users can learn to control the car in different ways before entering the different maps and game modes. The tutorial area will lead into different arranged maps, with different scale obstacles and different difficulties. There will also be options for the type of game-mode, including a time trial map or car controller.

A menu option will be available where there are different available choices to new players or existing players, the option that they pick influences the part of the game that they experience. The user activity menu controls the type of game experienced by the user; here they will decide on how to play the game specifically to them.

My game will only be single player offline and will not have an online opportunity to play online competitive, there is no option to play with bots either. The game will consist purely car controller around a selection of maps, the user will be the only active player on the map.

The game is a menu-based game, by opening the game a menu selected will pop-up, displaying the maps and possible game experiences.

Stake Holders:

My target audience for the game has no real age range, although it may be suited to people who enjoy car controller racing games, single player, who could potentially be young adults or older children; There is no ideal age demographic as the game is applicable to anyone who has an interest in racing games/ car controller games, and this is not a specific interest to a single age group of people.

The game is not targeted directly at a demographic there are no required skills to play the game. Its purpose is a car controller game which can be played by anyone with no skill, and therefore there is no target audience apart from those who take an interest.

Hardware and Software:

Hardware: Is hardware inputs(keyboard/mouse) and a PC with sufficient ram and CPU, the game will not have seriously intense graphics so a general-purpose PC can run the game perfectly.

Software:
Unity, c#, PHP, My SQL

Complexity:
Another complex aspect of the game is the use of connecting the SQL database to the unity game, by using PHP, the connectivity between the user logging into the game and displaying usernames on a leaderboard at the end of specific game modes, which will be a complex challenge using many different game modes.

Bubble sort for displaying the data in the table, this could be difficult when using SQL and c sharp, that the code uses the correct information about the user to display the leaderboard.

Binary search for displaying the correct game mode and menu could be difficult, by using a binary search effectively, the program must identify the game modes and scenery, and then actually display the object.

Analysis

Problem identification

Problem identification –

While looking currently at the racing car range of game I realised there was a surplus of competitive racing games, online and pvp especially. While researching there was no designated games specifically for car controller mechanics and time trials especially. Learning car controller basics was an additional feature of many other games (Forza horizon, Grand Turismo) which were specific to the games themselves and did not improve the skill level of the user. Car controller games are very intense online especially when the skill gap between newer players just joining the games and existing players who are experienced dominate over new players, with no real way of improving basic skills apart from practise.

The features of the car controller game target this problem between newer, unskilled players and existing, players. A general game which in cooperates the existing controls of many car controller games paired with different difficulty maps, and single player game modes.

By creating a game that allows users to practise car controller basics the problem of skill gap for racing games could potentially tighten making games for newer/older players more enjoyable.

Stakeholders

- Older children (possibly)
- Young adults (possibly)
- Gamers who enjoy racing games
- People learning to play racing games / car control games

The demographic for my car controller game is primarily focused on keen users of their PC, those who enjoy playing easy, self-competitive games which are unskilful to an extent. The stakeholders can range from people who enjoy light gaming and non-intensive racing to beginners learning the basics or car controllers and the principle of movement.

The stakeholders for this software game are young adults who are looking for a casual car controller game to play, mostly by casually gaming by themselves, not looking for a game with a real competitive aspect, on the other hand there are people representing the category who are looking to improve their developed skills of car controllers.

There is already a large amount of racing focused game available.

Why is it suited to a computational solution?

Many racing games now are starting to generalise in the principles of the game, including controls and basic tracks, also the developing technology surrounding racing games has drastically changed the mediums games are played on, most played on either controller or keyboard and mouse. A computational solution incorporates practise of mechanics and variation of gameplay on a single game platform.

Computational methods that lend itself to:

Problem recognition

Creating racing games on unity is relatively simple, including different aspects of a game, different maps, game modes and difficulties as well as custom car mechanics are a problem, which allows the user to change the sensitivity of the steering and power diverted to the car in game.

When this problem is overcome about the complexity of the interchangeable aspects inside of the game, the rest of the problem is making the settings usable at the user interface, and that the chosen game mode applies to the user.

Problem decomposition

The problem is complicated when trying to approach is due to interchangeable game modes.

- After the user has passed the login processes a menu will come up
- A menu will come up on the screen asking about the game mode
- This will pass into another decision about the map style
- The game will display the chosen choices by the user (calling different functions)

After each of option is picked, the game set-up has been completed. Unity can process each of the different options which either display the time-trial clock or not, onto the chosen difficulty of map.

Divide and conquer

Even though there is a small number of steps, the steps individually are very complicated. Each step requires a lot of attention due to the delicate code, if not written correctly there can be a lot of errors, the game is very versatile and therefore by using modular programming and functions the user can call the desired gameplay, therefore lots of smaller steps are important, the steps lead into each other, and the problem will be solved.

Abstraction

While there are lots of different ways to play the game, abstraction is necessary to keep the fundamentals of the game simple. Naturally, the user will be given choices for the different game modes.

A lot of current game incorporate game modes that are unneeded, the provide the same gaming experience; difficulty and some visuals and are present to make the game more diverse. My game will solely include maps that provide different challenges to the users and game modes. My game does not necessarily have to be big challenging other online games however it is going to include a diverse game experience.

By leaving out maps and game modes that provide the user with the same experience I can keep the internals of my game simple, by providing too many choices to the user the code is more likely to fail or include an error.

Having too many ways to play the game can cause confusion for the user, and an unworthy amount of time spent on developing code and backgrounds that are not going to be used.

I James Gammon – 27228 – 8292 – Abbeyfield school

Interview

I will email some different demographics about my PC game, enticing information about the ability, interest in the type of game and the uses. I will email the same set of questions to people of the same category, and different questions depending on the type of person being asked; there can be some follow up questions based on the replies of the original.

I will be giving my questions to 3 different demographics, to try and gain a wider perspective of the game I am creating.

- Casual gamers
- People leaning to play racing gamers
- Younger age demographic

I have asked casual gamers as they have experience with different mechanics in games, gaining their view on a dedicated practise/ free-roam game.

I have asked some people learning to play racing game to gain their view on the usage, whether they prefer to gain skill/experience playing against anyone, or a customizable single-player mechanic focused game.

Younger age demographic has been sent questions because of the variety of people playing racing games- there is no real age stereotype. Therefore, I will need to send my questionnaire also to a larger age-rage to gain ideas/views about the people using car

(Casual games)

1.While on racing games, do you feel like there is a lack of practice, and more competitive styled games.

2. While playing racing games, do you feel like there is enough attention paid to practice arenas?

3. Would you play a customizable, practice racing game to help yourself build the fundamentals of racing?

4.While games are becoming more competitively focused, do you feel there is a need for a time-trial, practice game.

5.Do you feel like a game of this style is too like current game aspects.

What do the questions mean

Questions [1+2] are asking the casual gamers if, when playing racing games, they feel like enough attention is applied to free roam arenas, and whether they are on par with the detail and incorporation included on the main focuses of the game – also gaining information on whether these areas are used.

Question [3] directly asks the casual gamer about their use of free roam/ offline areas of the game, from the answers I can plan to see how my game will be used, including the time-trial aspect, car controller and free roam around a map.

I James Gammon – 27228 – 8292 – Abbeyfield school

Question [4] is using my idea against already existing/ similar opportunities to be played, depending on the previous answers about the attention and quality of the car controller / fundamentals the casual gamer could respond about the desirability of this idea, or the opposite the fact they are not needed depending on the game; however, my idea included the mechanics of all racing games.

Question [5] asks about the similarity between my game and existing ideas, especially those already incorporated in popular racing games, whether the idea is way to like some games but just with a different focus.

Answers (Sam – keen gamer)

1. While on racing games, do you feel like there is a lack of practice, and more competitive styled games.

Yes, the game does not exactly tell you the best way to beat your opponents, and strategies that can sway the win in your favour if you play smart, for competitive side, yes and no, the race itself is competitive but there is no real ranked game mode where you compete to be the best of the best.

2. While playing racing games, do you feel like there is enough attention paid to practice arenas.

Yes, in most racing games you can put yourself on tracks against ai or not, this helps you learn new strategies and practice before you take on big boy players.

3. would you play a customizable, practice racing game to help yourself build the fundamentals of racing?

yes

4.While games are becoming more competitively focused, do you feel there is a need for a time-trial, practice game.

yes, time trails in racing games are good because it shows how much you improve and shows if certain strategies work better than others, can also show what cars and upgrades are better than other, practice games should be optional.

5.Do you feel like a game of this style is too like current game aspects.

I do not know, I do not play enough racing games

Answers (Rohan – casual gamer)

1.While on racing games, do you feel like there is a lack of practice, and more competitive styled games

Yes, I do think that racing games have a lack of practice. Practice is not introduced that much into many racing games and practice arenas are really useful to help you become better and help you and your patterns. In terms of competitiveness in racing games have become worse over years since those type of games is losing popularity

2.While playing racing games, do you feel like there is enough attention paid to practice arenas?

Practice arenas are really underrated in racing games and people do not take advantage in those arenas that the developers are providing. They keep many practice arenas quite basic as well if they do have it with sometimes a dull background and it just having a grey wall and a few places to drive around

3. would you play a customizable, practice racing game to help yourself build the fundamentals of racing?

Yes definitely, it would be a more fun experience for the game and would make players have a much more enjoyable experience and get the skill and practice out of the game

4.while games are becoming more competitively focused, do you feel there is a need for a time-trial, practice game.

Yes, since the skill gap of players are always becoming bigger and bigger a practice time trial

is needed to keep the players mind fresh in the game and for them to be getting much better and more advanced at those levels

5.do you feel like a game of this style is too like current game aspects.

In racing games, there is not many practice arenas to play in them and for you to get the time trial in the game. One game that introduces the time trial practice arena is Mario kart and Mario kart is one of the most popular games in the world

Analysis

Some of the casual gamers have used car game experiences before, other stakeholders have not. Rohan has had a lot more experience with racing games compared to Sam, Rohan could answer with better detail about the inclusion of different free-roam areas inside racing games. Compared to Sam who has not had as much experience with racing games Rohan talks about his inexperience with practice arenas, while Sam has not played enough games however says the practical use of maps and arenas is helpful when dealing with new/hard to grasp mechanics.

Both Sam and Rohan are stakeholders who frequently game; there answers suggested about the practical use of my solution, they both replied showing a lack of practice arena in racing games to build fundamental skill from the first question. There will be a clear path for users of my solution to better their skills at racing games through provided applications to these experienced gamers.

Interview questions (people learning to play racing games)

I will be asking these questions to people learning/ wanting to play racing games. If I have any further questions, I can email them more, the questions will help me better understand the demographic of newer players playing racing games.

I James Gammon – 27228 – 8292 – Abbeyfield school

(Learning to play)

1. Does the idea of new players racing online against already pro players off putting? Especially racing games which have a high skill gap?

2. Would a universal racing mechanics game (like aim trainer for shooting games) cold potentially be useful to you?

3. Are you interested at becoming better skilled at racing games using a different method to casually play online?

What do the questions mean

Question [1] is directly addressing new racing game players and any concerns they have about playing. When playing a game, it can be unenjoyable playing with people a much higher level than you, especially if this is the only way to practice becoming better.

Question [2] inquiries about the use of racing mechanic games, and how it can be useful to new players just starting out, to improve control of the car, speed management or for fun, focusing on building skill while existing games(forza) are there to be played seriously online. Asking the user would a relationship like this be useful to them.

Question [3] asks the user about the demand for a game like this to newer players, do they need an entirely new game to learn car controller or is this out of there league.

Answers

1. Does the idea of new players racing online against already pro players off putting? Especially racing games which have a high skill gap?

Yeah, pros playing games when you are trying to learn the mechanics are very annoying and I can only image how this is

2. Would a universal racing mechanics game (like aim trainer for shooting games) could potentially be useful to you?

Yeah, it would be cool, especially if the game is chill and you are the only person playing with no stress and simple focus.

3. Are you interested at becoming better skilled at racing games using a different method to casually play online?

Yes, sometimes by playing online can be repetitive and you cannot focus solely on the car control with lots of other factors. It would be good to see something incorporated.

Analysis

The newer demographic is going to approach a racing game wanting to gain as much experience and skill as possible as quickly as possible – which was shown in the answers. While maybe not playing driving games as an immediate, the first question implies they have had experience with a large skill gap plays and with pros. Promoting the idea and becoming familiar with a practice arena and skilled developer, a newer gamer would love to play as much as possible for as chill as possible.

In his answers Daniel says talks about his ideals to become better equipped at racing games, a practice arena aids his despise for pro players dominating over newer ones (possibly previous experiences), maybe an idea like my solution was useful in a different game and is keen to see something like that in cooperated into the driving game category.

Interview questions (younger demographic)

Younger demographic

1. When playing car games, does your ability you play at matter to you? e.g., your skill in handling, hand eye coordination
2. Do you enjoy playing racing games to improve? Or just for fun
3. Do you enjoy playing free-roam areas in racing?

What do the questions mean:

Question [1] asks the younger user if skill matters when they are playing racing games, whether they will therefore be interested in a way to improve/different style of game

Question [2] is asking the user how seriously they take racing games/ car games and whether therefore they will even want to aim to become better, separate from the interest of the idea.

Question [3] asks a younger user if they have any previous experience with car controller games, whether the younger age demographic is accessing them.

1. When playing car games, does the ability you play at matter to you? e.g., your skill in handling, hand eye coordination

Not really to be honest I just play a lot of the racing games for fun and the skill I play at does not matter to me too much if I'm having fun.

2. Do you enjoy playing racing games to improve? Or just for fun

Being good at a racing game is probably more fun than playing and being bad however I do not play racing games and set out to become the best at them, I play these games if they appear fun to me

3. Do you enjoy playing free-roam areas in racing?

Yes, sometimes free-roam racing games are more enjoyable then racing and competing with others, the focus of driving games is controlling the car so playing free-roam areas is also very enjoyable.

Analysis
From the answers I can immediately tell that stereotypically a younger demographic plays racing games for the experience and the competitive mindset has not been introduced to them as seriously just yet.

By answering the 3rd question, a younger demographic can apply themselves to my solution at a different approach from a more experienced gamer/ older user, by simply driving round and having fun with no set boundaries, unlike the uses from different stakeholders.

Sophie answered the questions with the same principle, driving games at her age are not necessarily something taken as seriously as possible and an in cooperation of a free roam/ mechanic developing area is a better/ funnier solution for them,

Research

Existing similar solution

- Free roam areas (forza horizon)
- Art of rally
- Inertial drift
- Track mania

Forza horizon (free roam)

Overview
Forza horizon is one of the most popular car controller games in the world, which can be played on almost all consoles and devices. By using different input devices, you can control the car on the screen around the forza map. Forza is an online game which puts the user in a server full of other people playing the game. Forza offers a free-roam area which is based on the same principle as my game.

The free roam area is very simple as it uses the exact same map as the online variation, the practice mode just removes all online players, races, and allows you to pick a car of your choice. There are some different tracks available inside of the large free-roam map which the user can pick from.

Parts that I can apply to my solution:
The base concept of the free-roam area is like my solution. The use of a car around a map, without competition, however my solution will emphasis more on the different maps and give more to the user about different ideas to incorporate in my solution, such as an added time-trial mode and different maps. From the idea I'd also like to incorporate the interchangeability of the different cars in to practice; some may be faster, some with better control, my solution will then be better suited to the user with a wider variety of vehicle types.

Art of rally

Overview:

Art of rally is a 3rd person stylized racing game which the user can drive stylized cars around real-inspired tracks inside of the game, online to compete in weekly challenges and offline against AI or bots. By using input devices, the user controls the car around the life inspired tracks, mainly focusing on the cars over ability to drift. The game can be used offline where maps and vehicle appearance can be changed, and online where trophies and challenges are completed.

Parts that I can apply to my soloution:

I will different aspects of this game to the soloution, the camera angles which the user views the car from is a good aspect of the game, meaning its different from a lot of other games on the market. Another aspect I can apply to my soloution is the intensity of the game, Art of ralley is a chilled car controller game when playing and no real compertition – this is how im aiming for my game to be like.

Overview:

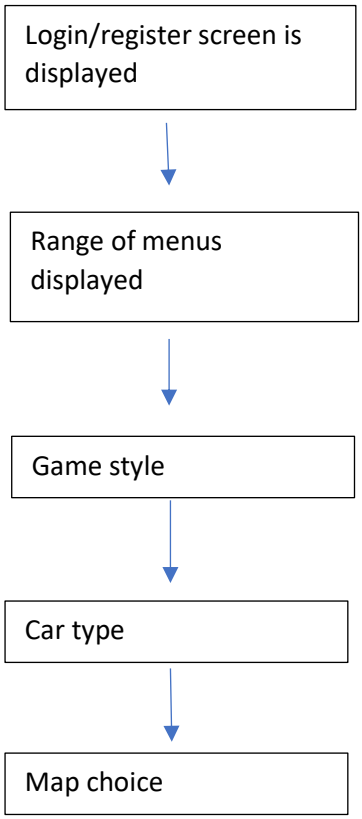
Inertial drift is like a lot of other racing games available to play.

Track mania:

Features of my posed solution:

Initial concept of my solution considering the research:

After considering research, my solution has advanced from the original idea, including similar ideas from research examples to become a better overall application.

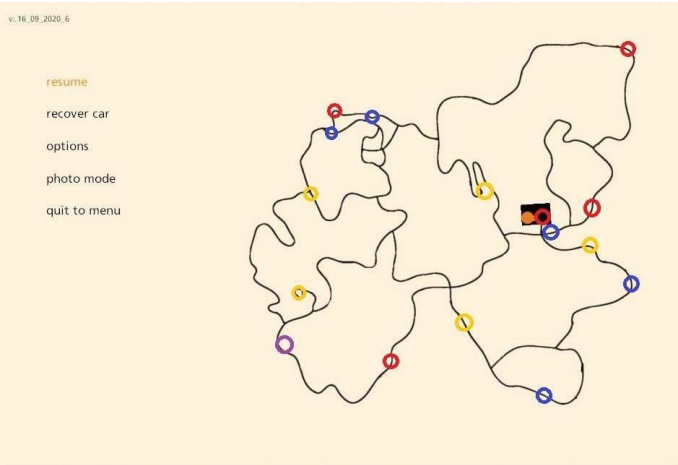


My soloution will begin by showing the user a screen with a login operation and register operation and when they are recognised in the system, the game graphics and menus will display else they are asked to repeat the login process. There will be 3 different menus incorporating a customizable game aspect: the game type (time-trial, car controller or explore , and interchangeable map choice: easy map, harder and hardest) The menus will come as different slides with multiple options per screen and the user picks the options.

The soloution will automatically pick up the keyboard/mouse inputs from the user; there will be an in game menu for further settings. The soloution will use the inputs from the computer to control the car on the screen around the setting.

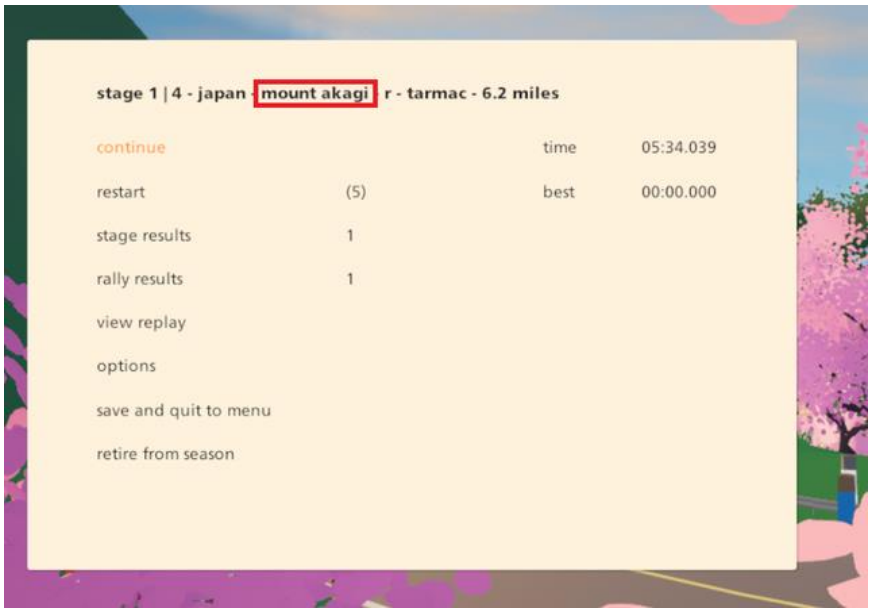
I James Gammon – 27228 – 8292 – Abbeyfield school

Art of Rally design GUI's



This is the art of rally pause menu, its menu features a simplistic design with options and pictures, so the game looks complete.

The game however is limited to this single track and is only a racer, I will incorporate this aspect of a simplistic design into my menus, however a more customization options will be present as a final concept of interchangeability.



This is the art of rally final screen menu and is an idea I would like to incorporate into my game, at the end of the stage at mount Akagi there is a time and a best time display text, with an options menu and a save and quit function.

I can build on this by adding a more cartoon is GUI, the functionality of the end game menu is idealistic for my own game, however this game is meant for a much younger audience, I am to tailor this functionality, though it seems a little bit overwhelming for a single screen to my game.

Taking away functions displaying the best time, save and quit to menu, viewing results and restart function.

Limitations of my solution:

One limitation of my solution will be the limiting input devices of the game (I am only using mouse and keyboard), making the number of potential players limited. A lot of racing games are on console (Xbox, PS) which use a controller to input racing commands, my game solely uses keyboard and mouse inputs. I could implement an algorithm which could include the inputs of a controller however this in unnecessary as my game is for PC players anyway, console players are a completely different demographic and I do not expect my game to be too large style, in the future this is sure a functionality I can add.

Another limitation of my solution is the desirability of the game, from the research there is a lot of other car controller games that are very similar to the concept of my game, where I specialize in the custom aspect. Even though it was my choice not to include an online option due to the complexity, it could however be a limitation of my game for users wanting to play with each other, they could potentially find a similar game with an online feature; limiting the use of my solution to plays wanting to play by themselves or from people wanting to play together.

In the future also, if my demographic is desirably, I can host my game to an online survey and implement a custom online map.

Requirements

Software and hardware requirements

Hardware

A PC with capability of running the software, (higher core processor, average graphics card, standard input devices e.g., QWERTY keyboard) the standard input devices are what drive the game from the users' inputs to the car controller on the screen. A fast enough PC with internals is required to perform the ongoing tasks of the game, each slight input from the user needs to be registered quickly and accurately for a game as high paced as a racing game to work, especially when there is pressure from time and racing difficulty.

Input devices (mouse and keyboard) are needed for the user to control the car and play the game.

Software

Any software to be able to run unity

A fully functioning operating system

Stakeholder requirements

I asked to type of stakeholders for requirements, my younger sister, and a friend of mine, I have combined both answers to produce the following tables.

(S) = sister

(F) = friend

Design

Requirements	Explanation
(S)Clear instructions on navigation	I am expecting a wide demographic to play my game so a clear navigation through the settings/menu
(F)Abstraction of unnecessary clutter on screen for understanding what needs to be done	Lots of interactivity on a menu can lead to a confusing menu
(F)Simplistic design not coming across as boring	Again, I am expecting a variety of people to use my game, on first glace I do not want it to appear like a children's racer as that is not the demographics I am aiming at.
(F)Representative of the game, possibly game graphics	I want my audience to know my game is not solely for children, I think having my Menu GUIs represent the game is a really good idea as it gives the game a type of feel. A simple Menu is not very representative.

Functionality

Requirement	Explanation
Simple walkthrough of menus	Having a simple walkthrough menu allows all ages to navigate the game simply and effectively. Simple menus are often better to look at and make the game appear to be more professional.
A back button in each of the GUI's	For a game to be successful and enjoyable there should be a lot of freedom in the menus, to travel back and forth between options and to pause the game at any time. A back button enables this movement to happen.
(F)Way of a 'back button' in the GUI to access each menu	'Back button' as when navigating through the menus there can always be mistakes made e.g., you click new user when you are an existing user
(F)Each menu or aspect of the game can be accesses even when playing, quit and save for example, a pause menu and logging out	I think this makes the game a lot more interesting to be able to stay into.

Hardware and software

Requirement	Explanation
Standard peripherals (keyboard, mouse)	The player needs a form of input devices to be able to control the car on screen.

The keyboard must have access to arrow keys and/or WASD as a controller, this will be defined inside of the game.	This should usually be done by the WASD keys or arrow keys.
Normal computer specs to run the game. 4GB of RAM Intel Pentium and above processor Working peripherals Clock speed of 2Ghz +	Games on unity need a lot of useable RAMS and a quick processor to be able to function.
An operating system	An operating system is required to be enabled to play the game.

Success criteria

Success criteria GUI (Menus)

Criteria	How I will evidence
Large buttons on each decision, simple decisions per screen or decisions are defined in the walkthrough area.	Screen shot the next pages in the GUI which show decision screens.
Simplistic design which is usable with the varied demographic (abstraction) of items which can target one or the other. This also includes non-confusing navigation panels	Screen shot windows which the user interacts with, include tip boxes and text boxes for a simplistic design
User input text boxes are clear and know when to be interacted with on the user authentication page	Screen shot examples of how the text boxes are designed to benefit the user.
Option to call menu pause menu in game which displays functions such as: Resume game Main Menu Quit and save	Showing facilitating code which allows the user to call the menu options.
Subroutines across the GUI are called without fail	Facilitating code showed where routines are called, and functions are defined.

Success criteria (Game play)

Option to change game map/mode in game //done by calling back to the map select function in the game menu which leads to the game mode	Code to call back the GUI menu will be screen shotted and shown.
Leaderboard is displayed to the user in game – this uses the users fastest time scores	Print functions of the sorted leaderboard i.e., the initiating code will be showed.

Each of the game modes are compatible with each of the game maps, meaning I can play a time trial on each map, a tutorial on each map and a explore function on each map.	Facilitating code showing where functions are called, transferrable functions are called across programs. Showing the code for calling functions and concurrent processing of the map and the game mode.
Game mode functions are specific to the chosen game mode, so modes do not dissolve into each other	Encapsulating code facilitating the characteristics of each game mode will be displayed. The text on the walkthrough page, as well as the leaderboard on the time trial example.
Stopwatch clock presented on the screen	Screen shots of the actual timer presented on the game (counting in seconds.)
Vehicle is completely controllable with arrow keys	Unity code which facilitates the movement of the car as well as the concurrent processing of multiple inputs at once.
When the vehicle crosses the finish line the timer/stopwatch resets for a second lap	Code that flags the variable as 'true' once a trigger is entered is screen shotted and evidence which in turn resets the stopwatch.
The car game object can drift in and around corners	Code facilitating the key movement is screen shotted and displayed.
Car game object has wheel colliders meaning the car does not get stuck in the surrounding environment if it crashed	The code for the wheel game object which creates colliders around the wheels is screen shotted. As well as the hitboxes of the wheels
Best time label presented on screen	Screen shots of the actual timer presented on the game (displaying the best times)
Each map has its own specific functions, unique to it.	Unity code which facilitates the specific functions of the map and screen shots of game evidence.
Time trial game mode accounts for the user's current game time, the users fastest lap time so far, and displays both values on screen with the addition of lap counter.	Unity code which monitors the users best time and example of this being added to the fastest time array is shown.
Explore game mode includes no parameters for time completion or a set rule to follow.	Screen shots of the GUI display of the car navigating this map
Tutorial game mode will display checkpoints throughout the map and current tutorials on each game aspect.	Unity code which facilitates the disappearance of checkpoints once the user has entered them also code with initiates the tutorials being active/disabled shown.

Success criteria (Database management)

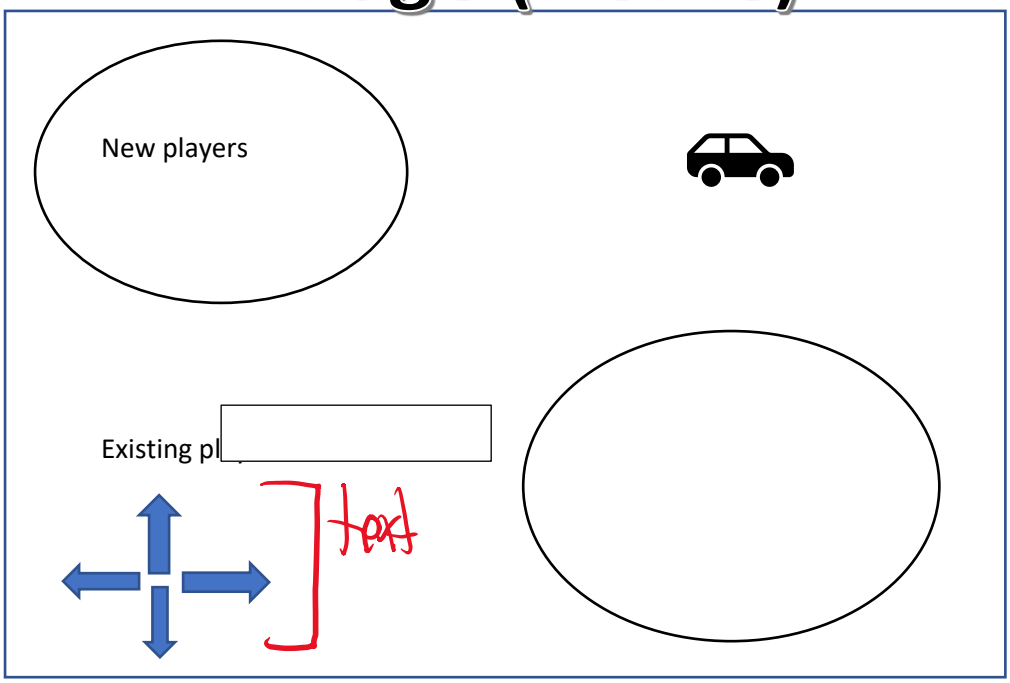
Criteria	How to evidence
Validation of user inputs compared to those stored in the array This includes a hash function for the password to compare the password entered to the user (hashing this password with the same characters) and comparing this one to the one stored in the database	Screen shots of the code which manipulates and compares user input data with the current data stored inside of the array

Leaderboard sorted via bubble sort	Code initiating bubble sort function will be shown, as well as how
The most recent user score is uploaded into the database	Code facilitating the input of the array and the manipulation of the data will be shown in screen shots.

Design

User interface design

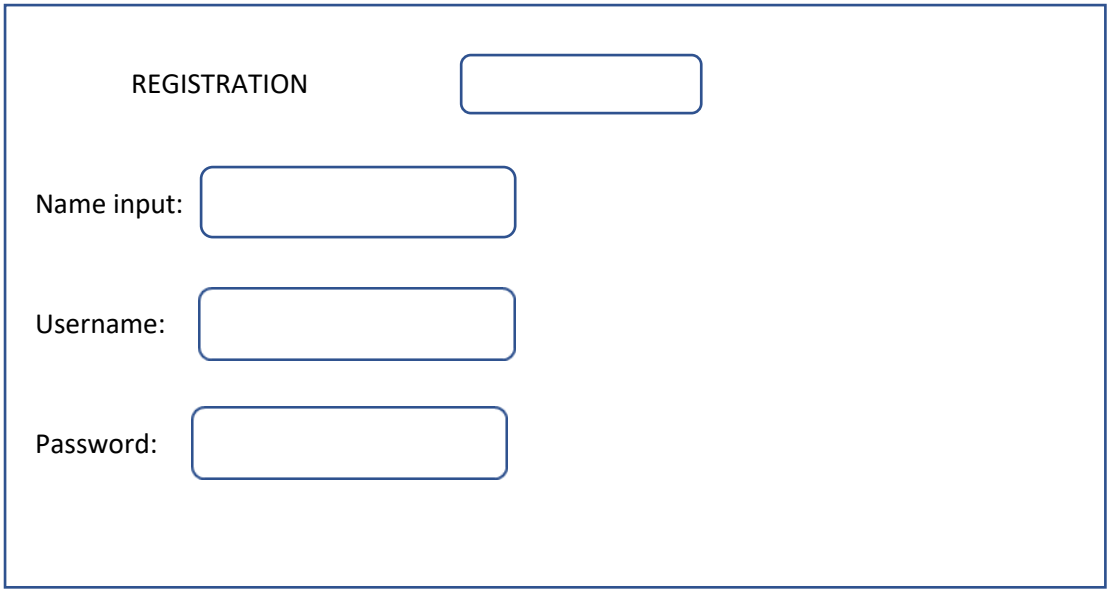
HOME PAGE **1st Page (Home)**



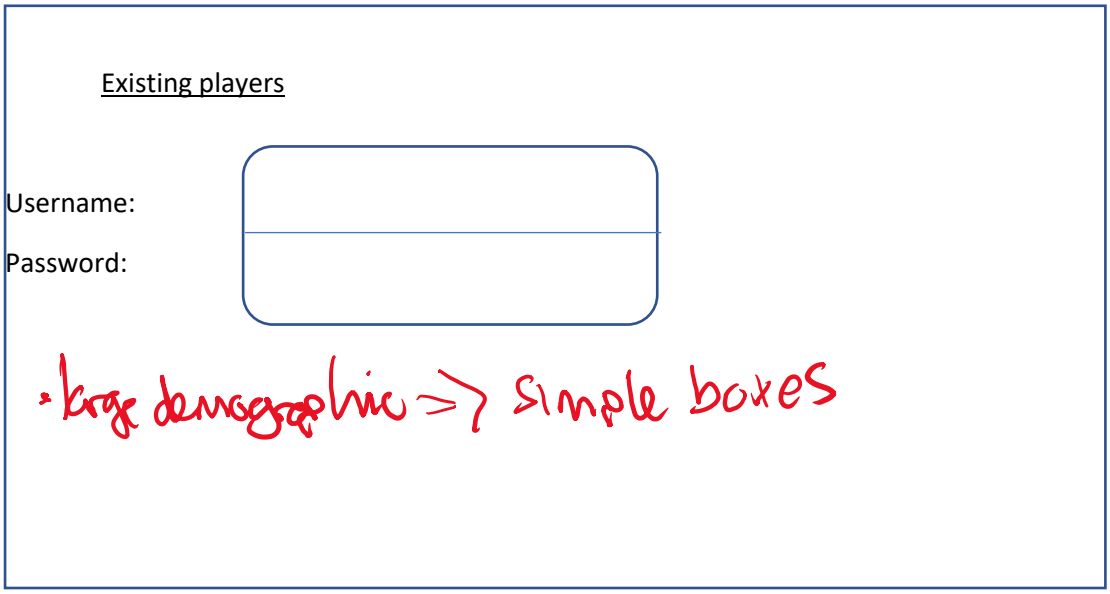
2nd page (NP)

I James Gammon – 27228 – 8292 – Abbeyfield school

NEW PLAYERS

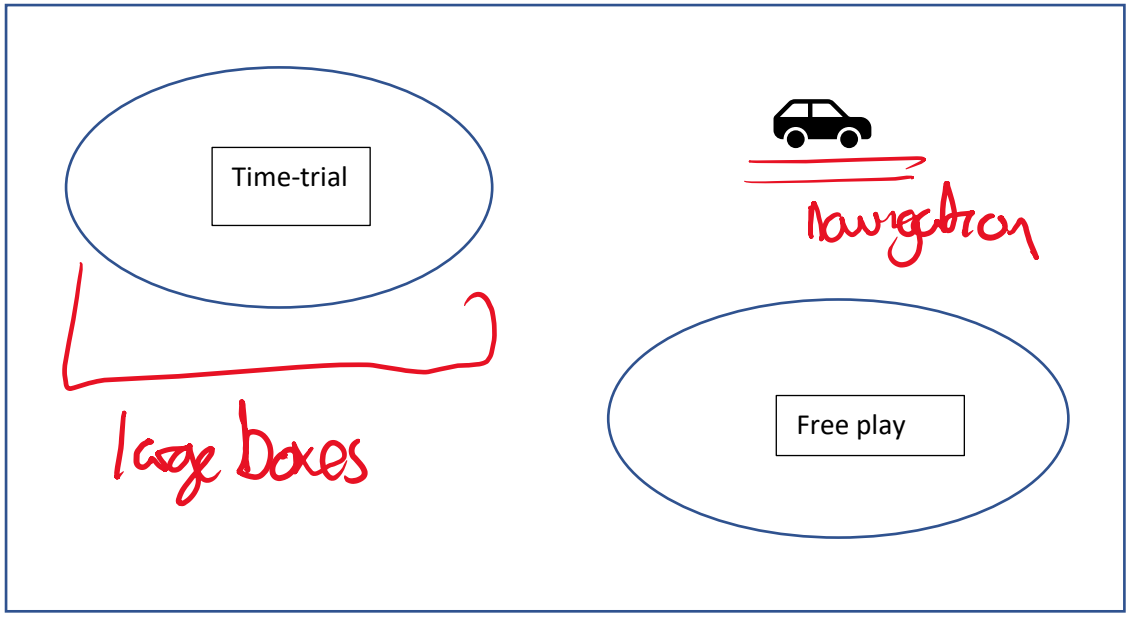


EXISTING PLAYERS **2nd page (EU)**



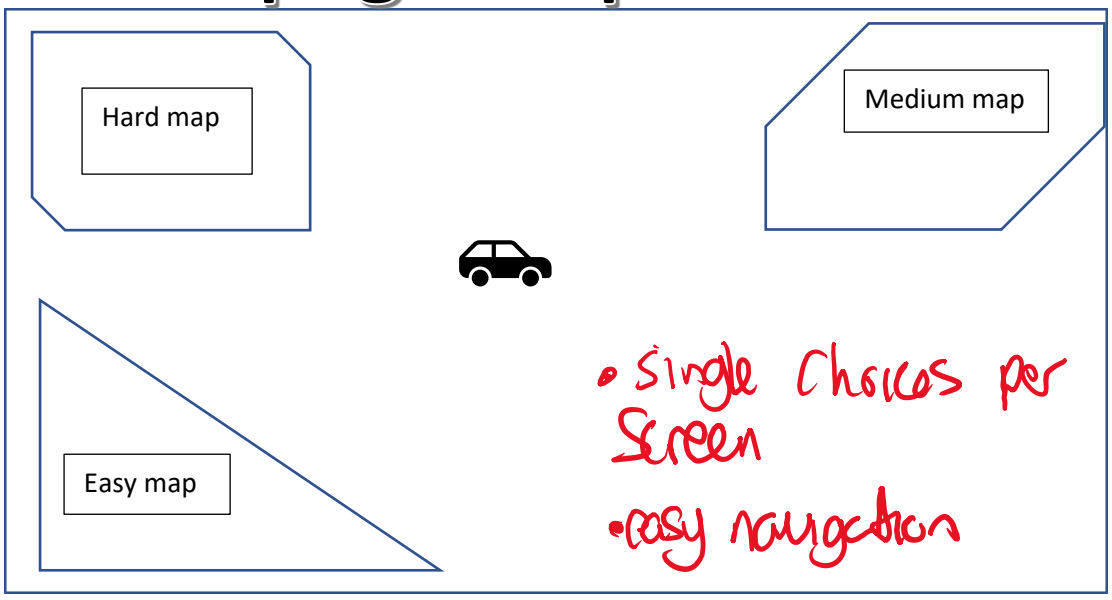
TYPE OF GAME MODE

3rd page game mode



4th page map

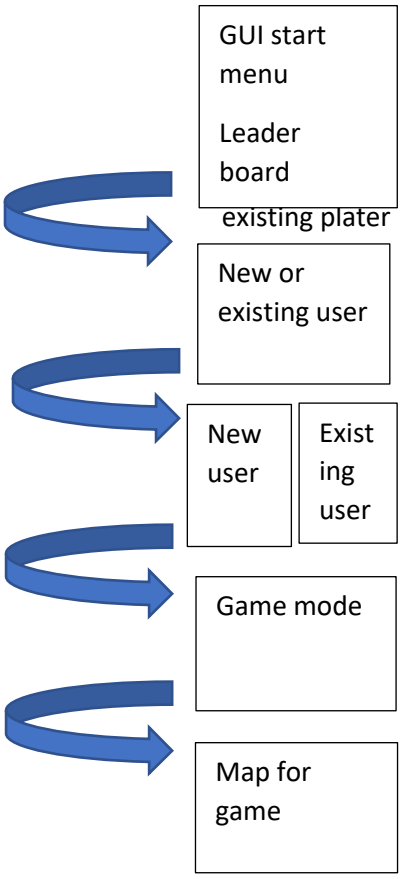
MAP

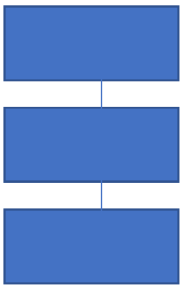


Usability features

A lot of the main program are slides of interface creating threads. The multiple page screens allow the user to customize the attributes of the game. Like OOPS there is a class of the game, each with an attribute of game stye (), map () and a user credential input.

Each option picked by the user creates a game matching to the user requirements and game experience.





Users desired game experience abstracted into game components vs how the linear menu sequence will be created.

The user has a version of the game that's personal to them; the menu breaks down the users desired game experience into sub-components to choose from; subcomponents such as gamestyle, game map will be customizable to meet the user requirement.

Having each attribute of the game at a different menu to be selected following a linear sequence, removing the complexity of customization, the user can be specific to each screen.

The GUI of the program needs to be accessible to all ages, hence the simplistic design. Structuring the GUI was important that users of all ages could navigate through the menu. The different screens minimize the amount of cluster on the user's screen, the user interface has been abstracted for all other unnecessary data.

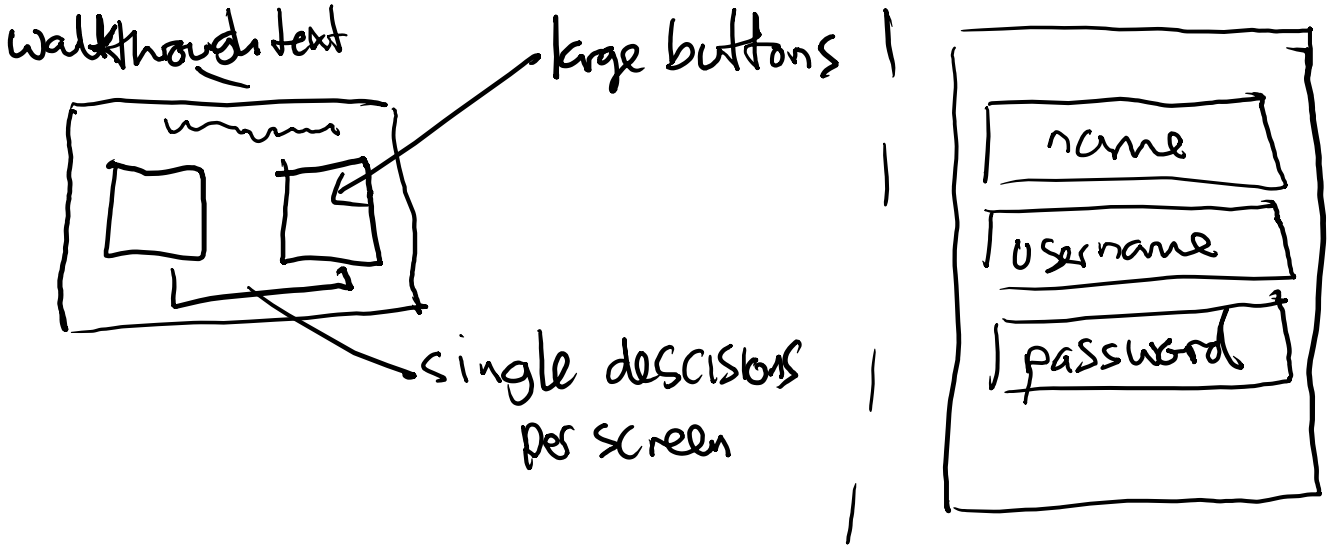
One of the biggest focuses is the specialization of the game. This is the most complex bit to code and to get right, the code is interchangeable. I will overcome this with a couple of different methods, first, abstraction. Unnecessary data that is not needed for the user requirements are taken out of the GUI to minimize choice's ability and lead to minimization of the code complexity.

My program must be suitably used for all the demographic playing the game as it must be easy to use for a younger demographic, this includes being simple and clear; as well as not being way too simple for an older which is a problem I do not want, to a more game veteran.

There are some distinct characteristics of my GUI that make the usability features better, these are the larger buttons, single decisions per screen to reduce complication, and a walkthrough box at the top of every screen to explain the options being presented, I have taken into account that not everyone who will play my game knows how games work or the functionality, this is my rationale for walkthrough boxes at each of the screen and when playing the game tutorials.

In the diagrams above, abstraction has been applied to the user interface to show the clear layout of the processes and what will happen on each screen, actual GUI functionality is included and not the appearance. I will pose this layout to my stakeholders as a blueprint on what my menus will look function and I can add smaller changes such as appearance later in the process.

Each new screen is presented in a similar format, so the user is familiar with with what is happening in the game. Large boxes for the user to decide detail as well on each screen so the GUI does not come across as boring for potential older users, and informative boxes dotted about.



Quickly, I posed this quick diagram to my younger sister, who often plays games on her laptop, including Roblox – which has some racing games. She is not very familiar with more complicated games such as registering and usability, however the proposition of the GUI really helped her navigate the game.

“I really enjoyed looking at james diagrams, it was very clear to me what needed to be done on first look. I don’t like playing some games as they are really complicated to work out how to play, the walkthrough texts are a really good idea.”

Link to the success criteria

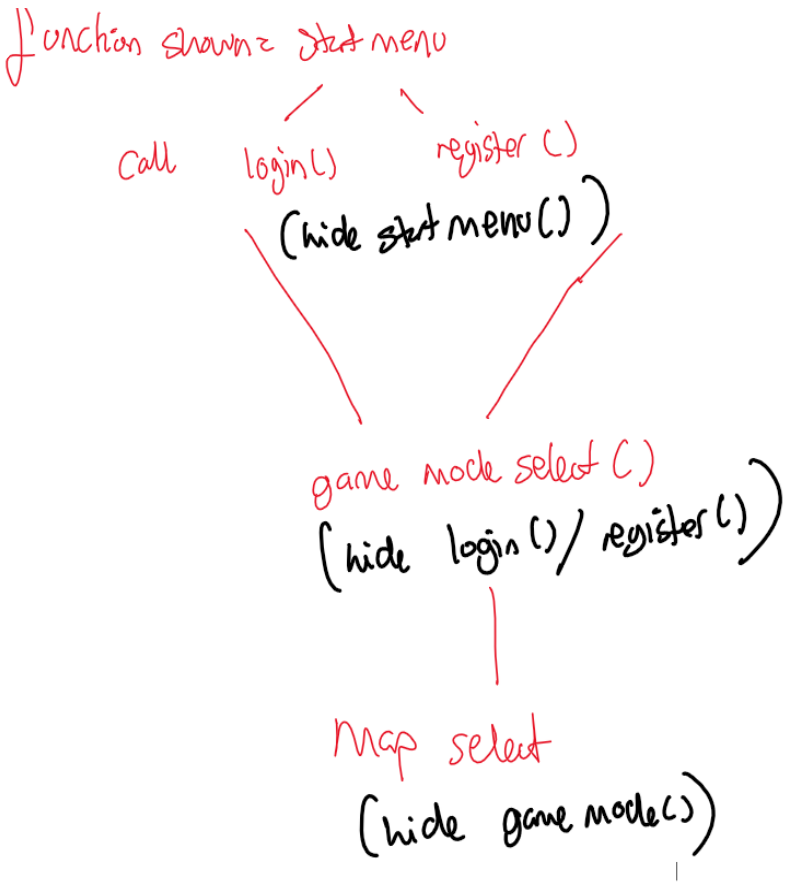
Start menu which is aesthetic to look at/ appears like a racing game.
Large buttons on each decision, simple decisions per screen or decisions are defined in the walkthrough area.
Simplistic design which is usable with the varied demographic (abstraction) of items which can target one or the other. This also includes non-confusing navigation panels

Setting up / Logging In

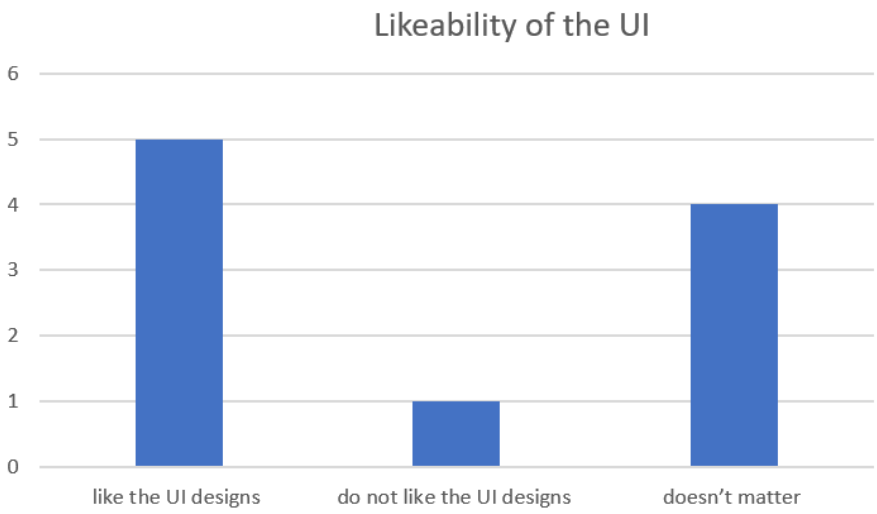
These 4 different pages are the first interfaces the user will interact with once the game is initiated; the user will begin making decisions on how they want the game to run. On each page there is a maximum of 3 possibilities per program choice; for users of all ages to have equal navigation.

Each of the menu interfaces will be called through functions in unity; each menu will call the next menu GUI in sequence.

These are defined through voids – which is a type of function



Stakeholder input



I asked 10 different people for their opinions of my UI design so far, here are the results.

I am pleased to see that majority of my results are optimistic about my design.

I have considered the answer which doesn't like the design and think this is because of the simplicity of the design, big buttons ect like it is a little boring.

The buttons seem to be in a linear format and are too simple – to stay with my stake holder idea about a simple design I have included a wider color palette for my Menus and GUI to satisfy the audience.

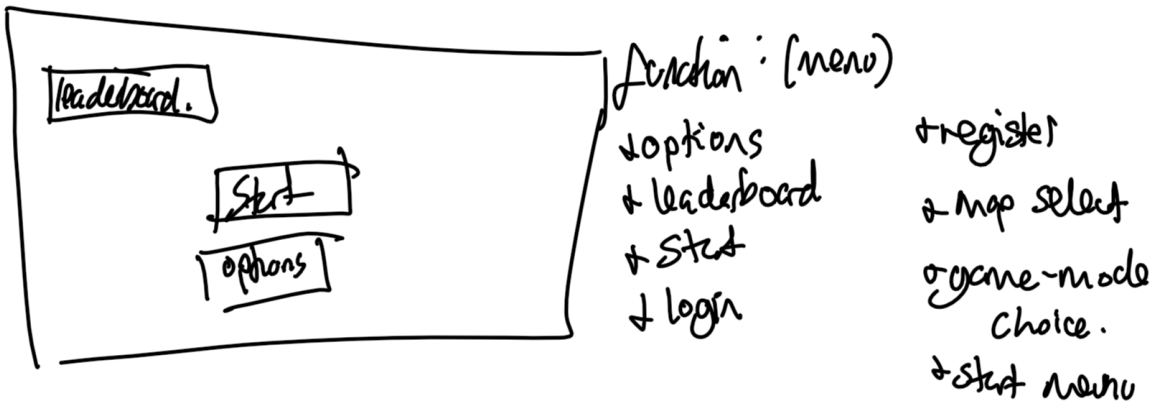
Start

The start box will be a simple box at the beginning of the game, the user will simply press the start button to initiate the game; this is like a lot of other games.

The GUI start menu will be the simplest design function with a simple command just to start the game. It will initiate the code to call the login/register button.

Start menu

The first screen the user comes into contact will set the stage for users. It will consist of a simple background with a singular start menu, any instructions or text boxes for the game will be included on this page.



New or existing user

The next screen displayed on the game with 2 different options in mind, a different menu screen will be called per the user's input: either (new user) or (existing user).

New user will call a database to input new data: username, name, password.

Or

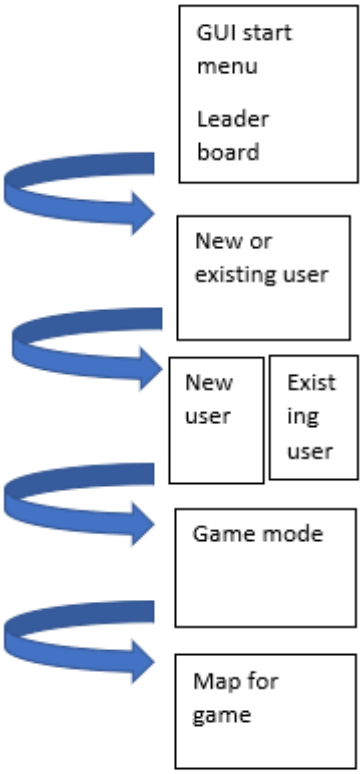
Existing user will call a database and compare input data to data currently stored (name, username, password).

For both options there are clearly defined input boxes for the user

Register Login

The second screen shows the first user activity page, which gives the users the options to log in or register, new users will input their name, username, and password into a database; existing users will have their credentials compared.

Swapping the map select and game mode select



In this earlier diagram it shows the game mode and map for game in that order, I thought that if I was a user, it would appeal to me more to decide what map I am playing first, compared to what I am doing on this map. Therefore, in my Menus I have changed the plans and chosen the latter way to set them out.

Game mode

This screen will be the first opportunity for the user to interact with the game. The options for map functions will be called depending on the decision the user takes. At this stage in the GUI the user decides between a Time-Trial game mode or a free play game mode or a walkthrough game mode.

The Screen will be simple (displayed above), the interact menu must be accessible for users – the only interactivity will be the choice of game mode.

The leaderboard for time-trials will be available to view in the initial menu.

Game mode Select

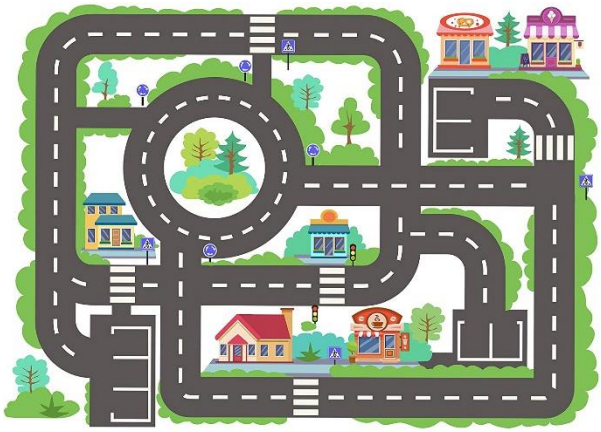
The third screen will show the game mode-select menu, which displays each of the options with a small description on what to expect

Game map

This screen will include 3 options for user choice, a game map which is hard to complete, one that is easier to complete, and another that is the easiest out of the 3 to complete.

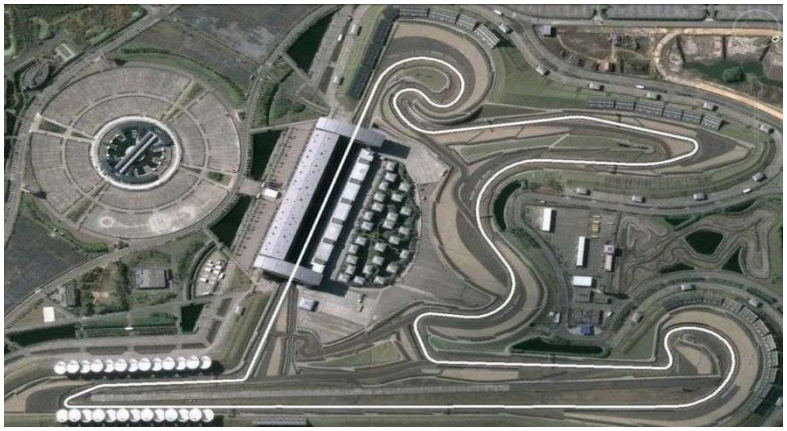
Map Select

The fourth screen will be the map select – as mentioned before there are 3 game maps to match the 3 game modes:



A hard map which is having a longer course and more turns, a more confusing aspect to be lost in.

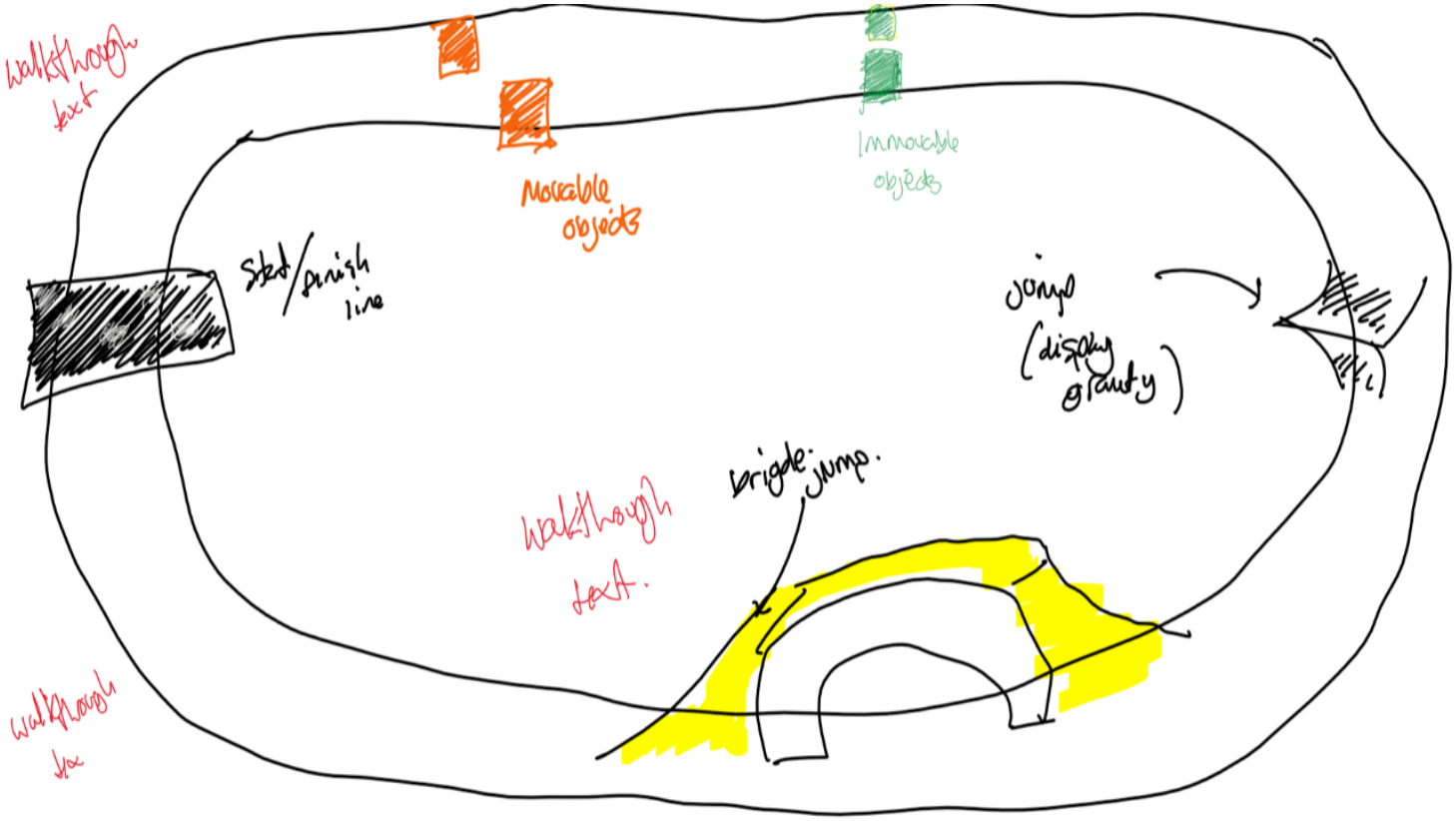
A medium difficulty course which is the smallest course however has the most turns – this is dedicated to a more game friendly user demographic who enjoy quick racing. Inspiration:



An easy course, basically a circle, with examples of objects to be found in the other two maps and no intense characteristics. Inspiration:

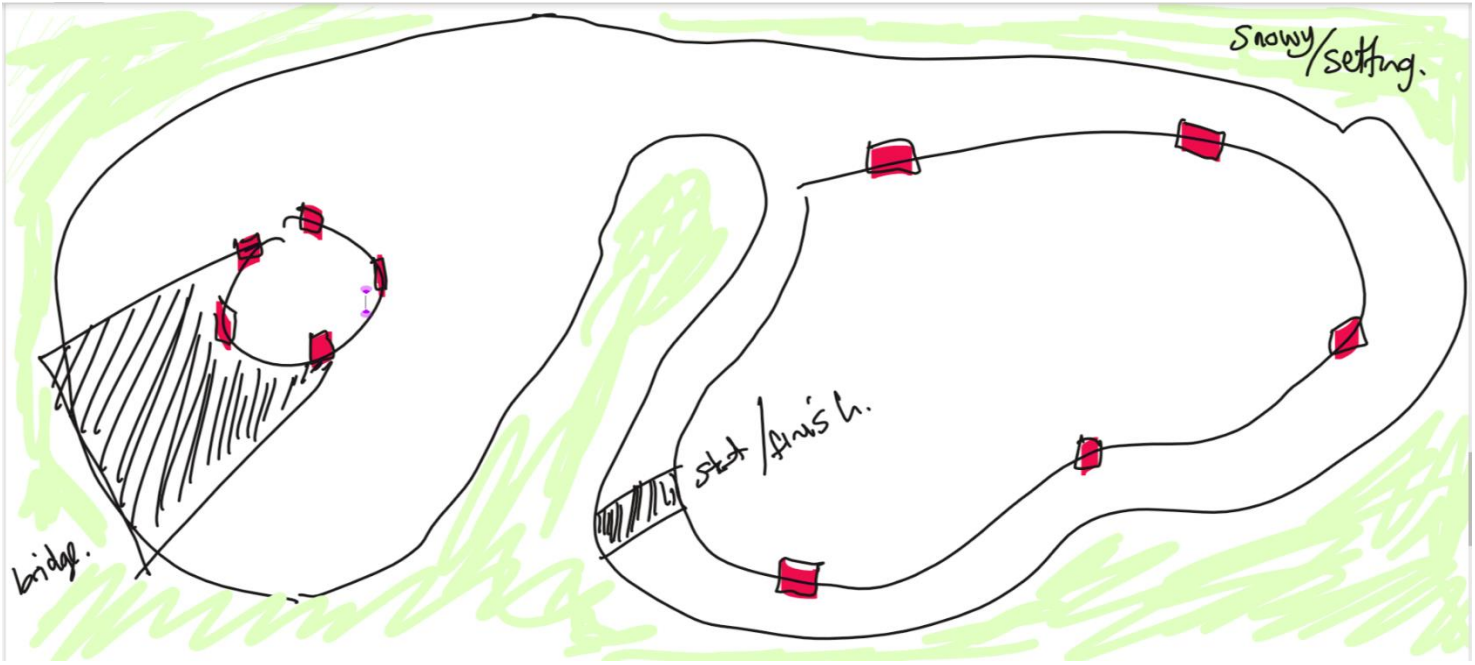
I James Gammon – 27228 – 8292 – Abbeyfield school

First map will be designed by me, it will be a walkthrough map of the different features of the game; this map should take about 1.5mins to complete

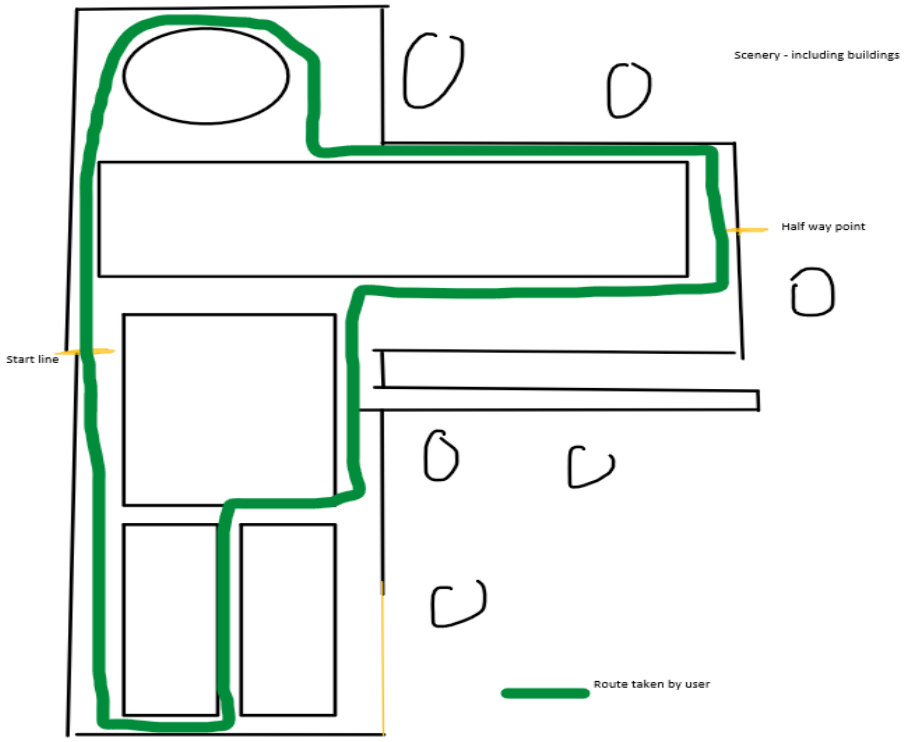


I James Gammon – 27228 – 8292 – Abbeyfield school

The second map can be played by users who have played racing games before – a simple racetrack game; this map should take around 1.5mins to complete



The third map will be the most complicated map for more experienced players inside of the game; this map should take around 2 mins to complete



Algorithms

My complexity in my Synopsis states that one of the hardest parts about my program is writing the bubble sort code for sorting the data inside data array (usernames/time-trails, and the time-trial score is the variable being used to sort).

This also includes the php language which I have never used before. My php will use an algorithm to search the database to return lines of ID's – testing to see if an input username matches another in the data base.

My algorithms will be composed of managing best score times operating lists inside of c# to produce and store real time lap times in the lap time format, changing scenes, setting game objects as Boolean for user interactivity.

Algorithms for front end	Algorithms for backend
Being able to input usernames and passwords and storing these in a private str variables	Checking if password and username are both above 8 chars for safety.
Menu select scene selects maps and time trials, combining the user choice of map as well as the user choice of game mode	Storing user input from unity and transferring this to a players database.
Pause menu algorithms which let the user (while playing) return to any menu screen, including quit, login and register.	Active scores inside of the game constantly being added into the database to print the best score.
Summon menu functions, setting menu panels active and disabling visited ones	Transferring the data variables from temporary stores in the c# script into variables from
Pause menu can be summoned at any point inside of the game, this allows the user to return to the menu for customization options.	Validating password and username variables are stored in the correct spaces

Start an elapsed Time variable when the user starts the lap	Validating there are no repeated usernames stored in the database, when c# checking for user input this as a Debug. Log.
Elapsed Time variable is restarted when the user finishes the lap	A leaderboard array is sorted and printed inside of the screen.
User elapsed Time for the lap is added to a float list with all time variables for this map and user; the list is sorted when lap is finished to display the fastest time on the GUI.	
Username is displayed from login/registration from the temporary value onto the GUI	
Pause menu can be summoned at any point inside of the game, this allows the user to return to the menu for customization options.	

Bubble sort for sorting best times in the leaderboard

Bubble sort for displaying the data in the table, this could be difficult when using SQL and c sharp and php, as I am using a completely different application software called MAMP and MAMP pro to conduct my PHP, I will have to make global variables so that the code uses the correct variable and time about the user to display the leaderboard. Inputting data variables as the correct forms is also tricky, data being presented on the lap time score must be a text, when it is inputted into the list array it will be a float, and when transferred into the php to be input into the data base it will be a string.

The bubble sort can be broken down into smaller problems to see its implementation on my code. The Bubble sort will be used for sorting the data to be able to go into a leaderboard, displaying the data in the table, using PHP for methods to assign the best scores into variables and then into my c# to be displayed by unity.

Bubble sort for displaying data inside of a table.

The fastest user lap time > Lap time is input into the array of 5 elements > Obtain the array with PHP > compare adjacent elements inside of the array > if time on the left is larger than the time on the right, swap > if time on the left is smaller than time on the right > do nothing > complete this n times (n number of elements inside of the array) > once a bubble sort is completed on the elements print the individual data elements as they are in ranking order, my bubble sort will perform n times, with n being the total amount of elements inside the array.

My bubble sort will be sorting the top 5 times, the rest will not be counted for.

Example bubble sort

(James [15]), (Charlie [7]), (Rohan [6]), (Aaron [9]), (Daniel [3])

Swap elements *(James [15])* and *(Charlie [7])* as adjacent nodes are in the wrong order

(Charlie [7]), (James [15]), (Rohan [6]), (Aaron [9]), (Daniel [3])

Swap elements *(James [15])* and *(Rohan [6])* as adjacent nodes are in the wrong order, Rohan has a quicker time than Charlie.

(Charlie [7]), (Rohan [6]), (James[15]),(Aaron[9]), (Daniel[3])

Swap elements (James [15]) and (Aaron [9]), as Aaron had a quicker time

(Charlie [7]), (Rohan [6]), (Aaron [9]), (James [15]), (Daniel [3])

swap elements (James [15]) and (Daniel [3]) as Daniel is smaller than James

Charlie [7]), (Rohan [6]), (Aaron [9]), (Daniel [3]),(James[15])

Swap elements Charlie [7]) and (Rohan [6]) as Rohan has a quicker time trial than Charlie

(Rohan [6]), (Charlie [7]), (Aaron [9]), (Daniel [3]), (James [15])

Swap elements (Aaron [9]) and (Daniel [3]) as Daniel has a much faster time trial then Aaron

(Rohan [6]), (Charlie [7]), (Daniel [3]), (Aaron [9]), (James [15])

swap elements (Charlie [7]) and (Daniel [3]) again Daniel has a faster time trial then Charlie

(Rohan [6]), (Daniel [3]), (Charlie [7]), (Aaron [9]), (James [15])

swap elements (Rohan [6]) and (Daniel [3]) as Daniel has the quickest time trial so he should be at the front

(Daniel [3]), (Rohan [6]), (Charlie [7]), (Aaron [9]), (James [15])

The elements inside of the list are sorted in ascending order of time trials, the first item would be an example of the quickest racer [ThereName] and [TimeTaken] to complete the track.

This here would be an example of a bubble sort performed on 5 sets of results inside of the stored array which will be printed in ascending order of fastest time to slowest time.

The elements of the array will be sorted via the time variable assigned to the user's username which will be a float variable in my code.

I James Gammon – 27228 – 8292 – Abbeyfield school

Code for the bubble sort

```
def bubble Sort(arr):
    n = len(arr)
    for i in range(n):
        for j in range(0, n-i-1):
            if arr[j] > arr[j+1] :
                arr[j], arr[j+1] = arr[j+1], arr[j]
arr = [james[15],Charlie[7],Rohan[6],Aaron[9],Daniel[3]]
bubble Sort(arr)
print ("Sorted array is:")
for i in range(len(arr)):
    print ("%d" %arr[i])
```

My bubble sort will sort the names and times of the players of the game with the player tag (Player) this will work as each player tag will have an associated time, the times will be sorted in turns of size, and the player names associated are displayed on the leader board for easy identification.

Algorithms for front end	Algorithms for backend
Being able to input usernames and passwords and storing these in a private str variables	Checking if password and username are both above 8 chars for safety.
Menu select scene selects maps and time trials, combining the user choice of map as well as the user choice of game mode	Storing user input from unity and transferring this to a players database.
Pause menu algorithms which let the user (while playing) return to any menu screen, including quit, login and register.	Active scores inside of the game constantly being added into the database to print the best score.
Summon menu functions, setting menu panels active and disabling visited ones	Transferring the data variables from temporary stores in the c# script into variables from
Pause menu can be summoned at any point inside of the game, this allows the user to return to the menu for customization options.	Validating password and username variables are stored in the correct spaces
Start an elapsed Time variable when the user starts the lap	Validating there are no repeated usernames stored in the database, when c# checking for user input this as a Debug. Log.
Elapsed Time variable is restarted when the user finishes the lap	A leaderboard array is sorted and printed inside of the screen.
User elapsed Time for the lap is added to a float list with all-time variables for this map and user; the list is sorted when lap is finished to display the fastest time on the GUI.	
Username is displayed from login/registration from the temporary value onto the GUI	
Pause menu can be summoned at any point inside of the game, this allows the user to return to the menu for customization options.	

I James Gammon – 27228 – 8292 – Abbeyfield school

Algorithms

One algorithm is about the time format of the game

When game starts start an elapsed time variable which is set at 0 > the user drives around the track and the time are stopped and added to an array

Another algorithm restarts the time variable when user enters the finished point trigger

When user completes a lap > elapsed time is set to 0 > background script keeps running and the time immediately starts counting up from 0

Another algorithm I need to complete is adding the car time into a list when completed, the list is sorted, best time displayed

User completes a lap > elapsed time added to a list > list is sorted on every lap complete after new variable added > if list has more than 1 elements > print the list > display best score on screen

Another algorithm is to display the user times inside of the array

When user enters end of lap trigger point > elapsed time from score is added to a static list > if times > 1 then the list is sorted

The final step process is transferring the best time score into the leader board

Once a variable is reached that the user does not want to play anymore (either saves and quits or completes a certain amount of laps the final score is added)

Is game finished? > prints the best score time into a variable > this variable is posted to a php script which assigns the best score variable from that game into the database at the current user position > Aswell as sending it to the leader board array > leader board array is sorted.

Each process in more detail

Time format of the game:
Start a timer

Timer set at null when the game starts

Record the timer in an update function every second

Timer will stop when a function is called to stop the timer or reset it to 0

Restarts the time variable when user enters the finished point trigger:
A Trigger function is called when the car enters the full lap trigger

The function:

Takes the current elapsed time > adds it to an array of times

The current value once recorded means elapsed time can be set to 0; and count up again

Elapsed time = 0f;

Car time into a list when completed, the list is sorted, best time displayed

I James Gammon – 27228 – 8292 – Abbeyfield school

The lap time has been added to best time list

List checks if it has more than one element

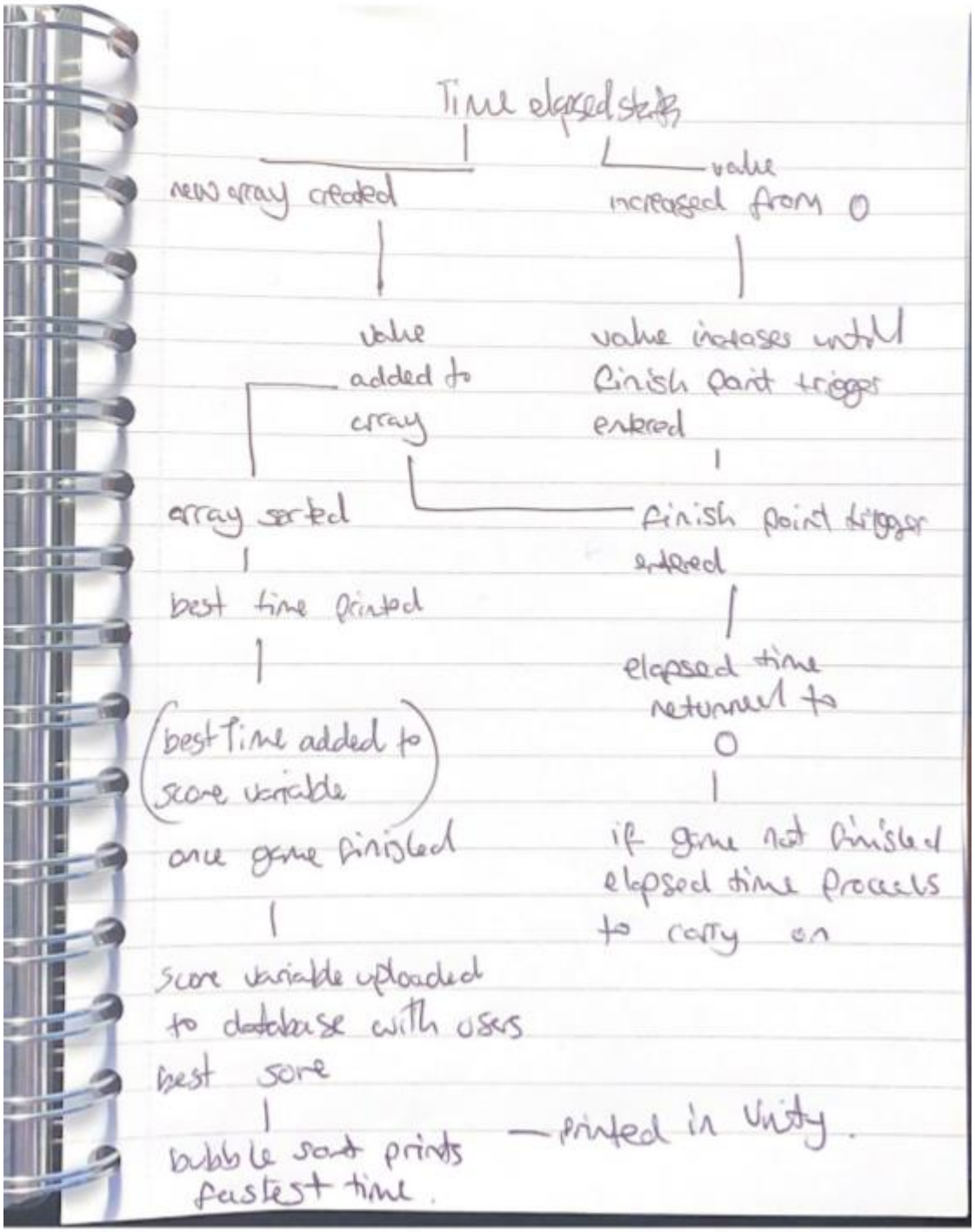
If list > 1 element the sort() function

List[0] the first element is retrieved

List[0] to string ()

Best time text = list[0]

How does this work:



The process follows a linear structure. There is a starting position where there the user starts the process and an ending position where the user ends and begins to play the game, on every user interaction with the game these steps are repeated.

Each step needs to be completed when going to play the game, once the user has registered, they will never need to register again, only if they register with a different username.

There are 4 different subroutines the algorithm can be split up into:

- New user
- existing user
- Game mode
- Game map

New user

- User has chosen new user function > initiates a new score, username, and password variable
- New user creates a new username, password, and score value in a database.
- The algorithm continues once the user is created > map select > time select

Existing user

- User has chosen login function > compares an SQL query to compare input username to existing username
- Existing user compares the entered data with data already held in the array – with a user check request in php and returning a value other that 1 is a failure.
- The algorithm continues once user is created

Game map

- Three different options displayed on the screen for the user
- Each map comes with its own set of pre-set functions. This creates the illusion there are interchangeable sections; the map function called portrays game map choices, each with their own speciality. If the user chooses a different map, this loads a different 3 sets of map decisions each with the functionality of their respective game mode.
- This limits the number of choices from 12 possibilities to 4 possibilities which is exponentially less coding and allows more accurate decisions.

Game mode

- Three different options displayed on the screen for the user
- When one of the options is picked it calls a function which carries out that game mode on the selected map, the game mode option buttons come with pre-defined map choices after each one.

New/ existing user

User comes to a decision:

Users will follow the game thread until they reach the options screen; the two options will be displayed on the screen. Each of the options will take the user to a separate page to carry out the function; *pick new user if never played the game*, pick *existing user if they already have a login*

New user creates a row in the database :

A singular row in the will holds all the data needed about the players.

Existing user compares the entered data with data already kept in the array:

pre-stored data inside of the database is compared with the users.

Algorithm is continued:

Once a flag variable is changed to true that the user Is logged/registered, this true change will call the next components in the algorithm, the game mode.

Game mode

Three different options displayed on the screen:

The different game mode will each be displayed on the screen for the user to choose. The boxes will be relatively large on the screen, and will each include a small description of the game mode i.e., features included such as time-trial table

When one of the options is picked it calls a function which carries out that game mode:

I will develop a program when the game mode is picked, it calls subroutines and functions associated with the game mode to picture together the game including the features. The time-trial game this will be the charts of quickest times, the free play will be a limitless exploration code and tutorial mode will include texts of how to:

Game map

Three different options displayed on the screen for the user:

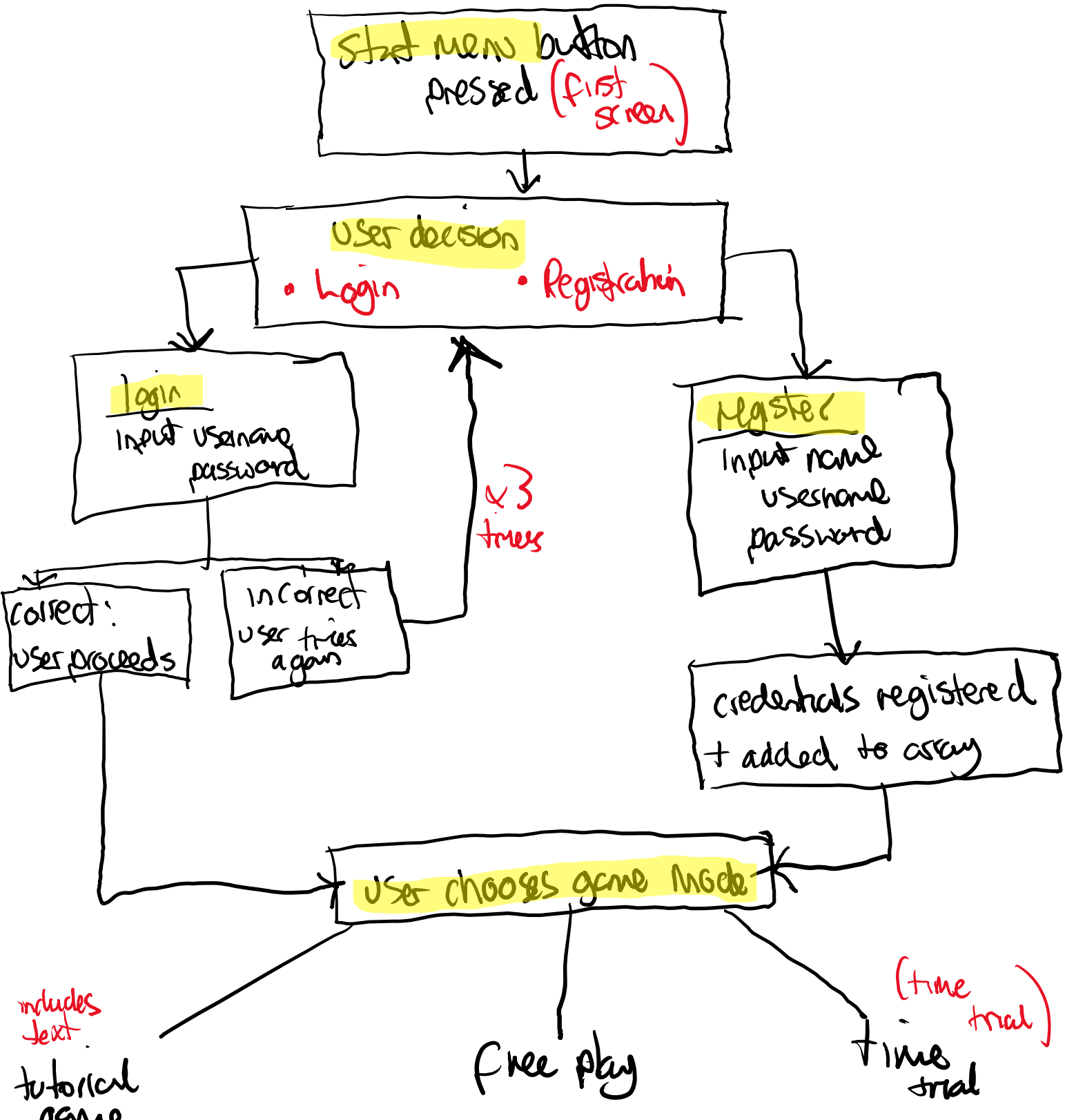
I James Gammon – 27228 – 8292 – Abbeyfield school

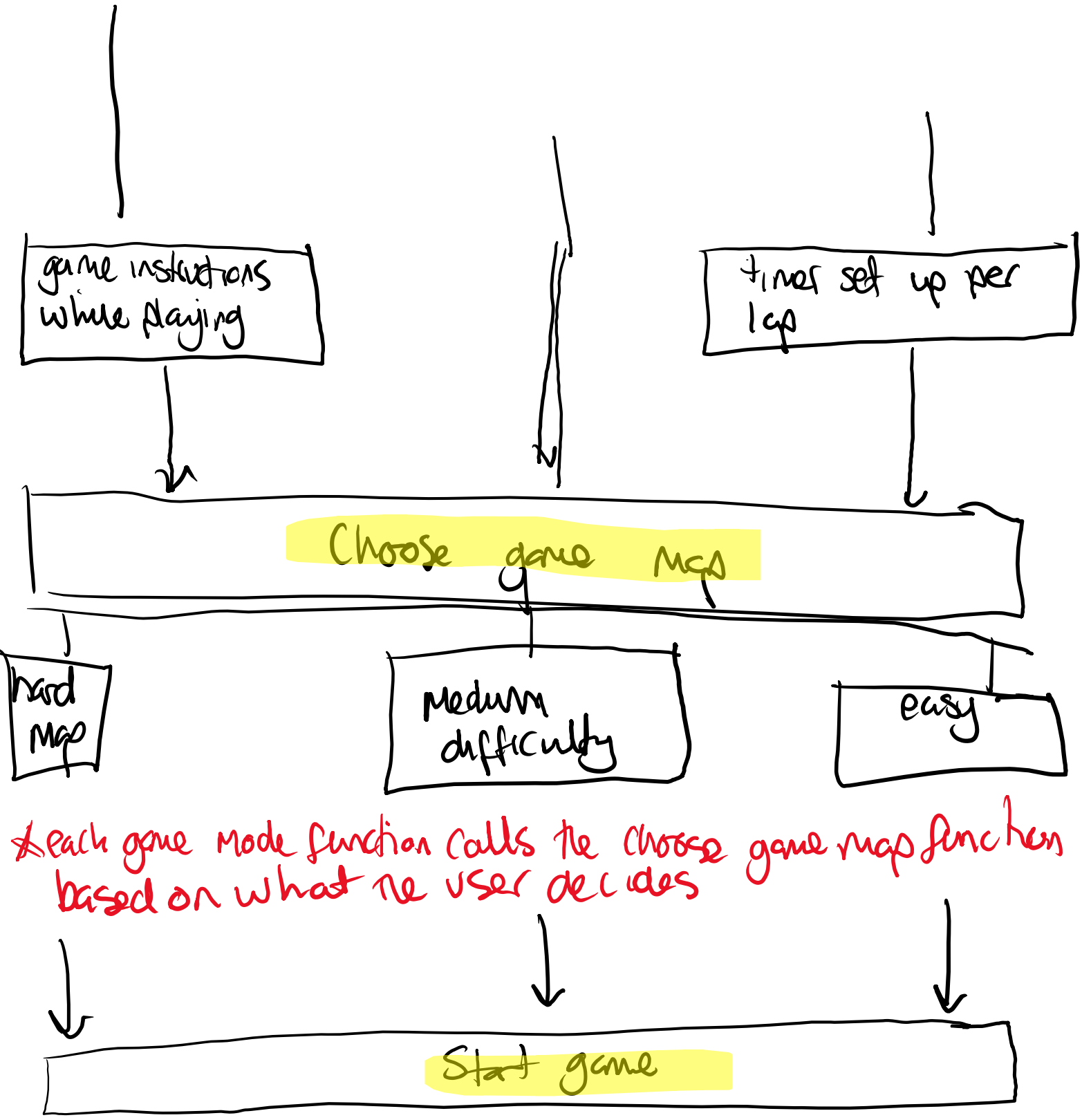
My GUI will include 3 different possible functions to call, each in turn calling a different level of map.

Game is started

Elapsed time started for Time trial game modes.

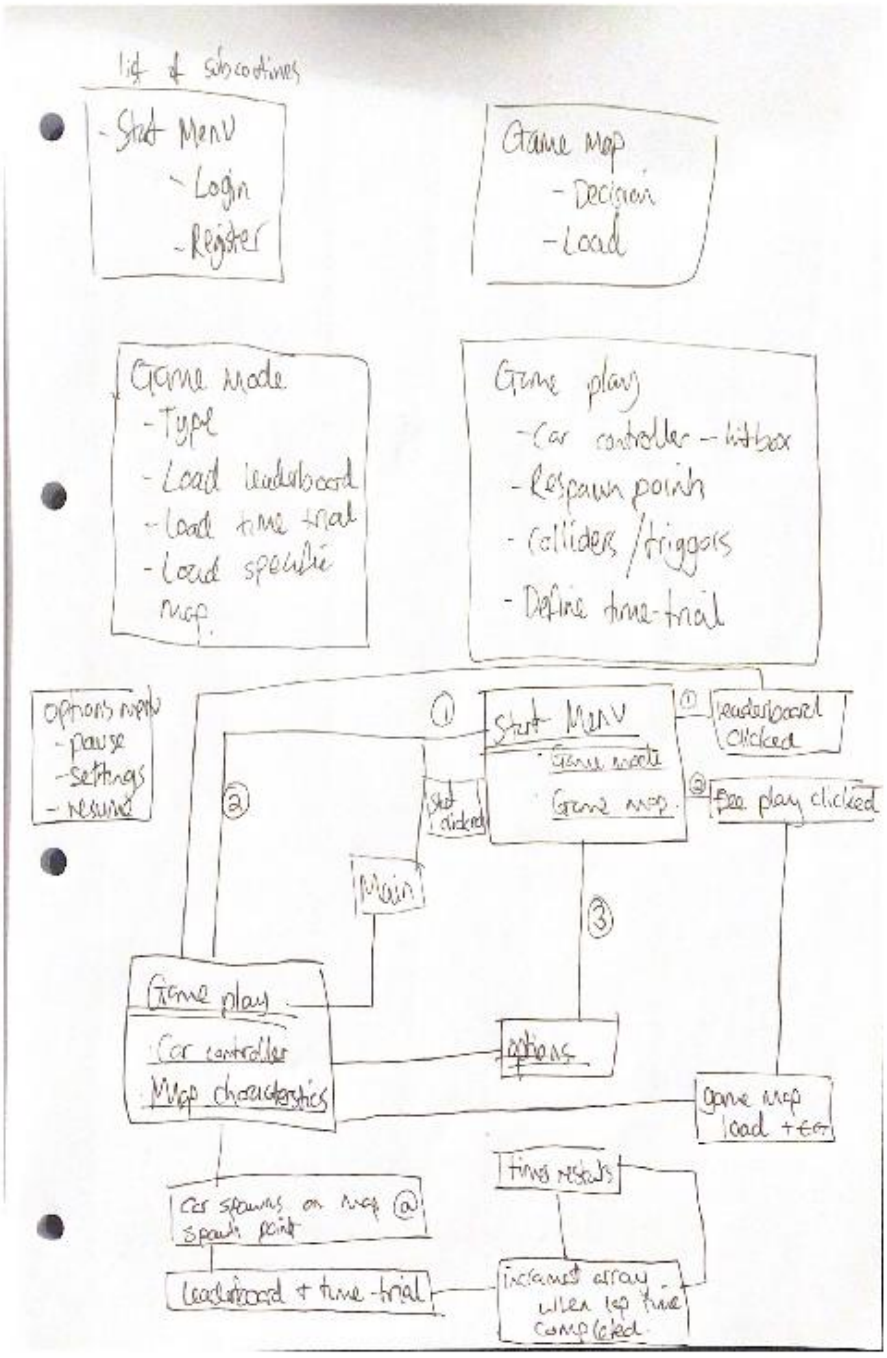
How does the menu work?





*each game mode function calls the choose game maps function based on what the user decides

Diagram to show how subroutines link



Subroutines

The program will be spilt into multiple different programs, each with multiple different subroutines on scripts which are applied to game objects.

The project will be organised through a variety of programs, where when the programs need to interact then global functions/subroutines are called.

A lot of my functions are subroutines as they are applied to multiple different maps; lap complete trigger which sets the instance of the elapsed time to 0 for example, is called on the hard map time trial, medium map time trial and easy map time trial.

Subroutines are:

C#

Menu GUI

Login player

Register user

Game Aspect

Setting a pause menu

Save player data

Increase Score

Database

On trigger enter for lap complete trigger

Sorting algorithm for leader board

php

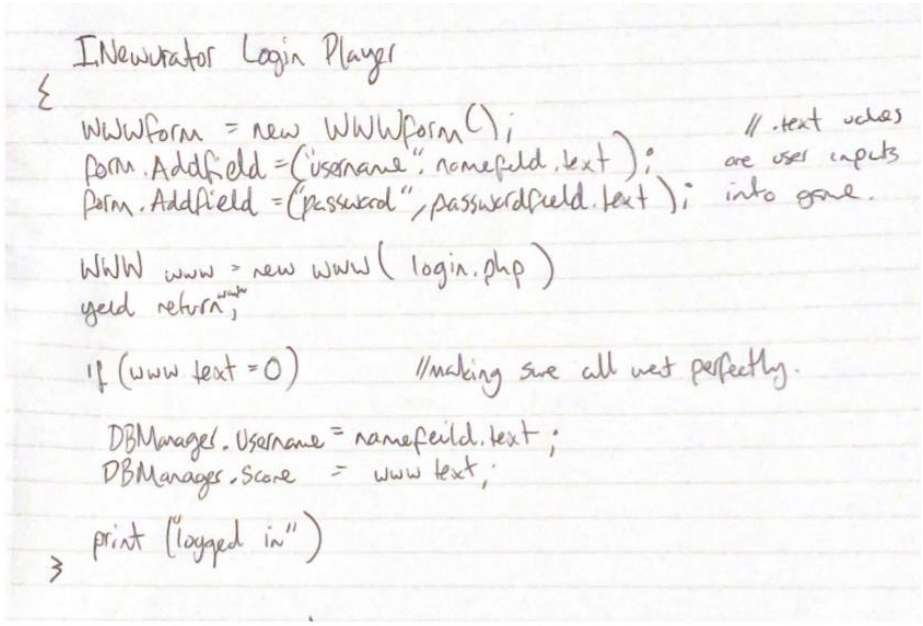
interactions with database

Name checking registering users to see if they already exist

Name checking logging in users to see if they are using a username which already exists

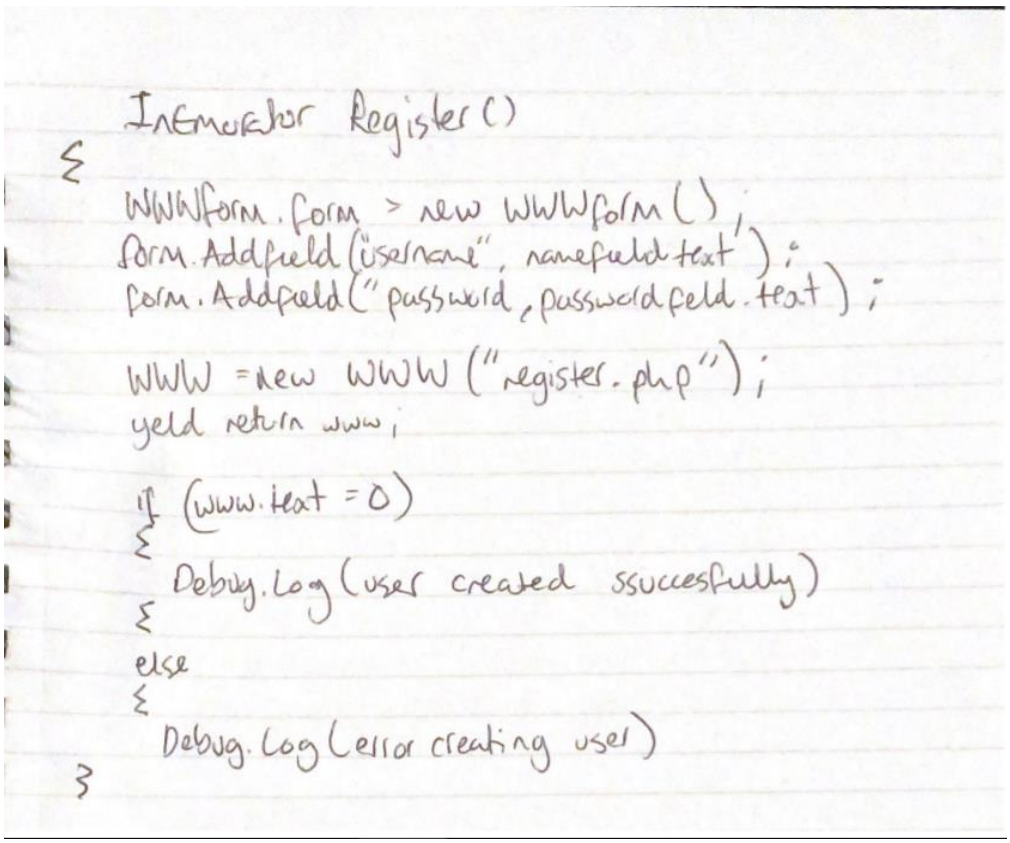
Menu GUI

Login player



This will be the function that logs in the player, the WWW form sends input data from the unity UI to a php file where inputs are validated for repetitions and compared to the database. It will take in 2 parameters which are the players username and password; and return in the Debug. Log the outcome of the process.

Register player

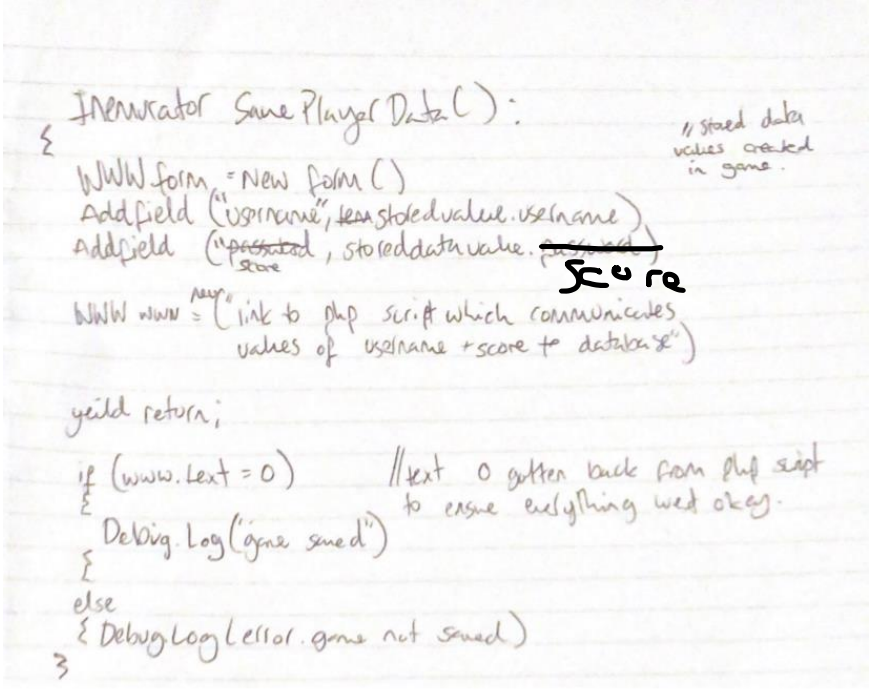


This function registers a player, playing a similar role to the login function, it takes two parameters form the GUI, the username and password. Both are uploaded into a php file Register, where length values of the password are validated, and no repetition in values of the database are created.

The value name field text can be displayed to any public Text field which I want to display the username value at.

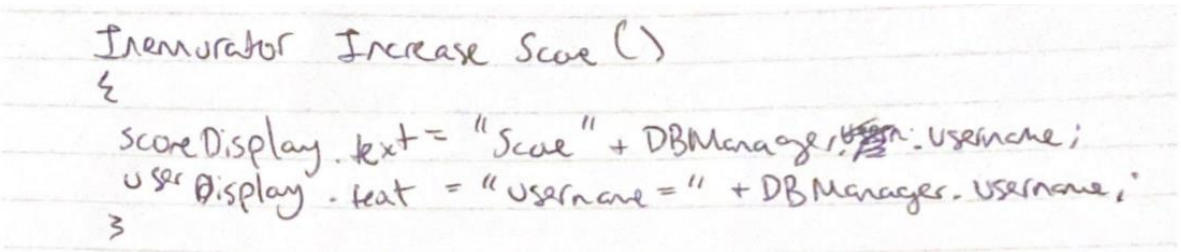
Game aspect

Save player data



This function saves the current username logged in from GUI and score achieved from best times string; the result from best time string will be passed through a global variable into the Add Field text of the WWW form, uploading this to the php script.

Increase score



This function updates the score on the final score page, once the user triggers this page the Increase Score function is performed, where the most recent DB Manager. Score value (what is being used as a filler variable currently) is updated from the best Times array. DBManger.username is a variable created when the user logs in/registers. From the php script back to c# this is the value which is referenced.

I James Gammon – 27228 – 8292 – Abbeyfield school

Pause Menu UI

```
void update
Updating every frame ()          gameispaused = false
  if (Input (keyCode = Escape))
  {
    if (gameispaused = false)
    { pause
      Pause();
    }
    else
    {
      Resume();
    }
```

This function checks public canvas gameispaused to see wether it is true or false once the initiate escape key is pressed. If gameispaused = false then its set to true to show the menu, and vice versa.

Public Enumerator Resume

```
public function Resume()
{
   pauseMenu.SetActive(true)
   gameispaused = true;
}
```

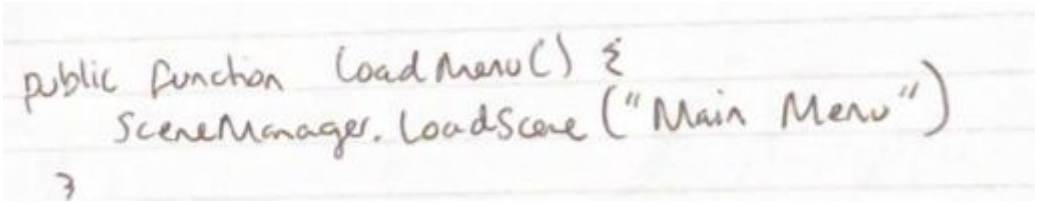
This function is a button on the main menu script, it sets active the opposite if the current variable gameispaused is set to false.

Public Enumerator Quit()

```
public function Quit():
{
   Application.Quit()
}
```

This function is another aspect of the pausemenu caused when a Quit button is pressed. Application.Quit is a function built into unity to stop the current running application.

Public Enumerator

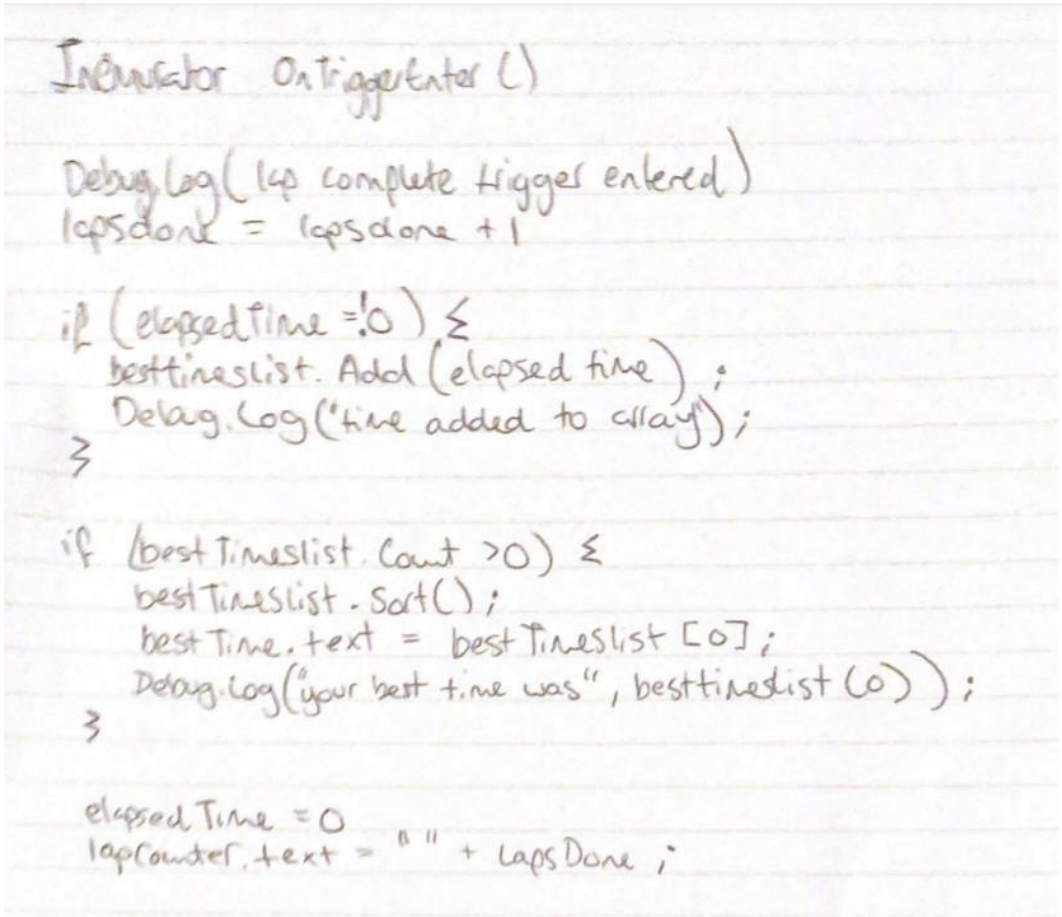


```
public function LoadMenu() {
    SceneManager.LoadScene("Main Menu")
}
```

This fits in my success criteria of changing the game map while playing the game once paused. Here the user can call back to the Main Menu while still being logged in, to set active a new map and game mode.

Database

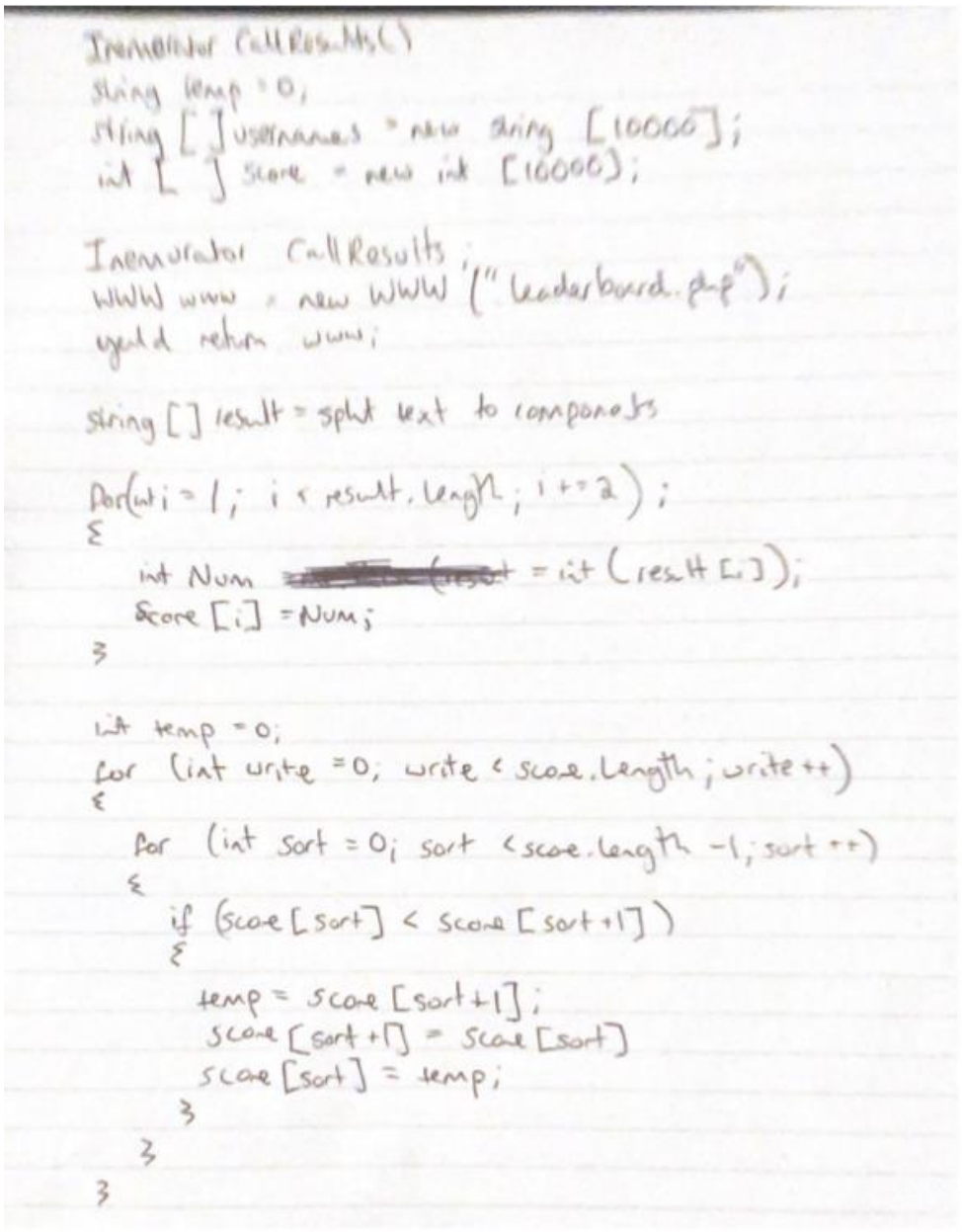
On trigger enter



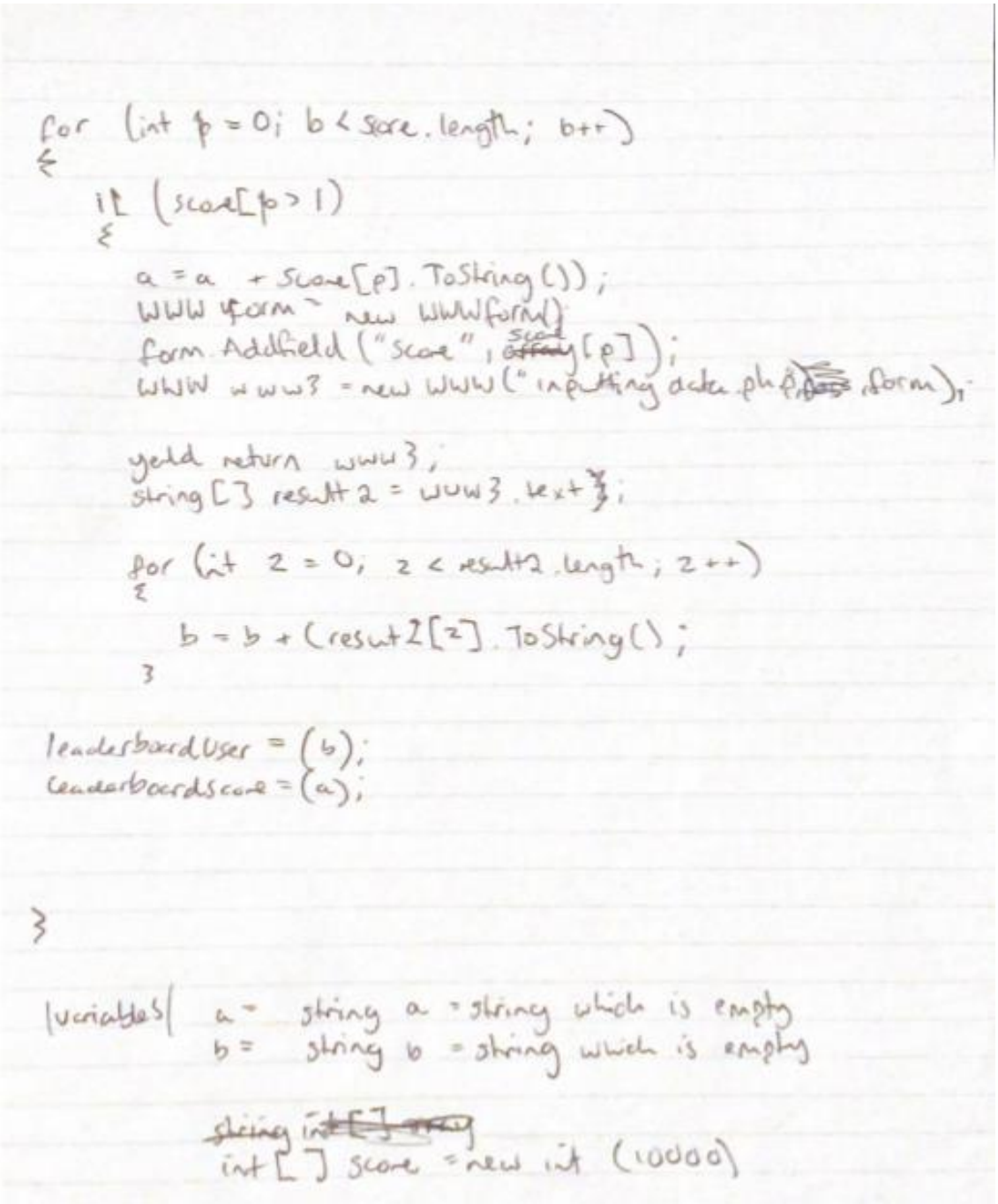
```
InCircuitor OnTriggerEnter()

Debug.Log(lap complete trigger entered)
lapsdone = lapsdone + 1

if (elapsedTime != 0) {
    besttimeslist.Add(elapsed time);
    Debug.Log("time added to array");
}

if (bestTimeslist.Count > 0) {
    bestTimeslist.Sort();
    bestTime.text = bestTimeslist[0];
    Debug.Log("your best time was", besttimeslist(0));
}

elapsedTime = 0
lapCounter.text = " " + LapsDone;
```

This function performs 4 tasks, firstly it increments the laps done variable to ++, it also adds the elapsed time of the lap into a public list called besttimeslist, where it will be sorted in the future. The function checks if the besttimes list has more values than 0 and sorts the list displaying the fastest one. Finally, the elapsed time is returned to 0 to indicate a new lap has been started.

Call Results()



```
IEnumerator CallResults()
string temp = 0;
string [] usernames = new string [10000];
int [] score = new int [10000];

IEnumerator CallResults;
WWW www = new WWW ("leaderboard.php");
yield return www;

string [] result = split text to components

for(int i = 1; i < result.Length; i += 2);
{
    int Num = int (result [i]);
    score [i] = Num;
}

int temp = 0;
for (int write = 0; write < score.Length; write++)
{
    for (int sort = 0; sort < score.length -1; sort ++)
    {
        if (score [sort] < score [sort+1])
        {
            temp = score [sort+1];
            score [sort+1] = score [sort]
            score [sort] = temp;
        }
    }
}
```



```
for (int p = 0; b < score.length; b++)
{
    if (score[p > 1)
    {
        a = a + score[p].ToString());
        WWW form = new WWWForm();
        form.AddField("score", score[p]);
        WWW www3 = new WWW("inputting data.php", form);

        yield return www3;
        string[] result2 = www3.text;

        for (int z = 0; z < result2.length; z++)
        {
            b = b + (result2[z].ToString();
        }

leaderboardUser = (b);
leaderboardscore = (a);


}

|variables| a = string a = string which is empty
          b = string b = string which is empty

          string int[]
          int[] score = new int (10000)
```

This function performs a bubble sort code on the available usernames and their partnered scores inputting these into text variables S1 and U1 to display on the leader board.

Inputs and outputs

The Areas where the user inputs into the game are menus and car controller inputs. Menus consist of inputting credential information to be saved aswell as function buttons. The game has inputs Car controller inputs.

The user inputs car controls through the WASD and arrow keys.

Above is an example of a potential menu design, with large clear buttons each with label's, and minimal activity in the background for easy user understanding.

Input	Process	Output
Start Menu button activation	Initiates the main program	Calls the User Activation subroutine
Choosing a new user The user is showed a screen with 3 input fields, name, username, password where they input a value into each one.	Username + password will connect to a database where their length and characters are validated and returned to the user.	If the user inputs are validated, the user is taken to the game map menu. A prompt is given to the user on both occasions. Else the user is shown an error message and is forced to register again.
Choosing an existing user The user is shown a login screen	credentials are compared from a user's database (username, password) to the ones input by the user.	If data matches, the user is taken to a map selection page A prompt is given to say they were logged in.

The user inputs into shown input fields > username, password		
Choosing a game map Easy Medium Hard	Choosing a game mode will call 1/3 different subroutines. Game map calls a subroutine, the subroutine contains the hyperlink to the game mode chosen. e.g easy map function Is picked > obtains hyperlinks in c# to load a new unity scene based on the game mode picked.	Each decision generates a different game mode option screen on the menu
Choosing game mode	Choosing a game mode will set active a choice of 3 different game modes. When a game mode is picked, a subroutine calling that unity scene and scripts is activated.	The chosen game map and output are set active on the unity game menu.
User inputting controls from the keyboard.	Each input is registered by unity, unity has a functionality which takes synonymous game controls and takes these as functions (arrow keys and WASD so the function is called, and they do not have to be hard coded)	The car on screen responds instantaneously to the user controls moving across XYZ coordinates in that direction.

Key variables

To split up the Game I have applied decompersition to the game process and understand which part they play.

Global variables (static variables) – called from c#file. Variable you want to access.

Using this diagram, I can assert global variables to my score function as they will be called cross script.

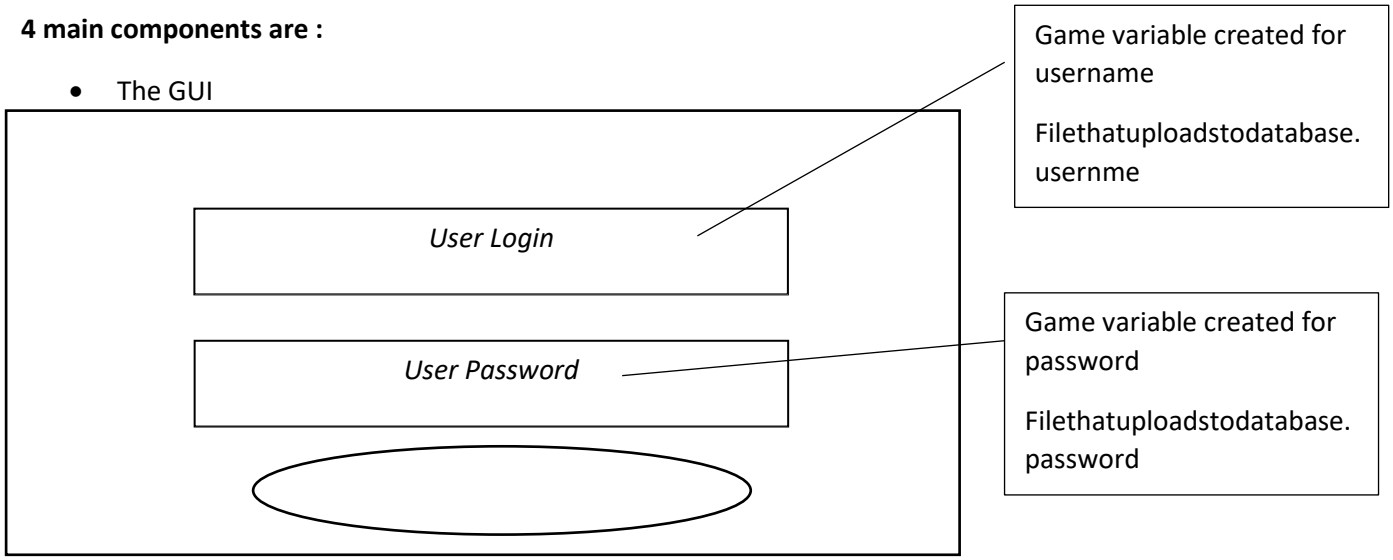
Public variables – this means that it will be visible from other classes and packages when called

I James Gammon – 27228 – 8292 – Abbeyfield school

Private variables – this Is invisible from other classes and packages from when being called.

4 main components are :

- The GUI



- The Game
- Php scrips
- Leader board printing

The GUI

Register

Name	Data Type	How it's used	Storied in the database
nameFieldUsername.text)	String	Store the users username	VARCHAR
nameFieldPassword.text	String	Store users password	VARCHAR

Login

Name	Data Type	How it's used	Stored in the database
UsernameField.text	String	User input to compare username to database's information	VARCHAR
PasswordField.text	String	User input to compares the input password to the databases stored	VARCHAR

The game

Name	Data Type	How its used
Timer going	Boolean	Checks to see if timer Is running or not
Elapsed time	Integer	Counts the current time playing in the game
CurrentTimePlaying	String	Prints the fastest lap time on the users screen

Half Lap Trigger	Boolean	Assigns the lap complete Game Object to true once this game object is entered (changed to false)_
Complete Lap Trigger	Boolean	Assigns the half lap point to true and this game object to false when entered

PHP scripts

Name	Data type	How it is used
Username	Input integer	The integer is used to reference the elements stored at that line in the array for logging in.
Password	Variable integer	Integer used to flag the amount of tries the user tries to login/ reach a stop point.
namecheck	String input	Checks if a player name, password is already stored in the database
Update query	String input	Inputs the new score, username, and password (possibly hash and salt into the database)
Con	VARCHAR	Connects the php scripts to the database (takes DB location, user + name and password)
MySQL Query	VARCHAR	Conducts an SQL statement which updates the users credentials.

Communicating bertween game and database

Name	Data Type	How its used
DBscript.score	Integer	Updates the most recent score form a time Trial level into the users score variable
DBscript.username	String	Updates the users username column in the DB when registered/logged in.

Leaderboard

Name	Data type	How it's used
Username	String	Once the scores with the associated usernames are sorted – the usernames are input into this variable which is printed.

Score	Integer	When scores are sorted, they are input into this variable which is then printed.

Validation

There is a lot of user inputs inside of my program. For my program to work fluently validation of each of the user inputs is required. Following will be types of data and how they can be input:

Inputs (controlling the car) via hardware

The user controllers are a function in Unity, so they are already a reliable input source. The nature of the controls is relatively simple and are commonly used across all programs.

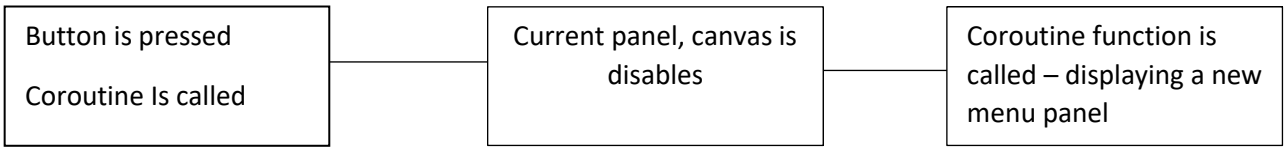
The controls very simple to gain control of and, if none are pressed the car will not travel, if multiple inputs are concurrently processed then commands are run at the same time and the car is controllable, there is no risk of mis-inputting or failing to control the car.

Its instinct for PC gamers to play on the keyboard arrows when playing any car controller game.

Pressing buttons to call menus, different game stages.

Interchangeability was a large aspect of my game and buttons are crucial to make this happen.

I would have to validate by making sure the commands for bringing up the menu are clear accurate when the game starts. Such as:



Text boxes in the GUI (entering string)

Entering string needs a lot of validation from the program as mis inputs are common for logins especially.

For Validation of both Logging in players and registering players I will validate both the username and password to be above 8char.

If the conditions are not met a print statement notifies the user about the failure to register/login.

Other validation includes input type; password input and username input will both be defined as VARCHAR so they can have a varied input character. I have limited these to numbers and characters for easily handling.

Validating logging credentials

Usernames

To validate users logging in a query will be performed in php to compare user input credentials to the ones stored in the database. The username is a string, and the password has been hashed. The username will be compared through an SQL statement and if the number of lines returned is anything other than 1 the username is incorrect.

Passwords

Passwords will be compared to the ones stored in the database. The salt will be applied to the hashed storage inside of the database. If the salt and hash match the password is correct, else the password is incorrect.

Testing method

There are many aspects to the game, it has many layers, coroutine, functions, panels, canvas, and algorithms. Each Testing method will be tested thoroughly to ensure its reliability. While creating code for the game, menu counters, scene changes and car controller scripts. I will use comments to indicate what function the code is performing and use comments printed in the Debug. Log to inform me which processes have happened, and which ones haven't.

If …. Happens print …. Else print …. (Will be used mainly in php which doesn't produce errors in the code, and some in c# for complex algorithms.)

My testing aims to protect against the vulnerability of game and its reliability to perform under stress (black box testing and white box testing) and to find bugs in the code before total release.

For testing I will:

Design a testing environment after the game grows

Test in two types of timeframes; during game development and after game completion

Make testable software requirement.

Perform deliberate errors to see how code reacts.

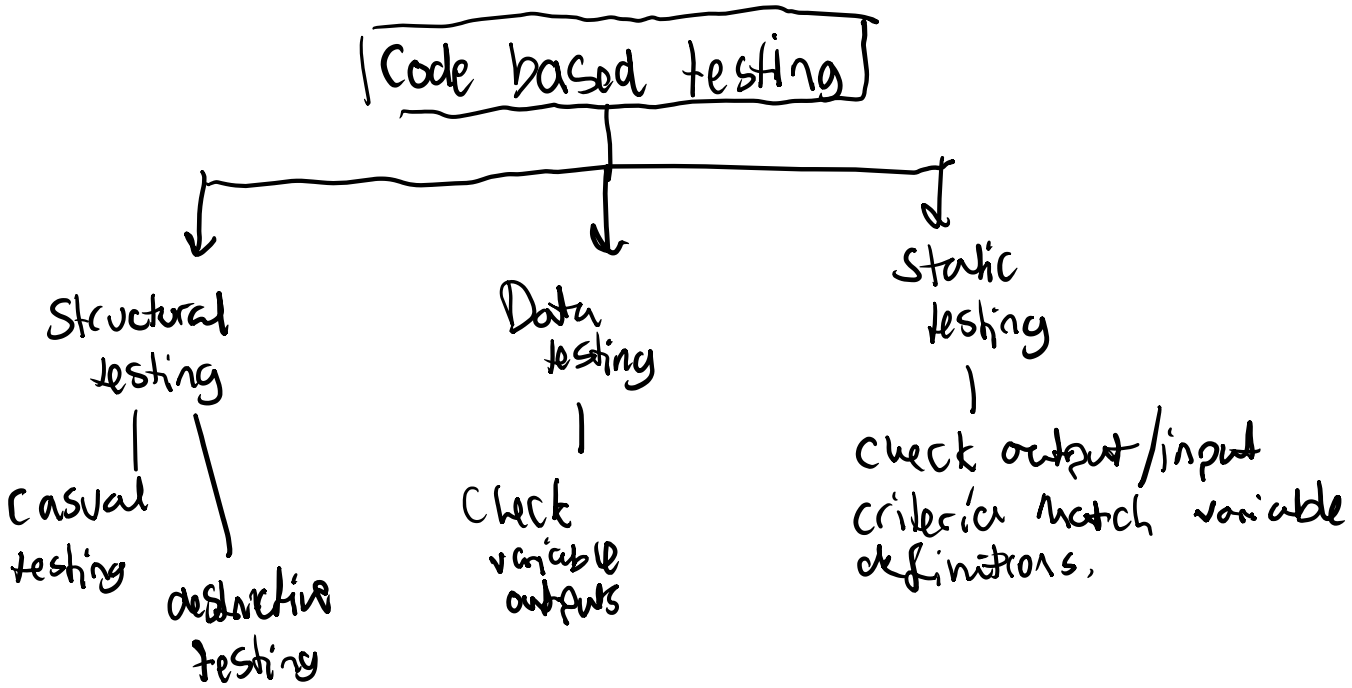
My game is going to be recursive in most areas in the sense that changing one area/ component will influence a lot of other components; this is due to global variables being used in different algorithms; so, I will design a testing environment later in the game to test individual components with more accuracy.

Through development especially in the game I will get different types of demographics to play my game to account for different user experiences.

- Choosing map and game mode
- Logging in and registering
- Using the specific features of each map
- Completing a game mode and Leaderboard
- Each module forms a 'process' in my game each with certain variables I can test.

During the testing

During the testing I will try two different methods. In the first I'll include the imputing data into the program (such as suitable registering names, maps, game modes and car inputs) and record the output result by the program. Any message outputs will be screen shotted; and the route I took to fix the problem. If there are no problems when naturally playing the game, I'll move to destructive testing to try and force an error from the game code.



Casual testing

Casual testing includes playing the game normally how it was designed to, including abiding to the walkthrough text on the screens, and following the game rules.

Casual testing is useful as it mainly focuses on the GUI of the program including making sure every scene and menu is correctly called, and text boxes are displaying at an appropriate size.

Destructive testing

The other method I will try when testing is destructive testing; I'll navigate through my game trying to find an error or create one by continuously inputting incorrect data or multiple functions at once; destructive testing will be useful in my game to reveal any problems not in the code, such as navigating menus and choosing game data. After intense destructive testing I should have unveiled enough forced problems and fixed them so when the game is released to my friends it runs smoothly.

Any problems that occur the error message will be screen shotted as well as the highlighted code to fix the problem.

Checking variable outputs

This testing method tests whether the output of a given variable is what it is supposed to be e.g., after the game is finished the print statement prints the time in the format of mm: ss. ff.

Checking the output input criteria:

I want to make sure that the variables in my code and their datatypes align the entire way through the game. By using my variable definitions diagram to compare input data types to output data types.

Iterative development

So far, I've been developing my project in an iterative manor. Tackling the large-scale project into ideal manageable components is much easier to code and test. Each smaller part is a specific component of the game which can be worked on separately.

- Choosing map and game mode
- Logging in and registering
- Using the specific features of each map
- Completing a game mode and Leaderboard

In iterative development my feature code is designed, developed, and tested in repeated cycles. (Each for each component) at the end of the end of the iteration, working code is expected that can be demonstrated in the game.

Testing checklist

To ensure I have met the functionality of the entire program.

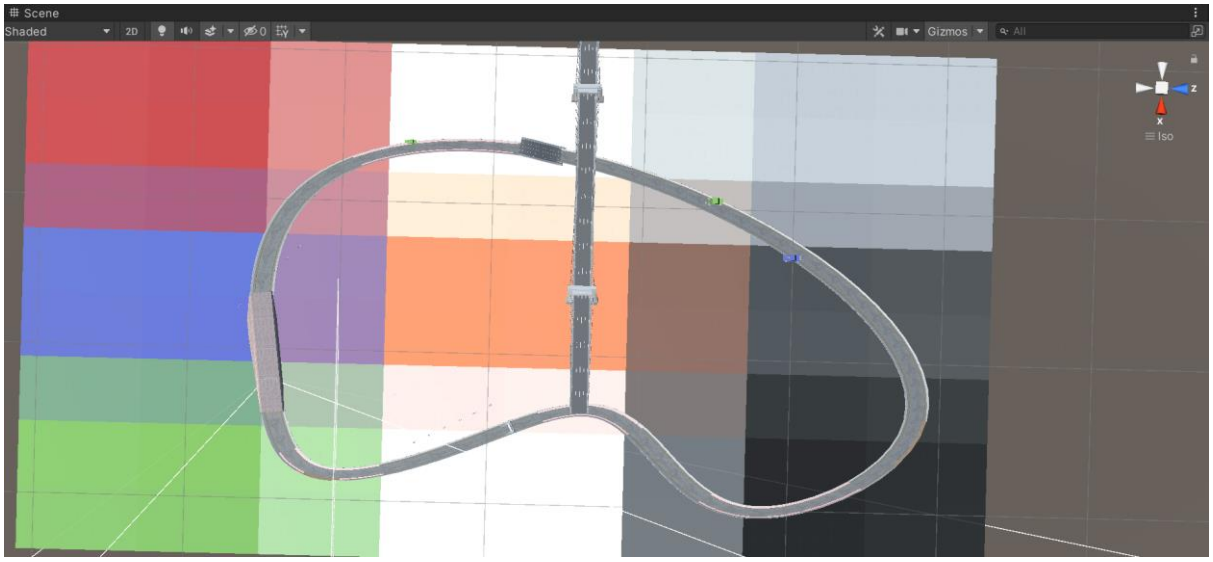
Functions to be tested	Is it working
Game buttons call and display the correct menus.	
Credential input boxes create a new row in a database storing username, password, score	
Password is stored as a hash inside of the database	
Credentials input into the user login page are compared to the database. If correct the user can proceed, else the user is denied.	
The user cannot login/register until the conditions for the username and password are met, the register/login button is not interactable	
Each map selected and game mode selected produces the correct selection on the game display.	
User tutorials are toggleable, for the game map, mode, and controls, they can be present on and off screen from a button.	
Racing game mode starts the timer once the game is started and is reset once the user enters the finish lap checkpoint.	
Each lap time is input into an array of times where the best time is produced onto the screen.	
The ESC button functionally calls the menu, each button on the menu is functional.	

Once the user has completed 3 maps, the finish game screen opened, and the user can see their best time and username.	
At the end of the time trial game mode the leaderboard menu is opened and displayed.	
The car is controlled by WASD or the arrow keys.	

I have included all of requirements of the project for it to be able to run successfully without any errors passed onto the stakeholders. Before the game is released to my stakeholders, I will forcefully try and find errors in the code to be able to identify any stakeholders may come across.

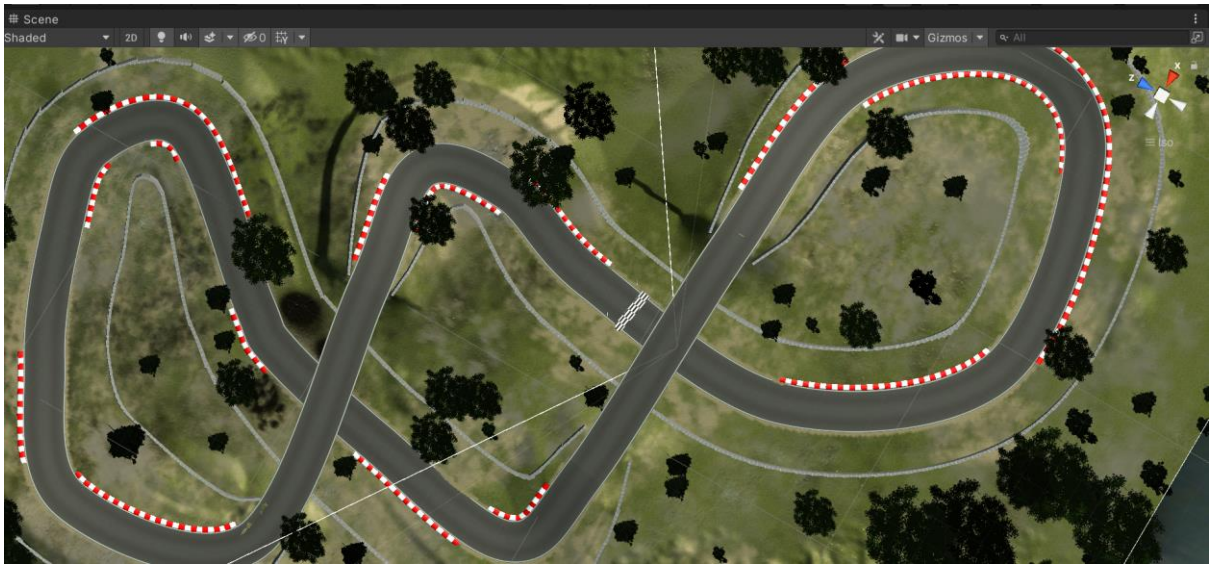
Game Maps

Easy Game map



I James Gammon – 27228 – 8292 – Abbeyfield school

Harder Game Map



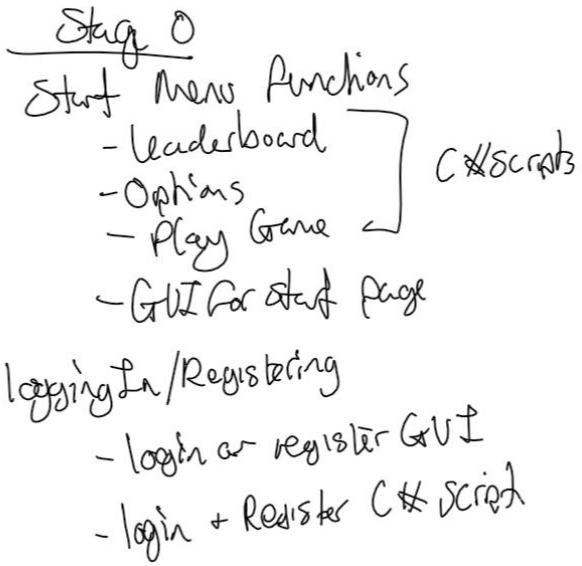
Hard Game Map

Development and testing

Stage 0: Start Menu
Stage 1: Creating the database
Stage 2: Registration
Stage 3: Login
Stage 4: Choosing Game map and Game mode
Stage 5: The Game
Stage 6: The Game - Time Trial
Stage 7: The Game - Tutorial
Stage 8: The Game - Free play
Stage 9: Completing the game and time display
Stage 10: Pause Menu
Stage 11: Bubble sort for leaderboard
Stage 12: Final Testing
Stage 13: Stakeholder Testing
Evaluation
Final Code

Stage 0 – Start Menu

This stage can be split up into two different sections – start menu functions and login/register functions.

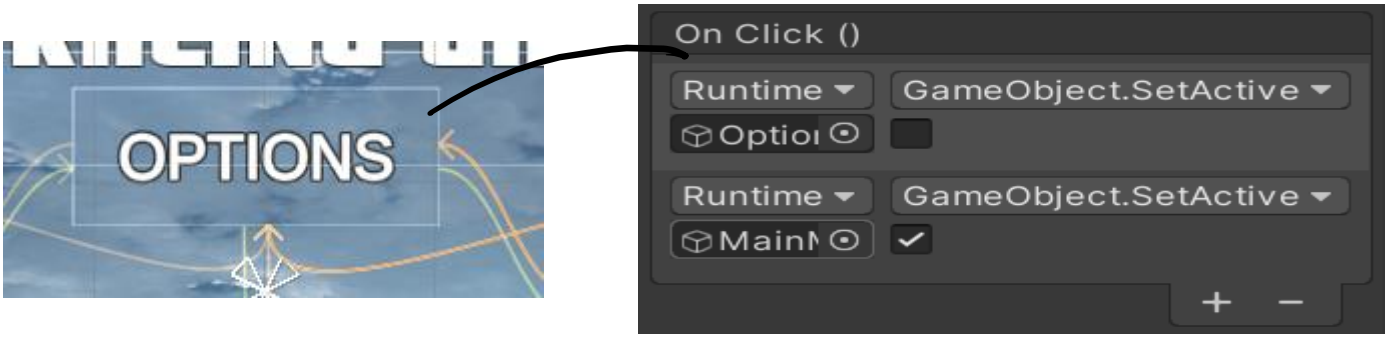


Start menu GUI

Adding functions to each of the buttons:

Options Menu button to Options Menu

When clicked the button options menu this disables the Main Menu game object (shown above) changing the canvas to the options menu. The next step is creating an options menu after creating its canvas.



Assigned to the options menu are the two function which call on Game objects. Each Game object in this instance is a canvas menu. So, when clicked, the main menu shown becomes false and option menu becomes true.

When testing the options menu, I was casually testing to see if the assigned menus to buttons where functioning correctly. I found the error below:

Error:

https://vimeo.com/663677083/65ef3039bb

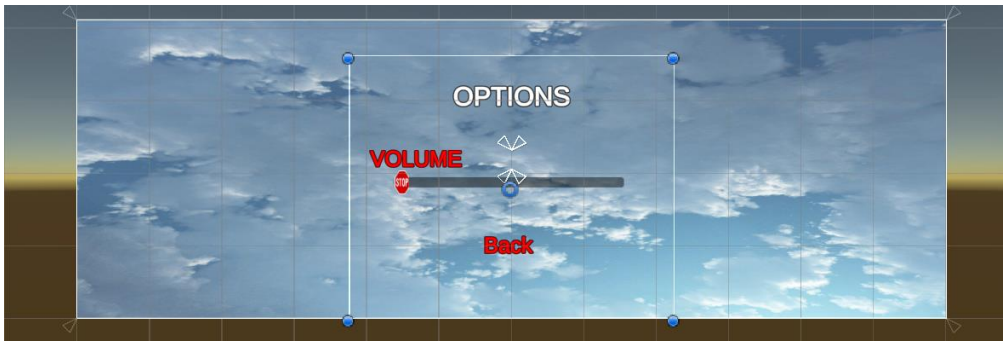
Fixed:

To fix this, the options menu was not being called by the button and the Main Menu was still being set as active.

Which then calls true the options menu

Options Menu:

Below is the options menu design.

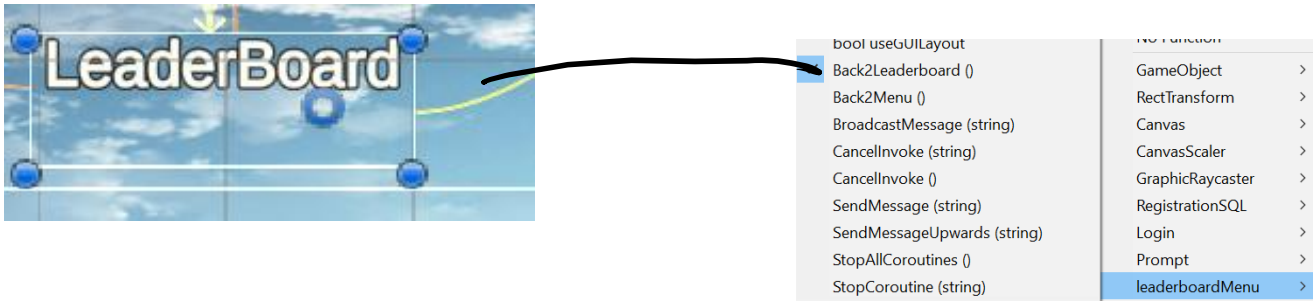


I James Gammon – 27228 – 8292 – Abbeyfield school

Video evidence:

https://vimeo.com/663678553/afec94aa6e

Leaderboard Button to leaderboard
This button directly calls the leaderboard menu from the main menu.



Above I assigned the leaderboard function to button which sets active the new scene leaderboard when clicked. Below is the code which branches into the subroutine Back2Leaderboard() which changes the scene.

```
Miscellaneous Files                                   leaderboardMenu
 1   using System.Collections;
 2   using System.Collections.Generic;
 3   using UnityEngine;
 4   using UnityEngine.SceneManagement;
 5
 6   public class leaderboardMenu : MonoBehaviour
 7   {
 8       public void Back2Menu()
 9       {
10           SceneManager.LoadScene(MainMenu);
11       }
12
13       public void Back2Leaderboard()
14       {
15
16           SceneManager.LoadScene(LeaderBoard)
17       }
18   }
19
```

This is the Scene Manger function which loads different scenes in the game. It takes the identity name of the scene and sets the value as active. Scene Manager is the overall function, LoadScene is the specific function and Leaderboard Is the value which we are setting true.

Error:

I found an error which did not set the leaderboard active, I checked the game object function but as this is the script it has no function.

Insert gif

Fix:

To solve the issue I used static testing, realizing that the input function for LoadScene was not string – and therefore the name of the scene was not being registered. To fix this I added quotations around the names of the scenes, which called the scene “Leaderboard”

I James Gammon – 27228 – 8292 – Abbeyfield school

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

public class leaderboardMenu : MonoBehaviour
{
    public void Back2Menu()
    {
        SceneManager.LoadScene("MainMenu");
    }

    public void Back2Leaderboard()
    {

        SceneManager.LoadScene("LeaderBoard")
    }
}
```

Leaderboard GUI

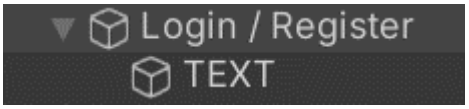
Play button initializing Login / Registration

I James Gammon – 27228 – 8292 – Abbeyfield school

To prevent bugs, I made sure that the MainMenu was set to “false” when clicked and the Login canvas set to “true”

Disables Main Menu GUI

Activates Login Registration GUI



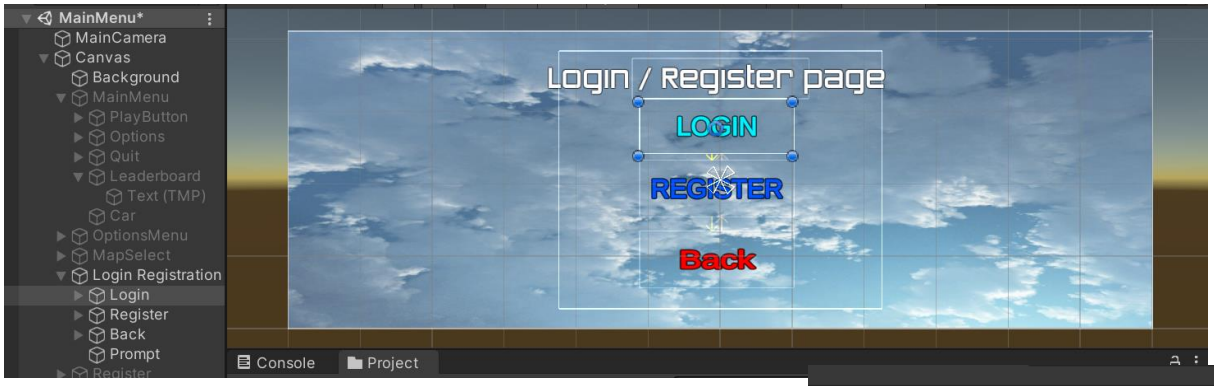
To test this, I created a temporary login page called Login / register which would act as a place holder for the real login/register

Link to view video evidence:

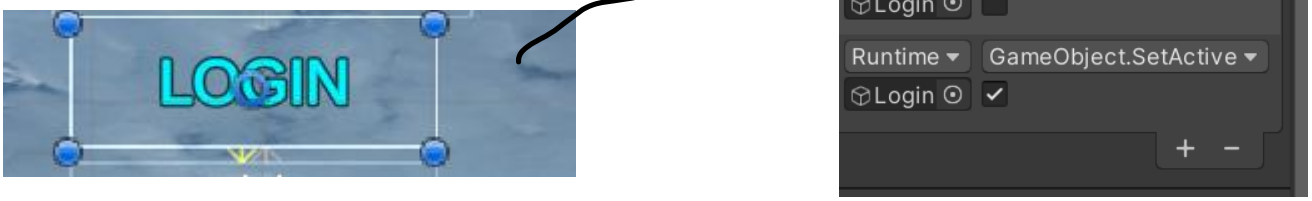
https://vimeo.com/663684939/b78f7295b6

Login

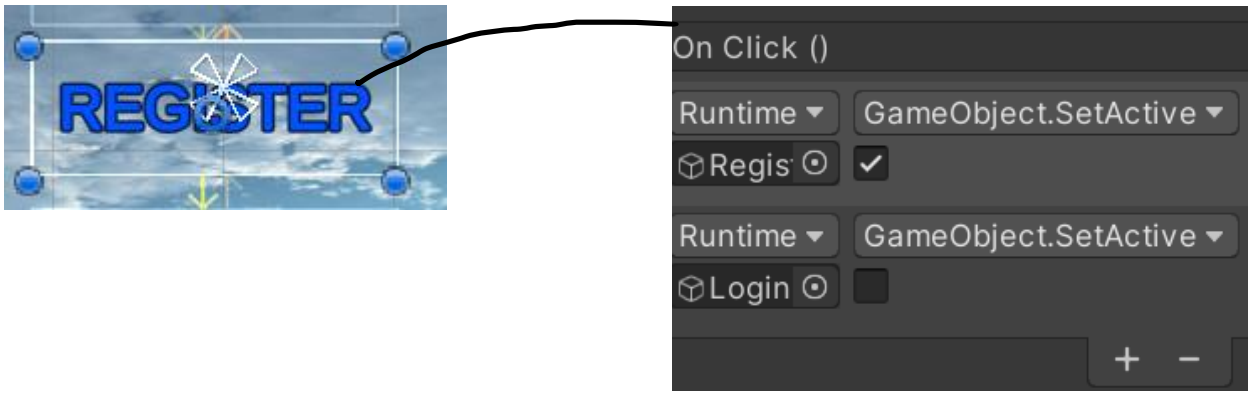
Login Registration GUI



Login Button



Register button



Stage 0 Review:

What has been done

In this stage I have created a start game function; producing buttons for the leaderboard, options, play game and Quit function. Each button takes the user into a different panel, such as the play button taking the user to the login/registration page and the Leaderboard button taking the user to the Login/Registration page.

How has it been tested

The variables have been tested as described. Most of the menus can only be tested by casual testing as no destructive testing can occur here. Most of the testing comes from casual testing to see if the buttons display the correct panel.

I found some errors with the assignment of buttons calling some functions, such as the options menu which was not calling the options panel when called. To debug I carefully looked around the code and realized it was assigned to the wrong panel, which ended up making the options panel work successfully.

There were minimal amounts of code in this section so syntax errors where not a large problem, it was more about the assignment of variables which I will be more careful of in the future.

Testing checklist

Functions to be tested	Is it working
Game buttons call and display the correct menus.	Yes

How it meets success criteria and user expectations

The areas of the success criteria which have been ticked off are:

Large buttons on each decision, simple decisions per screen or decisions are defined in the walkthrough area.	Screen shot the next pages in the GUI which show decision screens.
Simplistic design which is usable with the varied demographic (abstraction) of items which can target one or the other. This also includes non-confusing navigation panels	Screen shot windows which the user interacts with, include tip boxes and text boxes for a simplistic design

Changes in the design that have resulted from stage

A small change I have implemented are new menus: options menu was not originally planned in the design however I have implemented this as a new GUI to give the user some more exploration features.

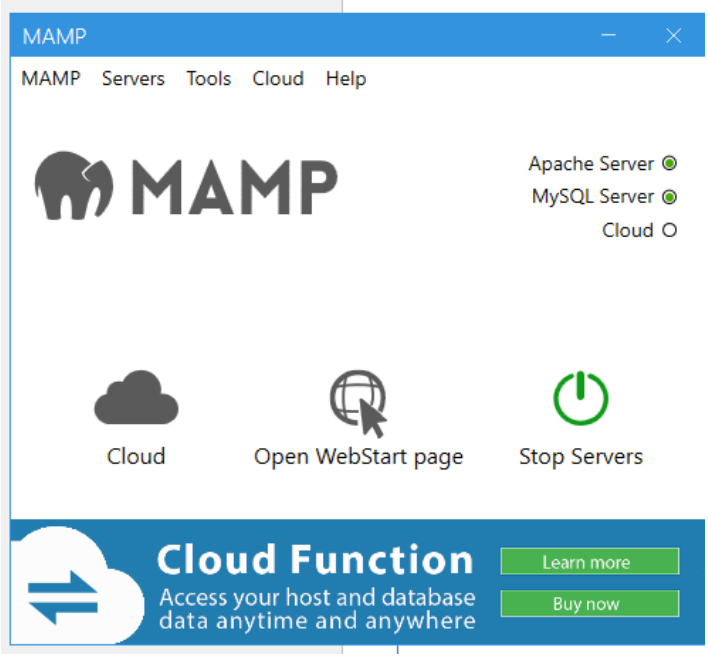
I have also changed around the Game Map select and Game Mode select Menus to give a better feel of the game GUI.

Summary of the project as a prototype at this stage

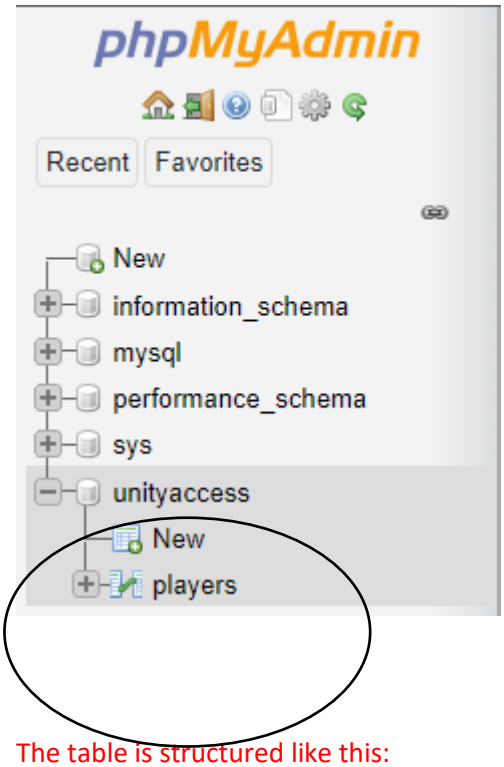
So far, my project is going okay and everything I have intended to complete in this stage has been completed.

Stage 1: Creating the database

This stage is where the I will create a local hosting database to store my user's username, password, and score. I will use MAMP; a local host web-server off my personal machine to run the software.



To store player data, I will be using a MYSQL server; therefore, to access data I will input SQL commands through a PHP script connected to the database. The database will be relational, although I am currently only having a single table this means I can expand the database for the future growth of the car racing game.



I have created a database called unityaccess to be accessed by the game, creating a table called players which will store the currently required data.

The table is structured like this:

	#	Name	Type	Collation	Attributes	Null	Default	Comments	Extra	Action
☐	1	id	int(10)			No	None		AUTO_INCREMENT	Change Drop More
☐	2	username	varchar(16)	utf8_general_ci		No	None			Change Drop More
☐	3	hash	varchar(100)	utf8_general_ci		No	None			Change Drop More
☐	4	salt	varchar(50)	utf8_general_ci		No	None			Change Drop More
☐	5	score	int(100)			No	None			Change Drop More

Having 5 columns, ID, Username, hash, salt, score.

ID = autoincrements the position of the index

Username = stores the username of the player

Hash = hashed password so password cannot be seen (cryptography)

Salt = used to decrypt the hash

Score = stores the score of the user

Success criteria (Database management)

Criteria	How to evidence
Validation of user inputs compared to those stored in the array This includes a hash function for the password to compare the password entered to the user (hashing this password with the same characters) and comparing this one to the one stored in the database	Screen shots of the code which manipulates and compares user input data with the current data stored inside of the array

My success criteria clearly states using a hash function to store encrypted password data in the database. When creating my table, I created a float integer column which will store the hash of the database.

The observable table looks like this:

```
SELECT * FROM `players`
```

When a user log in/ registers all data will be compared from the table players.

id	username	hash	salt	score

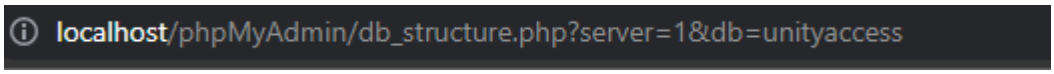
I have pre-planned the validation of users logging in by storing the salt, used to create the hash, this will be used to validate user passwords, by applying the salt (stored in the database) to the input password, compared with the hash. This validates the user input; I also do not have to store the hash in PHP scripts which can decrease security.

I have created this table as a sub-table so that its modular design means I can change and edit the database and database structure without changing the nature of the code.

Stage 1 review:

What has been done:

So far, I have created a local database accessibly from my machine using the software MAMP.



I have used a virtual local location, voluntarily hosting the database from my machine, and choosing when to run the servers.

The single table contains all of the immediate information needed inside of the game by using the success criteria and defined data stated by the criteria.

How has it been tested:

I have not been able to test the database in terms of inputs or outputs, nor connected it to a php script. When connected to a php scripts with inputs and outputs, the database will be stress tested destructively to try and find an error.

How it meets the success criteria and user expectations:

The database design has been created so it houses all of the data expected by the user including all inputs of credentials and scores, so when logging back in the user can review on the leader board their previous score which still remains, and the contents of the leader board do not disappear, so the data base continuously stores data without being changed when a new user signs up.

Success criteria:

The most recent user score is uploaded into the database	Code facilitating the input of the array and the manipulation of the data will be shown in screen shots.

The most recent score accomplished by the user can be uploaded and stored in the database by incrementing the integer value 'score'

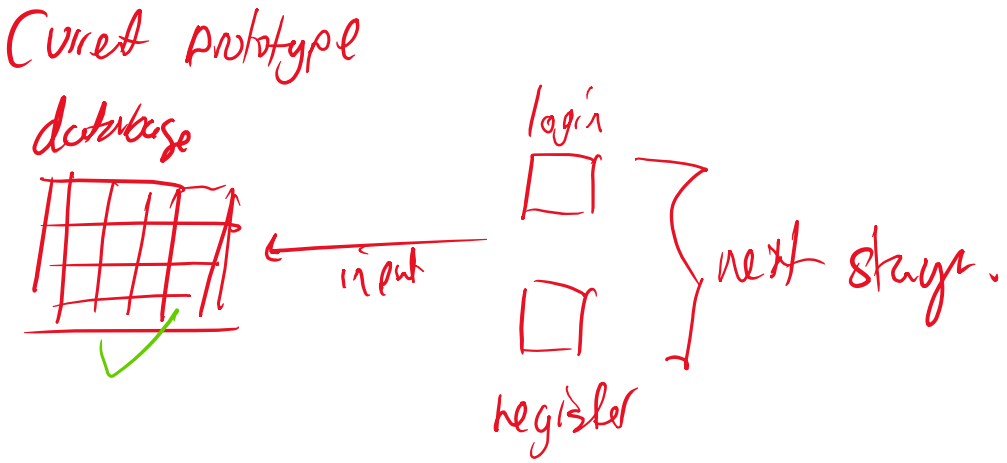
Changes in the design that have resulted from this stage:

There has been no changes in the design so far.

Criteria	How to evidence
Validation of user inputs compared to those stored in the array This includes a hash function for the password to compare the password entered to the user (hashing this password with the same characters) and comparing this one to the one stored in the database	Screen shots of the code which manipulates and compares user input data with the current data stored inside of the array

Summary of the project as a prototype at this stage:

So far, the project has been successful and followed the design well, I have created good foundations to be able to create login/register php files and create forms in C# which will carry data to the database.



Stage 2: Registration

The login / register pages are the first input pages where the user inputs their own username password inputs. This stage is also split up into 2 areas each with 3 different components:

Login

Login GUI

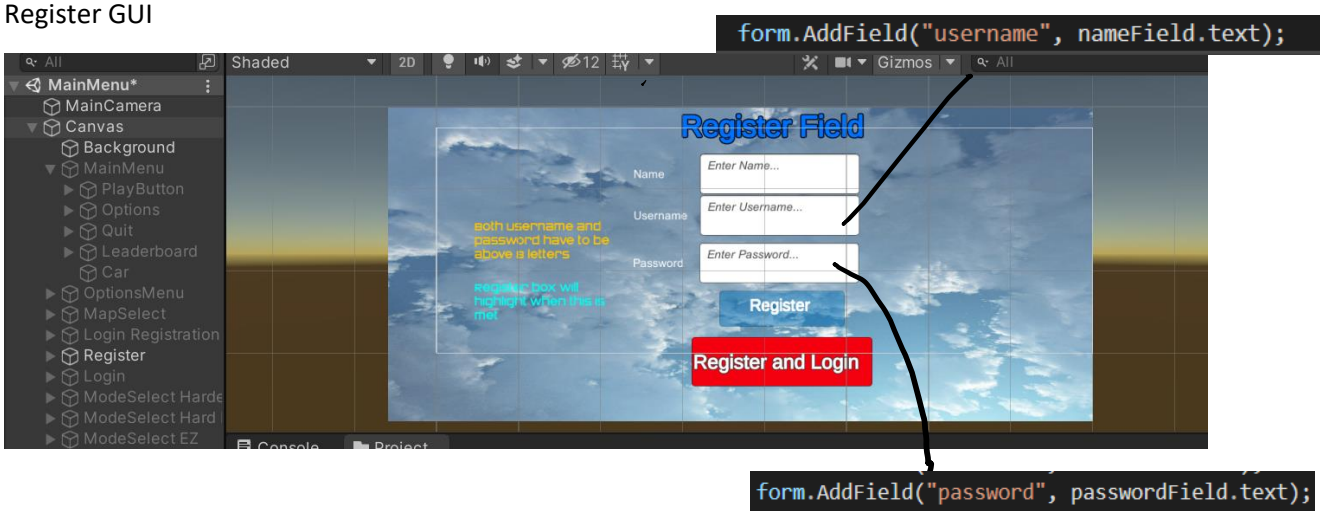
Login php

Login C#

Register page

- Register GUI
- Register php
- Register C#

Register GUI



Register button


```
public void VerifyInputs()
{
    submitButton.interactable = (nameField.text.Length >=3 && passwordField.text.Length >= 5
}
```

The register button has disabled as interactable to be able to let the player know that the inputs entered are not valid, there has been text walkthrough on the side of the Menu to let the player know the parameters of the password and what needs to happen. To aid all demographics then the button becomes interactable to show these conditions have been met.

Register C# script

What does the script do?

- Assigns the username and password field inputs into a variable
- Calls the coroutine to start the register script
- Produces a form to connect to the register php script
- Adds values username and password to php
- Adds the form to the php
- If no errors in the php script it suggests a user is created successfully, else it prints the error of the php.
- Validates the user inputs – making sure username length is more than 3 and password is more than 5

Assigns the username and password field inputs into a variable

```
public class RegistrationSQL : MonoBehaviour
{
    public InputField nameField;
    public InputField passwordField;
    public Button submitButton;
```

Declaring these public input fields means the entered credentials can be taken as inputs. These are shown in action on the diagram above.

Calls function to start the register script

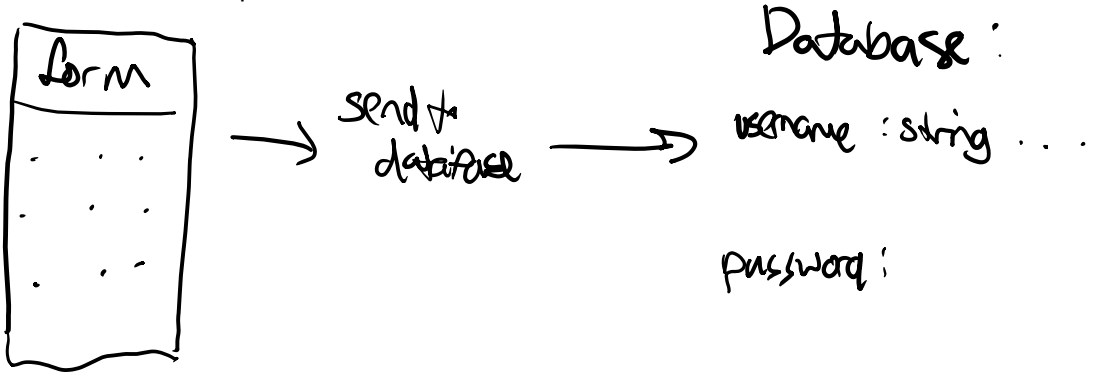
```
public void CallRegisterSQL()
{
    //starts coroutine to register user
    StartCoroutine(Register());
}
```

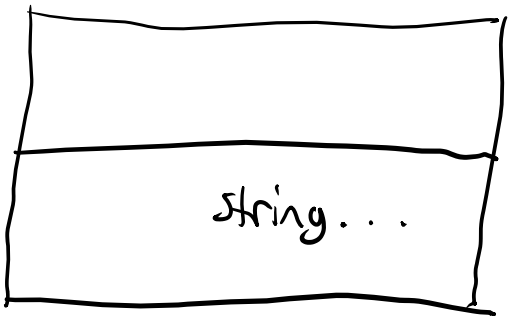
We will be communicating with the web through the www class, we need to create a coroutine which calls the www form and accesses the database. This calls a middleman class which starts the coroutine allowing this to happen.

Produces a form to connect to the register php script

```
//creating form and posting this to the surver
WWWForm form = new WWWForm();
```

New WWForm () is a built-in unity form which creates a page where I can input data, which then adds this to the respective fields on the database.





Adds values username and password to php

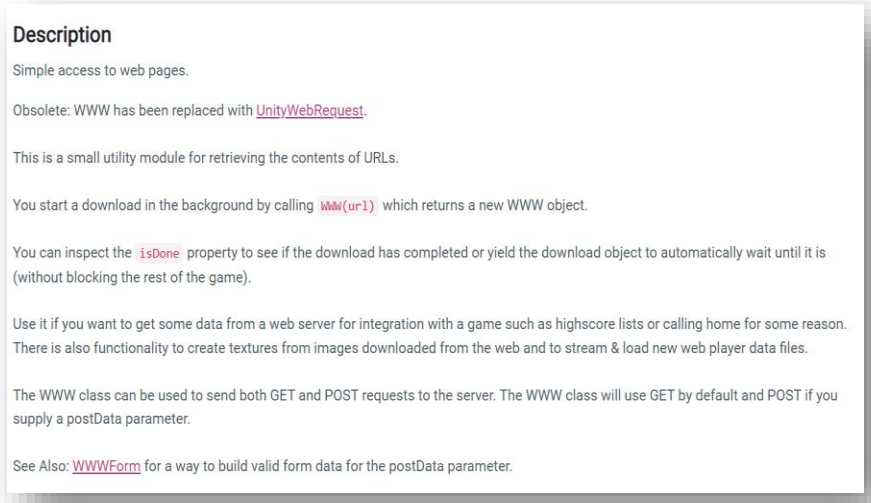
```
form.AddField("username", nameField.text);
form.AddField("password", passwordField.text);
```

These conditions add to the form sending to the unity web request and add this to the form created in the previous statement.

Adds the form to the php

I am using a function called WWW www which is a class in unity which accesses webpages. I am using it because I am using a local database on my machine which is accessed via the internet, shown on a webpage.

The Unity documentation gives examples of accessing scores from a webserver, basically accessing a data from an online server.



Description

Simple access to web pages.

Obsolete: WWW has been replaced with UnityWebRequest.

This is a small utility module for retrieving the contents of URLs.

You start a download in the background by calling `WWW(url)` which returns a new WWW object.

You can inspect the `isDone` property to see if the download has completed or yield the download object to automatically wait until it is (without blocking the rest of the game).

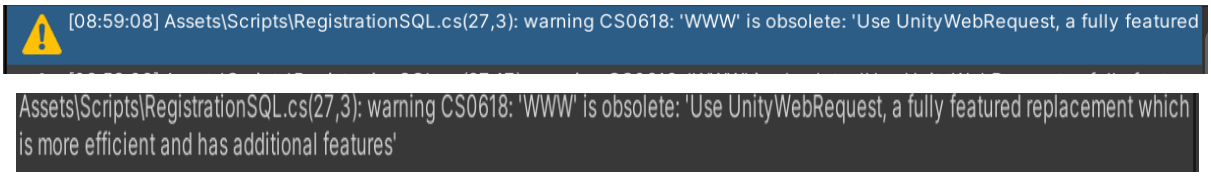
Use it if you want to get some data from a web server for integration with a game such as highscore lists or calling home for some reason. There is also functionality to create textures from images downloaded from the web and to stream & load new web player data files.

The WWW class can be used to send both GET and POST requests to the server. The WWW class will use GET by default and POST if you supply a postData parameter.

See Also: WWWForm for a way to build valid form data for the postData parameter.

```
WWW www = new WWW("http://localhost/sqlconnect/register.php",form);
```

This code then accesses my web-based database, calling the php script register.php to transfer my put username and password data.



When running my game to find any syntax errors in the code, this warning shows that WWW function is obsolete meaning it is unsupported. I had to improve my code to use the function UnityWebRequest.Post to access my webpage.

```
using (UnityWebRequest www = UnityWebRequest.Post("http://localhost/sqlconnect/register.php", form))
```

If no errors in the php script it suggests a user is created successfully, else it prints the error of the php.

```
yield return www;
//if everything in the php script
if (www.text == "0")
{
    Debug.Log("User created successfully");

}
else
{
    Debug.Log("User creation failed. Error #" + www.text);
}
```

Like before I found another warning in the Console projecting how WWW is obsolete, I again had to adapt my code to use the UnityWebRequest.Post function in replace of the www to produce:

This change looked like:

```
yield return www.SendWebRequest();
if (www.text = "0")
{
    Debug.Log(Debug.Log("User created successfully"); t);

}
else
{
    Debug.Log("User creation failed. Error #" + www.text);
}
```

The yield return function holds the function until the request is returned, I adapted this to include the function www.SendWebRequest (); since its counterpart was obsolete.

The errors appearing where:

```
1.cs(29,21): error CS1061: 'UnityWebRequest' does not contain a definition for 'text' and no access
```

Which was this portion of my text, I changed this to also support the UnityWebRequest.Post Function.

```
yield return www.SendWebRequest();
if (www.text = "0")
{
    Debug.Log("User created successfully");
}
```

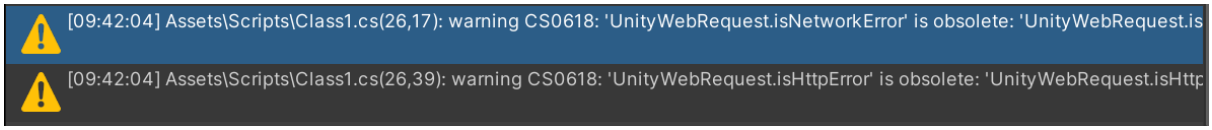
I changed the rest of my code to support this new function, which still displayed some obsolete functions

The rest of the code is displayed below

```
yield return www.SendWebRequest();
// isNetworkError always comes true, else is to check for logs
if (www.isNetworkError || www.isHttpError)
{
    Debug.Log("User creation failed. Error #" + www.text);
}

else (www..text == "0")
{
    Debug.Log("User created succesfully!");
    UnityEngine.SceneManagement.SceneManager.LoadScene(MapSelect);
}
```

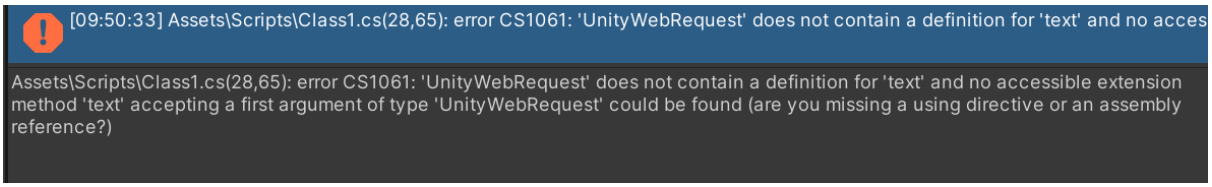
if (www.isNetworkError || www.isHttpError) is an iterative If statement saying if there is a www.isNetworkError or www.isHttpError print the user creation failed text, the error below shows the isNetworkError is obsolete as well, meaning that this is unsupported and also caused an error:

Error shown below:



Meaning that the code would not run. After formatting the code to incorporate the UnityWebRequest and finding some of the functions where obsolete meaning they were not able to download the echo '0' in the php to show no errors then the code couldn't run.

To attain the echo 0 showing no errors occurred in the php file, I reverted back to using (WWW



www) obsolete code as this still supported getting text from the php file even after trying to use new unity functions to stop this warning from happening.

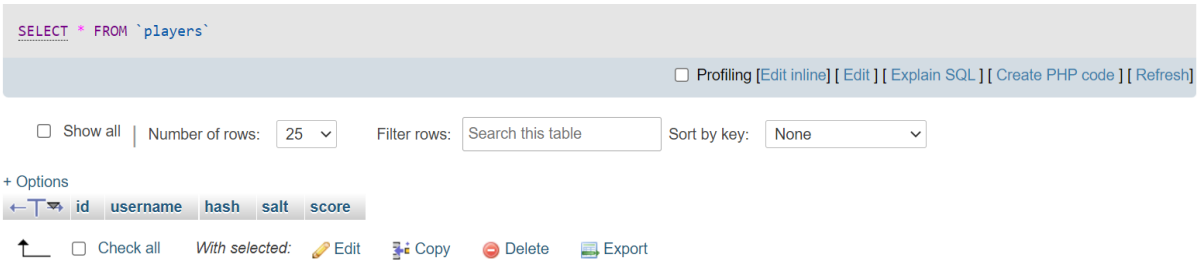
Validates the user inputs – making sure username length is more than 3 and password is more than 5

```
    //verifying inputs from the text fields called by the register button
    public void VerifyInputs()
    {
        submitButton.interactable = (nameField.text.Length >=3 && passwordField.text.Length >= 5 );
    }

}
```

Register php

What does the script need to do?

- Connects to the database // prints otherwise if failed
- Takes the values input from C# script and converts these to variables in php
- Selects the username row from database and performs a name check to see if that username already exists
- Hashes the password and stores this as salt, storing the hash in the database for user login
- Inserts these new values into a new row containing (username, password, salt, hash, score)
- Prints error if failed connection or prints 0 to show user success

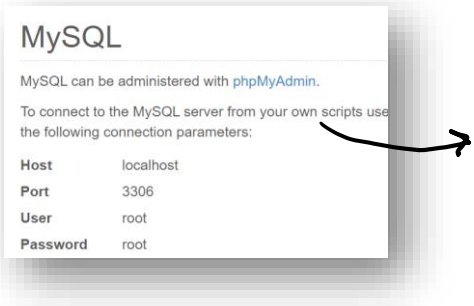


This is the php database where the players login/registration values entered will be stored // as well as the score

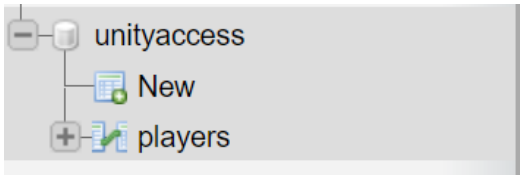
Connects to the database // prints otherwise if failed

```php
<?php


$con = mysqli_connect('localhost', 'root', 'root', 'unityaccess');
    // check for successful connection.
    if (mysqli_connect_errno())
    {
        echo "1: Connection failed"; // Error code #1 - connection failed.
        exit();
    }
```

The $con variable creates a new variable which connects directly to the web database. It's a local database and so the location is a localhost created off my laptop, root and root are both my username and password for the connection and unity access is the field where the database is stored.



This is the php admin variable where the SQL connects to my local host online servers searching the table unityaccess



```
//conveting to php variables
$username = $_POST["username"];
$password = $_POST["password"];
```

Takes the values input from c# script and converts these to variables in php. From the form we created earlier, using the _POST function built into php I have attained the username and password variables sent by the C# script.

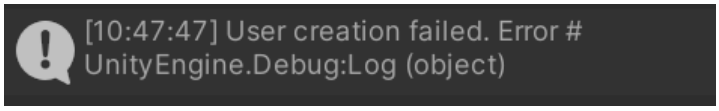
```
form.AddField("username",
form.AddField("password",
```

Selects the username row from database and performs a name check to see if that username already exists.

```
//selects username column from database, obtains all stored usernames
$namecheckquery = "SELECT username FROM players WHERE username='" . $username .;
```

Using SQL data, I have SELECTED the username column from the players table where the username is passed as the value $username defined earlier.

From casual testing to see if my username and password would register this error appeared in my Debug.Log. Printing no #Text after the Debug.Log statement means there is an error with my php syntax.



Because this error did not appear in the C# format, I know it was from the php script.

The error is shown below:

```
$namecheck = mysqli_query($con, $namecheckquery) or die("2: Name check failed - a username with this name already exists"); //
Error code # 2 - name check query failed.
```

I realized on my $namecheckquery the SQL query is not finished, there is no “;” used to end SQL statements query. I realized this syntax error by carefully looking through each section of the php script.

WHERE Syntax

```
SELECT column1, column2, ...
FROM table_name
WHERE condition;
```

This is a post from W3Schools showing the WHERE syntax and its need for the ";" at the end of the

```
//selects username column from database, obtains all stored usernames
$namecheckquery = "SELECT username FROM players WHERE username='" . $username ."';";
```

statement. To change this, I incorporated this into the code below: At the end of the line is a ";" in brackets which signifies the end of the SQL line's end.

Hashes the password and stores this as salt, storing the hash in the database for user login

```
$salt = "\$5\$rounds=5000\$" . "safetycharacters" . $username . "\$";
$hash = crypt($password, $salt);
```

First the $Salt creates a salt, then we are sing SHA-256 encryption method to encrypt the password, the rounds = 5000 shifts the password characters around 5000 times to create a mass of random character password. The /$5 indicates that this is the type of encryption I am using.

The "safety characters "text adds converts the password into the salt using these characters. I have passed the username and concatenated this to add some randomization to the password and the salts are not the same for each user. I won't pass the password in this text as it is visible inside of the database.

The $hash is a variable which stores the password variable mixed in with the salt, crypting is a built-in function, using the past parameters of $password and $salt to $crypt function into the $hash variable.

Inserts these new values into a new row containing (username, password, salt, hash, score)

```
$insertuserquery = "INSERT INTO `players` (`id`, `username`, `hash`, `salt`, `score`) VALUES (NULL, '" . $username . "', '" .
    hash . "', '" . $salt . "', '0');";
```

This statement adds the ID (which is incremented 1 per each row added), username, hash salt and score as parameters into the players table in the database. The VALUES statement passes the active values for each variable, the $username is defined in the C# script, the hash is the variable which we just created which is a mixture of the salt and password, the score is passed as the user needs to set a value of 0 to their score.

The $Salt is being passed as we will need this for user login, the salt will be applied to the user's entered password on login to see if it matches the one already stored in the database.

Prints error if failed connection or prints 0 to show user success

```
    mysqli_query($con, $insertuserquery) or die("4: Insert player query failed"); // Error code #4 - insert query failed.


    echo("0");
?>
```

This line of code is the final line which passes all the new data, mysqli_query enters a query to the database connecting through the $con variable, passing the parameters from the $inseruserquery variable.

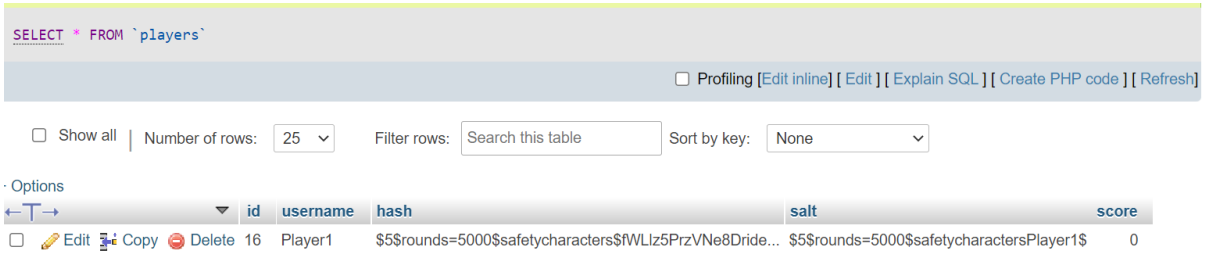
The die function occurs which is basically an else statement, printing the query failed, once this script is completed successfully the 0 value is passed.

Returning this value:
Allows the C# script to proceed, changing back to use the function WWW www to attain the www.text from the php file.

```
//if everything in the php script
if (www.text == "0")
{
    Debug.Log("User created successfully");

}
else
{
    Debug.Log("User creation failed. Error #" + www.text);
}
```

After Game Map mode created: (successful register)

https://i.gyazo.com/724e5ebd513605845f507ee87deb9ad1.mp4



After Game Map mode created: (unsuccessful register)

https://i.gyazo.com/f9177f26ce560080b2480bd5e192b7cb.mp4

Stage 2: Registration Review

What has been done

So far in this stage I have successfully created the Registration code which are now completely working; new users can successfully register into the database with usernames and passwords which are validated for length and see if they have been previously used before.

How has it been tested

Destructive testing

Debug.Log (print statements)

Constructive testing

I have tested my registration by creating a temporary user, Player1 and using this to monitor the processes happening.

While following the process, Debug.Log statements such as “User created successfully” and “Username already created” allowed me to identify what was going on behind the scenes and make sure the correct branches were being followed based on the inputs.

The final form of testing was destructive testing, where I input character strings outside of the variable length aswell as special characters to see how the C# script would respond; purposely trying to make the system crash or not know what to do so I would be able to fix the error.

Testing checklist:

Credential input boxes create a new row in a database storing username, password, score	Yes
The user cannot login/register until the conditions for the username and password are met, the register/login button is not interactable	Yes
Credential input boxes create a new row in a database storing username, password, score	Yes
Password is stored as a hash inside of the database	Yes

How it meets success criteria and user expectations

Screen shots from Success criteria:

Simplistic design which is usable with the varied demographic (abstraction) of items which can target one or the other. This also includes non-confusing navigation panels	Screen shot windows which the user interacts with, include tip boxes and text boxes for a simplistic design
User input text boxes are clear and know when to be interacted with on the user authentication page	Screen shot examples of how the text boxes are designed to benefit the user.

Subroutines across the GUI are called without fail	Facilitating code showed where routines are called, and functions are defined.

Algorithms for back-end:

Algorithms for backend
Checking if password and username are both above 8 chars for safety.
Storing user input from unity and transferring this to a players database.
Active scores inside of the game constantly being added into the database to print the best score.
Transferring the data variables from temporary stores in the c# script into variables from
Validating password and username variables are stored in the correct spaces
Validating there are no repeated usernames stored in the database, when c# checking for user input this as a Debug. Log.

Currently the combination of my register and login C# and PHP scripts has covered all the algorithms needed for back end.

The features of: Validating users' username and passwords has been implemented through a length validating technique, this validation also occurs through checking if the username already exists in the table as a primary key unique identifier.

Changes in the design that have resulted from stage

There were some errors I had to change with using obsolete code which led me to change my code when I received errors it was not supported. Instead of using www. to create a form, I used UnityWebRequest which then worked better.

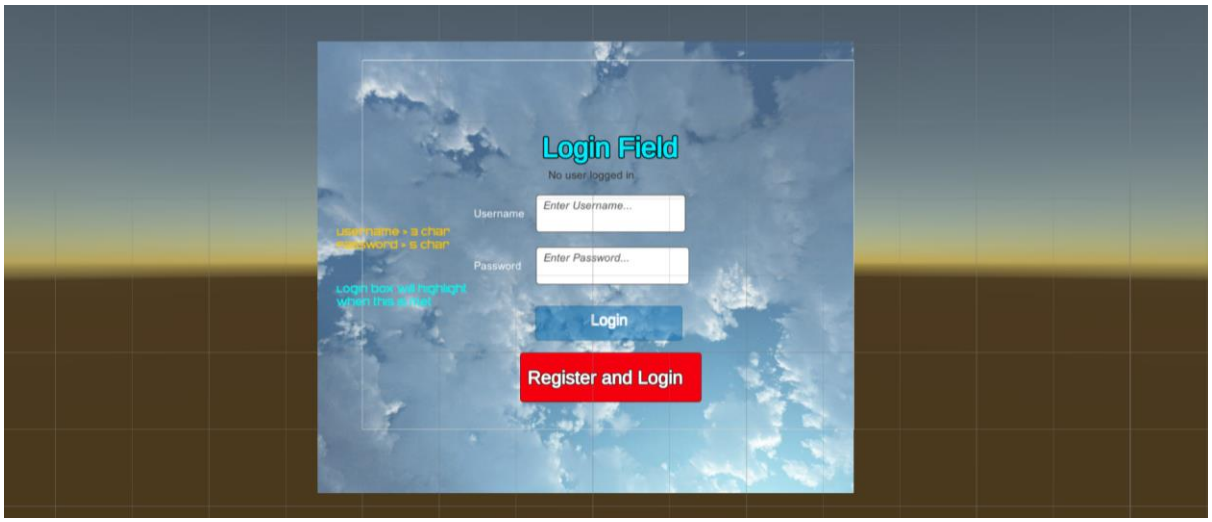
Summary of the project as a prototype at this stage

So far, the project as a prototype is really coming along. I have finished the first phase of the game which includes establishing a user connection to the database which can then be pivoted around for the user to play the game. For the future prototype, as the game maps have already been developed, I am looking to add functionality to each of them and an extension of the GUI to link the menu and playable car together.

Stage 3: Login

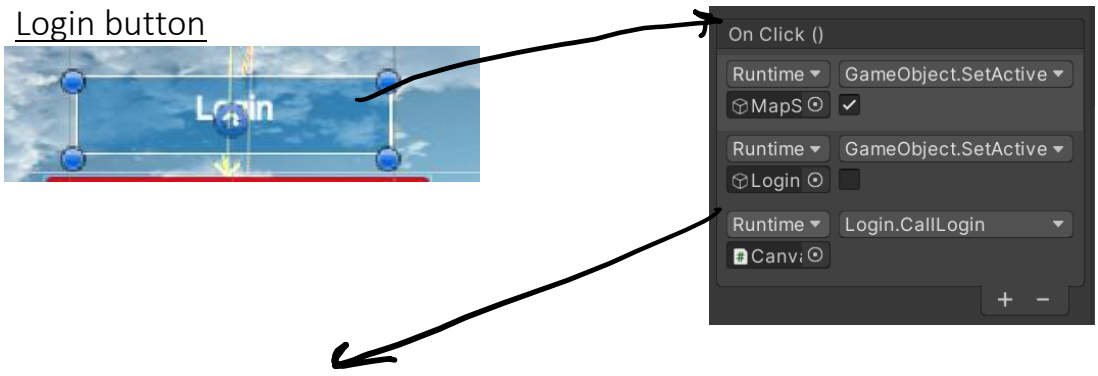
Login Field GUI

The login Field GUI is the interface where the user signs into the game. The GUI has two possible outcomes, the user enters incorrect username and or password and the login fail, or the user proceeds and chooses the map and game mode. The Login page will be like the Register page, activating the Login button once the conditions are met, using an SQLI query to communicate with the web database.



Meeting the success criteria, there is walkthrough text talking about the conditions to be met from the username and password, and an insight on activation on the Login button.

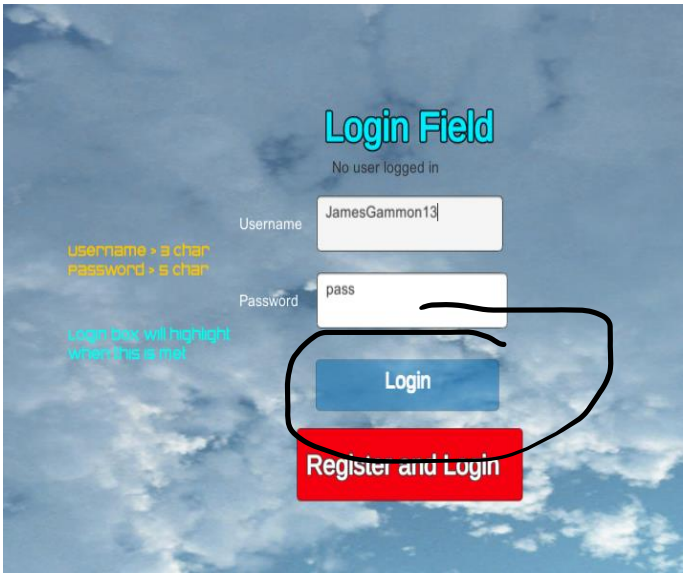
Login button



```
public void VerifyInputs()
{
    submitButton.interactable = (nameField.text.Length >=3 && passwordField.text.Length >= 5 );
}
```

The displayed Login button only activates once the conditions are met. The submitButton.interactable is a predefined function for the declaring the gameObject submit Button to be interactable.

Changing this value from true to false.



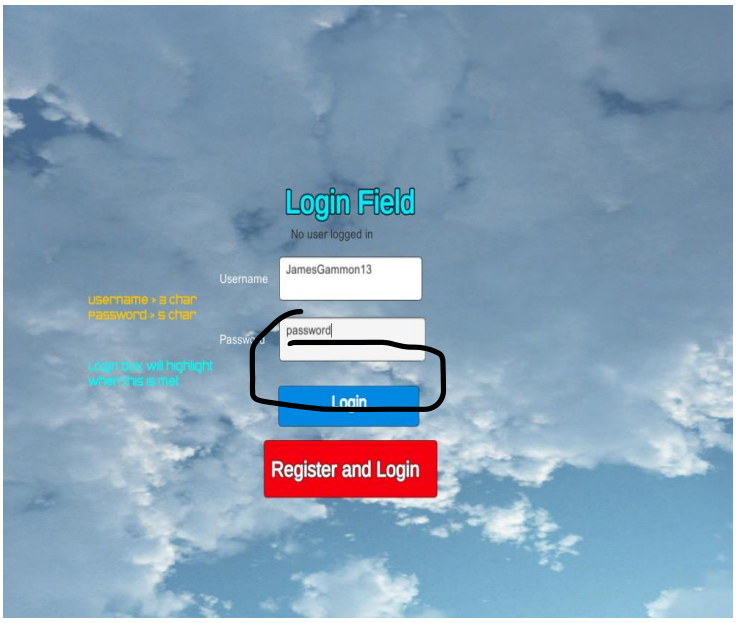


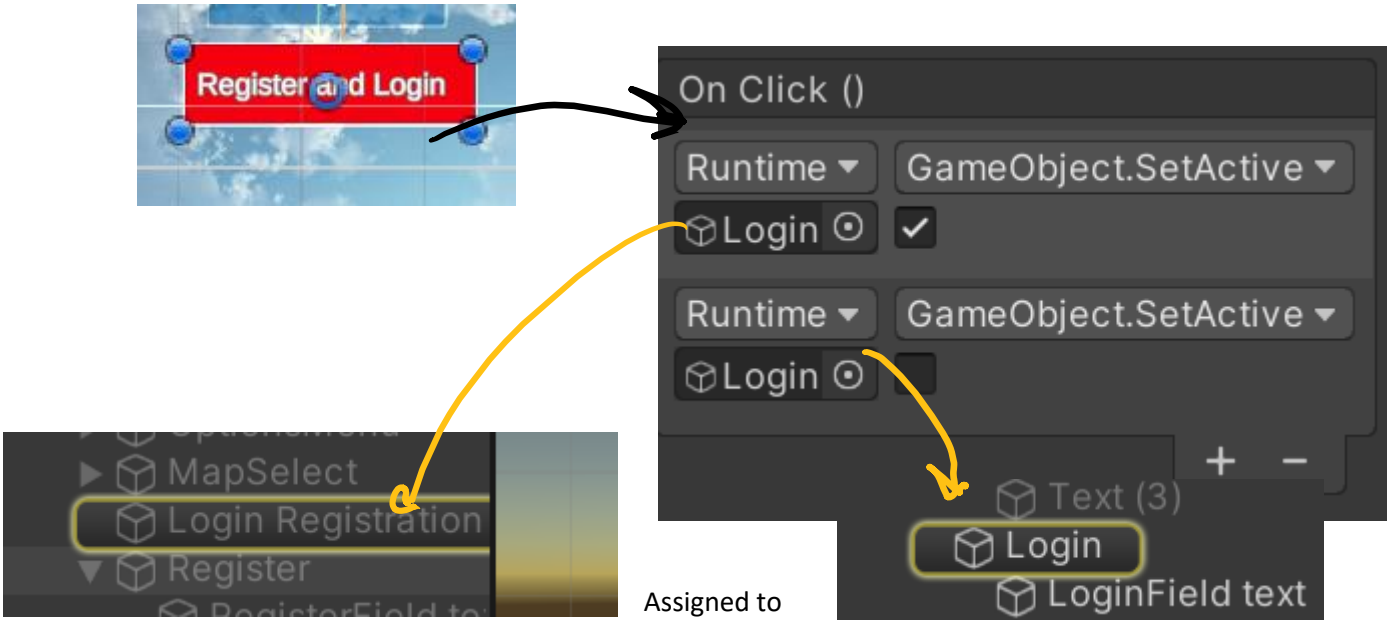
Figure 1

Figure 2

Figure 1 above showing password box is not above 5 char and the submit button is not interactable

Figure 2 above showing the button is interactable once the user has met the conditions.

Register and login button



Assigned to the Register and Login button are the GameObject.SetActive functions which are built directly into unity. They allow me to change panels by setting canvas active and not active (Boolean value)

In the diagram the Register login button (On Click) sets active the Login/Registration panel and deactivating the Login panel which is currently displaying on the screen.

Login C# script

What does the script do?

- Takes inputs of username and password
- Starts coroutine login player
- Adds the takes credentials and passes these as a parameter to the login script to communicate directly with the database
- Returns a logged in print statement if correct or user login fail error
- Verifies the inputs of username and password

Takes inputs of username and password

```
public class Login : MonoBehaviour
{
    public InputField nameField;
    public InputField passwordField;
    public Button submitButton;
```

This part of the code takes 3 inputs, the nameField for the username, passwordField for the password and submit Button for the Login button, the input nameField is defined as it is used in the php to be compared to the username stored, it is also assigned later to the variable DBManager.username to be printed on the game screen and leaderboard function. The passwordField is defined for the same reason but only for comparing with the database. The Login button is defined as once pressed – it calls the public void Call Login which starts the coroutine for the database.

Starts coroutine login player

```
public void CallLogin()
{
    //starts coroutine to register user
    StartCoroutine(LoginPlayer());
}
```

The Call Login void calls the coroutine as a function of a middleman – to be able to, when the Login button is pressed, Login in the player creating a new form and comparing this to values in the database.

Adds the takes credentials and passes these as a parameter to the login script to communicate directly with the database

```
IEnumerator LoginPlayer()
{
    WWWForm form = new WWWForm();
    form.AddField("username", nameField.text);
    form.AddField("password", passwordField.text);
    WWW www = new WWW("http://localhost/sqlconnect/login.php",form);
    //holds the function until rest of data is recieved
    yield return www;
```

Again, like the one used in Registering, New WWForm () is a built-in unity form which creates a page where I can input data, which then adds this to the respective fields on the database.

The functions form. Add Field are now used in a different context, they take the input username and password and add these to the form to compare them to the values stored in the database.

Next the form is called (login.php) and the values are added.

Returns a logged in print statement if correct or user login fail error

```
if (www.text == '0')
{
    DBManager.username = nameField.text;
    DBManager.score = int.Parse(www.text.Split('\t')[1]);
    Debug.Log("Logged in");
    GameMaps.SetActive(true);
}
else
{
    Debug.Log("User login failed. Error #" + www.text);
}
```

Once the users' credentials are input into the php, they are compared to existing users in the database, once the php is completed the value 0 is returned and the C# script proceeds. This is where the user is notified if the credentials are correct or incorrect.

DBManager.username is a variable where the correct username is assigned to, that's why it is assigned once the user is confirmed to be an existing user. The score is assigned to the DBManager.score variable as the user is just about to play the game.

Verifies the inputs of username and password

```
public void VerifyInputs()
{
    submitButton.interactable = (nameField.text.Length >=3 && passwordField.text.Length >= 5 );
}
```

The input values are again being tested through the ways described in the validation section testing the lengths of the username and password for them to be able to be secure.

Login php:

What does the php do?

- Connects to the database // prints error if failed
- Takes the values from the c# script (username, password)
- Checks if a username value already exists
- Checks if the hashed password input matches the hashed password stored in the database
- If the password is incorrect, it prints password wrong else it returns 0 which prints a login statement in the c# script

Connects to the database // prints error if failed

```
<?php

$con = mysqli_connect('localhost', 'root', 'root', 'unityaccess');

 //check if username exist

 if(mysqli_connect_error())
 {
  echo "1: Connection Failed";
  exit();
 }
```

Here the database connection is established, using local host as a location, root, root as username and password and unity access as the local location on the MAMP database.

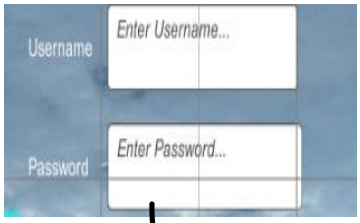
This area of code is copied and pasted from the Register.php as it performs the same function.

Takes the values from the c# script (username, password)

```
$username = $_POST['username'];
$password = $_POST['password'];
```

Like Register.php, the variables $username and $password are assigned values input from the text field in unity

Diagram of links shown below:




```
form.AddField("username", nameField.text);
form.AddField("password", passwordField.text);
```

```
$username = $_POST['username'];
```

I James Gammon – 27228 – 8292 – Abbeyfield school

Checks if a username value already exists

```
$namecheckquery = "SELECT username, hash, salt, score FROM players WHERE username='".$username."';";
$namecheck=mysqli_query($con,$namecheckquery)or die("2: Name check query did carry out ");//error code #2 name check qu

if(mysqli_num_rows($namecheck)!=1)
{
 echo "5: No username present or there is more than one row with that username";
 exit();
}
```

This section of code checks for the user's username, hash salt and score from the database, using $username as the primary for the database to uniquely identify the row, in which it is compared to the row already existing. If correct the user is a returning user.

I have used testing to identify whether the returning user is logged in using Debug. Log statements returned in the www.text print statement of the C# script.

Checks if the hashed password input matches the hashed password stored in the database, the value names are kept as "salt" and "hash" as the passwords input in the text field need to be combined with the same salt to form the same hash.

```
//get login info from query
$existinginfo = mysqli_fetch_assoc($namecheck);
$salt=$existinginfo["salt"];
$hash=$existinginfo["hash"];
```

If the password is incorrect, it prints password wrong else it returns 0 which prints a login statement in the c# script

```
$loginhash = crypt($password, $salt);
if($hash !=$loginhash)
{
 echo "6: Incorrect Password";
 exit();
}

echo "0\t" . $existinginfo["score"];
```

=

Printing this statement:

```
if (www.text == '0')
```

Errors and testing

Error 1

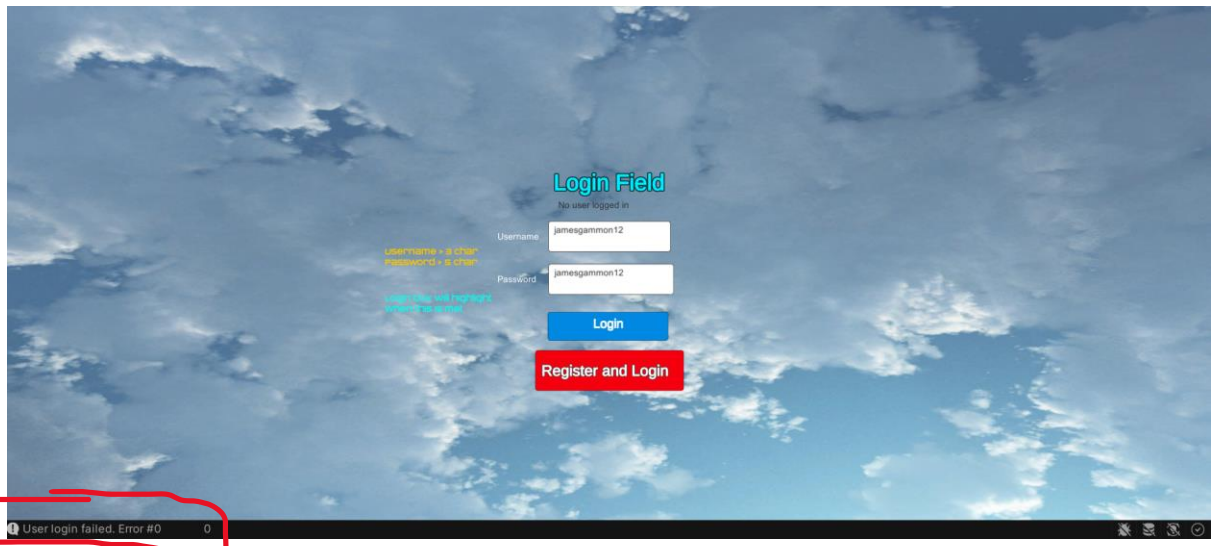


Figure 3

Figure 3 is an example of when the user successfully entered the correct credentials however the user login failed, printing the www.Text 0, which is supposed to initiate a scene change switch.

To fix this error, I had to change the following code to below:

```
if (www.text[0] == '0')
```

This is because the php code echo's 2 values at the end of the test, the 0 to indicate no errors and the score as the user has logged in, I added the increment 0 to define the position of the echo 0 statement in the text returned.

```
echo "0\t" . $existinginfo["score"];
```

My code previously was printing back the entire text: Error#0 and then another 0 to represent the score, which are split up by the '/t'. When calling back the data from the www.text == 0, I was not defining which index [0] for the 0 value or [1] for the score, hence it produced the error message as no singular 0 was being echoed.

Error 2

```
[19:19:42] IndexOutOfRangeException: Index was outside the bounds of the array.
Login+<LoginPlayer>d__6.MoveNext () (at Assets/Scripts/Login.cs:28)
```

The error above shows the list index which retrieves the 0 text returned from the login.php script is out of index range, meaning there is no '0' which has been printed or there is an element 0.

To try and solve the problem I added a print statement to print the value of www.text to see what was being returned.

```
    yield return www;
    Debug.Log(www.text);
    if (www.text[0] == '0')
    {
```

```
[20:36:14]
UnityEngine.Debug:Log (object)
[20:36:14] IndexOutOfRangeException: Index was outside the bounds of the array.
Login+<LoginPlayer>d__6.MoveNext () (at Assets/Scripts/Login.cs:29)
```

Error 1

This printed a null result, so I know that the php script isn't returning 0. To fix this I realised I had not incremented my code bertween the start and the end of the php file script; meaning no messages would be sent to the console to start the game.

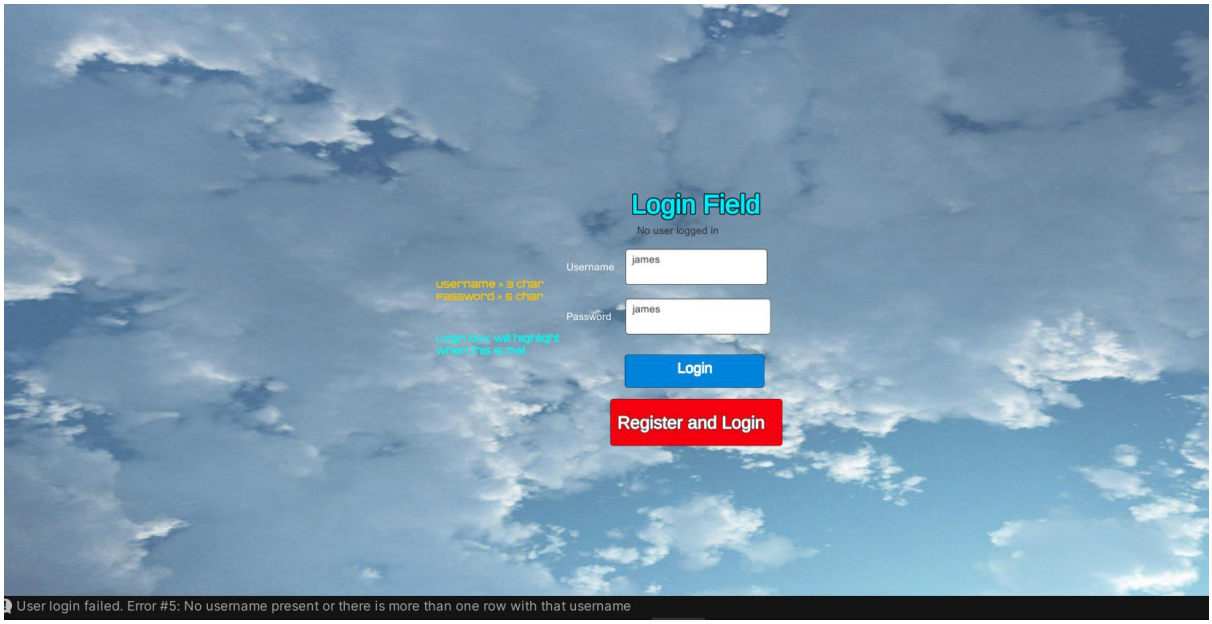


Figure 4

Figure 4 shows an example of the print statement where the username and password where not already registered, printing an error, and remaining on the screen still.

Test 2

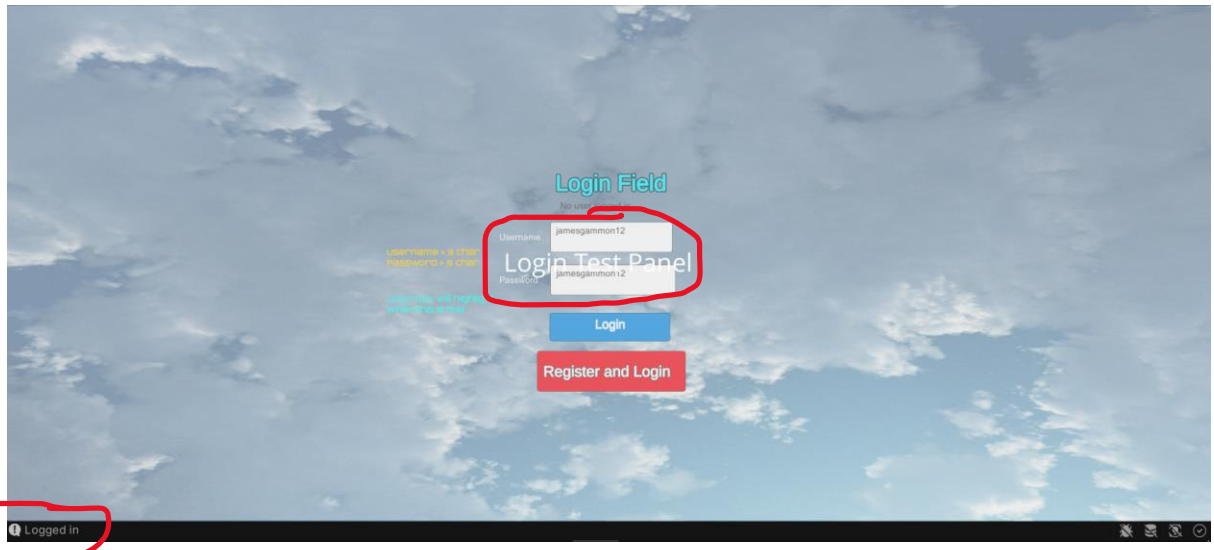


Figure 5

Figure 5 is a demonstration of a successful login stage. To test a Fake Login Test panel (seen on figure 5) was created to demonstrate the switching of canvas after the user Logged in Successfully.

Stage 3 Login Review

What has been done

So far in this stage I have successfully created the Login code which are now completely working; Users which are pre-existing inside of the database are able to validate credentials inside of the GUI and be accepted access if passwords are the same, where they are then advanced further. Logging in users is denied access if their credentials do not match the database and are then not advanced further.

How has it been tested

Stage one has a lot of different sets of code including a php files and C# files. I have used two types of testing methods: destructible testing which forces error out of the code, and Debug.Log print statements which print error messages from the php files which does not display its own testing errors, sublime text is not an IDE that has this function.

I had some large errors to do with the usage of obsolete code which led me having a wild goose chase to implement code that was supported. By trying www. And SendUnityWebRequest I was successfully able to implement the obsolete code into my game allowing the user to Log in.

Functions to be tested	Is it working
Game buttons call and display the correct menus.	Yes
Credential input boxes create a new row in a database storing username, password, score	Yes
Password is stored as a hash inside of the database	Yes
Credentials input into the user login page are compared to the database. If correct the user can proceed, else the user is denied.	Yes

The user cannot login/register until the conditions for the username and password are met, the register/login button is not interactable	Yes

How it meets success criteria and user expectations

Screen shots from Success criteria:

Simplistic design which is usable with the varied demographic (abstraction) of items which can target one or the other. This also includes non-confusing navigation panels	Screen shot windows which the user interacts with, include tip boxes and text boxes for a simplistic design
User input text boxes are clear and know when to be interacted with on the user authentication page	Screen shot examples of how the text boxes are designed to benefit the user.

Subroutines across the GUI are called without fail	Facilitating code showed where routines are called, and functions are defined.

Changes in the design that have resulted from stage

There have been some small changes from my original code which are apparent in this stage, instead of directly addressing the database through direct values, I have created a form that takes all data being changed in the data base and passing this to the login php, instead of directly writing each value from the C# script to the php then to the database.

Summary of the project as a prototype at this stage

So far, my project is growing at a standard rate, I am happy with the database aspect using usernames as a primary key which easily identifies rows, as well as the functionality and look of my menus really suits the style of game I am aiming for.

Stage 4: Choosing game map and mode

My game map and mode were chosen via the menus. After the user is logged in they are given a choice of a game map, and then an accompanying game mode for this game map. The selection choice is laid out as followed

Altogether there are 9 different game combinations to choose from; the diagram above is represented in unity below:

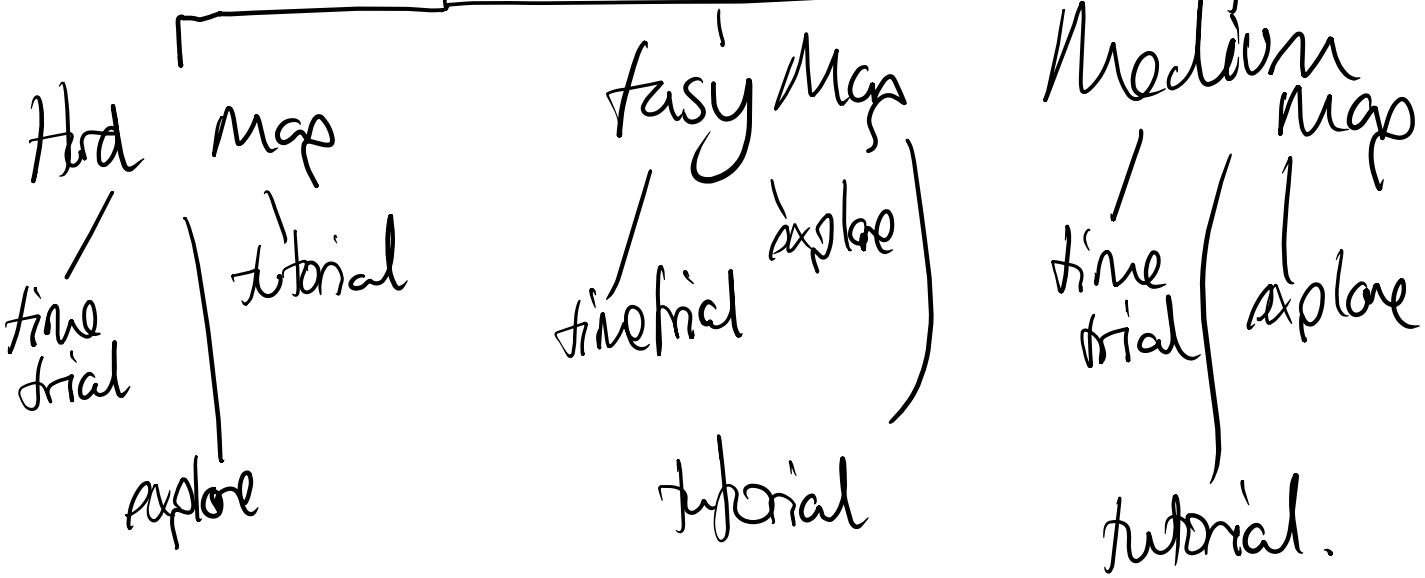
Stage 2

- Chosing game map + mode GUI
- C# script calling game decision subroutines which display maps.

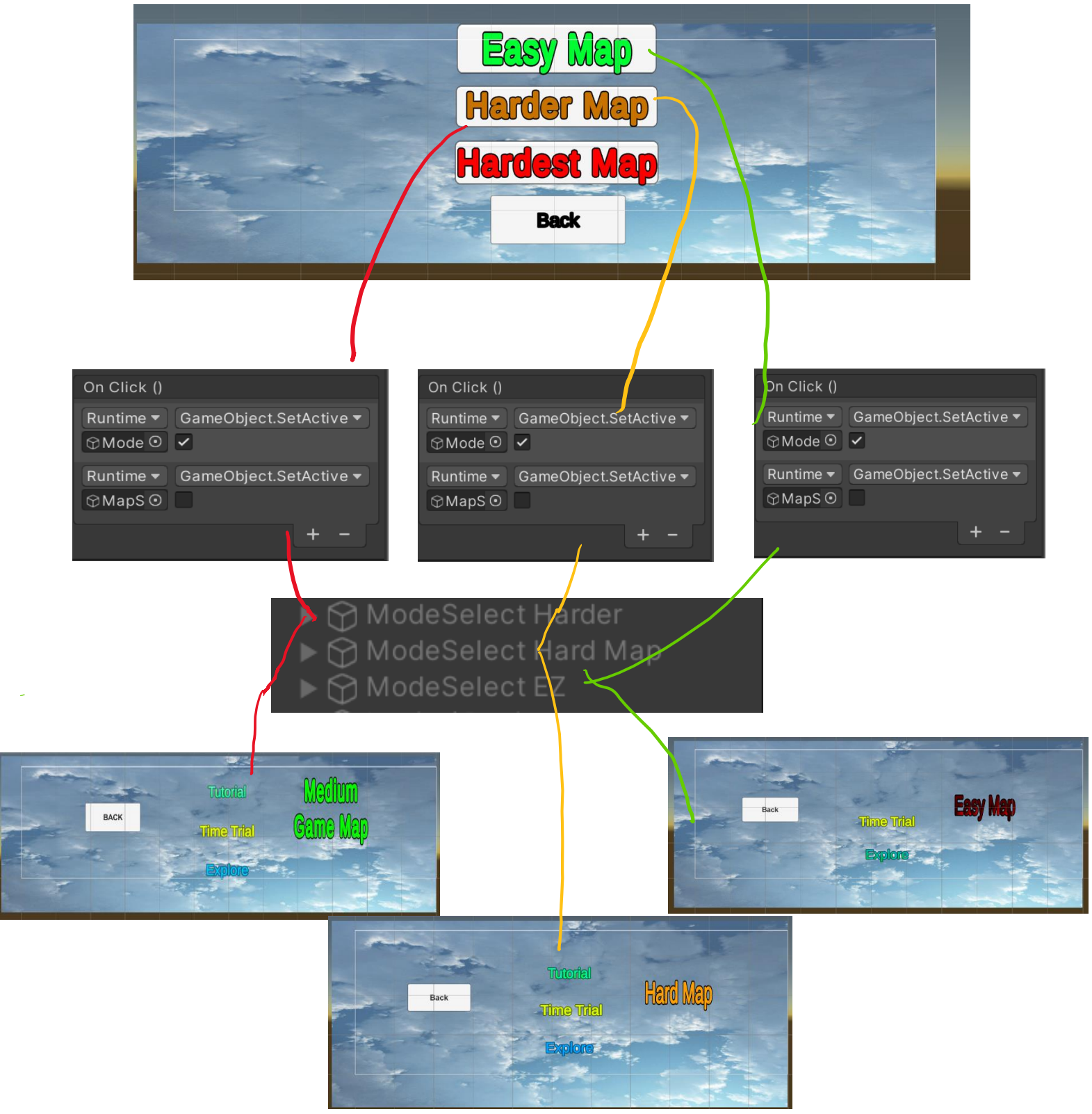
User registered / logged in.

↓

game maps displayed.



I James Gammon – 27228 – 8292 – Abbeyfield school



I James Gammon – 27228 – 8292 – Abbeyfield school

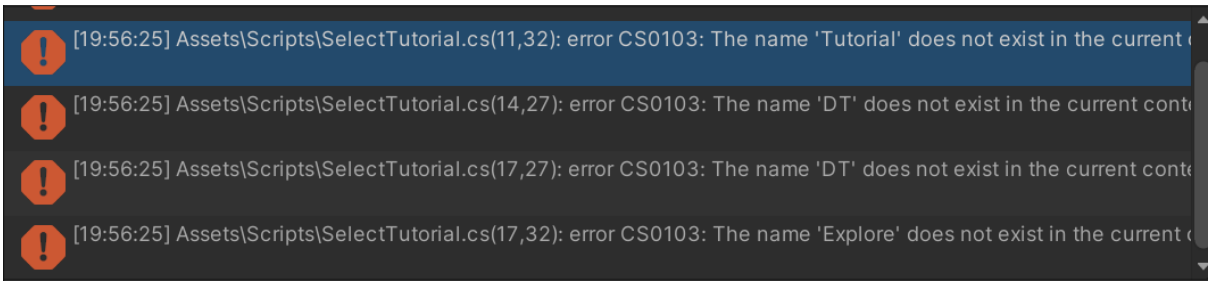
Video evidence of a successful run:
https://i.gyazo.com/23a67f93d8b6fd14a13f3382c97b82ca.mp4

Map Selecting code

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

public class SelectTutorial : MonoBehaviour
{
    public void selectScene(){
        switch (this.gameObject.name){
        case"Tutorial":
            SceneManager.LoadScene(DT - Tutorial);
            break;
        case"Free Play":
            SceneManager.LoadScene(DT);
            break;
        case"Explore":
            SceneManager.LoadScene(DT - Explore);
            break;
        }
    }

}
```

The Select tutorial is applied to the medium game map. The function case is specifically for buttons. When the Tutorial button is pressed the Medium Map Tutorial is pressed and the code is broke, the same applies for each of the maps.

Error produced:



I fixed this error by adding speech marks around each of the game maps so unity could identify them as assets.

I James Gammon – 27228 – 8292 – Abbeyfield school

Easy Map

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

public class EZSceneSelect : MonoBehaviour
{
    public void selectScene(){
        switch (this.gameObject.name) {
        case "Tutorial":
            SceneManager.LoadScene("EZ - tutorial");
            break;
        case "Explore":
            SceneManager.LoadScene("EZ - Explore");
            break;
        case "Time Trial":
            SceneManager.LoadScene("EZ - TimeTrial");
            break;
        }
    }
}

```

Medium map

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

public class SelectTutorial : MonoBehaviour
{
    public void selectScene(){
        switch (this.gameObject.name){
        case"Tutorial":
            SceneManager.LoadScene("DT - Tutorial");
            break;
        case"Free Play":
            SceneManager.LoadScene("DT");
            break;
        case"Explore":
            SceneManager.LoadScene("DT - Explore");
            break;
        }
    }

}
```

Hard Level

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

public class HLSceneSelect : MonoBehaviour
{
    public void selectScene(){
        switch (this.gameObject.name){
        case"Tutorial":
            SceneManager.LoadScene ("HL - Tutorial");
            break;
        case"Explore":
            SceneManager.LoadScene ("HL - Explore 1");
            break;
        case"Time Trial":
            SceneManager.LoadScene ("HL - Race");
            break;
        }
    }
}

```

Stage 4 review

What has been done

In my stage 4 I have further broken down my Menu into the map select and game select. I have created menus splitting down the map select to summon Hard, medium, and easy maps, each option summons the same map with different features, disabling the GUI and starting functionality.

The menus can be tracked back to a previous page by the users from the clear displayed back buttons on each screen.

The text boxes are clearly labelled and defined which the user can press, a color change present on the button indicates the choice being made.

How has it been tested

The testing has generally been carried out by casual testing, again there is limited code and mainly function buttons in the unity application. Code that has been included for the scene choice (map specifically) has had its input variables checked; menu button, and output variables checked: scene summon.

The code used for scene select was copied and pasted 3 times repeating, each with the scenes that match the map choice.

Testing checklist:

Each map selected and game mode selected produces the correct selection on the game display.	Yes

How it meets success criteria and user expectations

The users display GUI has large buttons which are each clearly defined on the game panel. Some walkthrough text has been shown on the menu screens to guide the user. Ultimately if the menus are clear and are not confusing the player will have a enjoyable interaction.

Simplistic design which is usable with the varied demographic (abstraction) of items which can target one or the other. This also includes non-confusing navigation panels	Screen shot windows which the user interacts with, include tip boxes and text boxes for a simplistic design
Subroutines across the GUI are called without fail	Facilitating code showed where routines are called, and functions are defined.

Changes in the design that have resulted from stage

There have been no changes in the design which have come from this stage.

Summary of the project as a prototype at this stage

At this stage the Menu GUIs are 90% complete, the menu to change map/game mode/ open the leaderboard/ options and quit functions are all completed and working successfully. I now need to complete the functionality of each of the game modes and after completing my leaderboard functions.

Stage 5 – The game

The game code can be split up into 3 different sections, the Time-trial game mode, the tutorial game mode, the explore game mode. Each has these components which are standard across each game mode:

Each map:

- Car controller
- Username display
- Count Down display

Car controller

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

public class finalcarcontroller : MonoBehaviour
{
    private const string HORIZONTAL = "Horizontal";
    private const string VERTICAL = "Vertical";

    private float horizontalInput;
    private float verticalInput;
    private float currentSteeringAngle;
    private float currentBreakingForce;
    private bool isBreaking;

    //[SerializeField] private float motorforce, brakeForce, maxSteerAngle;
    [SerializeField] private float brakeForce, motorforce, maxSteerAngle;
    [SerializeField] private WheelCollider frontLeftWheelCollider, frontRightWheelCollider, backLeftWheelCol
    [SerializeField] private Transform frontLeftWheelTransform, frontRightWheelTransform, backLeftWheelTrans
```

The car controller uses the front wheels to both steer and control the car. HORIZONTAL and VERTICLE are inputs straight from the keyboard. The Float currentSteeringAngle is the angle in which the cars wheels turn. Current breaking force is a component which neutralizes the momentum of the car. And isBreaking is a Boolean which checks if the car is breaking or not.

The [SerializeField] takes variables and allows me to input them inside of the inspector. Here I can add the wheel colliders for the 4 wheels, and the transform gameObjects of the wheels.

```
21
22      private void GetInput()
23      {
24          horizontalInput = Input.GetAxis(HORIZONTAL);
25          verticalInput = Input.GetAxis(VERTICAL);
26          isBreaking = Input.GetKey(KeyCode.Space);
27
28      }
29
30      private void FixedUpdate()
31      {
32          GetInput();
33          HandleMotor();
34          HandleSteering();
35          UpdateWheels();
36
37      }
38
```

Here the getInput function takes the horizontal inputs and verticalnput which allows the total steering angle to be applied to the vertical and horizontal inputs. The isBreaking bool receives an input once the space bar is pressed.

The FixedUpdate() subroutine is called immediately, and this calls each of the following subroutines.

```
private void HandleMotor()
{
    frontLeftWheelCollider.motorTorque = verticalInput * motorforce;
    frontRightWheelCollider.motorTorque = verticalInput * motorforce;

    currentBreakingForce = isBreaking ? brakeForce : 0f;

    ApplyBreaking();
}


private void ApplyBreaking()
{
    frontLeftWheelCollider.brakeTorque = currentBreakingForce;
    frontRightWheelCollider.brakeTorque = currentBreakingForce;
    backLeftWheelCollider.brakeTorque = currentBreakingForce;
    backRightWheelCollider.brakeTorque = currentBreakingForce;
}

private void HandleSteering()
{
    frontLeftWheelCollider.steerAngle = 30 * horizontalInput;
    frontRightWheelCollider.steerAngle = 30 * horizontalInput;
}
```

The handle motor subroutine applies the maximum talk to the vertical input, which is applied to the car.

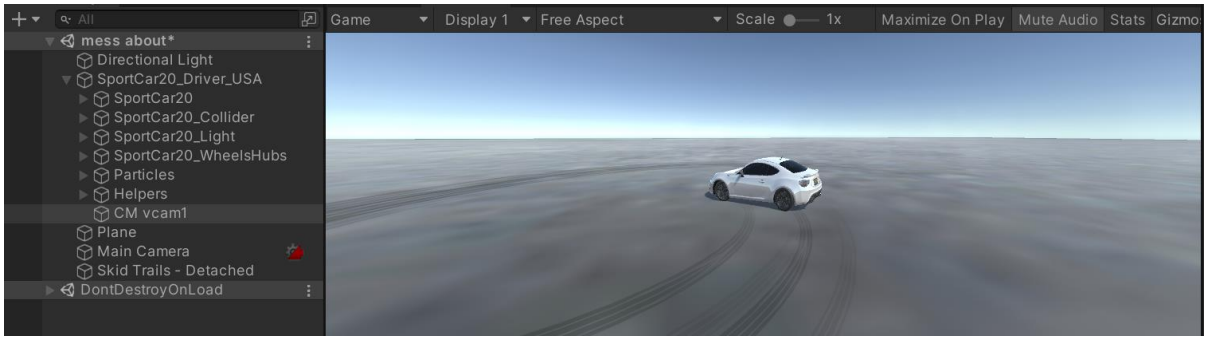
The currentBreaking will have a larger talk than the motortorque, therefore it will stop the car.

Handle steering of the front wheel colliders times this by the steering angle, which is then applied to the horizontal inputs.

```
64
65      private void UpdateWheels()
66      {
67          UpdateSingleWheel(frontLeftWheelCollider, frontLeftWheelTransform);
68          UpdateSingleWheel(frontRightWheelCollider, frontRightWheelTransform);
69          UpdateSingleWheel(backLeftWheelCollider, backLeftWheelTransform);
70          UpdateSingleWheel(backRightWheelCollider, backRightWheelTransform);
71
72      }
73
74      private void UpdateSingleWheel(WheelCollider wheelCollider, Transform wheelTransform)
75      {
76          Quaternion rot;
77          Vector3 pos;
78
79          wheelCollider.GetWorldPose(out pos,out rot);
80          wheelTransform.rotation = rot;
81          wheelTransform.position = pos;
82      }
83  }
84
85
```

Here the update wheel function passes the parameters of each of the wheel colliders to a routine called Update single wheel which takes the collider and the transform.

The quaternion is a rotation, and a vector is a speed with a direction; then the wheel collider outs the position of the wheel and rotation, and the wheel transform applies the rotation from the speed and position from the maximumTorque. These are calculated because of the wheel colliders are already being applied to.



In the image above we clearly see car's tires turning inside of the testing feature, with the toque and turning circle being applied to the vector3 position and quaternion rotation of the wheels.

Testing:
https://vimeo.com/700243323/93a685e3a6

Here the car rolls over when turning quickly, this is not an error in the code however the car is not supposed to behave like this. To counteract this, I developed a anti-roll script.

To fix this I this I experimented with the center of gravity, here the vector3 of the car sets the rigid body center of mass to the lowest point along the structure touching the car.

```
//testing centre of gravity
public Vector3 com;
public Rigidbody rb;

void Start()
{
    rb = GetComponent<Rigidbody>();
    rb.centerOfMass = com;
}
```

https://vimeo.com/700267935/e8fbfdd3fa

Above shows the failure of the lowering the centre of gravity.

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

public class AntiRollBar : MonoBehaviour
{

    public WheelCollider backWheelLeftCollider;
    public WheelCollider backWheelRightCollider;
    public float AntiRoll = 5000.0f;

    private Rigidbody Player;

    void Start()
    {
        Player = GetComponent<Rigidbody>();
    }

    void FixedUpdate()
    {
        WheelHit hit;
        float travelL = 1.0f;
        float travelR = 1.0f;


        bool groundedL = backWheelLeftCollider.GetGroundHit(out hit);
        if (groundedL)
        {
            travelL = (-backWheelLeftCollider.transform.InverseTransformPoint(hit.point).y - backWheelLeftCollider.radius) / backWheelLeftCollider.suspensionDistance;
        }

        bool groundedR = backWheelRightCollider.GetGroundHit(out hit);
        if (groundedR)
        {
            travelR = (-backWheelRightCollider.transform.InverseTransformPoint(hit.point).y - backWheelRightCollider.radius) / backWheelRightCollider.suspensionDistance;
        }

        float antiRollForce = (travelL - travelR) * AntiRoll;

        if (groundedL)
            Player.AddForceAtPosition(backWheelLeftCollider.transform.up * -antiRollForce, backWheelLeftCollider.transform.position);

        if (groundedR)
            Player.AddForceAtPosition(backWheelRightCollider.transform.up * antiRollForce, backWheelRightCollider.transform.position);
    }
}
```

The script above shows the code for the roll bar. First, we check whether the wheel colliders are grounded. I used the back left and front right wheel colliders.

```
//checking if grounded with bool
bool groundedL = backWheelLeftCollider.GetGroundHit(out hit);
if (groundedL)
{
    travelL = (-backWheelLeftCollider.transform.InverseTransfor
}
```

if the bool is true, I calculate the travel of the wheel collider from the ground.

```
{
    travelL = (-backWheelLeftCollider.transform.InverseTransformPoint(hit.point).y - backWheelLeftCollider.radius) / backWheelLeftCollider.suspensionDistance;
}
```

The amount of anti-roll required is then calculated by miniusing the difference between the left and the right collider.

```
float antiRollForce = (travelL - travelR) * AntiRoll;
```

If the grounded Left wheel is true, then the rigid body for the car is timsed by the collider, x – the roll force to ground the collider to ground again, the same applied to the right.

```
if (groundedL)
    Player.AddForceAtPosition(backWheelLeftCollider.transform.up * -antiRollForce, backWheelLeftCollider.transform.position);

if (groundedR)
    Player.AddForceAtPosition(backWheelRightCollider.transform.up * antiRollForce, backWheelRightCollider.transform.position);
```

https://vimeo.com/700271899/d868d21d00

Above shows the working anti-roll system under the same conditions, overcoming the rolling problem.

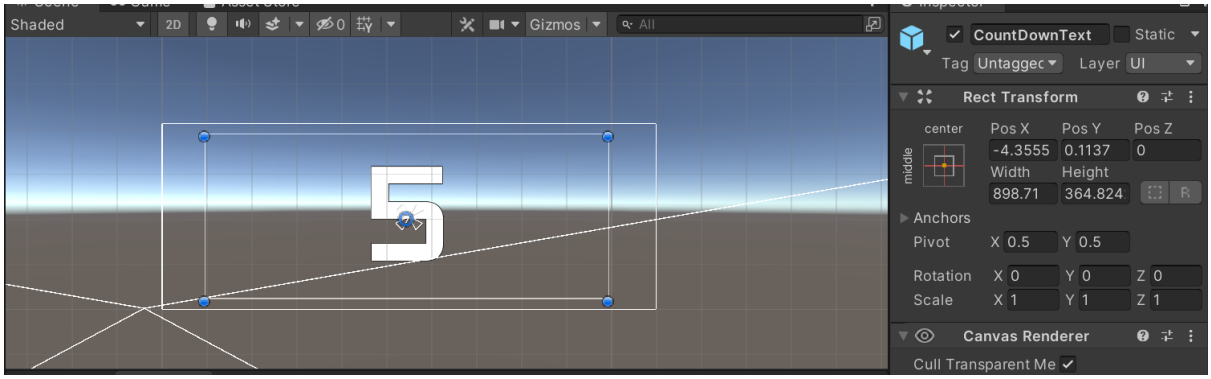
I James Gammon – 27228 – 8292 – Abbeyfield school

Count Down Menu

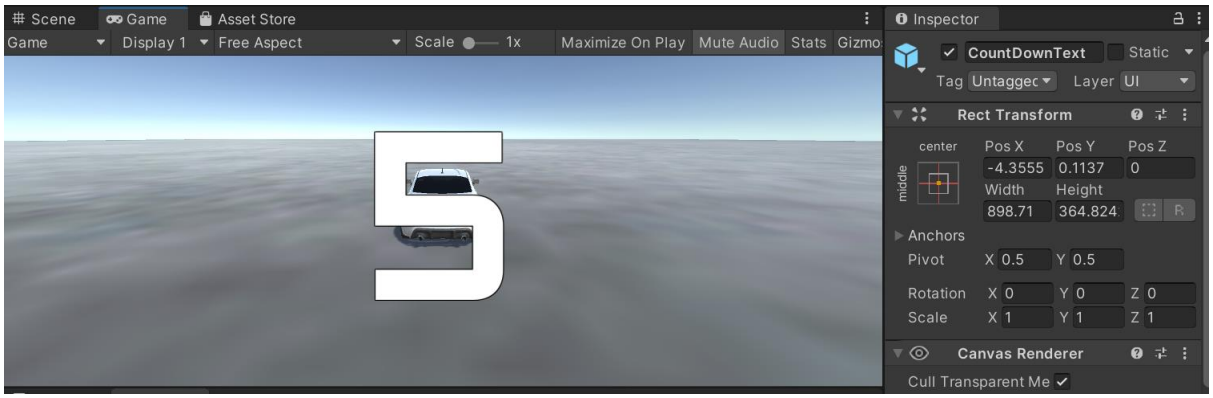
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;

public class CountDownController : MonoBehaviour
{
    public int countdownTime;
    public Text countdownDisplay;

    private void Start()
    {
        StartCoroutine(CountDownToStart());
    }
```

The class countdown controller defines two public game objects, the countdown timer which is the integer value displayed on the screen and the countdown display canvas.

On start (when the Scene the script is applied to is active)



On the game it is displayed as this:



The code initiating the timer

```
10
11      private void Start()
12      {
13          StartCoroutine(CountDownToStart());
14      }
15
```

Void start function

```
IEnumerator CountDownToStart()
{
    while(countdownTime > 0)
    {
        countdownDisplay.text=countdownTime.ToString();

        yield return new WaitForSeconds(1f);

        countdownTime--;

         if(countdownTime>=3.5f){countdownDisplay.color = Color.white;}
         if(countdownTime<3.5f){countdownDisplay.color = Color.red;}
    }

     countdownDisplay.text = "GO!";
```

On the countdown to start there are 5 processes;

- While the countdown timer is larger than 0; I am starting at 5
- The display text (defined as int) is converted to string to display on the screen
- After waiting and returning a value after 1 second.
- Countdown timer –- means to decrease the value by 1

CountDownText.Colour is a prebuilt function in C# which allows me to change the color of the text – for a better UI Experience.

Count down display function works: the timer turns to red when under the value of 3.5f (where f references to the frames passed per second)

Testing:

The Countdown text did not disappear when turning to the value two and disappearing after a second, I added this to the code:

```
26            if(countdownTime>=3.5f){countdownDisplay.color = Color.white;}
27            if(countdownTime<3.5f){countdownDisplay.color = Color.red;}
28        }
29
30        countdownDisplay.text = "GO!";
31        yield return new WaitForSeconds(1f);
32        countdownDisplay.gameObject.SetActive(false);
33
34    }
35}
36
```

By waiting 1f (one frame the countdown display is set to false and will not display on the screen)

Changes made after creating timer variable

After creating the timer variable, I created a duplicate of my current count down text and applied this to my tutorial maps. Changes made are:

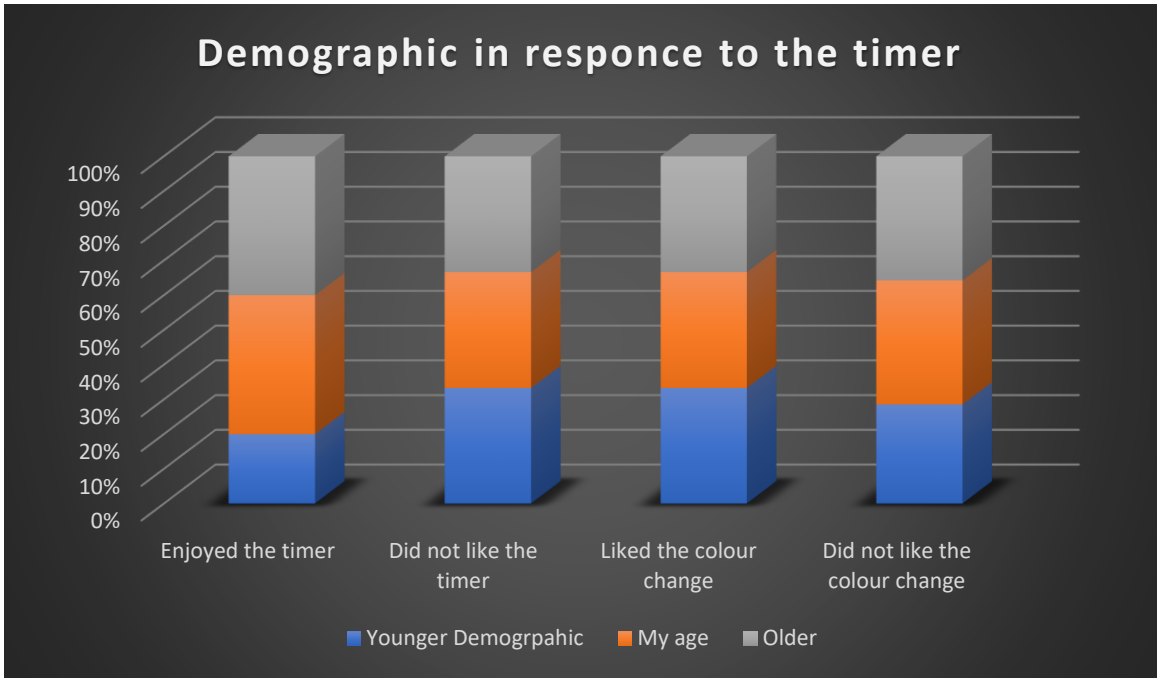
```
tutorialbuttons.gameObject.SetActive(true);
```

After applying my old countdown menu to my game tutorial maps, buttons to display the tutorial toggle's where still present, to improve I set them as false until the countdown timer ends which displays them as true, showing them on screen.

```
countdownDisplay.text = "GO!";
yield return new WaitForSeconds(1f);
tutorialbuttons.gameObject.SetActive(true);
countdownDisplay.gameObject.SetActive(false);
```

I James Gammon – 27228 – 8292 – Abbeyfield school

Presenting my Countdown Timer to my Demographic:



Username display

This code displays the user's username on the GUI menu

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

public class UserDisplay : MonoBehaviour
{
    public Text playerDisplay;

    private void Awake()
    {
        if (DBManager.username == null)
        {
            UnityEngine.SceneManagement.SceneManager.LoadMenu("MainMenu")
        }

        playerDisplay.GetComponent<Text>().text = "Player = " + DBManager.username;


        //DBManager score is always the best score from array bestTimes
    }
}
```

I am using the UnityEngine.UI; function here, I did with the timer as well, this assigns my code to any canvas, panel, or text box.

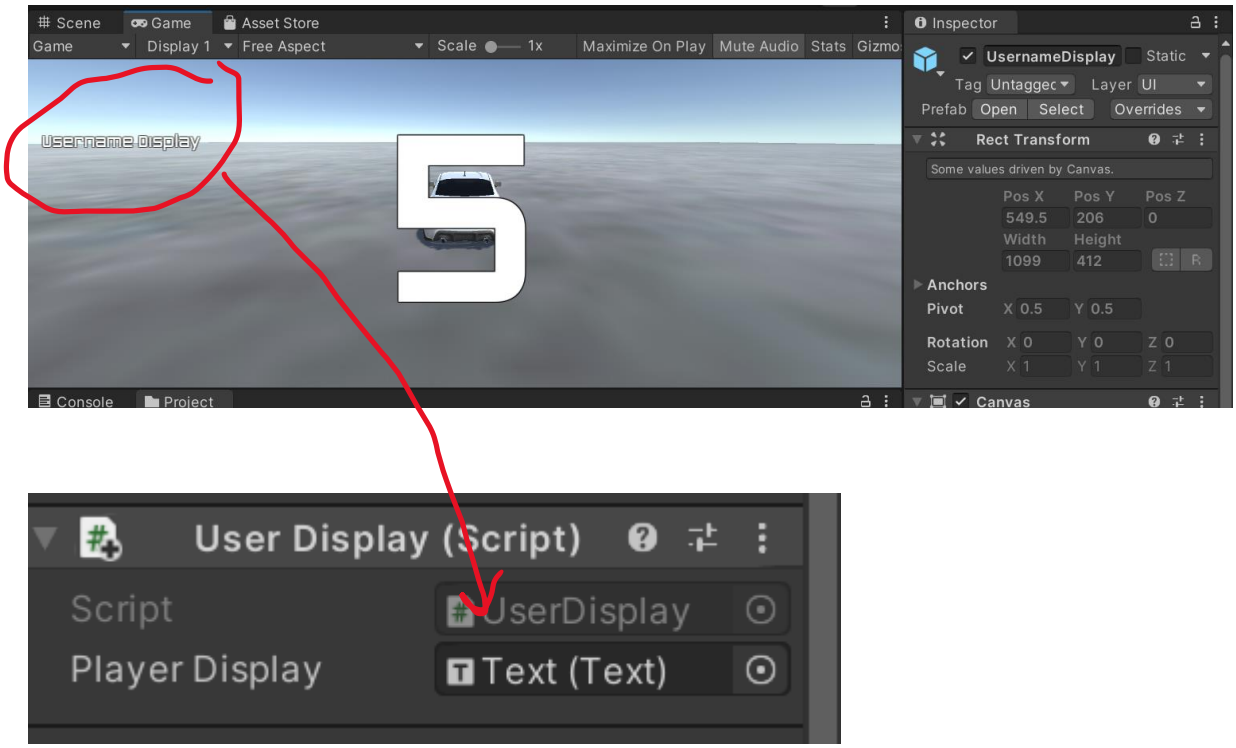
The public text player display is directly a game text box. In the if statement I am defining if the user has somehow gotten to this stage without logging in the username value would be null, they are returned directly to the Main Menu to be able to login.

The DBManager.username is a value assigned at the login/register functions.

Below is an example in the Login.cs.

```
yield return www;
if (www.text[0]== '0')
{
    DBManager.username = nameField.text;
```

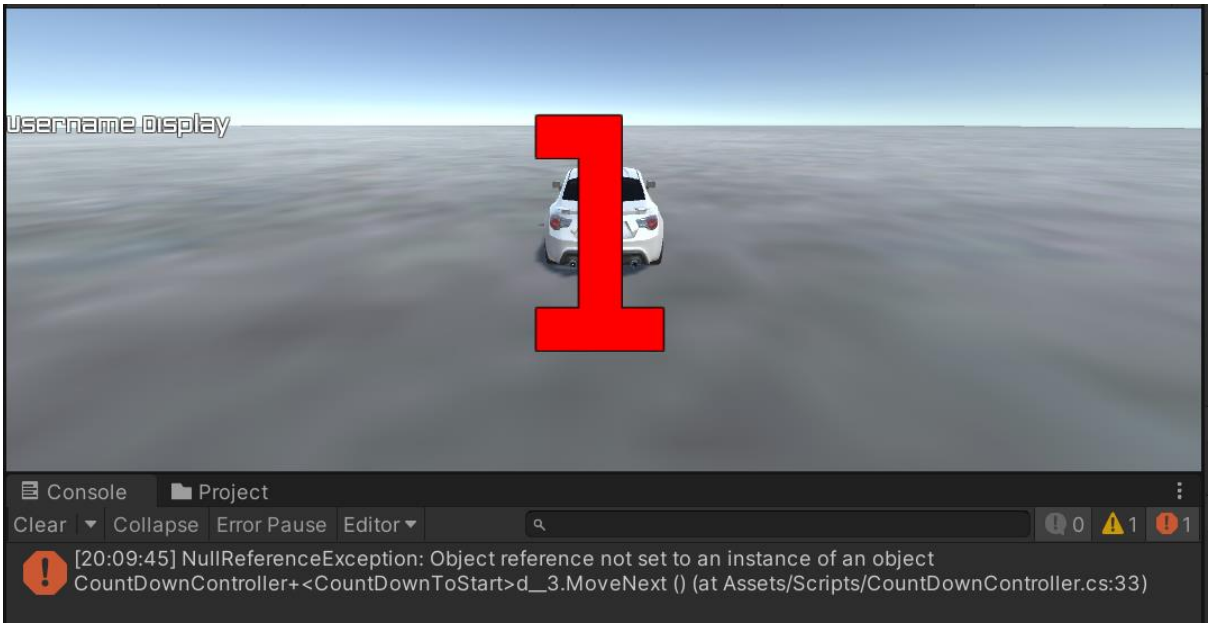
This is what the user display text looks like:



Assigned to the username display text is the UserDisplay.cs script which allows the text to change to the assigned username value.

I James Gammon – 27228 – 8292 – Abbeyfield school

Error:



Count down text stopped at value 1

https://i.gyazo.com/034e6778f376ef4ae0a3b70a0bab4fd6.mp4

Fix:

```
//starts the timer once the timer has reached GO!
 countdownDisplay.text = "GO!";
 yield return new WaitForSeconds(1f);
 countdownDisplay.gameObject.SetActive(false);
 ScoreTimeManager.instance.BeginTimer();
```

ScoreTimeManagr.instance.BeginTimer() was moved to the bottom of the text file as it was the last thing to execute

Stage 5 review

What has been done

In this stage I have added the fundamental car controller aspects which make up the ‘game’; these include the car controller, the username display and countdown menu. These are the components of the game which are transferred across each map choice that the user can pick. Now when the user picks a game mode and map, their respective username is displayed, the car is drivable, and a countdown text is produced.

How has it been tested

Each component has been tested differently and described in the stage. The countdown menu has been tested casually, on each game map I played casually to see if the countdown text was displayed and counted down to 1 printing GO! I cannot destructively test this piece of code as there are no user inputs.

The car controller has been tested through validating user inputs, I have forcefully pressed multiple buttons at once, to monitor the cars behavior in response to more than one input. I have tested each component individually, using both the WASD keys and arrow keys as input variables from the user, and validating their effect on the controller.

How it meets success criteria and user expectations

This section meets the users' expectations. A lot of my research and game development displayed count down texts and were synonymous across all driving games.

The success criteria in this match are creating the main game component, a drivable car which responds to user inputs.

Vehicle is completely controllable with arrow keys	Unity code which facilitates the movement of the car as well as the concurrent processing of multiple inputs at once.
The car game object can drift in and around corners	Code facilitating the key movement is screen shotted and displayed.
Car game object has wheel colliders meaning the car does not get stuck in the surrounding environment if it crashed	The code for the wheel game object which creates colliders around the wheels is screen shotted. As well as the hitboxes of the wheels

success criteria:

Changes in the design that have resulted from stage

There have been lots of changes in the design at this stage which I didn't plan for and developed on the go to improve the game. The first being the count down display at the start of each game, which I thought to include to raise the user's interaction with the game to give it a more professional feel.

The Second addition to the game is the userdisplay panel text which displays the players username inside of the game GUI, although I did not account for this in my planning and it was not a feature of other games, I felt it was a nice touch to add to the GUI to validate the logging in/registering process at the start of the game.

Summary of the project as a prototype at this stage

So far, my project is starting to enter the middle phase, each component which are applied to each of the 9 game maps has been tested and applied, though none of them are specialized yet they each have working car features which allow me to build on the functionality of the map.

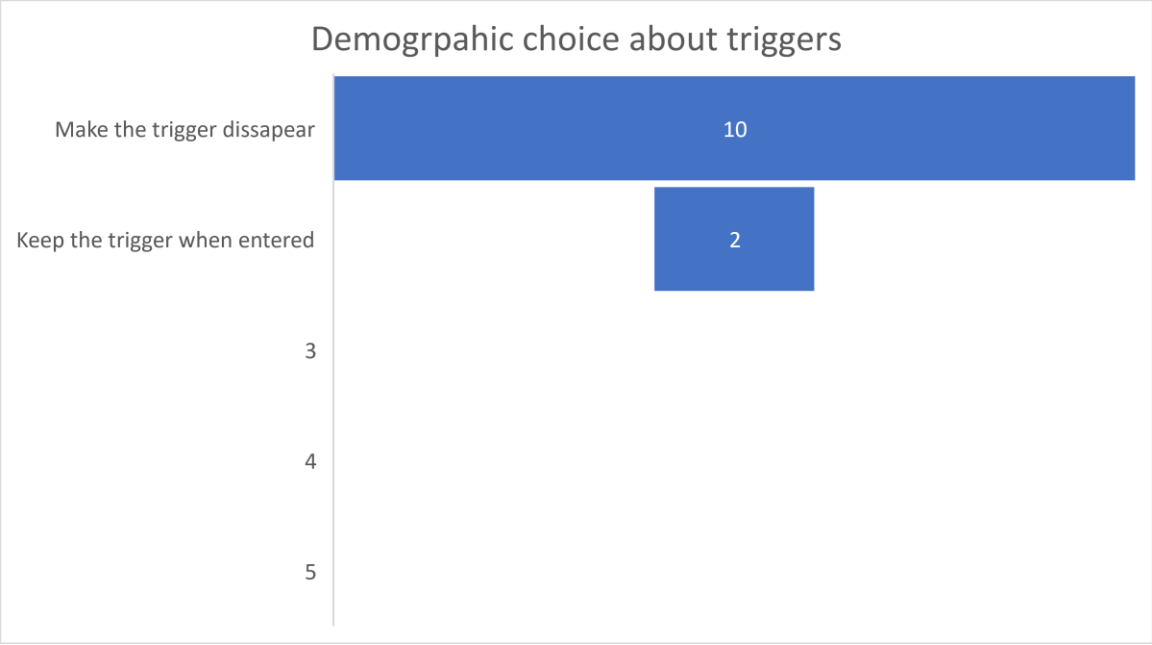
Next, I plan to incorporate specific features of the game and apply them to each of the maps for specialization of the racing game.

Stage 6 – The Game (Time Trial)

- Checkpoints
- DBManager
- Game
- Game.cs
- Map Half point triggers
- Score Menu
- Score Manager
- Score Time Manager Lap Complete

This is the display of the Hard Level Map, with a mixture of straits, turns, bridges and circles.

Checkpoints

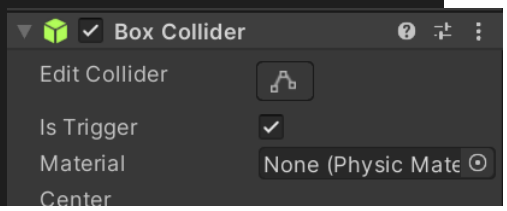


I James Gammon – 27228 – 8292 – Abbeyfield school

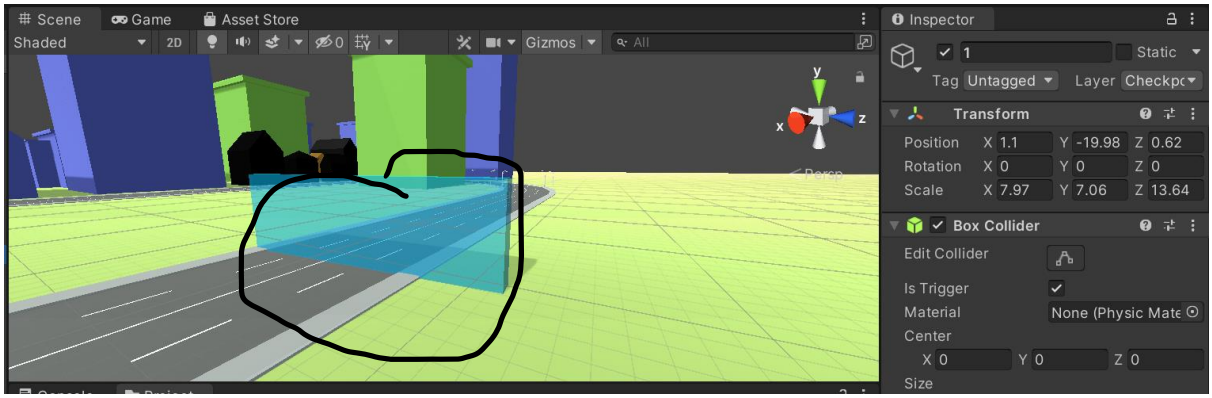
Stakeholder inputs:
Adrian, someone my age: “I’ve played racing games before… When the checkpoints disappear, I enjoy as it makes me feel I’ve completed something. This would be a really good idea to incorporate”

Code for checkpoint:

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

public class CheckpointsSystems : MonoBehaviour
{
    public GameObject CheckpointEnter;

    void OnTriggerEnter()
    {
        CheckpointEnter.SetActive(false);
        Debug.Log("Checkpoint Entered");
    }

}
```



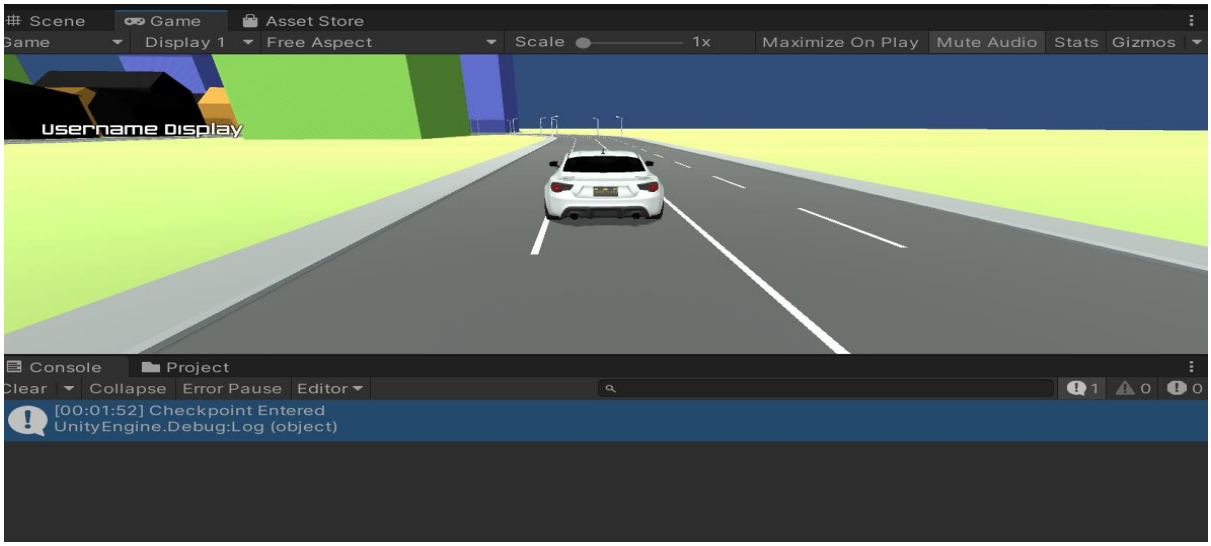
On enter as the game Object has been labeled a trigger; meaning that when entered the Checkpoint disappears to indicate to the user that the checkpoint has been entered as results from the query sent out.



This is an example of one of the checkpoints described in the tutorial game mode – the light blue color and size is universal around the gaming community making it much easier to identify as a checkpoint.

I James Gammon – 27228 – 8292 – Abbeyfield school

Checkpoint testing:



The image above displays the code performed; on entrance the trigger disappears and the Debug.Log statement is printed, if the checkpoint was not working the Debug.Log statement would not print.

DB Manager

DB manager was a script created at the start of the game, used to store the variables of the username, score of the user.

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

public static class DBManager {

    public static string username;
    public static int score;

    public static bool LoggedIn { get { return username != null;}}

    public static void LogOut() //username changed to null to log outt
    {
        username = null;
    }

}
```

The static description defines the current variable as global – meaning it can be referenced in multiple other scripts, this variable is used to store the Login and registration usernames and scores passed down, as well as the username display code.

example from login.cs script

```
    DBManager.username = nameField.text;
```

example from UserDisplay.cs

```
playerDisplay.GetComponent<Text>().text = "Player = " + DBManager.username;
```

example from RegistrationSQL.cs script

```
DBManager.username = nameField.text;
```

Ultimately these variables are going to be used to display the users' credentials in the leaderboard.

GameDB.cs

The game script is applied to the canvas of each of the racing games, Managing the score, it takes the score values and adds these to the final game menu, as well as printing the final score of the user and player and uploading this directly up into the database.

When developing my code, I came across an error, by applying my game script to each Time-trial map, the database would be receiving 3 sets of scores and sorting them representatively. This would be unfair as the shorter maps would have much shorter times compared to the longer hard maps; this mistake will be delt with later.

Functions of the code:

- Takes the score display and player display (score display is the fastest time and the player display is the username)
- The player Display variable text and score display variable text are substituted with the username and score of the player.
- The script starts a SavePlayerData Function
- The save Player Data script uploads the saved DBManager.username and DBManager.score passwords and uploads these to the database players.
- Debug.Log prints a statement Saying Game is saved if a 0 is printed from the php script, a user failed text is sent back to the user if any failure arises.
- Public void increases the score, sending the final score (min time-taken) and inputting this straight into the DBManager.score variable.

Takes the score display and player display (score display is the fastest time and the player display is the username)

```
public class game : MonoBehaviour
{
    public Text playerDisplay;
    public Text scoreDisplay;
```

This piece of code takes the text boxes on the user's screen and whatever outputs are presented on the screen, are presented in these two text boxes.

The player Display variable text and score display variable text are substituted with the username and score of the player.

```
private void Awake()
{
    if (DBManager.username == null)
    {
        UnityEngine.SceneManagement.SceneManager.LoadMenu("MainMenu")
    }

    playerDisplay.GetComponent<Text>().text = "Player = " + DBManager.username;
    scoreDisplay.GetComponent<Text>().text = "Score = " +  DBManager.scoreee;

    //DBManager score is always the best score from array bestTimes
}
```

I used this function to return the user back to the main menu if they are not logged into the game, and therefore cannot produce a username to associate the score with. UnityEngine.SceneManagement is a built-in function which changes the scene when called. To do this I created an if statement to say if DBManager.username ==null (meaning no registration or login, then the user is returned to the main menu).

When casually testing my game, I found a problem I was constantly being taken back to the Main menu when playing. I didn't want to have to login each time and so commented out this if statement to be able to test each map individually.

```
private void Awake()
{
    //if (DBManager.username == null)
    //{
       // SceneManager.LoadScene("MainMenu");
    //}
```

The script starts a SavePlayerData Function

```
public void CallSaveData()//starts the coroutine which saves player data
{
    StartCoroutine(SavePlayerData()); //uploads the username + score combernation to database
}
```

This function is a middleman function which calls the save data subroutine once the score is finished. I implemented this into a new script (instead of the ScoreLapTimeComplete.cs script as it is much easier to manage.

The save Player Data script uploads the saved DBManager.username and DBManager.score passwords and uploads these to the database players.

```
IEnumerator SavePlayerData()
{
    WWWForm form = new WWWForm();
    form.AddField("username", DBManager.username);
    form.AddField("score", DBManager.score);

    WWW www = new WWW("http://localhost/sqlconnect/savedata.php", form);
    yield return www;
```

Using the same methodology as login and registration this script also connects with the database. A new form is created but instead of passing new values of username and password it applies the current values to update the players scores and function just achieved. These are uploaded to the savedata.php script which will attain the username column from the players database, find the current username (logged in/ registered with) and apply the score achieved into the score value associated with this username.

Debug. Log prints a statement Saying Game is saved if a 0 is printed from the php script, a user failed text is sent back to the user if any failure arises.

```
if (www.text == "0")
{
    Debug.Log("Game is Saved");
}
else
{
    Debug.Log("Save Failed. Error no: " + www.text);
}

DBManager.LogOut();
{
    UnityEngine.SceneManagement.SceneManager.LoadScene("MainMenu");
}
```

These Debug.log functions are one of my testing methods to help me find errors, passing the value of 0 from the php file lets me know no errors have occurred, else an error message is printed with the type of error (used as an echo function in the savedata.php)

I James Gammon – 27228 – 8292 – Abbeyfield school

Testing for Game DB:

Unsuccessful game saved:

In this example, Debug. Log statement was not printed ; neither game saved, or game failed to save was printed.

Game.cs

To fix the miscommunication, I created another game script called Game.cs which would not upload scores to the database.

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

public class game : MonoBehaviour
{
    public Text playerDisplay;
    public Text scoreDisplay;


    private void Awake()
    {
        //if (DBManager.username == null)
        //{
            // UnityEngine.SceneManagement.SceneManager.LoadMenu("MainMenu")
        //}

        playerDisplay.GetComponent<Text>().text = "Player = " + DBManager.username;
        scoreDisplay.GetComponent<Text>().text = "Score = " +  DBManager.scoreee;

        //DBManager score is always the best score from array bestTimes
    }

    /*
     * public void CallSaveData()//starts the coroutine which saves player data
    {
        StartCoroutine(SavePlayerData()); //uploads the username + score combernation to database
    }
```

This script was copied and pasted from the GameDB.cs script and all the Database function were commented out with /* and */ functions.

On the awake value, the player. Display which displays the players username and score. Display is passed the variables DBManager.username and DBManager.scoreee to be displayed on the game

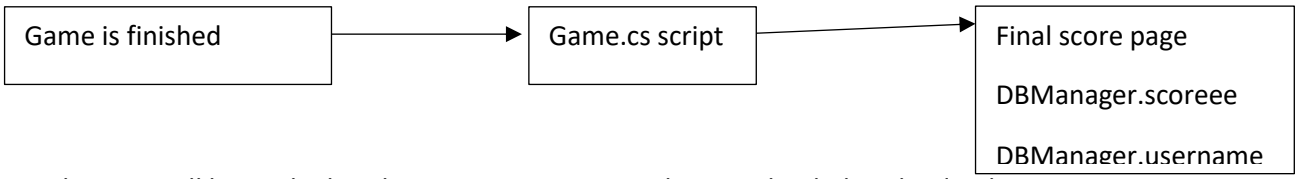
finished page, therefore there is no interaction with the actual values of the score. displayed on the game finished page, therefore there is no interaction with the actual values of the score.

```
//IEnumerator SavePlayerData()
{
    WWWForm form = new WWWForm();
    form.AddField("username", DBManager.username);
    form.AddField("score", DBManager.score);

    WWW www = new WWW("http://localhost/sqlconnect/savedata.php", form);
    yield return www;
    if (www.text == "0")
    {
        Debug.Log("Game is Saved");
    }
    else
    {
        Debug.Log("Save Failed. Error no: " + www.text);
    }

    DBManager.LogOut();
    {
        UnityEngine.SceneManagement.SceneManager.LoadScene("MainMenu");
    }
*/

public void IncreaseScore()
{
    scoreDisplay.GetComponent<Text>().text = "Score = " + DBManager.scoreee;
    playerDisplay.GetComponent<Text>().text = "Player = " + DBManager.username;
```

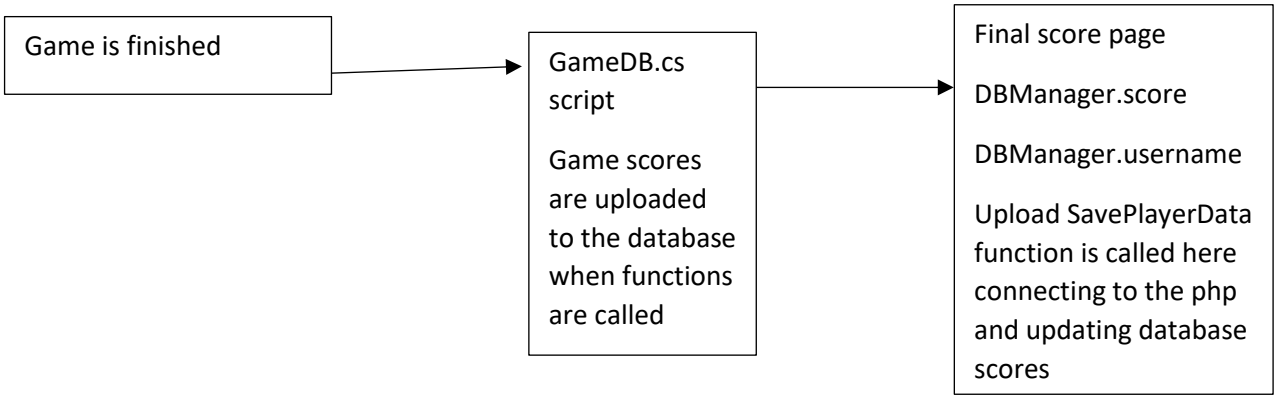
One change I made with this script is changing the DBManager.score global variable to DBManager.scoreee. By doing this the scores achieved on the shorter maps where still being passed as a value to the final game score page, however where not being passed to the database like the DBManager.score time function is.

The function Increase Score(will be applied to the increase score button once the game has finished, to input the users final score into the GameOver screen)

This step will be applied to the maps scores which are not being uploaded to the database



This step will be applied to the maps scores are not being uploaded to the database.



These non-database scripts will be applied to the EZ. Time Trial maps and the DT. Time Trial Maps ; the hardest level will be uploaded to the hardest time-trial maps.

Half point triggers

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class HalfPointTrigger1 : MonoBehaviour {
6
7       public GameObject LapCompleteTrig;
8       public GameObject HalfLapTrig;
9
10      void OnTriggerEnter (){
11          LapCompleteTrig.SetActive(true);
12          HalfLapTrig.SetActive(false);
13          Debug.Log("Halfway point entered");
14      }
15  }
16
```

Half point triggers are used for time-trial maps to indicate a halfway point and a final point in the code. The half point trigger and lap complete trigger will both be applied to the TimeTrial map.

As the game starts:

- The half point trigger is set to true (can be seen)
- Lap time manager is set to false (cannot be seen)
- The code above functions to set the HalfPointrigger to false once its passed through (OnTriggerEnter)
- The Lap time manager will have the Lap Finished Code – which will Debug.Log what has happened once the user has completed a lap.

It will activate this algorithm from design:

> Another algorithm I need to complete is adding the car time into a list when completed, the list is sorted, best time displayed
>
> User completes a lap > elapsed time added to a list > list is sorted on every lap complete after new variable added > if list has more than 1 elements > print the list > display best score on screen

Time Manager

This is the first-time manager of my script; it will use an iterative process to increase the score and display this text in a second part code. This script is solely for increasing the time.

The first part of this script defines the text boxes and triggers being used. The triggers are defined as public game objects, they do not have to be defined from the trigger script as the trigger game objects are being applied to this script as an object and the code for them is applied to each trigger.

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

public class LapTimeManager : MonoBehaviour {

    public static int MinuteCount;
    public static int SecondCount;
    public static float MilliCount;
    public static string MilliDisplay;

    public GameObject MinuteBox;
    public GameObject SecondBox;
    public GameObject MilliBox;
```

Game Objects Minute Display, Second Display and Mill Display are 3 separate game objects which will display the second timer, minute timer and milli second timer.

The code demonstrates three different processes:

Incrementing the milli second > once it gets to 9 the second count is increased and milli count is relayed back to 0

Incrementing the second count > once it gets to 9 the minute count is increased, and second count is relayed back to 0

Incrementing the min count > once it gets to double digits the minute count is incremented with a 0 in front of it.

```
17
18      void Update() { //counts the timer for the box
19          MilliCount += Time.deltaTime * 10;
20          MilliDisplay = MilliCount.ToString("F0");
21          MilliBox.GetComponent<Text>().text = ""+MilliDisplay;
22
23          if (MilliCount >= 10){
24              MilliCount = 0;
25              SecondCount += 1;
26          }
27
28          if(SecondCount <= 9){
29              SecondBox.GetComponent<Text>().text = "0" + SecondCount + ".";
30          } else {
31              SecondBox.GetComponent<Text>().text = "" + SecondCount + ".";
32          }
33
34          if(SecondCount >= 60) {
35              SecondCount = 0;
36              MinuteCount += 1;
37          }
38
39          if(MinuteCount <= 9){
40              MinuteBox.GetComponent<Text> ().text = "0" + MinuteCount + ":";
41          } else {
42              MinuteBox.GetComponent<Text> ().text = "" + MinuteCount + ":";
43          }
44
45      }
```

The milli count uses a Time.DeltaTime function built into unity which multiples the value by time Passed inside of the game. Second count is then increased.

```
if (MilliCount >= 10){
    MilliCount = 0;
    SecondCount += 1;
}
```

The second count here is formatted, if a single digit it is printed with a 0, else the same process as the milli second happens, however this increments the minutes.

```
if(SecondCount <= 9){
    SecondBox.GetComponent<Text>().text = "0" + SecondCount + ".";
} else {
    SecondBox.GetComponent<Text>().text = "" + SecondCount + ".";
}

if(SecondCount >= 60) {
    SecondCount = 0;
    MinuteCount += 1;
}
```

Again, if the minute count is a single digit its formatted with a 0 on the end, else its formatted with no 0 to show the double number.

```
    if(MinuteCount <= 9){
        MinuteBox.GetComponent<Text> ().text = "0" + MinuteCount + ":";
    } else {
        MinuteBox.GetComponent<Text> ().text = "" + MinuteCount + ":";
    }

}
```

Time Managers lap complete

Time manager lap complete script takes the game Objects defined in the Lap Time script and displays them on screen.

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

public class LapComplete : MonoBehaviour {

    public GameObject LapCompleteTrig;
    public GameObject HalfLapTrig;

    public GameObject MinuteDisplay;
    public GameObject SecondDisplay;
    public GameObject MilliDisplay;


    public GameObject LapCounter;
    public int LapsDone;
```

This time manager laps complete script is called when the LapCompleteTrigger is activated. It summons the static variables from the Timer Script, the displays, and new objects called Lap Counter and Lap Complete to show the user the number of laps completed to show them on screen.

The laptime completed script enables the formatted times of minute display and second display to

```
    void OnTriggerEnter () {
        LapsDone+=1;

        if (LapTimeManager.SecondCount <= 9) {
            SecondDisplay.GetComponent<Text> ().text = "0" + LapTimeManager.SecondCount + ".";
        } else {
            SecondDisplay.GetComponent<Text> ().text = "" + LapTimeManager.SecondCount + ".";
        }

        if (LapTimeManager.MinuteCount <= 9) {
            MinuteDisplay.GetComponent<Text> ().text = "0" + LapTimeManager.MinuteCount + ".";
        } else {
            MinuteDisplay.GetComponent<Text> ().text = "" + LapTimeManager.MinuteCount + ".";
        }

        MilliDisplay.GetComponent<Text> ().text ="" + LapTimeManager.MilliCount;
```

have the appeared times of mm:ss:ff on the user's screen, inputting ':' and '.' For it to look like this.

At the end of the script the success criteria stated to restart the timer after every lap to display 00:00.00 after completion to start again. The variables LapTimeManager set the original values to 0 for them to proceed to count again. The half lap trigger and LapCompleteTrigger are inversed, so the half lap trigger is true and LapTimeTrigger is false.

```
36
37
38          //uploads laps to Laps done
39          LapCounter.GetComponent<Text>().text = "" + LapsDone
40
41
42          //setshalflaptrig
43          HalfLapTrig.SetActive(true);
44
45          //sets lap complete trig
46          LapCompleteTrig.SetActive(false);
47
48          //changes timer back to 0
49          LapTimeManager.MinuteCount = 0;
50          LapTimeManager.SecondCount = 0;
51          LapTimeManager.MilliCount = 0;
52
53      }
54  }
```

On completion of the scores, adding them to a database as an integer and reformatting them again was a big problem, I found that coding each score as an individual component was difficult to take the entire value and manipulate this was difficult. The values were formatted as mm: ss. ff with changing string values '0' and ':' and '.'; to overcome this I reworked my scoring system completely, parsing the score as an integer value, using it as a float to sort the best times, and to display this on the board, where I will pass this value to the database and return it back to an integer.

I created new script files called ScoreTimeManager and ScoreTimeManagerLapComplete; with these I retained the functions of the two previous scripts in terms of the formatting and the trigger points, however the score value was changed completely.

Score Time Manager

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

public class ScoreTimeManager : MonoBehaviour
//public class TimeManager : MonoBehaviour
{

    public static ScoreTimeManager instance;
    //public static TimeManager instance;

    public Text timeCounter;

    public TimeSpan timePlaying;

    public static bool timerGoing;

    public static float elapsedTime;

    public int score;

    private void Awake()
    {
        instance = this; //allows to call functions from outside of the class
    }

```

To create this new script I added new variables, I created a Boolean variable to control the timer display a built-in float called elapsedTime and built in TimeSpan variable measuring time called TimePlaying.

The TimeSpan variable would aid me as I am able to format the time taken as mm:ss:ff keeping the actual value as an integer.

Timer Going allows me to access the timer going from the ScoreManagerLapComplete script and public Text is what is displayed on the users GUI.

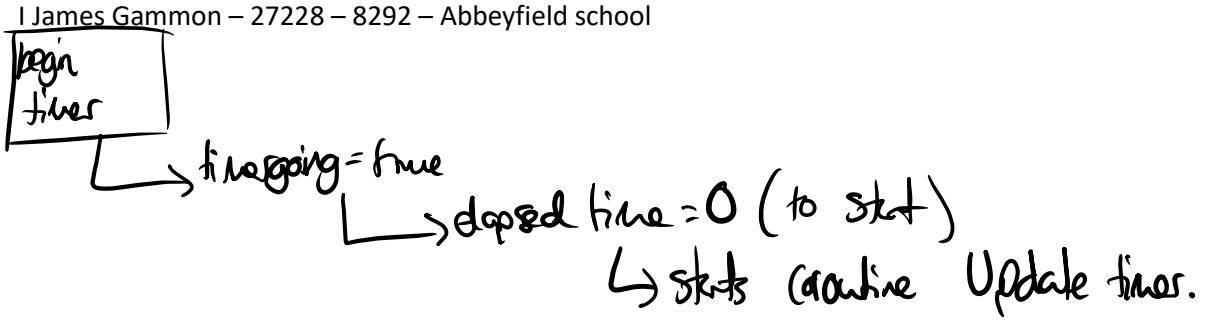
Private void awake () is a built-in function which activates the script once the game is called which allows me to call each function from different scripts, not just the variables.

On the script the start() time as 0:00.00 once the game starts, not until the game starts is the timer

```
// Start is called before the first frame update
public void Start()
{
    timeCounter.text = "Time: 00:00.00";
    timerGoing = false;
    Debug.Log("Timer Reset");


}

public void BeginTimer()
{
    timerGoing = true;
    elapsedTime = 0f;

    StartCoroutine(UpdateTimer());
    Debug.Log("Timer began");
}
```

changed to true.

Begin timer is a static void which allows me to start the game timer at any point by any other script




```
52      private IEnumerator UpdateTimer()//counts the timer for the box
53      {
54
55          while (timerGoing)
56          {
57              elapsedTime += Time.deltaTime;
58              timePlaying = TimeSpan.FromSeconds(elapsedTime);
59              string currenttimePlayingStr = "Time: " + timePlaying.ToString("mm':'ss'.'ff");
60              timeCounter.text = currenttimePlayingStr;
61              yield return null;
62              //return to this point on the next frame checks the while loop again
63          }
64      }
65  }
```

The process uses a while loop to constantly update the timer, ElapsedTime Variable is added to the Time.DeltaTime variable (built in function to unity) to create the total time playing.

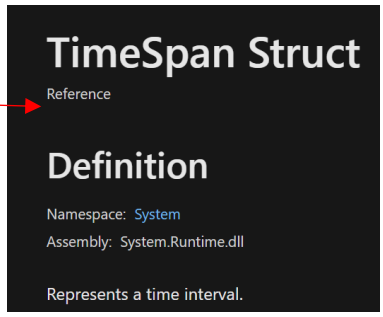
The timePlaying variables which takes the time interval between the 0 function in void start and the

deltaTime	The interval in seconds from the last frame to the current one (Read Only).

elapsedTime function (included with Time.DeltaTime) and creating the variable which is displayed on the screen.

Displaying the timePlaying variable as mm:ss:ff I formatted it to be a string, using a function ToString passing a parameter (the style I desired to display)

```
16      public TimeSpan timePlaying;
```



I applied this to the time Counter which is displayed on the GUI.

Score Time Manager Laps Complete

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

public class ScoreTimeManagerLapComplete : MonoBehaviour
{

    //publicDisplay
    public Text BestTimeCounter;

    //sorting bestTime
    public static TimeSpan mintimePlayed;
    public static List<float> bestTimes = new List<float>();

    //laptriggers
    public GameObject LapCompleteTrig;
    public GameObject HalfLapTrig;

    //laps counter
    public GameObject LapCounter;
    public static int LapsDone;

    //defined for the pauseMenu
    public GameObject IncreaseScore;
    public GameObject UserNamePanelUI;
    public GameObject TimertextUI;
    public GameObject LapLabelUI;
    public GameObject BestTimePanel;

```

Like the score time script all my variables are defined, including my triggers, lap counter display and integer, and the total GUI for the score menu pre-defined for my increase Score panel.

I have created a static float list best times which will be used to sort all the times and print the best one.

```
    void OnTriggerEnter()
    {
        //if the laps completed
        if (LapsDone == 1)
        {
            IncreaseScore.SetActive(true);
            TimertextUI.SetActive(false);
            UserNamePanelUI.SetActive(false);
            ScoreTimeManager.timerGoing = false;
            LapLabelUI.SetActive(false);
            BestTimePanel.SetActive(false);
            Debug.Log("Laps Finished");
        }
        //entering the trigger
        Debug.Log("LapCompleteTrigger Entered");
        //increasing the amount of laps
        LapsDone += 1;

```

On the final Lap trigger Enter: (I have used a value of 1 for quick testing) which disables the GUI for the pause Menu.

If the lap count does not = 1 (3 when the game is released) then the LapsDone Intiger is incramented by a value of 1.

```
        //adding time to the array
        if (ScoreTimeManager.elapsedTime != 0)
        {
            bestTimes.Add(ScoreTimeManager.elapsedTime);
            Debug.Log("Time added to array");
        }
```

This piece of script carries out the same function as the previous score script on.

However now I can apply the intiger value into the bestTimes list to be sorted and printed as a whole intiger instead of 3 different values.

```
47
48          //changes timer back to 0
49          LapTimeManager.MinuteCount = 0;
50          LapTimeManager.SecondCount = 0;
51          LapTimeManager.MilliCount = 0;
52
```

```
58
59          //upload laps to laps done
60          if (bestTimes.Count > 0)
61          {
62              bestTimes.Sort();
63
64              mintimePlayed = TimeSpan.FromSeconds(bestTimes[0]);
65              DBManager.scoreee = "" + mintimePlayed.ToString("mm':'ss'.'ff");
66              BestTimeCounter.text = DBManager.scoreee;
67
68              Debug.Log("The fastest score was " + DBManager.scoreee + "Seconds... Well Done");
69
70          }
71
72
73          //set timer back to 0
74          ScoreTimeManager.elapsedTime = 0;
75
76          //set half lap trigger
77          HalfLapTrig.SetActive(true);
78          // set lap complete trigger
79          LapCompleteTrig.SetActive(false);
80          LapCounter.GetComponent<Text>().text = "" + LapsDone;
81      }
82  }
```

I then use an if statement applied to the List to check if the number of variables >0; if so, the list is sorted using the prebuilt function.

Like the ScoreTimeManager, the minTimePlayed is the minimum time played, sorted from array of the besttimes, this is formatted and applied to the variable used for the GameMaps which will not be applied to the Database: DBManager.scoreee.

The next part of the script carries out the same function as the previous score script; to reset the timer and start again.

Previous script functions

```
47
48          //changes timer back to 0
49          LapTimeManager.MinuteCount = 0;
50          LapTimeManager.SecondCount = 0;
51          LapTimeManager.MilliCount = 0;
52
```

The LapTriggers are then set Active for the next Lap and the Lap counter text is incramented onto the screen.

Score Time Manager Laps Complete DB

For uploading my game score to the database, a separate Score Time Manager Lap Complete DB script has been created. It carries out the exact same functions as the other script, however, the function in the previous script applying the score to DBManager.scoreee has changed to DBManager.score.

```
//upload laps to laps done
//printing the best time
if (bestTimes.Count > 0)
{
    bestTimes.Sort();
    mintimePlayed = TimeSpan.FromSeconds(bestTimes[0]);
    string timePlayedStr = "" + mintimePlayed.ToString("mm':'ss'.'ff");
    BestTimeCounter.text = timePlayedStr;
    DBManager.score = int.Parse(ToString("mm':'ss'.'ff"))
    //printing the DBManager.score value in the Debug.Log
    Debug.Log(DBManager.score);


    //assigning the best player score to DBManager to be uploaded
    Debug.Log("The fastest score was " + DBManager.score + "Seconds... Well Done");

}
```

However, this error occurred:

```
Assets\Scripts\ScoreTimeManagerLapCompleteDB.cs(69,32): error CS1501: No overload for
method 'ToString' takes 1 arguments
```

```
[18:31:37] Assets\Scripts\ScoreTimeManagerLapCompleteDB.cs(69,32): error CS1501: No ove
```

Stating that the ToString only takes one argument. This is because I have now added the format int. Parse and removed the component minTimePlayed. Now the ToString component is taking itself as a parameter and the component text while the mm:ss.ff is not converting to an integer because of the ':' and '.'.

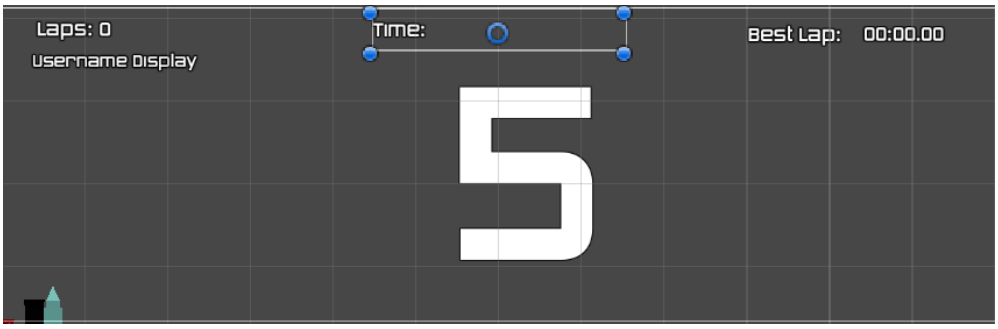
To fix this I added the minTimePlayed variable back as an integer; formatting this to a straight string as mmssff and converting this as an integer to:

```
if (bestTimes.Count > 0)
{
    bestTimes.Sort();
    mintimePlayed = TimeSpan.FromSeconds(bestTimes[0]);
    string timePlayedStr = "" + mintimePlayed.ToString("mm':'ss'.'ff");
    BestTimeCounter.text = timePlayedStr;
    DBManager.score = int.Parse(mintimePlayed.ToString("mmssff"));
    //printing the DBManager.score value in the Debug.Log
    Debug.Log(DBManager.score);


    //assigning the best player score to DBManager to be uploaded
    Debug.Log("The fastest score was " + DBManager.score + "Seconds... Well Done");

}
```

This is the code that has been changed; the time is still formatted as a string, however the line of code changed:

```
DBManager.score = int.Parse(mintimePlayed.ToString("mmssff"));
```

Changes the on-screen display text to an integer, concatenating the values mm:ss.ff to mmssff which also takes into account the milliseconds.



Here is the timer text, lap, best Lap, and username display panel

Here we can see an error:

The timer has begun while still counting down the lap Time Manager.

To fix this I have added a ScoreTimeManager instance which begins the timer function once the Timer reaches go.

```
32      //starts the timer once the timer has reached GO!
33       countdownDisplay.text = "GO!";
34       yield return new WaitForSeconds(1f);
35       countdownDisplay.gameObject.SetActive(false);
36       ScoreTimeManager.instance.BeginTimer();
37   }
```

Stage 6 Review

What has been completed:

This stage has added the specialty of the game maps. For Time-trials; the game now has an efficient scoring system; displaying the best time on the screen which is updated after each of the laps. I have created an effective checkpoint system to guide the users on the tutorial maps.

How has it been tested

Checkpoints – Checkpoints have been tested by driving the car through the checkpoint system making sure it disappears on entry.

DBManager – This has been tested by screen shotting the users display name in the Database players.

Game – Has been tested by through the Debug.Log; printing the values passed to the database to make sure they are okay.

Laps Done – The lap's done variable was tested at a lower number, once the predefined number of laps has been completed there was a new aspect of a final page score display which appeared when the number of laps was completed.

Half Point triggers – Half point triggers have been tested through Debug.Log statements, using print statements once the trigger is activated and displaying this to the user.

Time Manager – Time manager has been tested on the user screen through casual testing, the users current time has been tested to upload to the TimeManager.text, this has been tested by waiting for 10s, 1min and 60s to make sure that the time is formatting correctly.

Upload to the database – Again, Debug.Log statements have been applied to the variables DBManager.score and DBManager.username.

Testing checklist:

Racing game mode starts the timer once the game is started and is reset once the user enters the finish lap checkpoint.	Yes
Each lap time is input into an array of times where the best time is produced onto the screen.	Yes

How it meets success criteria and user expectations

The game

Name	Data Type	How its used
Timer going	Boolean	Checks to see if timer Is running or not
Elapsed time	Integer	Counts the current time playing in the game
CurrentTimePlaying	String	Prints the fastest lap time on the users screen
Half Lap Trigger	Boolean	Assigns the lap complete Game Object to true once this game object is entered (changed to false)_
Complete Lap Trigger	Boolean	Assigns the half lap point to true and this game object to false when entered

From the research from other games there are a clear set of standard attributes and aspects which include a visible timer, a best lap timer, a lap complete indication and a username display. Usually, the player expects these in any standard game and are what I've included in mine. Shown on the GUI there are clear timers are clear for the user to experience.

Changes in the design that have resulted from stage

Changed the timer to start after the countdown has begun

Added a final score page

Initially I did not think far enough ahead about starting my timer after my countdown because the countdown was a new aspect I assigned to my game in a previous stage. On implication the timer and countdown menu began at the same time which was not right, I changed this by setting the timer to begin once the integer of the countdown timer reached 0, leading from one to another.

Summary of the project as a prototype at this stage

The prototype is growing so far, the initial game development has created a root that these different game aspects can grow out of. The prototype has a modular development application which means that it is interchangeable and can be applied to the 3 different game maps. The prototype is almost ready for stakeholder testing specifically on time-trials. Compared to other games, my game is mimicking the fundamentals of these which highlights the complexity of the growth and allows a modular development in the future.

Stage 7 – The Game tutorial

The game tutorial will be a transferrable component of tutorials on the screen, there will be 3 aspects, a game tutorial (how to play), a map tutorial (using the map) and an options menu (using the game) and then activating and deactivating checkpoints progressing around each map.

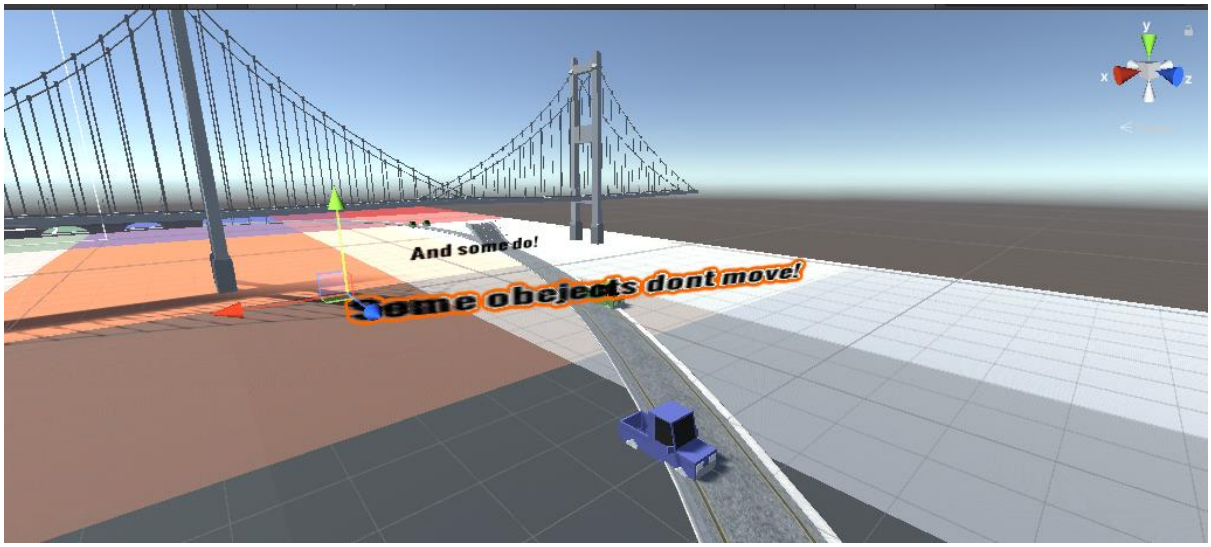
I will create the tutorial as a prefab in unity (which is a transferrable asset) and apply this to each of the maps. There are 3 different steps:

Creating the tutorial panel

Creating the checkpoints

Apply the checkpoints and tutorial to each of the different maps

This will conclude the ‘tutorial aspect of the game’



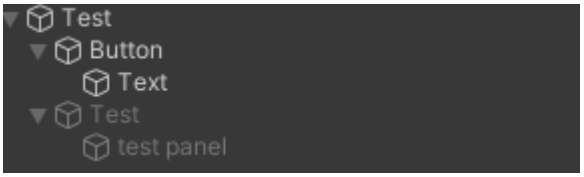
The image above is a screen shot of 3D text implanted into the easy game mode, this will be reflected throughout each of the three game maps. I created a 3D text object supported by unity and placed these systematically around the map.

Video below:

https://vimeo.com/693181292/d4f26953dd

Creating the tutorial panel:

To create the tutorial panel, I wanted to create an opaque text panel which would display text boxes on the user’s screen.



By creating these game objects: The Test Canvas to hold the button and panel. The button is the input from the user which calls the function to set the panel.

The ‘Test’ in grey is the panel which holds the test text, displaying on the screen.

The panel is set as false

The button is clicked

Calls a subroutine

Displays text

Button is clicked again to make this disappear.

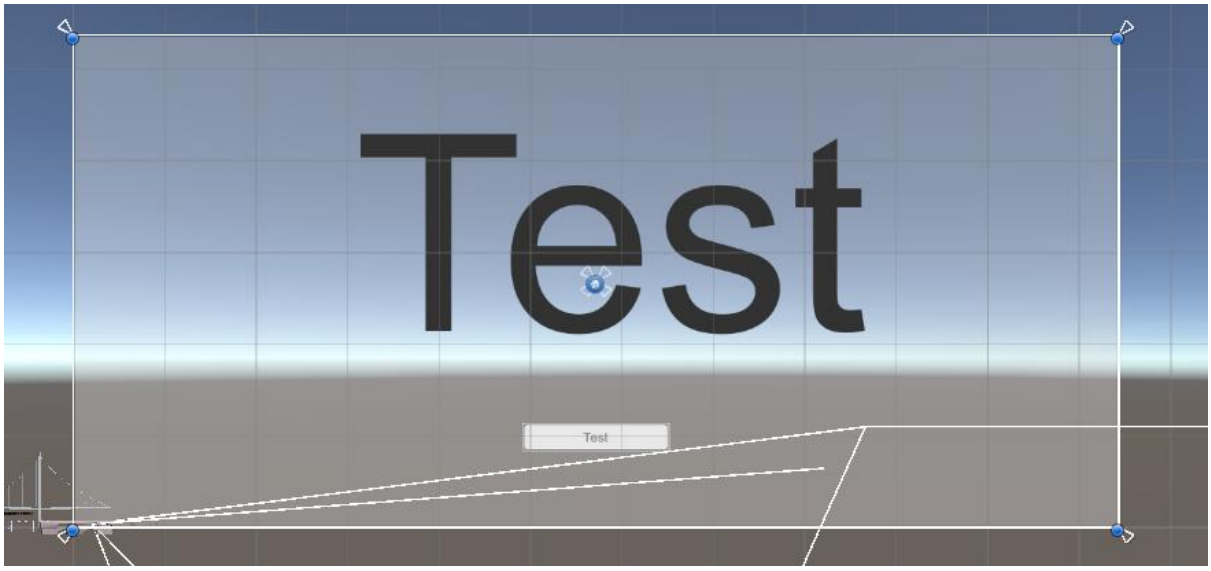
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class testscript : MonoBehaviour
{
    // holds the panel with text
    public GameObject TutorialPanel;


    // function to open the panel
    public void OpenTurorialPanel()
    {

        // checks if the panel is active
        if (TutorialPanel != null)
        {
            //activates the panel
            bool isActive = TutorialPanel.activeSelf;
            TutorialPanel.SetActive(!isActive);
            Debug.Log("Opening Tutorial Panel...");
        }
    }
}
```

Above is the prototype used in the code, activating, and deactivating the panel script. The sub-function ‘isActive’ is built into unity which checks if the game object is currently present or not.

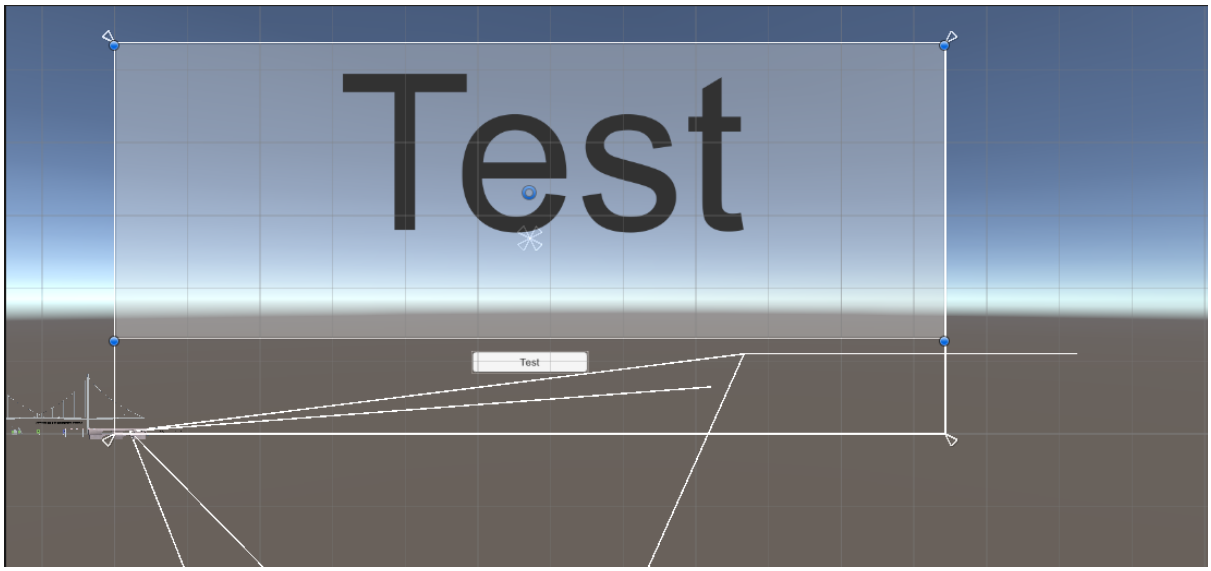
The function will be tested below:

https://vimeo.com/693194629/4a405faed0

Error: The code worked to open the script but did not close the script once the button had been pressed again, therefore constantly displaying the panel.



To fix this I had to change the panel from covering the button, to being just above the button, as being over the button prevented the button from being registered as clicked.



This is now a test with this change:

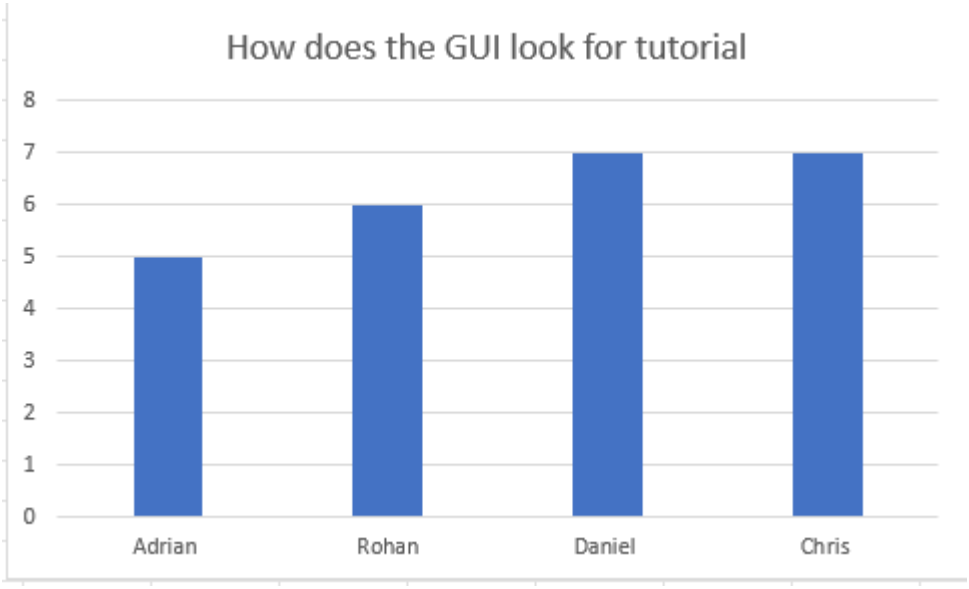
https://vimeo.com/693201163/bed7509ae7

above is a working test in which the testing button successfully actives and deactivates the testing panel which contains text.

As this successfully works, I can use this prototype to create an actual working menu, with colored tutorial text and buttons.

Above is the changed GUI buttons with some small text at the bottom to indicate what to do, now I will duplicate and change the earlier code to produce the working text.

I James Gammon – 27228 – 8292 – Abbeyfield school



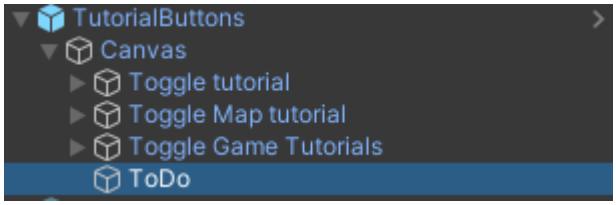
I am happy with the feedback from the questionnaire, both Adrian and Rohan spoke about the inability to see the black text.

Adrian: “The buttons are a really cool idea; however, I was a little confused on what to do and the black text did not help as it didn’t stand out and I couldn’t see it; this would be better if it was a brighter color ”

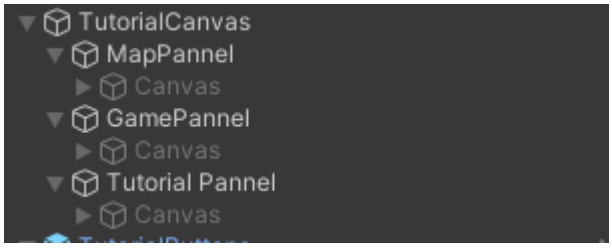
I changed the GUI to the above color which improved Adrian’s feedback

Adrian “The color looks really good now, I can clearly see what to do”

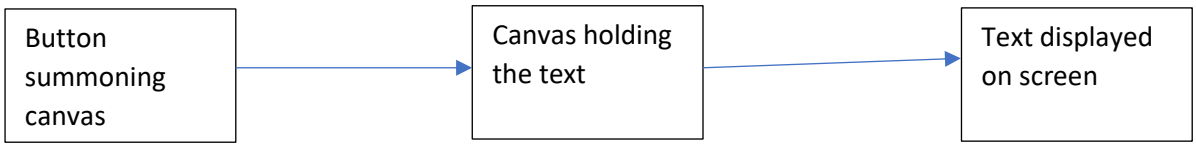
Next, I developed the script for the multiple button's panels.



The canvas holds the buttons for each of the summoning texts, the ToDo object is the text at the bottom of the screen.



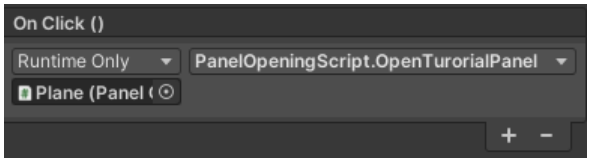
Here are the panels which each of the buttons will be summoning to display the text,



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

public class PanelOpeningScript : MonoBehaviour
{
    public GameObject TutorialPanel;
    public GameObject MapPanel;
    public GameObject GamePanel;

    public void OpenTurorialPanel()
    {
        if (TutorialPanel != null)
        {
            bool isActive = TutorialPanel.activeSelf;
            TutorialPanel.SetActive(!isActive);
            Debug.Log("Opening Tutorial Panel...");
        }
    }

    public void OpenGamePanel()
    {
        if (GamePanel != null)
        {
            bool isActive = GamePanel.activeSelf;
            GamePanel.SetActive(!isActive);
            Debug.Log("Opening Game Panel...");
        }
    }


    public void OpenMapPanel()
    {
        if (MapPanel != null)
        {
            bool isActive = MapPanel.activeSelf;
            MapPanel.SetActive(!isActive);
            Debug.Log("Opening Map Panel...");
        }
    }
}
```

Here is the script that activates and deactivates the panels, each panel is assigned by a game object. Using the logic of the prototype code then it's checked to see if its already active, if it is then it is deactivated or else it is activated.

Each subroutine mimics the same logic but is self-similar to the others being specific to the text/ panel it is wanting to open.



Each button is assigned the script which allows them to call the subroutine.

This is what the text looks like after being called by the button.

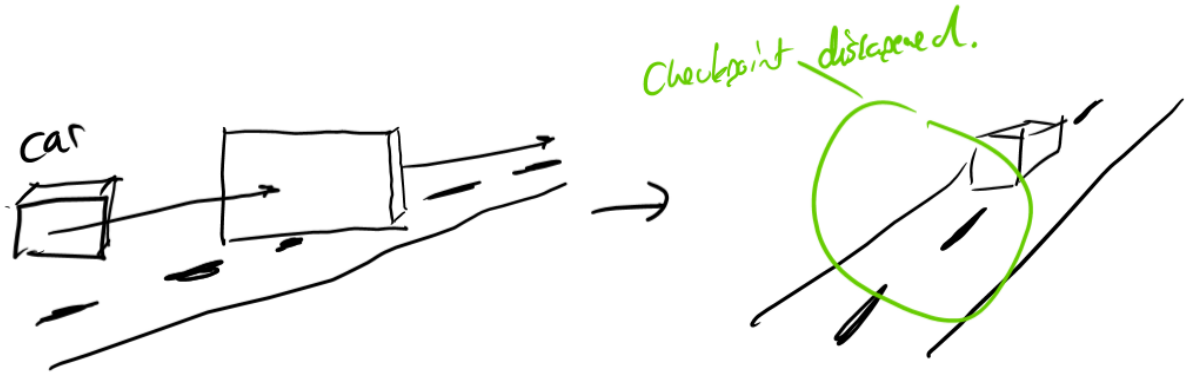
https://vimeo.com/693214780/0dd51e0eb3

I James Gammon – 27228 – 8292 – Abbeyfield school

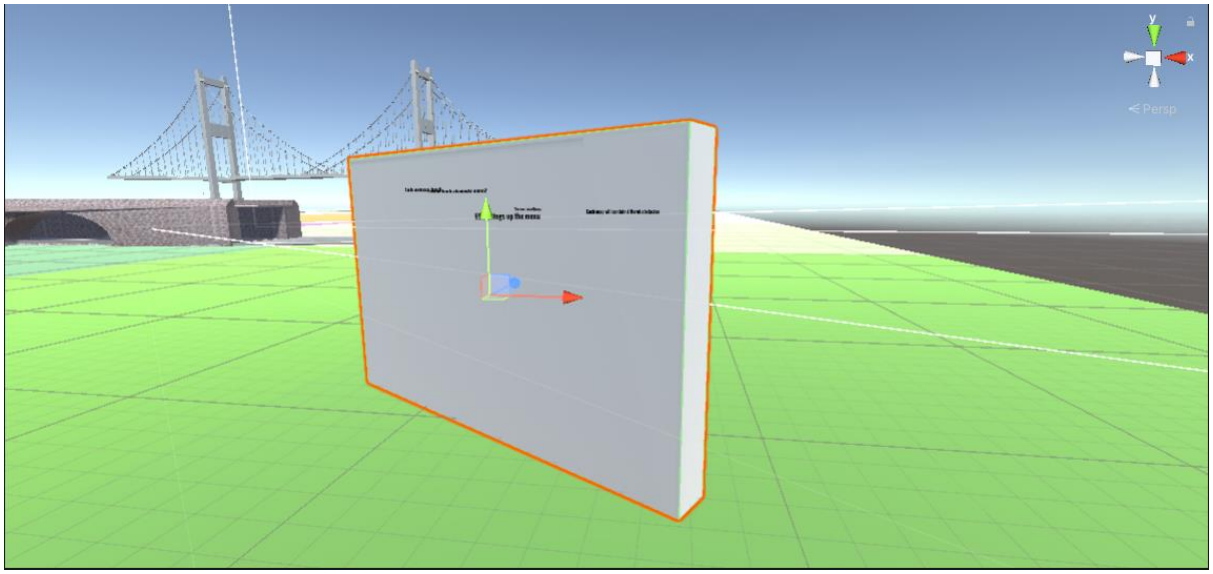
This is a video of the text being called by the buttons, correctly working in the game, this has now been applied to each of the current game maps (with different map texts to describe the maps and how to play.)

Creating the checkpoints:

The checkpoints should look like this:



Firstly, I created a game object which would pose as a check point



Here is the initial copy of the game object that will be used for the checkpoint.

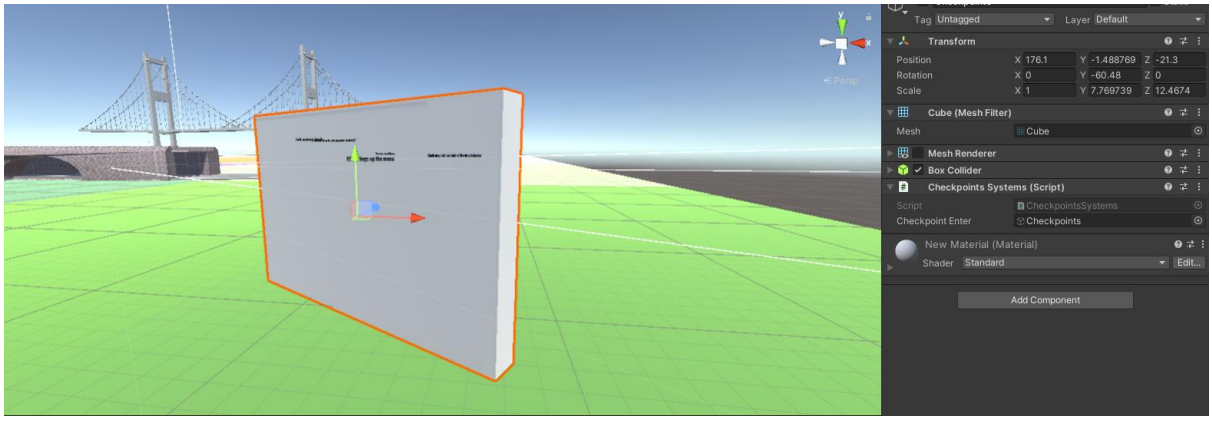
```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class CheckpointsSystems : MonoBehaviour
{
    public GameObject CheckpointEnter;

    void OnTriggerEnter()
    {
        CheckpointEnter.SetActive(false);
        Debug.Log("Checkpoint Entered");
    }
}
```

I created a GameObject (checkpointEnter) which was a variable representing the boolean state of the object. By being true the checkpoint would be present on the user's screen, and false would deactivate the game object.

The void OnTriggerEnter is a function which tracks whether the game object has been entered by my car (this is the trigger motion) causing the game object to become false and disappear.

I have planned for each trigger to be already present on the screen before the user enters them, this is to show the user where to travel in the long-term.

I James Gammon – 27228 – 8292 – Abbeyfield school

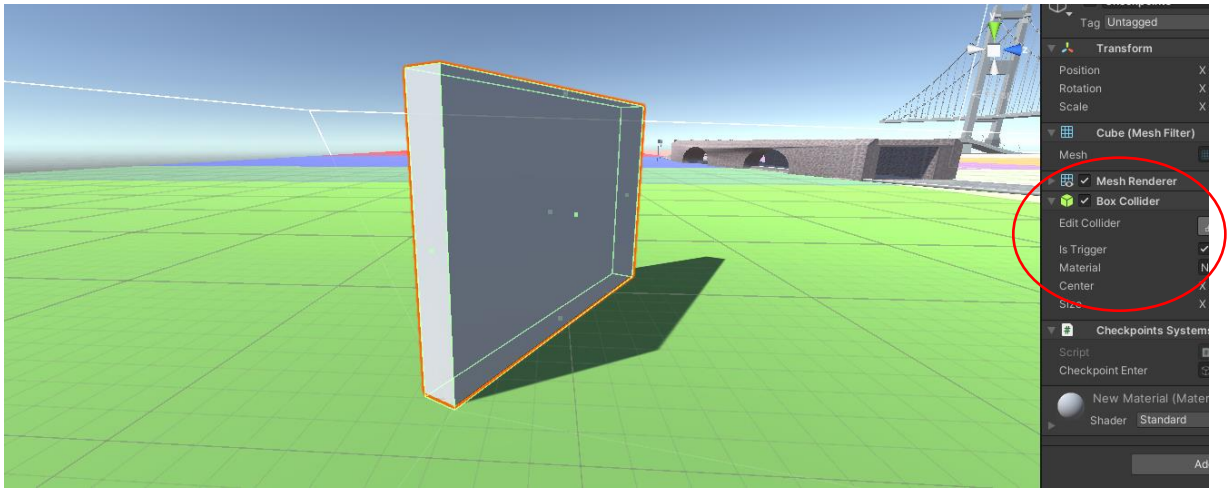


On the game object I applied the C# script and assigned the game object as the current trigger.

https://vimeo.com/693247562/2db3697791

An error occurred above; this shows that the Checkpoint was not registered as a game object; not printing the 'Checkpoint entered' statement.

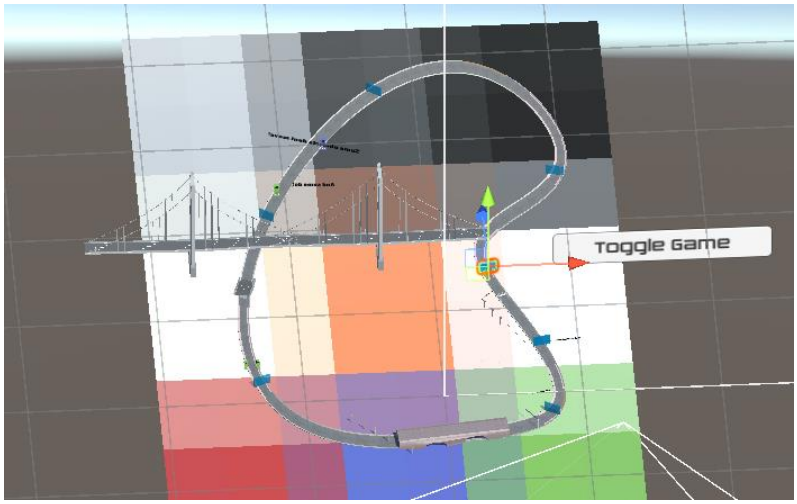
I found out I had to add a box collider to the object, when collision is detected in this area it counts as a trigger set.



Once the collider had been created, I then tested again:

https://vimeo.com/693251902/b6a38c0207

Which correctly identified the box collider as a trigger and printed the print statement 'checkpoint entered' in the dialogue box so I knew the correct branch of code was being followed.



the checkpoints have then been applied all around the maps for the tutorial game mode

Stage 7 – The Game tutorial – Review

What has been done ?

I have created and applied the tutorial aspect to my code, creating a user checkpoint system, and a togglable tutorial text screen teaching the user about the map, the game, and the gamemode they are playing, where each of these has been added to the 'time-trial' mode of each map.

How has it been tested ?

The mode has been tested through trial and error, while also creating prototypes. I used constructive testing to mimic how the player would use the game in a playing situation, and for destructive testing I was clicking on multiple boxes/buttons at the same time to see how the program reacted.

How it meets success criteria and user expectations?

Tutorial game mode will display checkpoints throughout the map and current tutorials on each game aspect.	Unity code which facilitates the disappearance of checkpoints once the user has entered them also code with initiates the tutorials being active/disabled shown.

In this stage of the game all the success criteria have been met and completed, the checkpoints are implemented in a way that carefully guide the user around the map, however this is not limited to speed and users can travel as fast as they desire. Users often expect game tutorials to cover the basics of the game and let the player naturally find out more about the game, however I chose to lead the player closely in the tutorial which means all players can adapt quickly.

Criteria being met?

(F)Abstraction of unnecessary clutter on screen for understanding what needs to be done	Lots of interactivity on a menu can lead to a confusing menu
(S)Clear instructions on navigation	I am expecting a wide demographic to play my game so a clear navigation through the settings/menu
Simple walkthrough of menus	Having a simple walkthrough menu allows all ages to navigate the game simply and effectively. Simple menus are often better to

	look at and make the game appear to be more professional.

The criteria in the analysis stage state that the abstraction of unnecessary clutter on the screen is vital for the aesthetics of the game. In the tutorial game mode this is demonstrated through simplistic buttons with a clean information statement, so it does not confuse the wide range of users for the game.

Changes in the design that have resulted from stage?

There is one change in the design that occurred in this area and that is the changing of the layout of the menu. Originally, I had designed a menu that required 2 clicks of buttons to pull up the walkthrough text, one to open the menu then another to lead to the desired menu choice. I thought this was sufficient in the design as it saved space, however, did not account for the fact some users may have minimal mouse movement and not have time to click on buttons on the screen; so, I changed the design to a simple 3 button approach on the GUI which I felt worked better.

Summary of the project as a prototype at this stage ?

Overall, the prototype is developing well. I have added the second stage of map uniqueness which really adds to the game's depth, playing multiple game modes on the same or different maps (something which was vital). The prototype at this stage is becoming really well structured with clear modular components. For the future I am looking to include the 3rd and final development portion (explore / free play) and focus on linking the current game scores to the database.

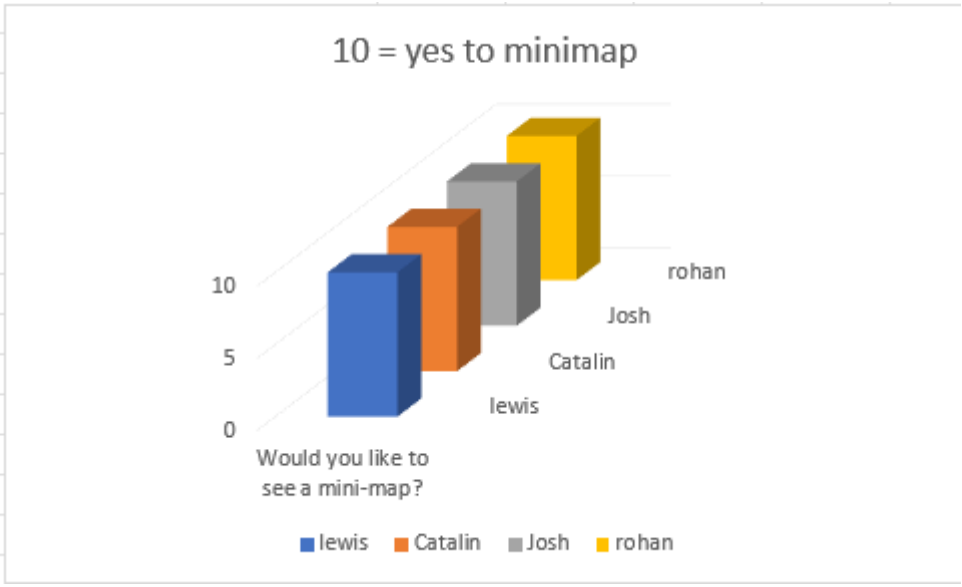
Stage 8 – Free play

My stage 8 free play was designed to be a map explorer with a simple count down menu, player, and map. I originally designed this to just include these features which are already pre-made into the game.

However, I wanted to ask the stakeholders what they would like to see Included in the mini-map and this was said:

Lewis:	Josh:	Rohan	Catalin:
I think the the explorer game mode is good and I like how it is different from the other modes.	I think this is okay, however I would like to see something to define this so it's not as boring.	I think that a mini map would be a cool idea:	I think it's a good idea but needs something to specialize it from being a little bit dull, like an additional feature.

I James Gammon – 27228 – 8292 – Abbeyfield school

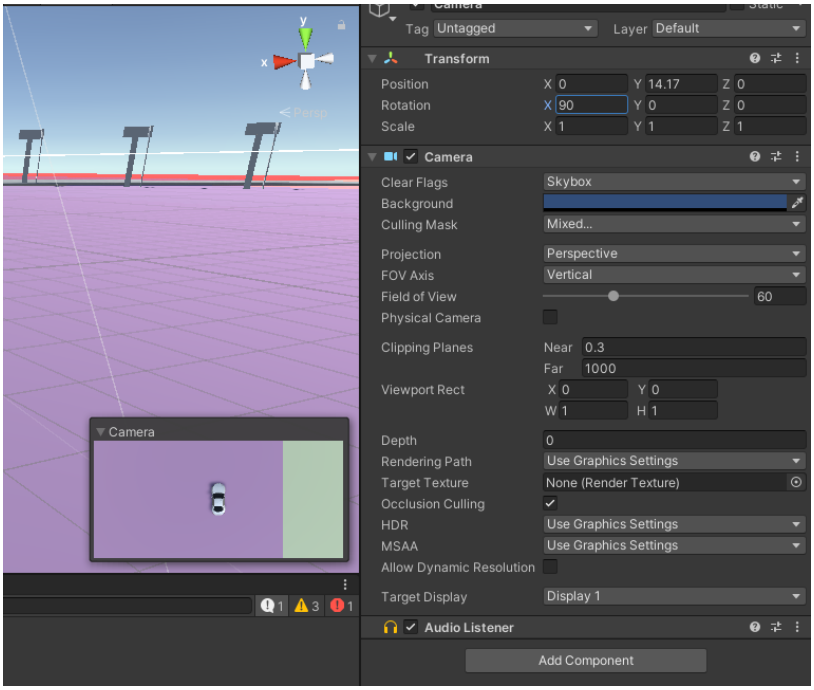


On reflection, all the stakeholders where completely in favor of a mini-map idea.

Creating the camera:

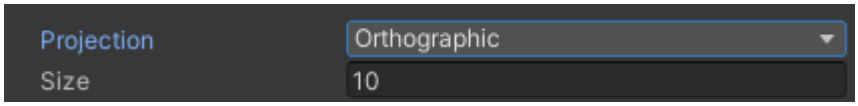
First, I created a new game object of camera:

This functions as a separate view from the main camera which operates to follow the car.

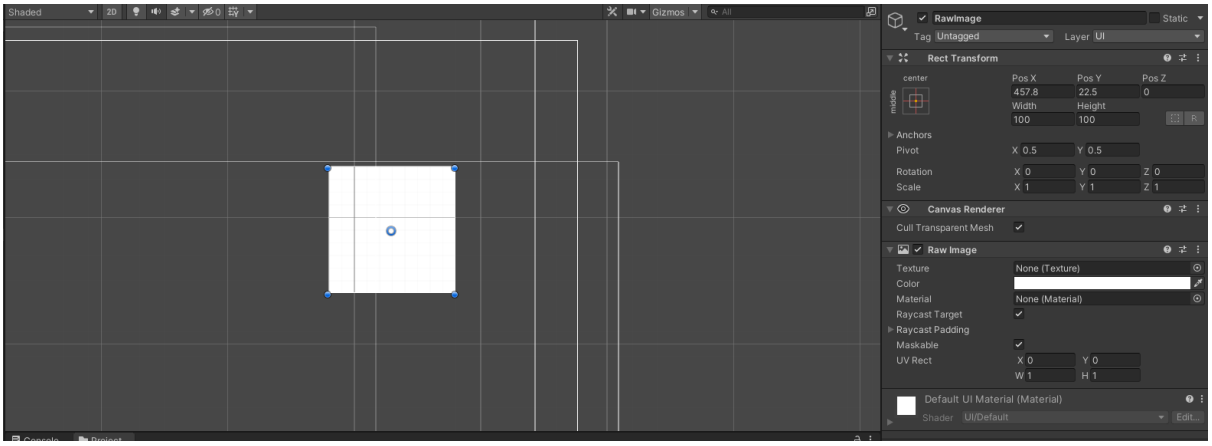


I then transformed the camera to X+Y+Z coordinates of the car, and flipped the camera 90 degrees on the X-axis to point directly on the car.

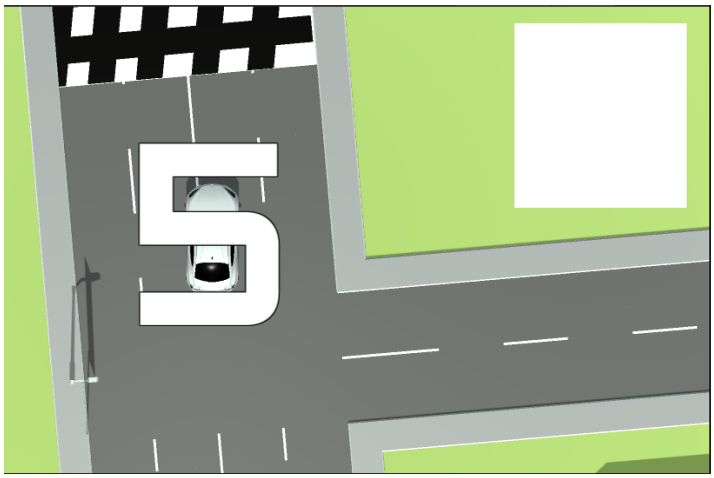
I then changed the projection to Orthographic, which abstracted any unnecessary values of the map appearing it to look 2D.

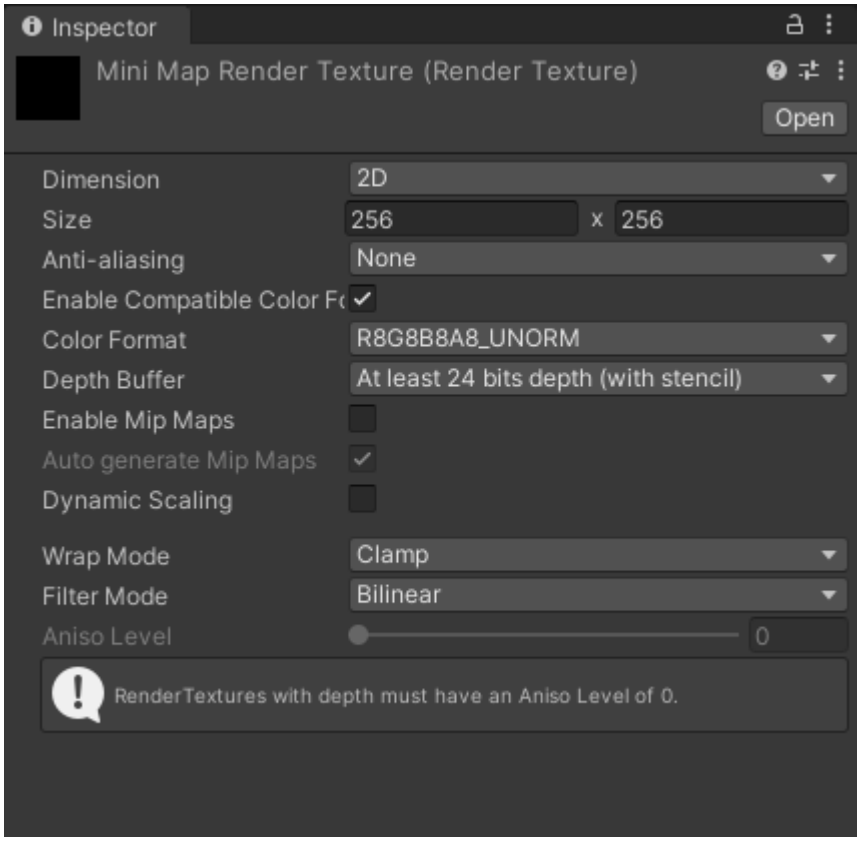


I then created a raw image component on the user interface, this will be a canvas for the camera projection. I will take the Texture for the raw image and apply the camera source feed created earlier to the raw image, displaying on the UI.



On the game view, this appeared to be a blank square. I now need to input the source feed into the raw image to display a camera.

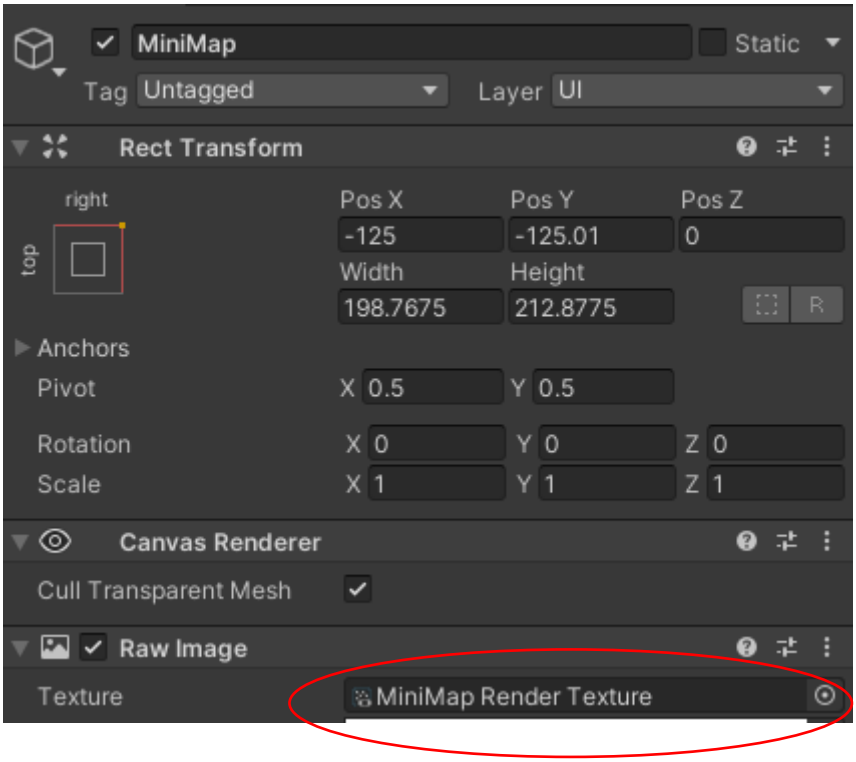




I then created this render texture, rendering images built into unity. I created one the same size as my raw text, this is a middle man between my camera and my game UI display.

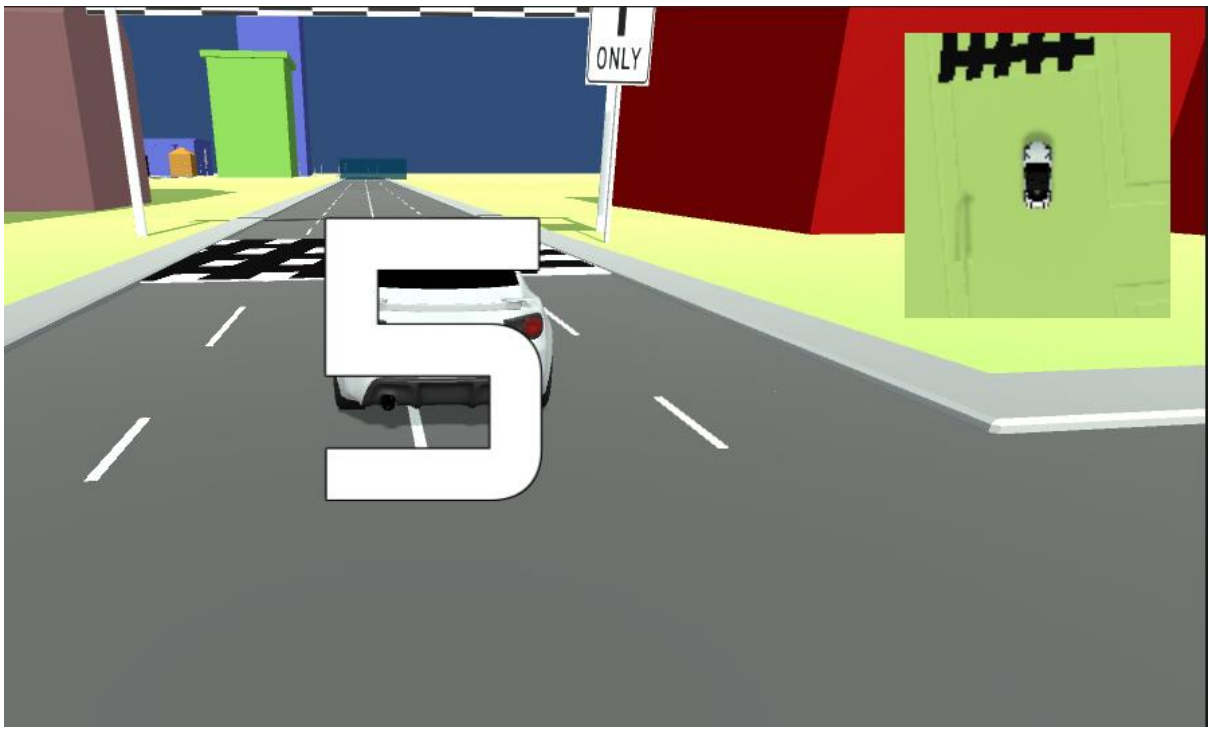
Now, my original game camera is the main focus, and the contents of the mini map camera are being fed into the Mini Map render texture.

Here is the Raw image, the Render texture and Camera have not been assigned to it yet.



I then assigned the MiniMap render texture to the raw image on the UI.

This produced a MiniMap photo on screen, shown below:



https://vimeo.com/693630232/f74020a731

Error: The mini-map camera did not follow the user's car, and only the user.

Because of this I created a script with a transform to the player, this transform will then be copied over to the position of the main camera, following the player.

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

public class MinimapScript : MonoBehaviour
{

    public Transform player;

    void LateUpdate()
    {
        Vector3 newPosition = player.position;
        // causes the y position to stay the same
        newPosition.y = transform.position.y;
        transform.position = newPosition;

        transform.rotation = Quaternion.Euler(90f, player.eulerAngles.y, 0f);
    }

}
```

The script updates the vector3 position, which is the cars current position. The next line then continues the camera distances y a set distance away from the car, no matter the change in the players gradient.

Error:

Originally, I thought that by assigning the main camera to the player this would work, but it removed the mini map of the users UI when playing.

https://vimeo.com/693648029/5e63b42219

To fix this, I assigned the script to the mini map camera, instead of the GUI which then worked, following the car.

Evidence:

https://vimeo.com/693658558/442fe9960d

I then applied this to each of my explore maps in the game.

Example of the mini map on the medium difficulty game

Stage 8 – Free Play – Review

What has been done?

During this stage I have completed the free play mode arena, which is the final design completion of my game with its 3 different modes. I have added a special feature to each adding diversity to the game, I have also worked closely with the players of my game which has allowed development to be in the favor of the players.

How has it been tested

The game mode has been tested in a couple of different ways; firstly, I have used constructive testing to mimic the players of the game. There was not much to test in terms of baseline components as I was re-using other from different stages, such as the car ect.

On the new component added, I tested this by recreating every change in direction and height that the user could go by, aswell as following steps the user would take when playing the game. This led me to fix the errors of the mini map disappearing from the GUI once the game had started and the camera not following the car.

How it meets success criteria and user expectations

I the design element, I designed that the free-play arena lets the users freely explore the 3 different maps with no time restraint or specific route to follow. I picked this idea up from the other examples of games, which each included a free play arena. The success criteria states that: the free play arena must have no restraints.

I have completed this through the only things on the screen being the mini-map (to aid exploration), the users username and the countdown menu.

Criteria being met

Explore game mode includes no parameters for time completion or a set rule to follow.	Screen shots of the GUI display of the car navigating this map

Changes in the design that have resulted from stage

I have created a completely new element in the free play stage which I did not account for in the design. This was the mini-map menu displayed in the corner of the screen; this came from user feedback about the uniqueness of each game mode, and each game mode having something special to the user to add.

The mini map includes a second camera focusing on the player, displaying this on the users GUI.

Summary of the project as a prototype at this stage

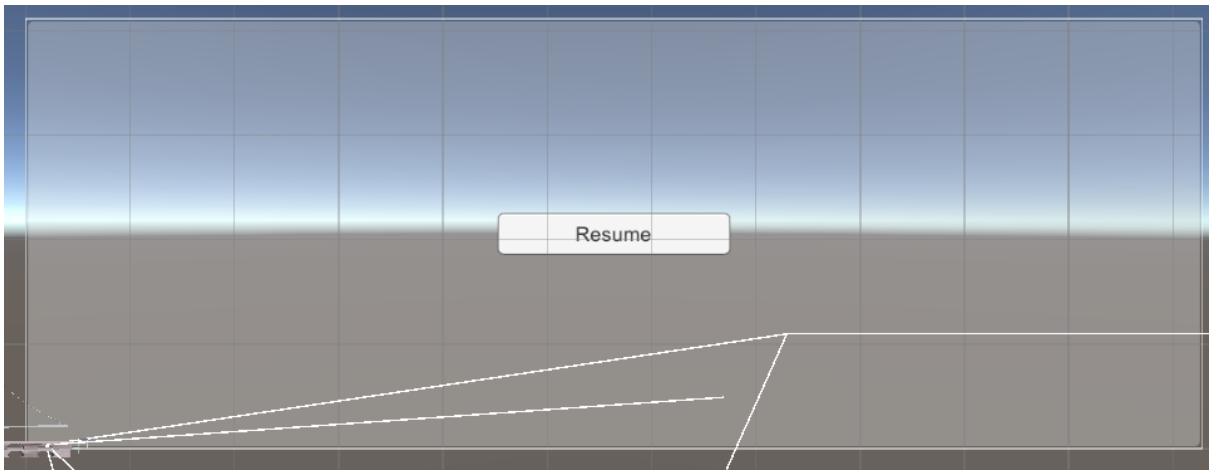
So far, the prototype of the project is very successful, I have completed each of the map's functionality; the time-trial. Free play and tutorial, each being fully accessible through the game menu. My next stages include the development of a menu GUI which allows players to navigate between different game modes and game maps and also accessing the menu.

Stage 10 – Pause Menu

Currently, once the player has entered a level / map they have no way of exciting unless they want to exit the game. I am going to create a pause menu which will be applied to each of the maps which provides a GUI interface allowing the user to travel back to the main menu – so they can choose a different game customization.

The pause menu will work by the user pressing the ESC key (synonymous to most laptop games) which will bring up the menu.

Firstly, I created a new game object called PauseMenu, with a button, labeled resume game. This is acting as my prototype .



Here is the pause menu.

To be able to interact with the game menu I created a script.

```
public class PauseMenuScript : MonoBehaviour
{
    public static bool GameIsPaused = false;

    // pause menu which will be activated / deactivated
    public GameObject pauseMenuUI;

    // Updates every second, activating once ESC is pressed
    // decides wether to activate the pause Menu or deactivate the pause menu e
    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (GameIsPaused)
            {
                Resume();
            }

            else
            {
                Pause();
            }
        }
    }
```

I created a main script which uses a Boolean value GameIsPaused to verify whether the GUI menu is active or not. The GUI menu (created and shown above) is the GameObject pauseMenuUI. To check whether the ESC key has been pressed the void update subroutine is checked every second.

If the game is paused the sub-routine Resume() is called which is defined later, else Pause() is called which pauses the game.

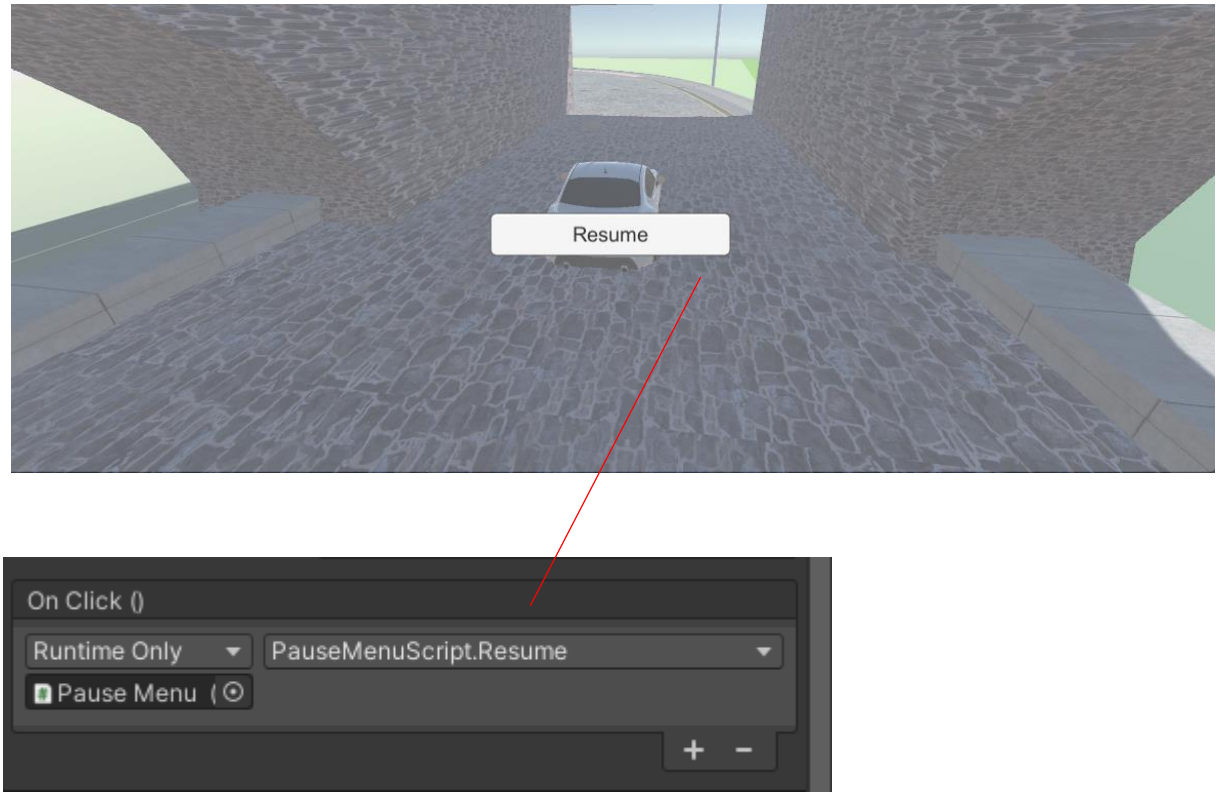
```
// resumes the game
public void Resume()
{
    Time.timeScale = 1f;
    GameIsPaused = false;
    // testing with debug.log statement to check wether subroutine is called
    Debug.Log("Game Resumed");
    UserNamePanelUI.SetActive(true);
}
```

The subroutine Resume changes the Boolean value to false telling the subroutine the game is not paused. This subroutine will be tested by a print statement bring printed if the subroutine is called.

```
42
43      // pauses the game
44      void Pause()
45      {
46          pauseMenuUI.SetActive(true);
47          Time.timeScale = 0f;
48          GameIsPaused = true;
49          Debug.Log("Game Paused");
50          UserNamePanelUI.SetActive(false);
51
52      }
53
54  }
```

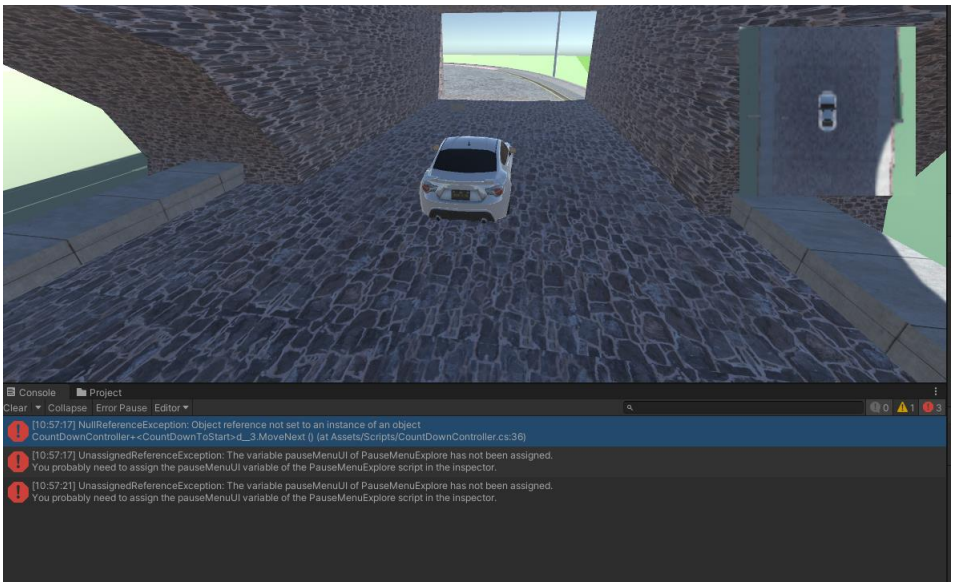
Here is the final subroutine that has been called, the pauseMenu.UI.SetActive(true) calls the GameObject menu and sets it to true which displays this on the user's screen. This subroutine is also tested by a Debug.Log statement.

The usernamePanelUI.SetActive(false) disables all of the other UI elements that are on the screen.

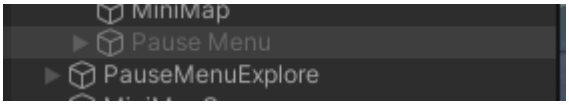


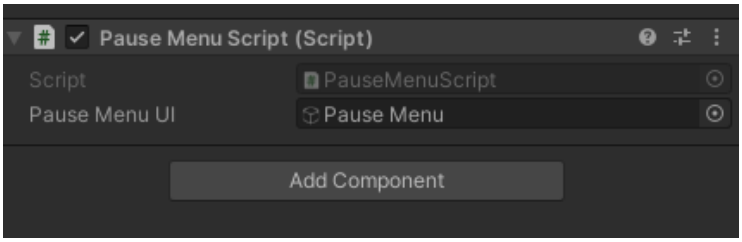
The PauseMenu button on the UI has been assigned the Resume function, so I can test wether the Resume function works.

Testing:



I got an error when testing the pause menu initially, the error states the Object reference is set to an instance of a null object, meaning a value that is assigned does not exist currently.





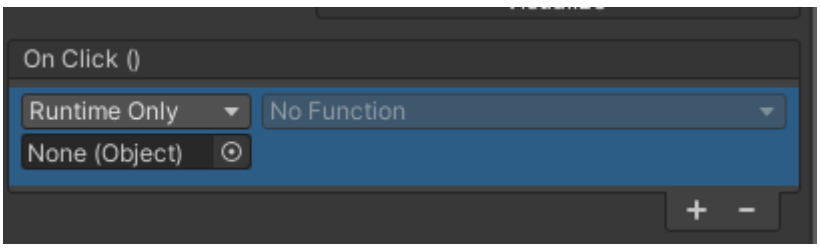
This error is because my Pause Menu script is assigned to my PauseMenu GUI which is deactivated when not called, so the script object is not appearing when ESC is being pressed. To fix this I added the PauseMenu script to the canvas element.

Error:

https://vimeo.com/695444395/78be69dc22

Calling the pause menu was not a problem and worked well, however the Resume button did not work and the only way I was able to disable the pause menu was to press escape again, this is a function I wanted but I need the resume button to work.

When I moved the script to the canvas, the function button deleted off of unity and therefore the button wasn't assingned to do anything.



I fixed this by assigning the subroutine on click function (Resume()) to the button. Where the pause menu then worked:

https://vimeo.com/695448453/5f9ce0d563

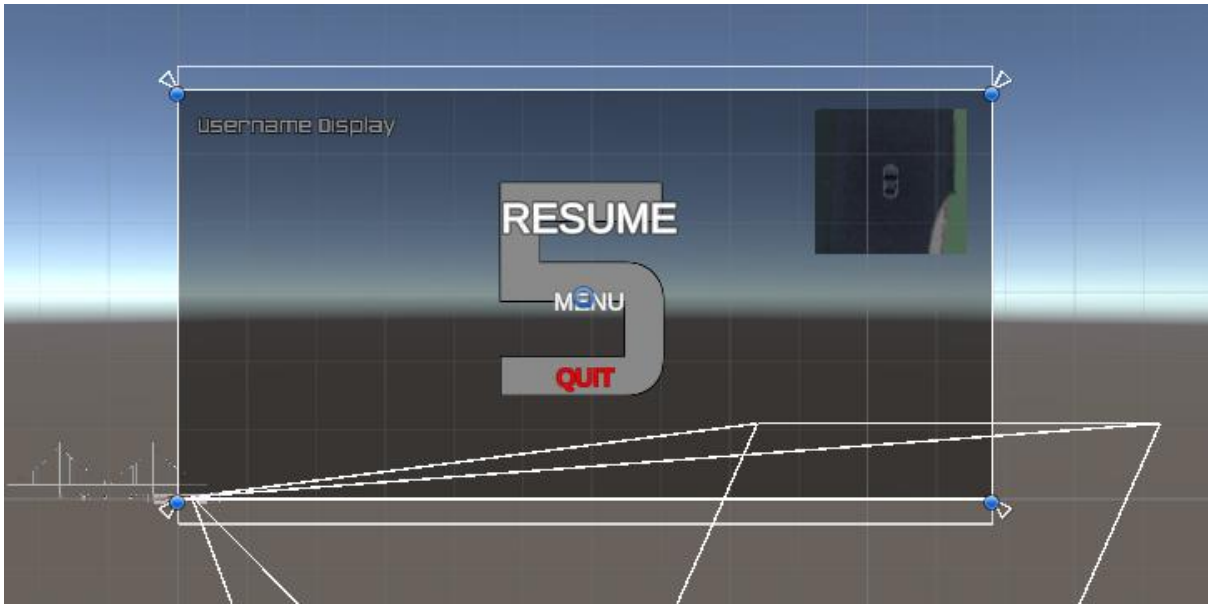
Above is the pause menu working as a prototype, now I need to add other functions.

```
49      public void LoadMenu()
50      {
51          Time.timeScale = 1f;
52          Debug.Log("Loading Menu...");
53          SceneManager.LoadScene("MainMenu");
54      }
55
56      public void QuitGame()
57      {
58          Debug.Log("Quitting Game...");
59          Application.Quit();
60      }
61
62  }
```

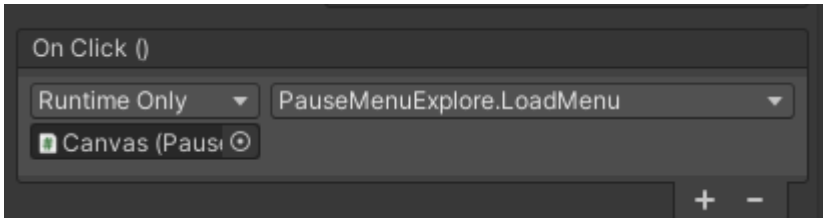
Here I added to extra buttons, a Load Menu function, and a Quit game function, these will be applied to the Menu button and Quit button.

SceneManager.LoadScene is a built in unity function which allows me to change the scene without pausing the game.

Application.Quit is another built in Unity function which quits the current open app (unity)



Here shows the new PauseMenu, with a new UI design and added buttons.



Above is an example of the Menu button, which has been assigned the Menu subroutine to be called when clicked.

Testing

https://vimeo.com/695450536/a70a98c34d

Here is the PauseMenu being tested on the countdown where it is working perfectly, the Countdown timer stops and the pause menu activates.

Menu button:

https://vimeo.com/695450949/fda6a2c757

By clicking the menu button, the game takes me back to the time-trial selection successfully.

Quit:

```
public void QuitGame()
{
    Debug.Log("Quitting Game...");
    //Application.Quit();
}
```

Here I have commented out the Quit button, so the application does not quit, being tested by the print statement.

Here is the print statement to show how the game would have quit. As this menu is working completely, I added this to each of the Explore Maps. I made changes to the code for the tutorial Maps and timed maps as they have different components on their UI.

Tutorial Maps:

```
public class PauseMenuTutorial : MonoBehaviour
{
    public static bool GameIsPaused = false;
    public GameObject pauseMenuUI;
    public GameObject UserNamePanelUI;
    public GameObject tutorialbuttons;


    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (GameIsPaused)
            {
                Resume();
                tutorialbuttons.SetActive(true);
            }

            else
            {
                Pause();
                tutorialbuttons.SetActive(false);
            }
        }
    }
```

Here, for the tutorial buttons I made the UI which housed the tutorial buttons and text on the screen and made this a collective game object called tutorial buttons. While the Resume function was being called aswell the buttons where being displayed. While the game was paused (while the pause routine was being called) I disabled the canvas housing this GUI; so, it was not seen when the menu was called. This leads to a cleaner menu and a better user experience.

Evidence:

https://vimeo.com/700237738/725db7d0e2

Timed maps:

```
public class PauseMenu : MonoBehaviour {

    public static bool GameIsPaused = false;
    public GameObject pauseMenuUI;
    public GameObject UserNamePanelUI;
    public GameObject LapLabelUI;
    public GameObject BestTimePanel;

    void Update() {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (GameIsPaused)
            {
            Resume();
            } else
            {
                Pause();
            }
        }
    }

    public void Resume ()
    {
        pauseMenuUI.SetActive(false);
        Time.timeScale = 1f;
        GameIsPaused = false;
        Debug.Log("Game Resumed");
        LapLabelUI.SetActive(true);
        UserNamePanelUI.SetActive(true);
        BestTimePanel.SetActive(true);
    }

    void Pause ()
    {
        pauseMenuUI.SetActive(true);
        Time.timeScale = 0f;
        GameIsPaused = true;
        Debug.Log("Game Paused");
        UserNamePanelUI.SetActive(false);
        LapLabelUI.SetActive(false);
        BestTimePanel.SetActive(false);

    }
```

Here, I added Game objects to account for the timer, best time lap and current lap counter. These elements were set active and inactive when the pause menu was abled and disabled. For this I used a separate script.

Stage 10 – Pause Menu review

What has been done?

In this section I have created and implemented the pause menu for each of the maps and game modes. At the start I created a prototype to show how the menu would work, then built upon this prototype to specialize to each of the maps (activating and deactivating the special GUI components) which led to a better user experience.

I have also created an escape from each of the other game modes, as previously there wasn't one. This allows the players to completely end the game or, move back into the menu, allowing the user to pick another game style/ map.

How has it been tested?

The game has been tested by several ways. I tested this through:

- Calling the menus at different points
- Displaying print statements at called subroutines so I knew where the program was and what iterative pattern it was following.
- Destructive and constructive testing

For example, at one point I called the menu during when the game was being played, this allowed me to get a feel for the user experience. Then, to find an error I called the menu when the countdown display was active, to see how it would react to force an error from the system.

Another example is where I printed print statements on routines, so I knew at which the point they were getting called and if they were getting called. This was tested in the Quit Game menu button as I was testing whether it performed.

How it meets success criteria and user expectations ?

The ESC button functionally calls the menu, each button on the menu is functional.	

The user always expects there to be a pause menu for each game so I have included one in mine. As the user is constantly told about the customization of the game, I added this function while inside of a game so it was not only implemented into the menus specifically. .

Criteria being met?

Option to call menu pause menu in game which displays functions such as: Resume game Main Menu Quit and save	Showing facilitating code which allows the user to call the menu options.
Subroutines across the GUI are called without fail	Facilitating code showed where routines are called, and functions are defined.

There are two pieces of criteria being met on this stage, the first is the actual option to call the game menu, which I have tested and implemented and the second is that subroutines across the GUI are called without fail.

The menu design closely related to the design feature, as this is the most optimal design for the user demographic. I did not implement a save option however as this is only displayed on the leaderboard.

Changes in the design that have resulted from stage ?

There have been no specific changes to the design so far, however as the game developed with specific GUI features, I adapted the facilitating code for each game map to account for the different components on the GUI. This creates an uncluttered screen with simple decision boxes to stop the user from being confused – which is important aspect in the game.

Summary of the project as a prototype at this stage?

So far, the project is around 80% complete, the game can fully and functionally work, navigating through starting menus, customizing the game to the user's choice. I now need to add the complexity which is a searching algorithm for the leaderboard and to print this in the leaderboard GUI.

Stage 10 - Bubble sort for leaderboard

The bubble sort for the leaderboard is the most complex part of the game, it occurs at the end of the Hard level time trial map, taking the best score out of the 3 possible laps. The final time is then converted from a mm:ss:ff format to a mmssff format inputted into a c++ bubble sort, connecting to the database and times.

The bubble sort performs on 3 inputs, the current users score (just achieved) the scores already stored in the database (for every user) and the username that each of the scores is assigned to.

The bubble sort will work in descending order, with the smallest time (the fastest) being printed at the top of the leaderboard and so on.

Leaderboard GUI

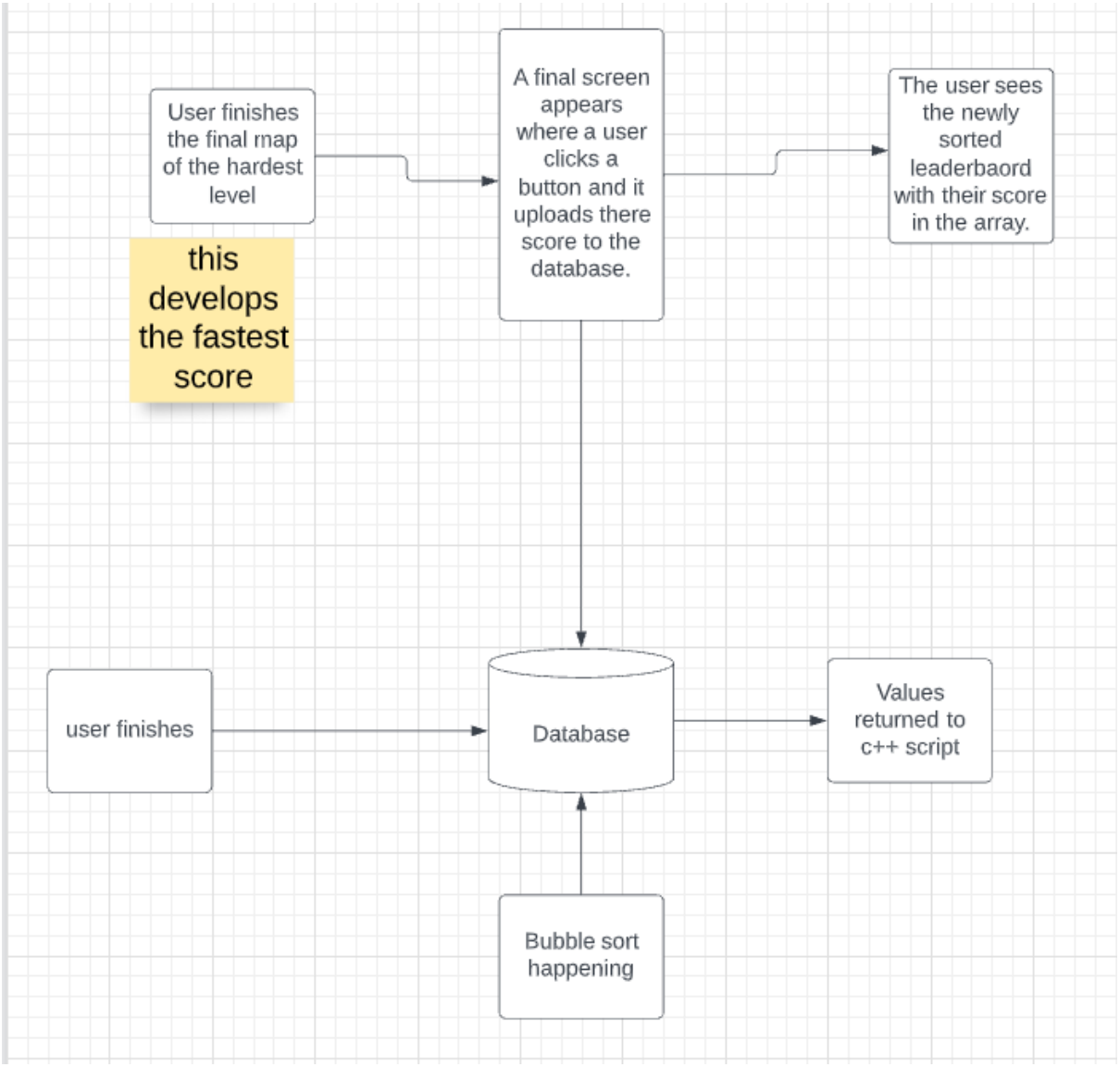
Here is the current GUI menu developed previously which houses the users score and input. The bubble sort will directly print values into this table.

Below there is a flow diagram facilitating the processes for the bubble sort.

Key aspects:

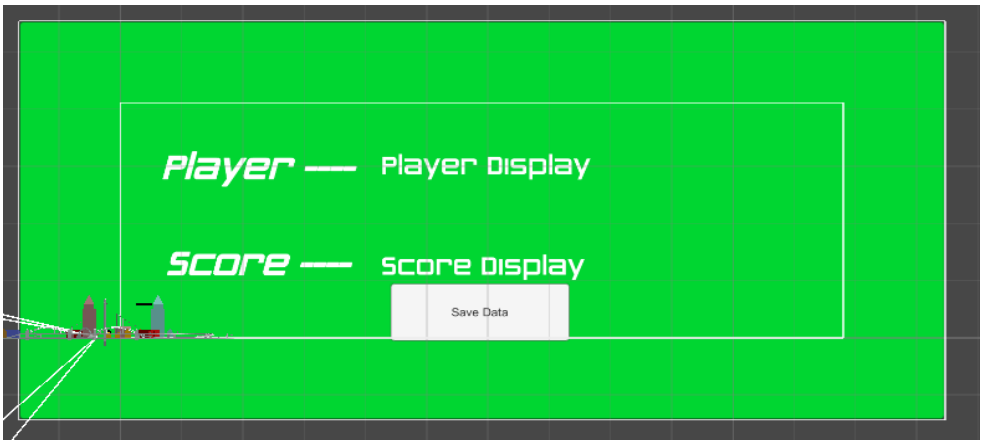
- Updating the database
- Connecting the bubble sort to the database
- Performing the bubble sort on the inputs
- Applying the outputs of the database into the leaderboard

Updating the database



On the ScoreTimerManager.DB script there is a value which when changes to 3 laps disables all the GUI relevant to the time trial and displays the ScoreMenuUI gameobject; this GUI being called will have a button which uploads the users score.

This was planned in the design section and implemented into the time-trial script as I knew this was going to occur.

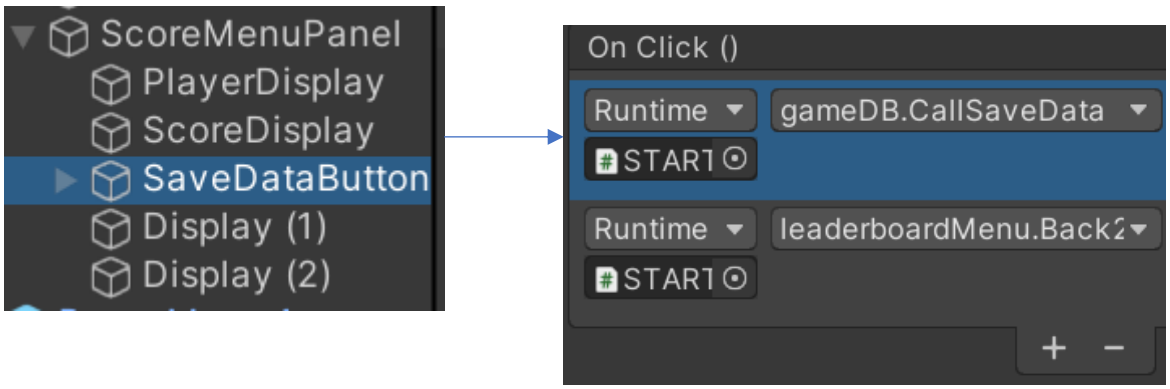


Here I have designed the final GUI menu to display the users time and name, the time comes from the best time; displayed here in the scoretimemanager.db script covered previously.

```
if (bestTimes.Count > 0)
{
    bestTimes.Sort();
    mintimePlayed = TimeSpan.FromSeconds(bestTimes[0]);
    string timePlayedStr = "" + mintimePlayed.ToString("mm':'ss'.'ff");
    BestTimeCounter.text = timePlayedStr;
    DBManager.score = int.Parse(mintimePlayed.ToString("mmssff"));


    //assigning the best player score to DBManager to be uploaded
    Debug.Log("The fastest score was " + DBManager.score + "Seconds... Well Done")
}
```

By taking this best score and the DBManger.username I can change the values on the final GUI.

On the GUI; the save player button has an attached script which enables it to communicate with the database.



```
public void CallSaveData()
{
    StartCoroutine(SavePlayerData());
}

IEnumerator SavePlayerData()
{
    WWWForm form = new WWWForm();
    form.AddField("username", DBManager.username);
    form.AddField("score", DBManager.score);

    WWW www = new WWW("http://localhost/sqlconnect/savedata.php", form);
    yield return www;
    if (www.text == "0")
    {
        Debug.Log("Game is Saved. ");
    }
    else
    {
        Debug.Log("Save Failed. Error no: " + www.text);
    }
}
```

the script then described in the gameDB.cs file earlier which was tested is then applies to this button, so I know it already connects to the database.

Back to the leaderboard script:

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.SceneManagement;
5
6   public class leaderboardMenu : MonoBehaviour
7   {
8       public void Back2Menu()
9       {
10          SceneManager.LoadScene("MainMenu");
11      }
12
13      public void Back2Leaderboard()
14      {
15          SceneManager.LoadScene("LeaderBoard");
16      }
17  }
```

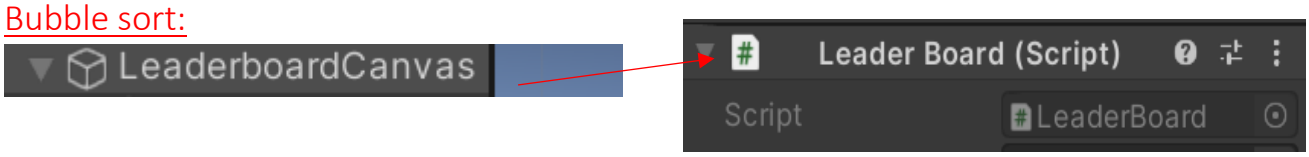
Here I created this script which takes the user back to the leaderboard. The leaderboard was already designed and created in the first stage at the creation of the GUI, so I connected this leaderboard scene to the button.

Test:

https://vimeo.com/695670987/c7e6f12ad7

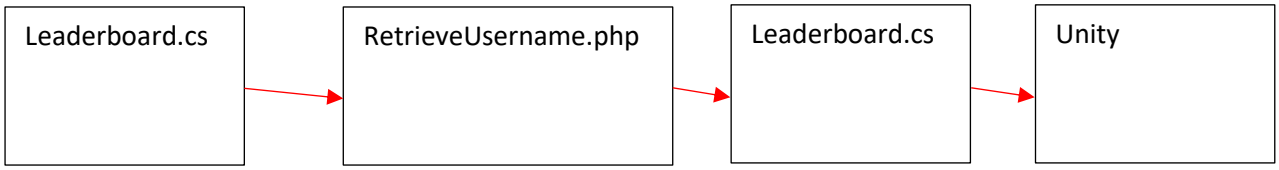
Above, the transition between the final page menu and the leaderboard works fine.

Bubble sort:



On the leaderboard canvas there is a script assigned called leaderboard; this contains the bubble sort.

The bubble sort uses 2 different scripts to communicate with unity and the database, these are Leaderboard.cs and RetrieveUsername.php



Leaderboard.cs:

```
8   public class LeaderBoard : MonoBehaviour
9   {
10      // creates variables to upload into unity
11      string a = string.Empty;
12      string b = string.Empty;
13
14      public Text Score;
15      public Text Username;
16
17      // creation of an array of usernames from the leaderboard
18      string[] array2  = new string[1000];
19
20      // array creation of for the scores inside of the leaderboard as initgers
21      int[] array1 = new int[1000];
22
23
24      List<int> list = new List<int>();
25      string temp;
26
```

Here is the class leaderboard. The class has multiple game objects which will be used to store the username and the associated score. The Text score + Username are the physical game objects in unity which will display the scores to the user.

These are both stored in arrays, array 2 stores an array of usernames and array 1 stores an array of passwords. The indices of the array are mimicked between the username and associated score.

The list and temp define where the elements wll be stored and the current element which is being sorted.

```
// immediately after opening the leaderboar script the resutls are imported and updated
void Awake()
{
    StartCoroutine(BubbleSort());
}
```

As the leaderboard script is applied in the leaderboard scene, it is immediately called and the BubbleSort is performed.

```
35      IEnumerator BubbleSort()
36      {
37          // creates a form that will retrieve the current scores and inputs from the leaderboard and input them
38          // into the array
39
40          WWW www = new WWW("http://localhost/sqlconnect/leaderboard.php");
41
42          yield return www;
43
44          // splits the usernames from a single line of string into differnet lines
45          string[] result = www.text.Split('\t');
46
47          //takes the first element in th results and while is less than total results
48          // then adds 2 to get to the next username
49
50          // score array
51          for (int i = 1; i < result.Length; i += 2)
52          {
53              // changes the position of score to an intiger
54              // passing the value of the leaderboard by value
55              int Num = int.Parse(result[i]);
56
57              // array elements are now value numbers
58              array1[i] = Num;
59          }
```

Here the BubbleSort() function creates a form which connects to the ‘players’ database and retrieves all information It stores; these are passed as usernames and passwords which are parsed into separate lines.

A score array is then created which takes the 1 element passed (the first score) and the third element passed (the third score) where these score values are formed into an integer and create an array called array1; with all the individual score times.

```
60
61          int temp = 0;
62
63          // for every element in the array
64          for (int write = 0; write < array1.Length; write++)
65          {
66              // Last i elements are already in place
67              for (int sort = 0; sort < array1.Length - 1; sort++)
68              {
69                  // checks the position of the score, if it is smaller than than the one above
70                  if (array1[sort] < array1[sort + 1])
71                  {
72
73                  // temp is the current element we area comparing
74                      temp = array1[sort + 1];
75                      // moves onto the next element
76                      array1[sort + 1] = array1[sort];
77                      // makes the current element to the adjacenet one
78                      array1[sort] = temp;
79                  }
80              }
81          }
```

Each of the elements are now being sorted. For the length of the array1 (every element in the array) (write var) the is being compared the number of times in the array. If the current element (array1[sort]) Is smaller than array1[sort+1] then there is no swap and the current element being compared increases, using the same fundamentals as the bubble sort.

```
82          //code to implement into the unity leaderboard
83
84          // for each element in the score array
85          for (int j = 0; j < array1.Length; j++)
86          {
87              // if the element is larger than the first one
88              if (array1[j] > 1)
89              {
90                  // a = string variable inputting the score into
91                  // a now becomes the current score in the array1[j] the current second best
92                  a = a + (array1[j].ToString() + "\n");
93
94                  // creating a form to submit to the database
95                  WWWForm form = new WWWForm();
96                  // incramenting the score variable to have the sorted score
97
98                  // posts the score and variable to the form
99                  form.AddField("scoreUnity", array1[j]);
100
101                 // php script which assigns the score variable to the username stored in the databa
102                 WWW www2 = new WWW("http://localhost/sqlconnect/RetrieveUsername.php", form);
103
104                 // returns of the usernames from the php script from the database
105                 yield return www2;
106
107                 // array which stores the string of usernames with associated scores
108                 string[] result2 = www2.text.Split('\t');
109                 // partitioins the values returned from the query
110
111                 for (int q = 0; q < result2.Length; q++)
112                 {
113                     // b (the usernames) are converted to string and assinged to the b array
114                     b = b + (result2[q].ToString() + "\n");
115                 }
116             }
117         }
```

Here is the function which implements the sorted elements into arrays which can be displayed in unity. For every element after the second one, the index position is partitioned, and a new form is created. This list of sorted scores will be used as a search comparison inside the database to find the associate usernames.

The form takes and inputs a variable called scoreUnity, inputting the current score element into the php script which makes this a variable.

```
104             // returns of the usernames from the php script from the database
105             yield return www2;
106
107             // array which stores the string of usernames with associated scores
108             string[] result2 = www2.text.Split('\t');
109             // partitioins the values returned from the query
110
111             for (int q = 0; q < result2.Length; q++)
112             {
113                 // b (the usernames) are converted to string and assinged to the b array
114                 b = b + (result2[q].ToString() + "\n");
115             }
116         }
117     }
```

The return WW2 is composed of a sorted list of usernames with reference to scores. (Usernames are found in descending order), for each username returned this is placed into the string array b (defined earlier)

Php scripts:

```
1   <?php
2
3     $con = mysqli_connect('localhost', 'root', 'root', 'unityaccess');
4
5     if (mysqli_connect_errno())
6       {
7         echo "1: conection failed"; // error code #1 = connection failed
8         exit();
9       }
10
11    $sql = "SELECT username, score FROM players";
12
13    $result = $con->query($sql);
14
15    if ($result->num_rows > 1) {
16     // output data of each row
17     while($row = $result->fetch_assoc()) {
18
19       echo $row["username"]."\t".$row["score"]. "\t";
20       }
21
22    }
23
24    $con->close();
25
26  ?>
```

This is the first php script used.

Leaderboard.php is what attains all of the current database information and supplies this to the script. A $con variable hosts the connection between the database and the php script.

An SQL statement selects all of the usernames stored inside of the database and their associated information which includes score.

I have used the username as a primary key because there is a chance that some scores can be the same, but none of the usernames can be the same.

RetrieveUsername.php:

```php
<?php

$con = mysqli_connect('localhost', 'root', 'root', 'unityaccess');

//check that conection happened
if (mysqli_connect_errno())
{
    echo "1: conection failed"; // error code #1 = connection failed
    exit();
}

// takes score input from the c++ script and creates a variable in php
// usernames with this score are then printed back to the c++ script in order

$scoreUnity = $_POST["scoreUnity"];
$int = (int)$scoreUnity;

// sql statement which connects to the database to find associative scores
$sql = "SELECT username FROM players WHERE scoreUnity = $int";

// sent as a query , using the $con variable to find the relational database inside of UnityAccess where the players table is
$result = $con->query($sql);

// if there is more than 0 rows then there are users in the database with this score (input from the SaveData courutine)
if ($result->num_rows > 0) {
 // output data of each row
 while($row = $result->fetch_assoc()) {

    // returns the row with username as the primary key
    echo $row["username"]."\t";
 }

 // else there are no scores of this value inside of teh database
} else {
  echo "No results in the database";
}

$con->close();

?>
```

Here is the php script which fetches the associated usernames from the score. Initially a data connection is also established.

$scoreUnity is the parameter passed from the unity script which passes the current array time into the script, and this is converted to an integer (stored as an integer in the database).

An SQL statement is then again used to find all of the usernames which have the score = scoreUnity.

```php
// sql statement which connects to the database to find associative scores
$sql = "SELECT username FROM players WHERE scoreUnity = $int";
```

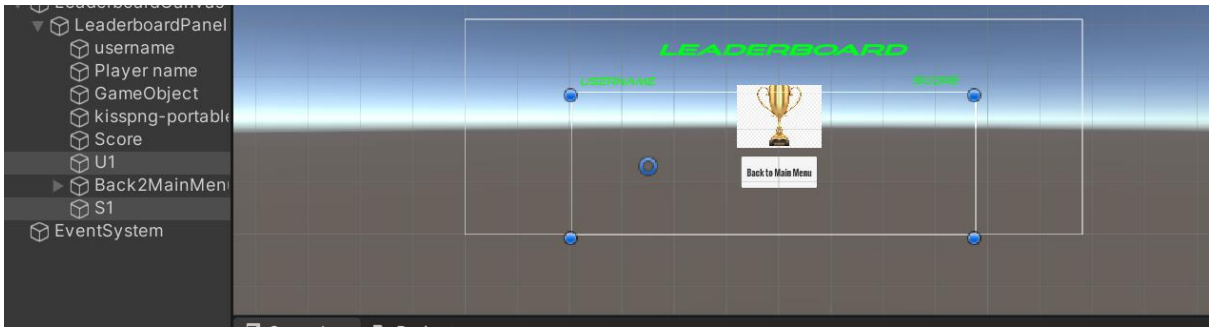
The number of rows with this score are then returned as an output and echoed to the unity script as username.

This area will be tested by output statements back to the unity log to indicate where the error is based on what branch.

Finally:

```
116         }
117     }
118
119     // assings these to game objects inside of unity
120     Score.text = (a);
121     Username.text = (b);
122
123
124
125
```

Finally, the outputs of the scripts are formatted as a + b and then applied to the game objects defined at the start.



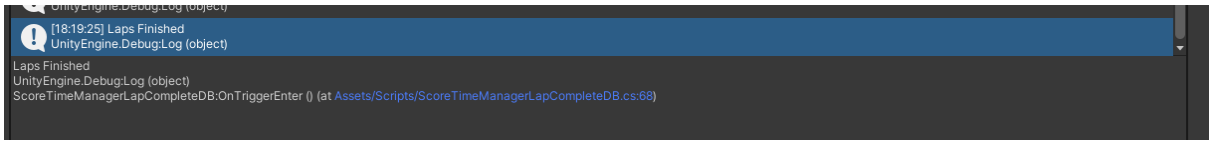
These game objects are S1 and U1 for the score and username.

Testing:
https://vimeo.com/695796662/d76b3d8c7d

The test below shows me logged in as test1 completing the map, the game over screen shows up successfully with my credentials however, the leaderboard does not display any information about my game.

This could be two errors:

- The score is not being submitted to the database
- The php script is not retrieving any information.



There is no Game Saved Debug.Log print statement so iknow the game data is not being uploaded to the database.

https://vimeo.com/695800772/cc995b4bb8

Here, the bubble sort receives the inputs, however the webpage document is displayed instead of the usernames. On the right, the players scores are displayed (test1 (previous) and test2 (current)). Now there is a problem with the usernames.

I James Gammon – 27228 – 8292 – Abbeyfield school

A screen shot shows that the scores on the bubble sort are being sorted ascendingly and not descendingly, so the fastest score is at the bottom.

There are 2 errors:

- Ascending bubble sort
- Usernames not showing

Usernames not showing:

Here, I changed the colour of the background to read the text and loaded the menu separately, here the php script is returning ‘no results In the database’ so the usernames cannot be found.

Error fixed:

To fix the error; I changed each of the 'scoreUnity' variables to 'score', as the leaderboard script uses 'score' as a variable it was not being transferred from the leaderboard to the username and score associated correctly; therefore, echoed no results in the database.

```
18      $score = $_POST["score"];
19      $int = (int)$score;
20
21
22      // sql statement which connects to the database to find associative scores
23      $sql = "SELECT username FROM players WHERE score = $int";
24
25
26      // sent as a query , using the $con variable to find the relational database inside of UnityAccess where the players table is
27      $result = $con->query($sql);
28
29
30      // if there is more than 0 rows then there are users in the database with this score (input from the SaveData courutine)
31      if ($result->num_rows > 0) {
32       // output data of each row
33       while($row = $result->fetch_assoc()) {
34
35         // returns the row with username as the primary key
36         echo $row["username"]."\t";
37       }
38
39
40       // else there are no scores of this value inside of teh database
41      } else {
42        echo "No results in the database";
43      }
44
45      $con->close();
46
47  ?>
```

Second error, the bubble sort sorting the wrong way:

Upon testing the bubble sort, I realized that the bubble sort was being sorted the complete wrong way, with the largest elements at the top and the shortest times at the bottom.

To fix this I must change whether the bubble sort swaps elements if the current is bigger than the latter.

```
// if (array1[sort] < array1[sort + 1])
if (array1[sort] > array1[sort + 1])
{
```

To try and fix this, I reversed the operation which swapped the elements if the current element was larger than the next one.

https://vimeo.com/695818644/42aae6cbe0

Above shows how the change correctly inversed the list and It now sorts correctly.

Further testing:

https://vimeo.com/695819593/0a3f0314da

Here, the leaderboard can be accessed while not being logged into, meaning it works correctly.

Stage 10 – Review

What has been done

Each game element is now completed with the game fully operational with each component. In this stage the complexity has been added. A bubble sort is performed on the current database elements in descending order with the fastest time at the start. This is in unison with 2 php scripts which supply all information to the bubble sort, and then request usernames. This stage follows on from the final panel stage which was developed in the game stage.

How has it been tested

The bubble sort has been tested in 2 different ways; I have been simulating the input of different users by registering which tests the integrity of the bubble sort with a larger amount of data elements.

I have used times which are very close together in time to see to what degree they are sorted to.

I have also used destructive testing to force an error from the GUI's; I have logged out of the account and accessed the Leaderboard directly from the start menu, seeing if all the times are displayed. A

Finally, I have used constructive testing which mimics the behavior of the player; this means the leaderboard is being called directly from the end of the game mode.

How it meets success criteria and user expectations

User expectations are that the Leaderboard is easy to access and clearly displaying all the metadata. The focus of the game is a clean, easily accessible, and easy transitional game. This has all been met by the fluent transition from the game and to the leaderboard, where the leaderboard is automatically sorted from the user entering the scene.

A leaderboard array is sorted and printed inside of the screen.

This screen shot is from the design element which highlights the functionality of the bubble sort. The criteria retrospectively states the array is sorted and then printed on screen.

Criteria being met

Once the user has completed 3 maps, the finish game screen opened, and the user can see their best time and username.	
At the end of the time trial game mode the leaderboard menu is opened and displayed.	

Here is the success criteria for the bubble sort, the first instance shows how the bubble sort is updated with the most recent score through the final game screen.

The second instance describes the natural flow of the leaderboard menu, which reduces the complexity of the transitions between Menus.

Leaderboard sorted via bubble sort	Code initiating bubble sort function will be shown, as well as how
The most recent user score is uploaded into the database	Code facilitating the input of the array and the manipulation of the data will be shown in screen shots.

Here are the success criteria for the bubble sort. The leaderboard is sorted through a bubble sort and there is a specific coroutine called 'SavePlayerData'.

Changes in the design that have resulted from stage

There have been no changes in the design stemming from this current stage.

Summary of the project as a prototype at this stage

The game is now complete with every functionality, and every stage being completed. Now what is needed is for final testing and testing from players. The game now has 3 fully unique, customizable maps, with a link to the leaderboard from the end of the hard level time trial and initial GUI. Each map has access to a pause menu which establishes better interconnectivity within the game.

Now, I need to look for improvements to make on the stakeholders and improvements for the future.

Stage 11 – Final Testing

Testing checklist:

Functions to be tested	Is it working
Game buttons call and display the correct menus.	Y
Credential input boxes create a new row in a database storing username, password, score	Y
Password is stored as a hash inside of the database	Y
Credentials input into the user login page are compared to the database. If correct the user can proceed, else the user is denied.	Y
The user cannot login/register until the conditions for the username and password are met, the register/login button is not interactable	Y
Each map selected and game mode selected produces the correct selection on the game display.	Y
User tutorials are toggleable, for the game map, mode, and controls, they can be present on and off screen from a button.	Y
Racing game mode starts the timer once the game is started and is reset once the user enters the finish lap checkpoint.	Y
Each lap time is input into an array of times on the Time Trial Map we qwhere the best time is produced onto the screen.	Y
The ESC button functionally calls the menu, each button on the menu is functional.	Y
Once the user has completed 3 maps, the finish game screen opened, and the user can see their best time and username.	Y
At the end of the time trial game mode the leaderboard menu is opened and displayed.	Y
The car is controlled by WASD or the arrow keys.	Y

Game buttons call and display the correct menus:

The first thing to test is the user interface which is the first thing the user meets.

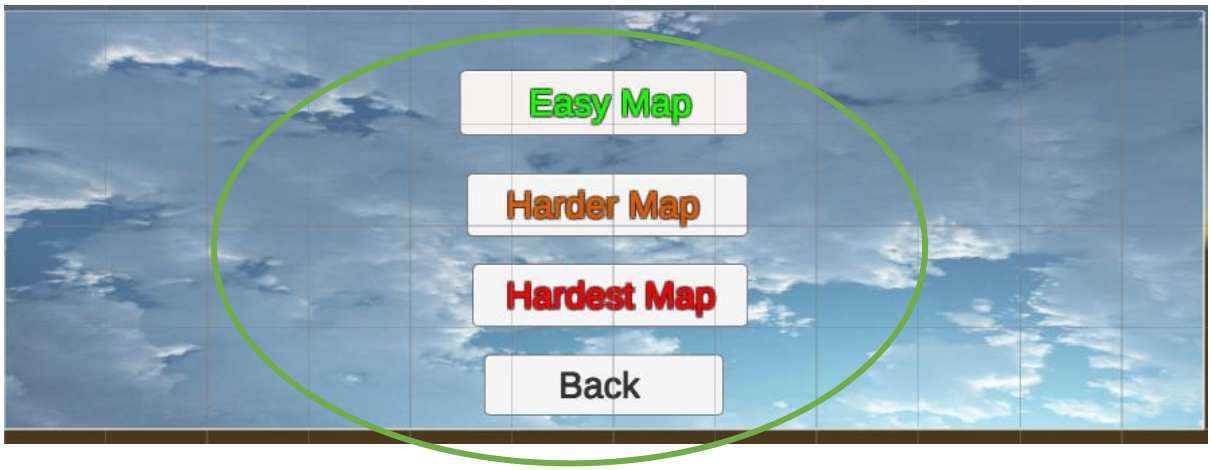
https://vimeo.com/695873711/001c744eeb

A test above shows that all the buttons for the game work successfully. The sliders for the option menu show a graphical change and hovering over each button shows a change in color to a darker color which indicates to the user that the button will be pressed.

The register field shows that some of the text has different fonts which I would like to change so it has a single format; these are highlighted in red.

These changes where made in red to show that standard text across the GUI.

On the map select GUI, the text was not the same which had been used on the other slides.

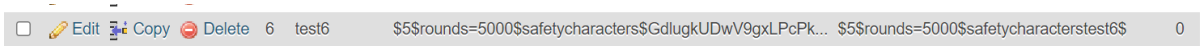


These changes above are made in green.

Credential input boxes create a new row in a database storing username, password, score

https://vimeo.com/696021948/81fb96f2fe

The video above shows a complete run through of the credential input boxes creating a new row in the database, the username is clearly stored under the username attribute, and so on for password (hashed) and score.



Password is stored as a hash inside of the database

https://vimeo.com/696021948/81fb96f2fe

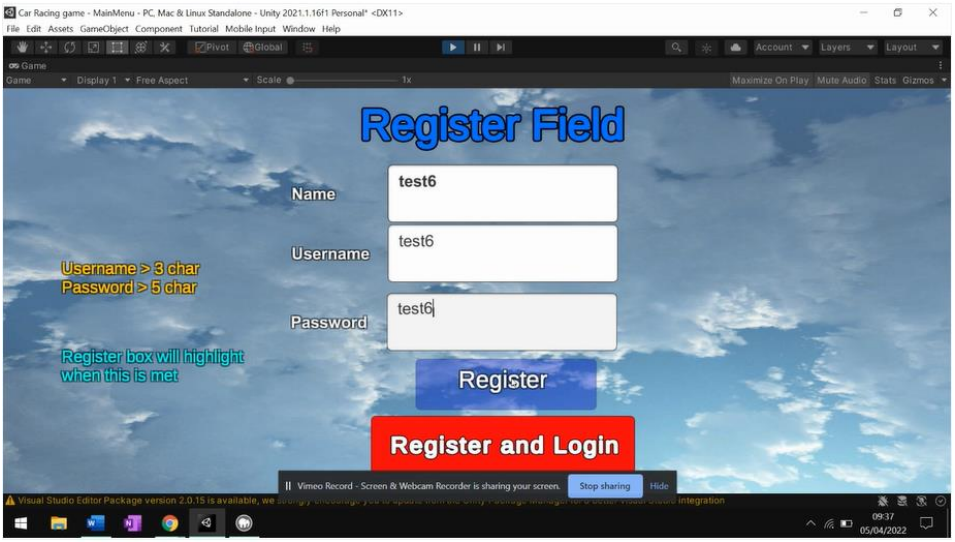
The clip above also clearly shows the hashing of the password again in the database. The hash function is a one-way encryption of data inside of the database, and to compare the input password we apply the hash to the input and compare if the values are the same. The hash encrypts the plain text.

	id	username	hash
Edit Copy Delete	1	test1	5rounds=5000$safetycharacters$fWLlz5PrzVNe8Dride...
Edit Copy Delete	2	test2	5rounds=5000$safetycharacters$2eNLdoPU6LVD0Catk5...
Edit Copy Delete	3	test3	5rounds=5000$safetycharacters$2eNLdoPU6LVD0Catk5...
Edit Copy Delete	4	test4	5rounds=5000$safetycharacters$j8oycr8tygq8QLj12w...
Edit Copy Delete	5	test 5	5rounds=5000$safetycharacters$fWLlz5PrzVNe8Dride...
Edit Copy Delete	6	test6	5rounds=5000$safetycharacters$GdlugkUDwV9gxLPcPk...

Credentials input into the user login page are compared to the database. If correct the user can proceed, else the user is denied.

https://vimeo.com/696025681/3ae6c7bb2d

The clip above shows a successful login demonstration using the same credentials input earlier, this process works by comparing the username initially, and then comparing the input password, with the hashed password, by applying the hash function to the input password.



Here in this screen shot from earlier we see that the username and password both are test6, and the login function recongnises me by inputting the correct password 'test6' instead of the stored hash in the database.

https://vimeo.com/696029456/a7f481bee4

Above shows the denial of the user entering the wrong password. In response to this, a Debug.Log statement printed to show the user that the password was incorrect (in this case it was).

https://vimeo.com/696029998/bd17562e39

In the case above: the username was now incorrect, a different Debug.Log statement showed that the username was incorrect.

Destructive testing:

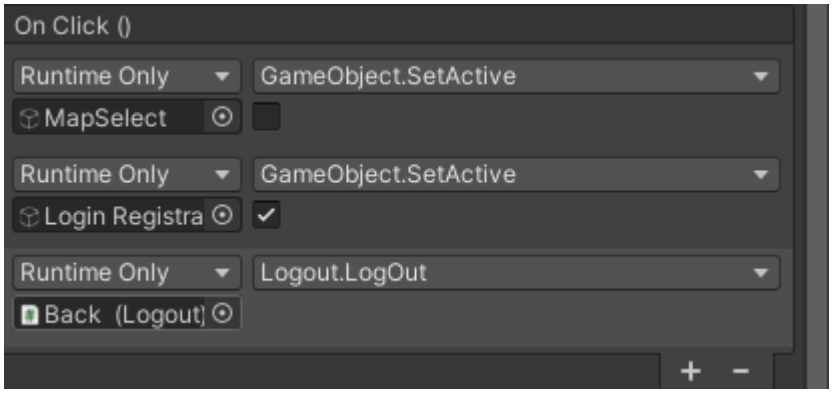
To destructively test this; I logged in with test9, then pressed the back buttons until I could log in again. An error surfaced that when pressing back on the game map selection (the first after login) it did not log me out.

https://vimeo.com/696046176/c1366ae346

The video above displays this well, I was stuck and only able to log back into test9.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class Logout : MonoBehaviour
6   {
7       public static void LogOut() //username changed to null to log out
8       {
9           DBManager.username = null;
10      }
11  }
```

To fix this, I created a new script called logout which creates a function that when called, changes the DBManager.username (username variable) to null.



I then applied this to the Back button on the MapSelect, which calls the function.

https://vimeo.com/696049849/57a0cc8215

Above shows when pressing the back button, the display text shows no user logged in, fixing the error.

```
public class Logout : MonoBehaviour
{
    public static void LogOut() //username changed to null to log out
    {
        DBManager.username = null;
        Debug.Log("User Logged out")
    }
}
```

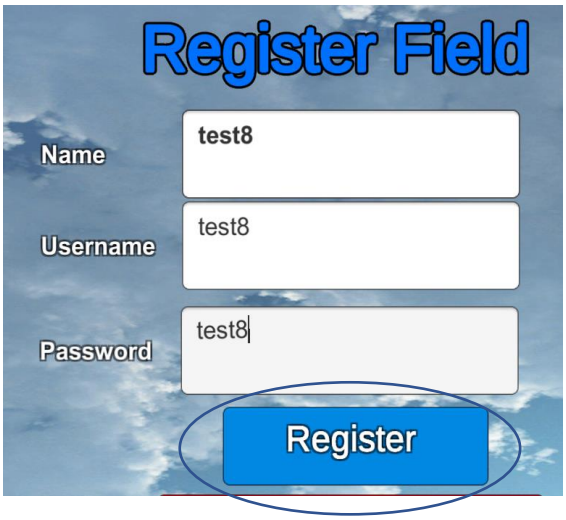
For a better user experience, I added a Debug.Log statement which provides this information to the user.

The user cannot login/register until the conditions for the username and password are met, the register/login button is not interactable

Below shows a clip of the login button being uninterruptable because the conditions stated on the left-hand side are not correct. Though the user is logging in, so they have already registered with these conditions being met, they are still there for a prompt. This functionality is completely the same for the registration menu.

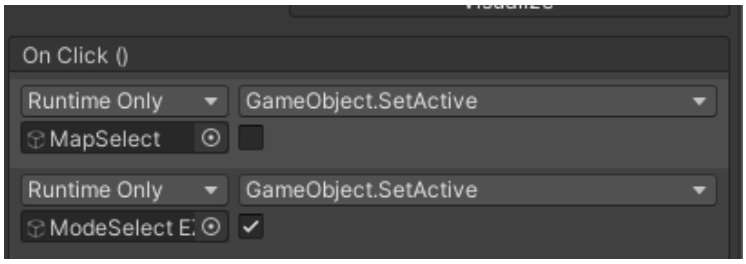
To make it easier for all users of different ages, there is a clear difference in colour between an interactable button and a button that isn't.

https://vimeo.com/696032137/c9fd7eaf20



Above shows screen shots from the register page with a clear color difference.

Each map selected and game mode selected produces the correct selection on the game display.



Each of the different map selects uses the same function, when the map button is clicked, this decision takes the user to a separate panel which holds the customizations of that game map.

Working example:

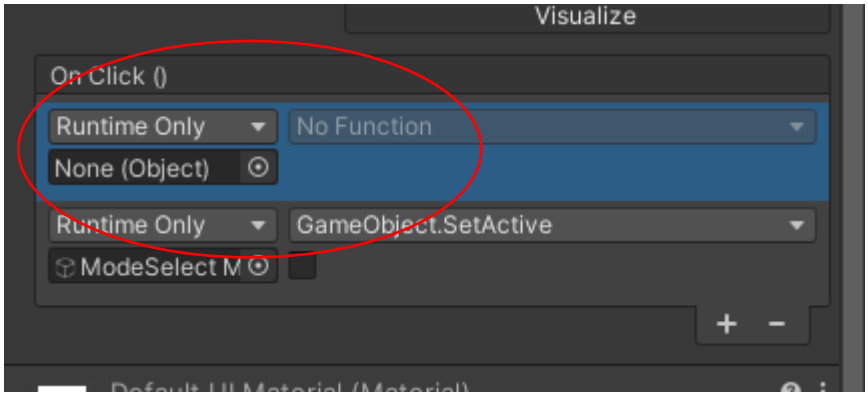
https://vimeo.com/696041727/75bb5de5c0

Error:

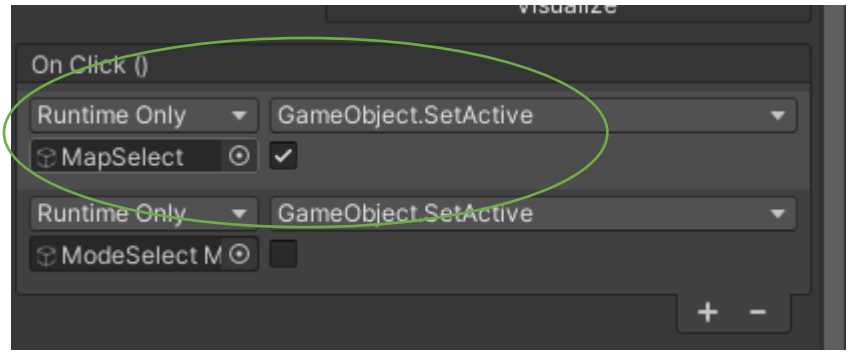
https://vimeo.com/696042111/47df5a83ab

When working I found this error: that when pressing the back button for the medium menu, the previous panel (map select) did not show but the medium map decision disappeared.

I realized that this was because of the back button in the menu was assigned to reactivating the previous panel.



The bottom element correctly disables the mode select medium game map, but the top element is unassigned. This may be because an object was removed from its original position and therefore the game could not find it through the previously established path.



To change this, I reassigned the map select panel and it worked.

The rest of the map selects worked well without errors.

Testing Game mode Selects:

By testing the game mode selects, I was only able to constructively test as there was only 1 input which allows the user to click.

Below is the process of the correct calling of a game map, as I previously said, there is only 1 input and that is a button click on screen, so I will repeat this test for each map.

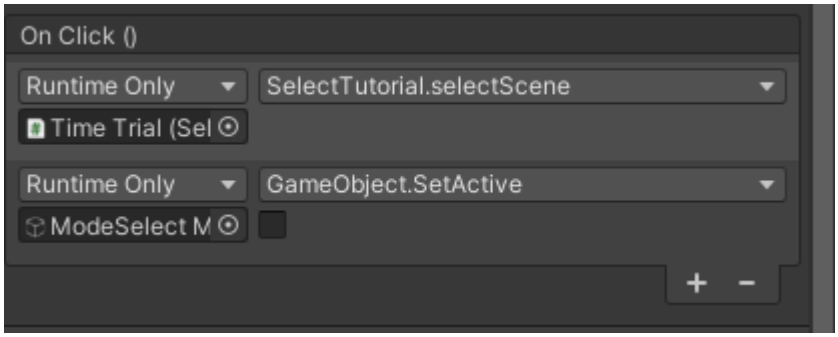
https://vimeo.com/696058903/230cb87653

Below is an error in the calling of the map in the medium game mode – time trial, the map did not call, and the screen was left blank.

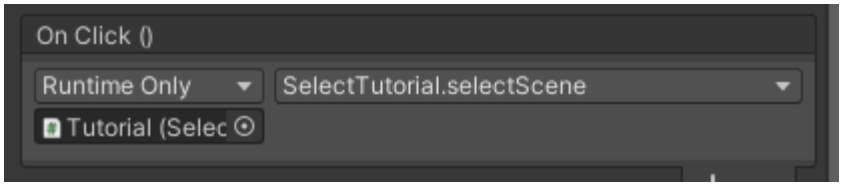
https://vimeo.com/696061826/b9622fa97e

```
6   public class SelectTutorial : MonoBehaviour
7   {
8       public void selectScene(){
9           switch (this.gameObject.name){
10          case"Tutorial":
11              SceneManager.LoadScene("DT - Tutorial");
12              break;
13          case"Free Play":
14              SceneManager.LoadScene("DT");
15              break;
16          case"Explore":
17              SceneManager.LoadScene("DT - Explore");
18              break;
19          }
20      }
21  }
```

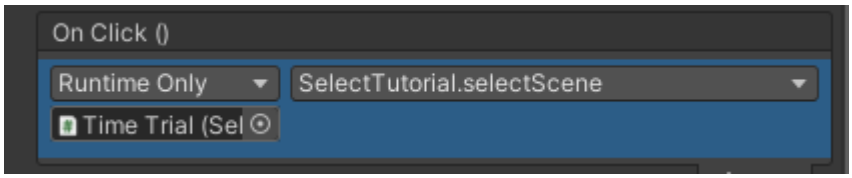
Here is the code, nothing seems to be wrong with the code so it must be the game object.



I noticed that time-trial disabled the Mode Select Menu, when none of the other's buttons had this, for example: 'Tutorial':



I edited the GameObject, and tested again:



https://vimeo.com/696064416/33f72ac167

Still the game map was not being called: I then added Debug.Log statements to the scripts.

```
public class SelectTutorial : MonoBehaviour
{
    public void selectScene(){
        switch (this.gameObject.name){
        case"Tutorial":
            SceneManager.LoadScene("DT - Tutorial");
            break;
        case"Free Play":
            SceneManager.LoadScene("DT");
                Debug.Log("Calling time trial map");
            break;
        case"Explore":
            SceneManager.LoadScene("DT - Explore");
            break;
        }
    }
}
```

When clicking the button, statement did not come up, so I know the function was not being called.

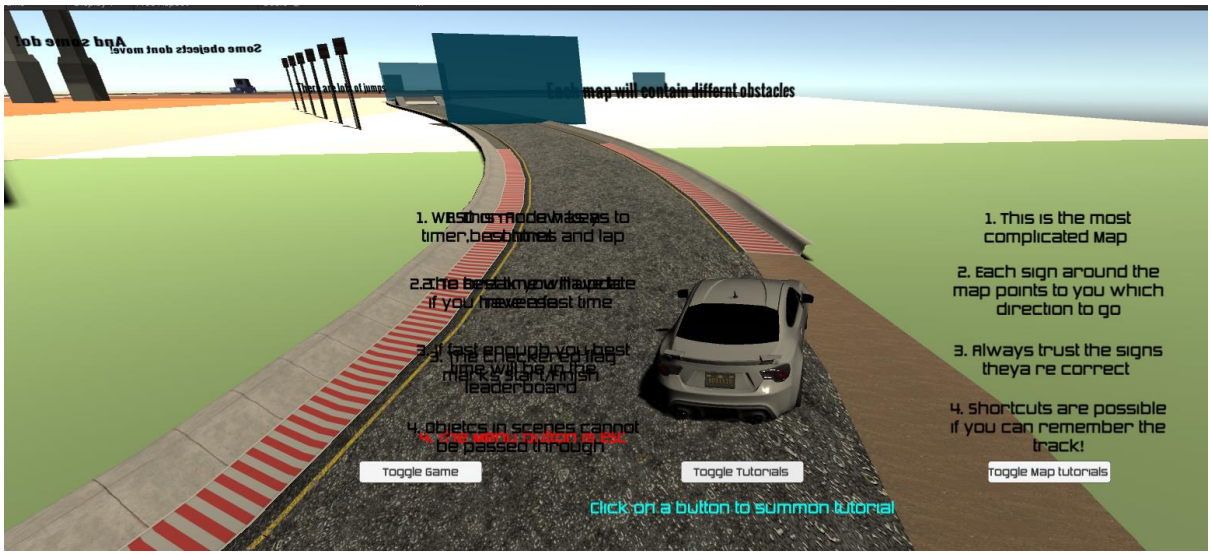
I then realized it was because the case was called ‘Free Play’ and not ‘Time Trial’ labeled on the button.

```
public class SelectTutorial : MonoBehaviour
{
    public void selectScene(){
        switch (this.gameObject.name){
        case"Tutorial":
            SceneManager.LoadScene("DT - Tutorial");
            break;
        case"Time Trial":
            SceneManager.LoadScene("DT");
                Debug.Log("Calling time trial map");
            break;
        case"Explore":
            SceneManager.LoadScene("DT - Explore");
            break;
        }
    }
}
```

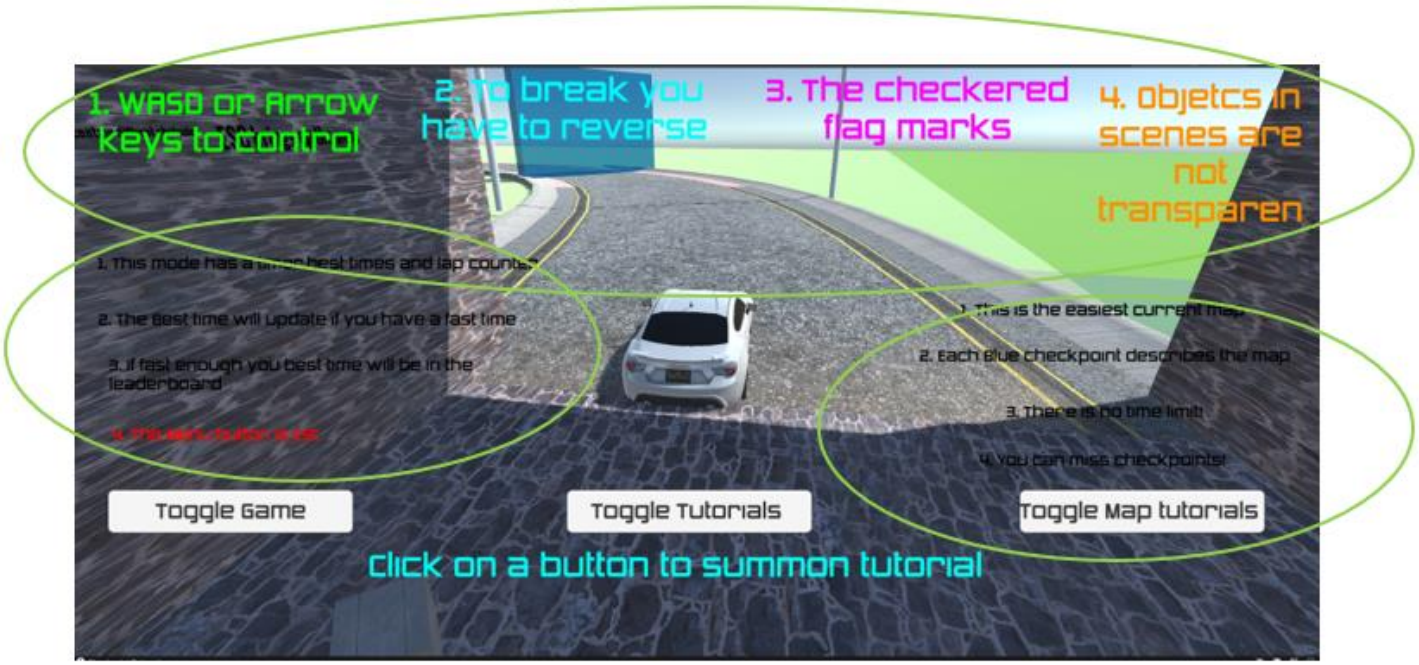
I then tested the code, where the scene was called correctly, the rest of the calls are working correctly now.

https://vimeo.com/696066262/064c1c9b4b

User tutorials are toggleable, for the game map, mode, and controls, they can be present on and off screen from a button.



Here is a screen shot of the easy map tutorial which shows some overlapping of the walk-through text which is not good for a user experience. Aswell as this, the right-hand side text is not accurate for the map; this is meant for the final side of the map. In the green areas, I have updated the color



so that they are more easily observed to the user. Then I moved the texts so that if they were all active at the same time then they would not over-write each other and could be read at the same time.

To destructively test, I have played the game on not full screen, all the texts still scaled to screen size and worked okay. This was tested on the medium difficulty map.

Racing game mode starts the timer once the game is started and is reset once the user enters the finish lap checkpoint.

First, I am checking that the timers check at each of the maps.

Here is the medium map:

https://vimeo.com/696173724/e8755913d1

Here is the easiest map:

https://vimeo.com/696175638/a508ac14ae

Here are two examples of the timer starting at the start of the map. Th

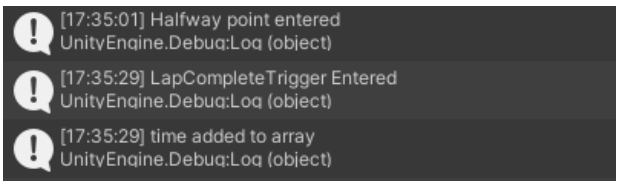
Finishing lap checkpoint restart timer:

https://vimeo.com/696187436/91dd971ddc

Each lap time is input into an array of times on the Time Trial Map where the best time is produced onto the screen.

```
void OnTriggerEnter()
{
    //entering the trigger
    Debug.Log("LapCompleteTrigger Entered");
    LapsDone += 1;


    //adding the score to the aray
    if (ScoreTimeManager.elapsedTime != 0)
    {
        bestTimes.Add(ScoreTimeManager.elapsedTime);
        Debug.Log("time added to array");
    }
```

Here the code facilitates the use of the ScoreTimeManager variable which prints the time added to the array of bestTimes.



Here the time is added to the array on the trigger enter.

https://vimeo.com/696193795/4f40922806

Destructive testing:

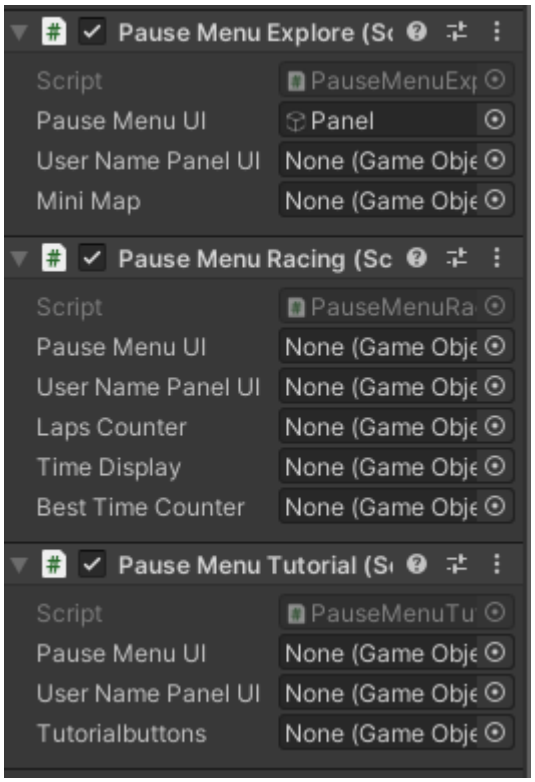
Here is evidenced that the best time is produced on screen. The first time around 30s was currently the best time, then the second time which was a minute was counted as a time and added to the array but was not printed to the best time once sorted.

The ESC button functionally calls the menu, each button on the menu is functional.

There were some problems with the game being paused and that was with the size of the pause menu, and some of the GUI elements not being disabled.

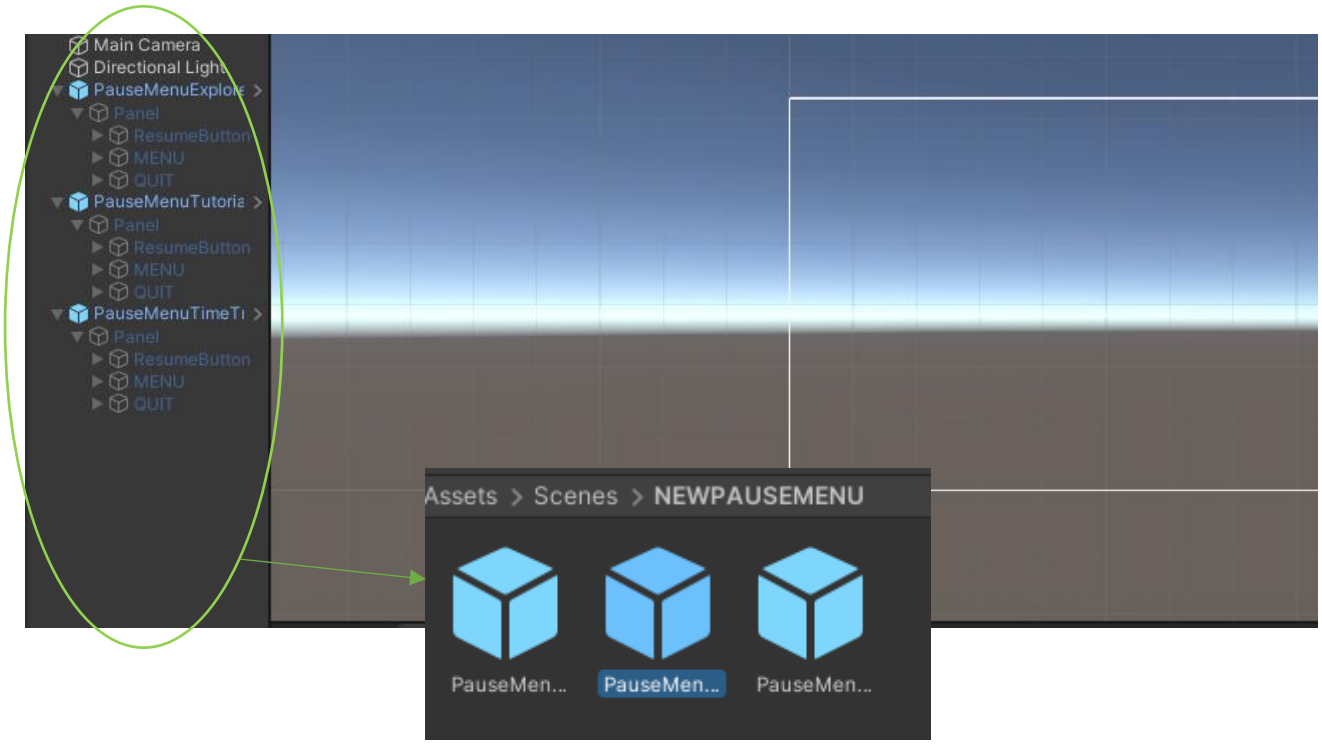
Here is a screen shot of the GUI not loading fully, not disabling the countdown when called and the current time is still displayed on the screen while the other GUI elements are disappeared.

To fix this error across all maps; I created a new prototype with 3 different scripts:



For each of the different game modes; I created a new script which enabled and disabled different GUI menus for a clean look, then applied this to all of the related maps for that category of game mode.

In a prototype scene I then created 3 new game objects to test them, I made sure that each filled the entire menu and testing them to disable the current GUIs.



https://vimeo.com/696209856/c6ccbfdf28

Here is an error; the pause menu activated correctly but then did not correctly re-start the count down menu.

This was fixed by applying the countdowntext, instead of the panel it was on. As the panel did not reenable the script when disabled which controls the countdown text, but the actual text object does.

https://vimeo.com/696212158/d1af1608ab

**Clip starts at 8s the computer was slow

Error:

While destructive testing, I opened a panel while opening the pause menu to test whether it was working correctly, and it did not disable the tutorial. which does not look good and can confuse the users.

To fix this I added another 3 game objects in the tutorial script (game, map, controls) and enabled them and disabled them so they would disappear with the pause menu.

Here are the added game objects in green:

```
public class PauseMenuTutorial : MonoBehaviour
{
    public static bool GameIsPaused = false;
    public GameObject pauseMenuUI;
    public GameObject UserNamePanelUI;
    public GameObject tutorialbuttons;
    public GameObject CountDownDisplay;
    public GameObject GamePanel;
    public GameObject MapPanel;
    public GameObject TutorialPanel;
```

```
44          // elements being enabled
45          UserNamePanelUI.SetActive(true);
46          tutorialbuttons.SetActive(true);
47          CountDownDisplay.SetActive(true);
48          GamePanel.SetActive(false);
49          MapPanel.SetActive(false);
50          TutorialPanel.SetActive(false);
51
52      }
53
54      void Pause()
55      {
56          // elements being enabled
57          pauseMenuUI.SetActive(true);
58
59          Time.timeScale = 0f;
60          GameIsPaused = true;
61          Debug.Log("Game Paused");
62
63          //elements being disabled
64          UserNamePanelUI.SetActive(false);
65          tutorialbuttons.SetActive(false);
66          CountDownDisplay.SetActive(false);
67          GamePanel.SetActive(false);
68          MapPanel.SetActive(false);
69          TutorialPanel.SetActive(false);
70      }
```

Elements are kept false in the Resume() function because this disables them after the menu is Resumed() if they was set to true they would all be enabled after resuming the game.

Once the user has completed 3 maps, the finish game screen opened, and the user can see their best time and username.

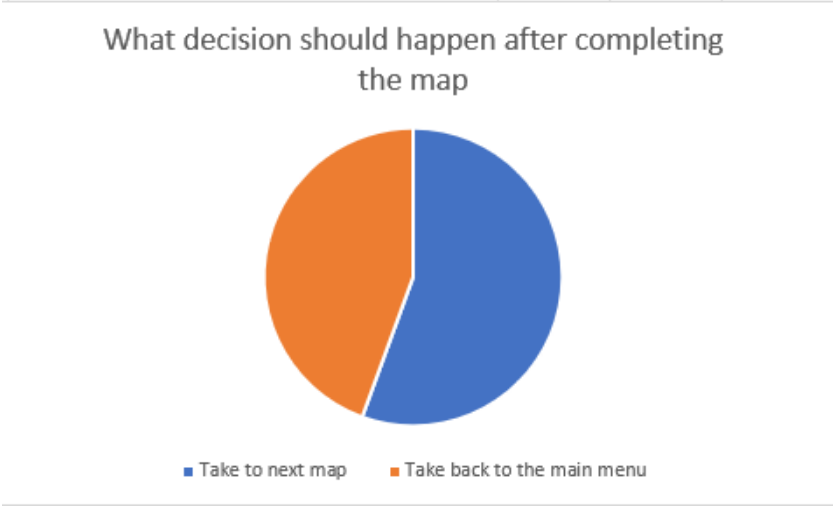
I am testing the finishing 3 laps on each of the maps through mimicking the user's experience. This starts by entering the finishing checkpoint on the map and following the automatic GUI menu and single options given to the user.

I have tested this by finishing 1 lap, so I did not have to complete 3 every time: the time and . username display correctly on the clip (I am not currently logged in), but the save data function does not do anything currently. When being pressed.

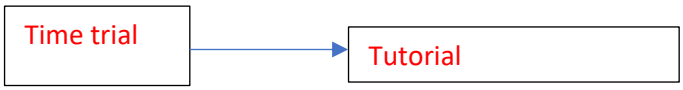
The final score display works perfectly fine: (example of being logged in)

https://vimeo.com/696284189/d22ed1615d

However, the save data feature does not do anything:



Here I am going to take the player to the next consecutive map.




```
public class Movingtomenus : MonoBehaviour
{
    // EZ time trial
    public void EZTimeTrial()
    {
        SceneManager.LoadScene("EZ - TimeTrial");
    }

    // DT Tutorial
    public void DTTutorial()
    {
        SceneManager.LoadScene("DT - Tutorial");
    }

    // DT TimeTrial
    public void DTTimeTrial()
    {
        SceneManager.LoadScene("DT");
    }

    // Hard Level Tutorial
    public void HLTutorial()
    {
        SceneManager.LoadScene("HL - Tutorial");
    }

    //Hard level TimeTrial
    public void HLTimeTrial()
    {
        SceneManager.LoadScene("HL - TT");
    }
}
```

This script Changes scenes to the next progressive difficulty. This is applied to the appropriate button at each level.

https://vimeo.com/696294489/eac4b4f39f

Here is evidence of the next level button.

There is also a move to next map feature that has been applied to the end of every TimeTrial Map – when the Green final score panel appears:

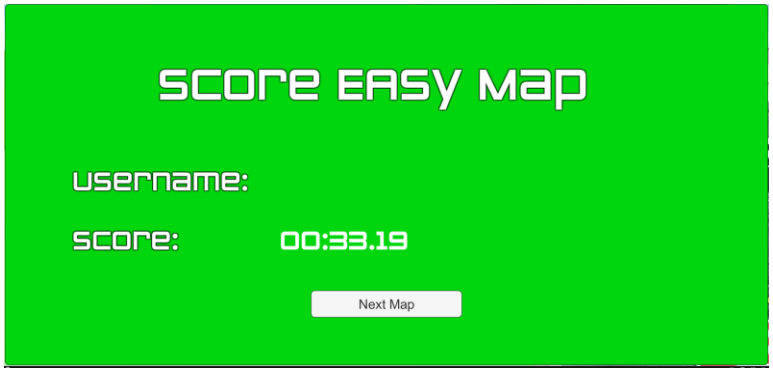
Below is an example of the button in action inside of the tutorial map.

At the end of the time trial game mode the leaderboard menu is opened and displayed.

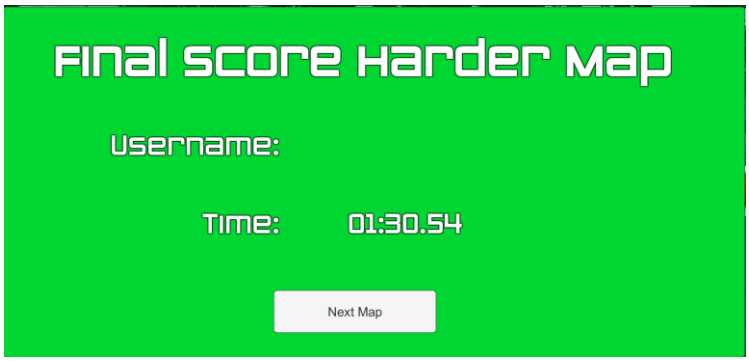
At the end of every time-trial game mode there is a Green final panel screen which displays the user's information and their final score. There is an option to move onto the next level for the first two maps, and the final map takes the user to the leaderboard section.

This section will be tested on each map that has a final score panel, and the leaderboard menu is displayed on the final (hardest level map).

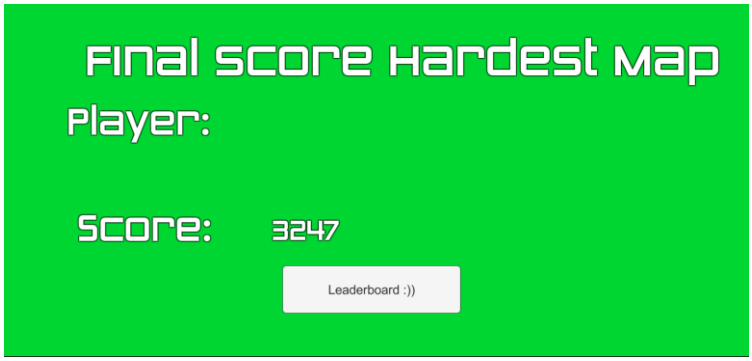
Score Easy map



Score harder map



Score hardest map



The car is controlled by WASD or the arrow keys.

Over the course of the entire testing sections the clips of the car moving inside of the game maps highlights that the car can be maneuvered using the arrow keys and WASD. WASD is more recognized, but arrow keys are also used naturally.

The car moves well with no problems.

Stage 12 – Stakeholder Testing

Now I am confident that there are no errors in the Final Game, I am now able to pass the game over to stakeholders for their final testing. The stakeholders will play the game how they would approach any other without background knowledge, this is for a raw user experience which highlights as many inconsistencies as possible.

I am going to ask my users to fill out a questionnaire for their feedback, and then the users will have an opportunity to give open feedback and any criticisms.

Questions:

How did you find the menu navigation experience?

How did you find process and customizability of the map and game mode?

How did you find each of the game modes with maps?

Was the functionality of the pause menu?

How was the leaderboard display? Did this work well?

Stakeholders:

I will be asking different demographics of stakeholders to get a wider representation/opinion of this game.

Daniel – Teenager user who plays games

Sophie – Younger user has never played a game before

Lewis – Teenager user who does not play a lot of games

I James Gammon – 27228 – 8292 – Abbeyfield school

Question answers:

How did you find the menu navigation experience?

Daniel: I found the user experience overall fluid and aesthetic. The game menus looked good with different colors, and these did not take away from the easiness of the game to navigate. I found that for, each color for the combination of game mode and map choices where different for each game modes which made the navigation of the menus easy especially with the large amount of choice. For logging in and registering, having the text small and at the bottom is really hard to see, if this text was displayed on screen it would be much easier and more prominent to read.

Sophie: Immediately the navigation of the menus was clear, and I had a good idea on what to do. The clear and large buttons where self-explanatory which meant I was able to open the options menu and register without getting lost or too confused. Because there was not a lot of decisions on screen this made it really clear on what to do.

Lewis: I really liked the menu's as they were clear, and I was able to navigate them without having a lot of precursor knowledge on overall game menus. What I found useful was the clarity of the logging in/ registering panels. The prompts on the screen were able to provide me with some clarity on what was happening, which really made it easier.

How did you find process and customizability of the map and game mode?

Daniel: I thought that the process was straight forward which made it easier for me. It followed the layout and process of a lot of other games on the market and their menu layouts, functions which makes adaptability to a new game much easier.

Sophie: I thought that the menu customizability was okay, there was 3 clear decisions on the screen, so it was self-explanatory on what I had to do. There could be a little note by the side of each map and game mode decision to say what that includes.

Lewis: I was able to grasp well on what to do, it was not overcomplicated from the start with clear sequential steps in picking my desired game customization, which was a good user experience. However, I would have liked the game to appear more like a car racing game as I couldn't tell it was only from the title.

How did you find each of the game modes with maps?

Daniel: The customization of different game modes and game maps worked well, I liked that you could have a time-trial, free play, and tutorial all on the same map I was able to grasp the mechanics of the game well. As this was displayed over the course of 3 different maps, I felt like there was a high ceiling for customization and enjoyment.

Sophie: I don't play a lot of these types of games, so I chose the tutorial map. I really liked that the buttons displayed on screen, and they stayed there once clicked so I could take my time. Calling the buttons worked well on all the maps.

Lewis: The game modes and maps were very different. I liked that each game mode can be carried out on all the maps. I especially liked the medium difficulty map, I liked how it subtly increased in difficulty and I didn't have to go back to the menu all the time to change map, this could be done in game.

Was the functionality of the pause menu helpful in the game?

Daniel: Yes, the functionality was very useful to me, when I first used the pause menu it logged me out when clicking back to the menu, but when I logged back in and used it a second time it worked

well. I think that the button to switch maps progressively as well was a good idea as it proves clear indication on what to do.

Sophie: I think that the pause menu was a good addition. At first when on the tutorial game mode, I did not know how to move onto the next map, then I checked the tutorial again and it was evident I needed to use the pause menu. This made a lot of sense and took me back to the map selection point.

Lewis: I think that using the pause menu to navigate the game customization was a good idea. It enables me as the user to navigate well inside of the game and allows me to pick and re-pick different maps and customs which is what I expected from the game. The pause menu worked well on each of the different maps and game modes. I liked it especially when it disabled all the GUI features available in the different game modes; the screen wasn't cluttered and made the game feel a lot cleaner.

How was the leaderboard display? Did this work well?
Daniel: At first, I did not know only the last map inputs scores into the database, so I was confused when I first visited this. Otherwise, I liked having the scores of previous users and there usernames inside of the leaderboard, it makes it look very professional.

Sophie: I thought that the leaderboard aspect was a good idea to show where the users scores go at the end of the game and adds an aspect of competition to see who can get the best scores. The Leaderboard was fluid to access from the initial start menu.

Lewis: The leaderboard menu was a good idea as it adds a final finish point in the game like you are working towards something progressing up each of the game maps. The leaderboard was successful as my fastest score on the hardest map was uploaded to the leaderboard in the correct position.

Evaluation

Changes made to the game after stakeholder feedback:

Game appearance:

I took Lewis's feedback and thought about how I could make the game more stylized like a racing game.

Here I added a new 'car themed' background.

Here are some examples of the improved GUI. It looks more like a low-poly racing game now, meeting user expectations.

Here shows the

- Leaderboard
- Login panel
- Final score panel

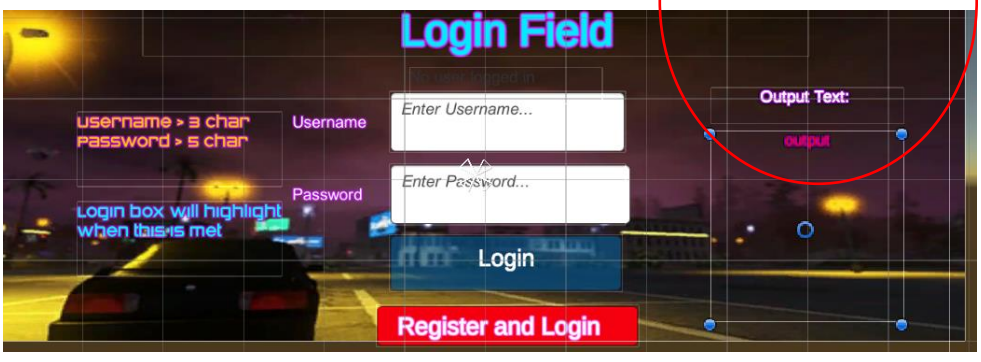
Logging in and registering text on screen:

To improve this, I created a blank public text placed this on the side of the login/registration screen, this blank text was made a game object where I would then be able to upload error messages for the user to see much clearer.

```
public Text Output;
```

```
41          else
42          {
43              //Adding text so the user can read more clearly
44              Output.GetComponent<Text>().text = "User creation failed. Error #" + www.text;
45              Debug.Log("User creation failed. Error #" + www.text);
46          }
```

On the else routine if everything does not align, the error message is now printed to the Debug.Log statement as well as the new text game object. This code was added both to the login/registration scripts.



Here I added the plane text box which is currently blank, when an error message appears it will show here clearly for the user to see. I only need to create this box for errors because if the user passes through successfully, they are advanced. Below is evidence and feedback.

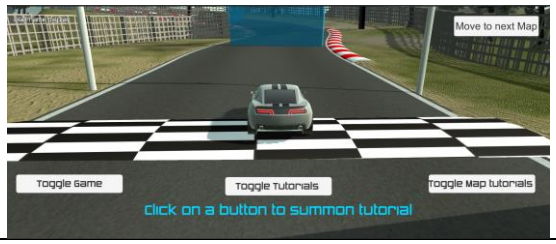
https://vimeo.com/700019253/a161869609

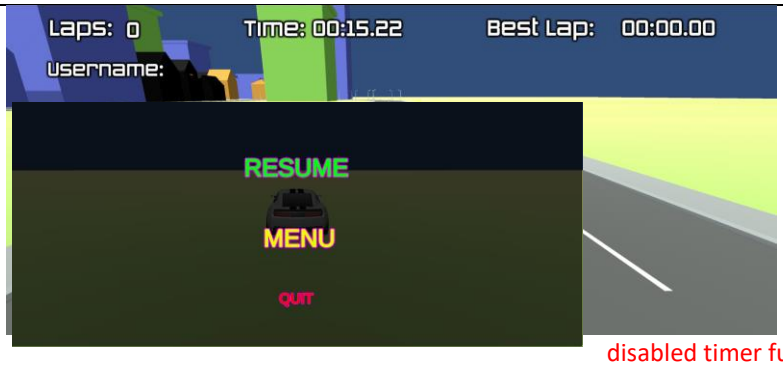
Have the success criteria been met?

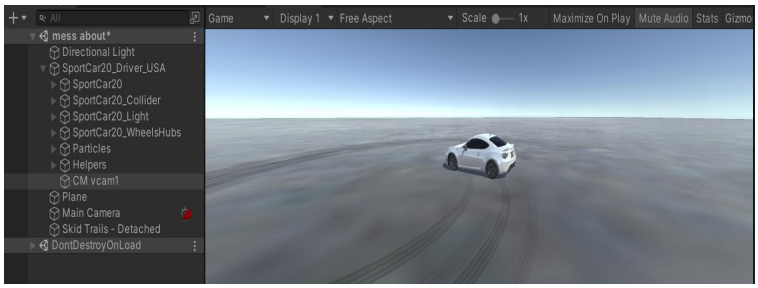
Criteria	Has this been met?	Evidence:
Large buttons on each decision, simple decisions per screen or decisions are defined in the walkthrough area.	Y This criterion has been fully met. My stakeholders showed positive feedback on my menu GUIS.	
Simplistic design which is usable with the varied demographic (abstraction) of items which can target one or the other. This also includes non-confusing navigation panels	Y This criterion has been fully met. My stakeholders commented on the clear and defined paths in the game. Especially from the non-gamers.	
User input text boxes are clear and know when to be interacted with on the user authentication page	Y Originally this criterion had only been part met, however feedback from players encouraged to create a easily identifiable and clearly read output and text boxes	https://vimeo.com/700236755/f90fc4f96a
Option to call menu pause menu in game which displays functions such as: Resume game Main Menu Quit and save	Y The Pause menu completes this functionality and has fully met the success criteria. It was very popular with the stakeholders.	https://vimeo.com/695450949/fda6a2c757
Subroutines across the GUI are called without fail	Y This success criteria had been fully tested and is fully met. In each stage (especially the first and second) the routines are tested with prototypes and debug.log statements.	https://vimeo.com/700238050/6dc7b5f0af

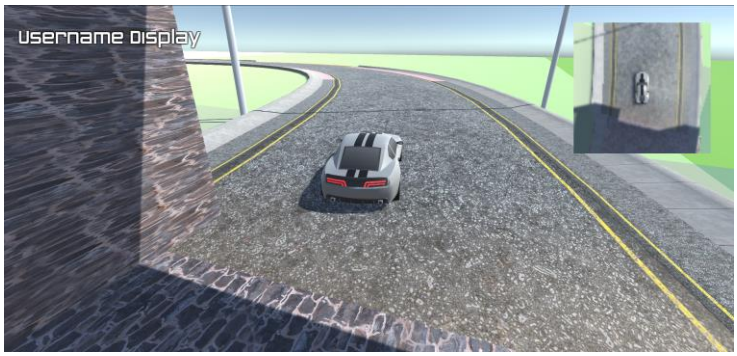
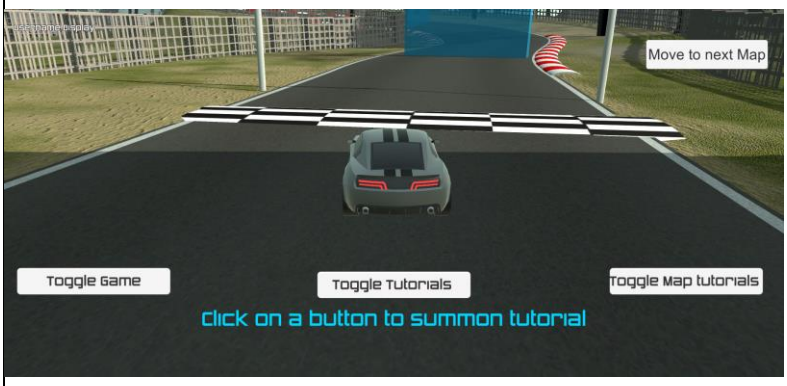
I James Gammon – 27228 – 8292 – Abbeyfield school

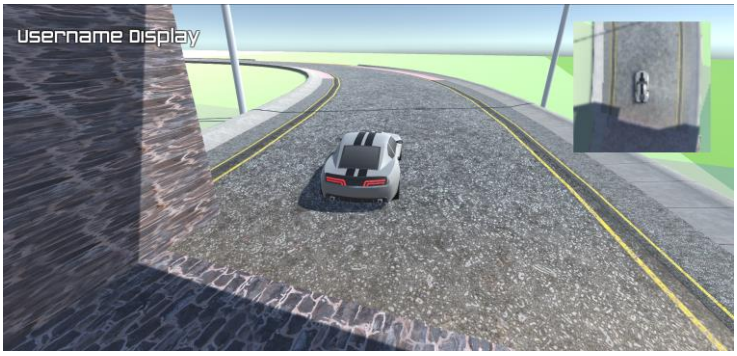
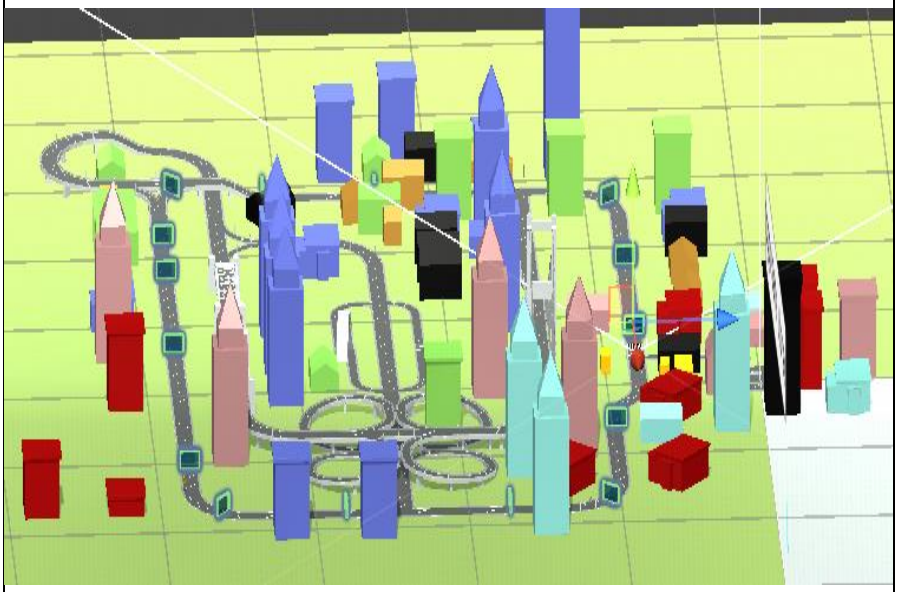
Success criteria (Game play)

Option to change game map/mode in game //done by calling back to the map select function in the game menu which leads to the game mode	Y This was met originally; however, it was tedious to always travel back to the main menu. From feedback I improved on this criterion by adding on a button on the GUI advancing the user	Evidence: Next level button: (from game) https://vimeo.com/696294489/eac4b4f39f
Leaderboard is displayed to the user in game – this uses the users fastest time scores	Y This criteria has been fully met and is integrity, it can handle a lot of input data laps.	https://vimeo.com/700273333/cf9b08cdd0 Above shows the best user time displayed on the fastest time scores inside of the game.
Each of the game modes are compatible with each of the game maps, meaning I can play a time trial on each map, a tutorial on each map and a explore function on each map.	Y This criteria has been fully met, there was 9 different combinations of map that can be played by the user. On the right, displays the 3 different possibilities of the harder scene select.	 

		
Game mode functions are specific to the chosen game mode, so modes do not dissolve into each other	Y Each game mode has the same fundamentals and layout as 'UI at the top' and benefitting the user's experience. However, each game mode is completely different from the others.	Above on the 3 different levels, shows the differentiation of the maps. The timer, the buttons, and the game map. Each with a specific benefit to the user with no-self similarities.
Stopwatch clock presented on the screen	Y The stopwatch is successfully printed on the screen in all the time-trial game modes. It disables and enables with different GUI. This criterion has been fully met.	 Here is the disabled timer function which disables the ongoing clock. Fulfilling the success criteria.
Vehicle is completely controllable with arrow keys	Y This criteria's been fully met, the car	https://vimeo.com/700910099/114d43aa6b

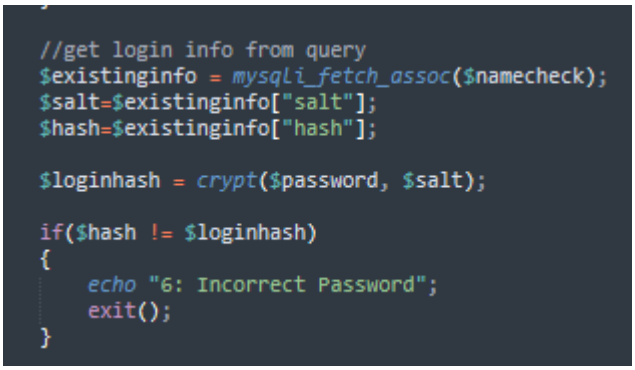
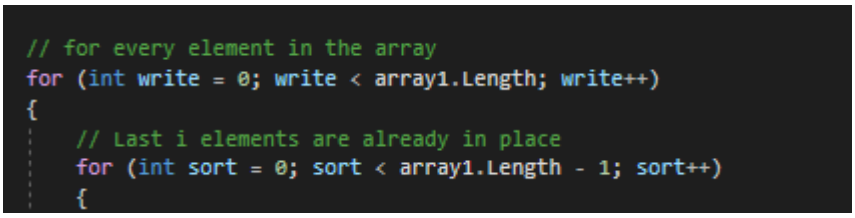
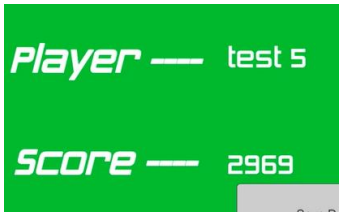
	is also controllable with WASD.	
When the vehicle crosses the finish line the timer/stopwatch resets for a second lap	Y This criteria has been met fully and has been tested by all the stakeholders.	https://vimeo.com/700918561/6a41fca694 Here the current timer was 4:14, resetting after the lap is completed.
The car game object can drift in and around corners	Y This criteria has been part met. The car does not drift round corners intestinally on every go and is variable on the type of corner; this can't be helped.	 Here is a screen shot from car testing clearly showing the cars ability to drift around corners
Car game object has wheel colliders meaning the car does not get stuck in the surrounding environment if it crashed	Y This criteria had been fully met, the cars wheels are colliders with a rigidbody, meaning they do not get stuck.	https://vimeo.com/700915619/1aedad9135 Above shows the car not being stuck when on a curb.
Best time label presented on screen	Y This criteria has been fully met in the time-trial game mode.	https://vimeo.com/700918561/6a41fca694 The best time screen is printed on the top left, the best time remains there if it is still the fastest.

Each map has its own specific functions, unique to it.	Y Each of the maps are incredibly different, the Harder maps and easy maps are extremely well to differentiate because of the characteristics of the course and surroundings, such as the topography and jumps.	Screen shot from explore  Tutorial Map  Time Trial 
Time trial game mode accounts for the user's current game time, the users fastest lap time so far, and displays both values on screen with the addition of lap counter.	Y The criteria have been fully met well; each value is clearly defined on the top of the users screen.	

Explore game mode includes no parameters for time completion or a set rule to follow.	Y This criterion has been fully met. I have surpassed these criteria and added a mini-map function which differentiates it from the other games modes. Left is a screen shot from testing	
Tutorial game mode will display checkpoints throughout the map and current tutorials on each game aspect.	Y This criteria has been fully met and improved with varying stakeholder feedbacks.	

I James Gammon – 27228 – 8292 – Abbeyfield school

Success criteria (Database management)

Criteria	Has this been met?	Evidence:
Validation of user inputs compared to those stored in the database This includes a hash function for the password to compare the password entered to the user (hashing this password with the same characters) and comparing this one to the one stored in the database	Y This criterion has been met and fully tested using different salts and hashes, the process works like other hash functions by hashing the input password and then a comparison is made.	 Here is the code facilitating the criteria. The hash is retrieved from the DB from existing info.
Leaderboard sorted via bubble sort	Y This criteria has been fully met as the leaderboard has been sorted with an ascending bubble sort.	 Here is the code facilitating the bubble sort. It functions self-similar to a bubble sort, however, sorts the fastest at the end which is then displayed at the top.
The most recent user score is uploaded into the database.	Y This criteria has been fully met, when accessing the players database, the most recent score	https://vimeo.com/695818644/42aae6cbe0 Here is a video from a previous test.  

	can be seen there.	

Usability is evaluated more effectively, although it would have been nice to see some discussion of the end user comments linked in or referenced to reinforce the statements made.

The limitations and maintenance sections are well-written and address future developments and improvements

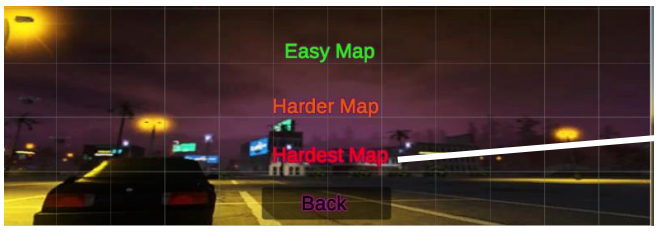
Usability features:

As shown in the screen shot the buttons are very large and clear cut defined which helps the user identifying the button. The screen menus are easy to navigate as colors for important buttons are labelled in a visible color.

The buttons take up a large portion of the screen.

The design of the GUI menus is consistent throughout shown in the background which makes easier navigation.

On different game modes the colors of the menus are blue for time trial and green for map select for example, this helps the user knowing what decisions they have made.



There are clear instructions for game navigation inside of the tutorial game map, this helps the users get the bearing for the game. It includes knowing about the pause menu, and how to navigate the game.

Playing the game is very easy, once the user finishes the game, they are given a single decision game mode to move onto the next map (Or leaderboard), the user does not have to do anything technical for this process as it is already completed for the user.

Limitations:

One of the biggest limitations is that the users cannot play against other users as there is no multiplayer feature. As the game is made on unity it ran on a local machine and is not hosted on a server so users cannot be inside of the same game, the closest way a user can get to this multiplayer as if the game is available on 2 machines and each running the game and comparing manually.

In the future, I can transform the game to an online server; this would substitute for the need to run MAMP as a localhost database and allow the leaderboard to take input scores from a range of different machines displaying the true fastest scores.

Another limitation of the game is that there are only 3 maps, and each map repeats 3 different game modes which can be repetitive for the user if they don't find the characteristics of the different game modes entertaining enough and find the maps to become repetitive. Though this was done specifically as it accounts for a wide variety of users playing the game, this does not fully account for experienced game players who may become too familiar with each of the maps.

And finally, to use the software, you need specific environment, you must have MAMP (and be logged into) the Unity Database which causes safety implications as users' authoritative position over users database; some players may not be able to download MAMP on their machines as hosting online databases takes up a lot of RAM. Finally, the players must have received the game files by me to play the game, meaning it is not open for users to play, once the users have received the game files, they can change the game in the unity inspector as there is no current way to make the application closed source.

Overcoming limitations:

To kind of tackle the limitation of no multiplayer an AI car can be initiated inside the game to add a competitive aspect to the user. This could be a separate game feature that the users can apply to any of the maps, increasing the total customization to 12 different combinations. Although this does not account for the lack of multi-player some users may desire it does still enforce that competitive nature. Aswell, users who want to compare scores and play together can always play on the same machine, logging out is a feature which enables different users to play on once local machine under different registrations. Adding a real multiplayer function is not feasible right now as there is not enough users playing the game.

The limitation of 3 maps can simply be overcome by designing more. Firstly, users may play the free play arena once, the tutorial once, and the time-trial 3 times in a sitting which is 5 per game map. Times this by 3 maps there are 15 'goes' at the game before the user may stop playing. The game currently is targeted at a very small demographic and therefore currently exceeds that. In the future with the growth of the game, if players demand for more game maps increases this can easily be accounted for.

To overcome the specific environment setting I can load the game onto an online cloud server which makes the game accessible to play over multiple-devices which unified access to the leaderboard.

With this, the game can be closed source (the fact it won't be able to change the game constructs) and be released on a game download software.

Improvements for the future:

There is a very high development ceiling for first person racing games as there are so many currently in the world right now and each adds its own unique taste. As players are getting better at racing games and more people are venturing out into racing improvements primarily can be made to make the game harder through:

- Adding difficult maps to overcome
- Adding a time limit to some of the maps; if the user does not complete 3 laps within the time limit, then they fail the map.

Characteristics like this are relatively easy to implement and can really boost the games functionality.

Another improvement mentioned earlier is the addition of an AI car performing as an extra 'level' for the user to overcome after advancing the time-trial game mode. An AI version can be programmed to have different levels of difficulty for the player depending on there skill level. The AI could work in 2 ways.

- It could be used to educate inside of the tutorial-game maps as a 'ghost' displaying to the user the fastest way to drive around the track. If the user strafes to far away from the AI this distance can be monitored, and the AI will slow down/stop.

- The second is after the time-trial game mode. Depending on the time users complete the final lap equates to the difficulty of AI they experience in the next level. A faster time (under a certain threshold) would lead to a AI which has a faster speed around the track compared to one which was slower (paired with a user who finished after the threshold)

Maintenance:

The greatest limitation of the game currently is its inability to facilitate multiple users wanting to play together across different machines. The program is run on a local host basis which prevents interaction of; 2 users using the game at the same time on the same machine, 2 users using different machines wanting to access the same game. If users want to access the same game, they cannot. Solving this in the program I have specified to the user that this is a single player practise game mode, and there is no need for multiplayer games.

Future maintenance could begin to change the dynamics of the game where it can become more competitive orientated, including the original focus of a different game mode racer to gain experience, it can evolve to orientate more around racing other players competing for the best time score. This can be run from an online server which oversees the current MAMP server hosting the leaderboard, and the connection to the game can be hosted from the same online game server instead of the local machine.

It could add new features such as timer limits for racers, leaderboard for each of the racing map and a competition of a 3-game map sprint for the for players.

The stakeholders also have a large part to play in the future of the game, this decision includes whether to maintain the game in its current state or to orientate around a new focus. As each of the game mode UIs are prefabs these can be applied to a range of maps and are modular in designed – game modes are self-contained. The menu also follows procedures which are heavily repeated throughout the game allowing the total game to be easily expandable.

The software has been designed in a range of different programs but each of these programs are self-explanatory and have a mono-function which can be understood relatively easily. Scenes and game modes can be easily duplicated to account for additions of an AI for example, just by duplicating the scene the assigned scripts and game components are mirrored also.

For the code, it is annotated heavily throughout especially in the more complex parts such as the leaderboard and the time-trial aspects, therefore they are easily maintainable if they are then re-visited after a long-time.

Overall, the game is very easily expandable because of its modular design and is currently at a pivot point where it can focus into a new direction without leaving the prior ‘versions’ of the code to be behind and outdated with new features. The game can be built onto with new features incorporated into each map and the GUI, for example added features of the pause menu can be included and a respawn point in maps can be added.

Final code

Stage 1: Creating the database

Stage 2: Registration

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



public class RegistrationSQL : MonoBehaviour
{
    public InputField nameField;
    public InputField passwordField;
    public Button submitButton;
    public GameObject GameMaps;
    public GameObject Registration;
    public Text Output;

    public void CallRegisterSQL()
    {
        StartCoroutine(Register());
    }

    IEnumerator Register()
    {
        //creating form and posting this to the surver
        WWWForm form = new WWWForm();
        //takes inputs from the text fields username and password
        form.AddField("username", nameField.text);
        form.AddField("password", passwordField.text);
        WWW www = new WWW("http://localhost/sqlconnect/register.php",form);
        //holds the function until rest of data is recieved
        yield return www;
        //if everything in the php script
        if (www.text == "0")
```

```
        {
            DBManager.username = nameField.text;
            Debug.Log("User created successfully");
            Output.GetComponent<Text>().text = "User created successfully";
            GameMaps.SetActive(true);
            Registration.SetActive(false);
        }
        else
        {
            //Adding text so the user can read more clearly
            Output.GetComponent<Text>().text = "User creation failed. Error #" + www.text;
            Debug.Log("User creation failed. Error #" + www.text);
        }
    }

    //verifying inputs from the text fields called by the register button
    public void VerifyInputs()
    {
        submitButton.interactable = (nameField.text.Length >=3 && passwordField.text.Length >= 5);
    }

}
```

I James Gammon – 27228 – 8292 – Abbeyfield school

Registration php:

```php
<?php

$con = mysqli_connect('localhost', 'root', 'root', 'unityaccess');
    // check for successful connection.
    if (mysqli_connect_errno())
    {
        echo "1: Connection failed"; // Error code #1 - connection failed.
        exit();
    }

    //conveting to php variables
    $username = $_POST["username"];
    $password = $_POST["password"];

    // check for if the name already exists.

    //selects username column from database, obtains all stored usernames
    $namecheckquery = "SELECT username FROM players WHERE username='" . $username ."';";

    $namecheck = mysqli_query($con, $namecheckquery) or die("2: Name check failed - a username with this name already exists"); // Error code # 2
    - name check query failed.


    if (mysqli_num_rows($namecheck) > 0)
    {
     echo "3: Name already exists"; // Error code #3 - name exists already.
     exit();
    }

    $salt = "\$5\$rounds=5000\$" . "safetycharacters" . $username . "\$";
    $hash = crypt($password, $salt);
    $insertuserquery = "INSERT INTO `players` (`id`, `username`, `hash`, `salt`, `score`) VALUES (NULL, '" . $username . "', '" . $hash . "', '" .
        $salt . "', '0');";
    mysqli_query($con, $insertuserquery) or die("4: Insert player query failed"); // Error code #4 - insert query failed.

    echo("0");

?>
```

Stage 3: Login

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


public class Login : MonoBehaviour
{
    public InputField nameField;
    public InputField passwordField;
    public Button submitButton;
    public GameObject GameMaps;
    public GameObject LoginScreen;
    public Text Output;

    public void CallLogin()
    {
        //starts coroutine to register user
        StartCoroutine(LoginPlayer());
    }
    IEnumerator LoginPlayer()
    {
        WWWForm form = new WWWForm();
        form.AddField("username", nameField.text);
        form.AddField("password", passwordField.text);
        WWW www = new WWW("http://localhost/sqlconnect/login.php",form);
        //holds the function until rest of data is recieved
        yield return www;

        if (www.text[0] == '0')
        {
            DBManager.username = nameField.text;
            DBManager.score = int.Parse(www.text.Split('\t')[1]);
            Debug.Log("Logged in");
            GameMaps.SetActive(true);
            LoginScreen.SetActive(false);
        }
```

```
        else
        {
            Debug.Log("User login failed. Error #" + www.text);
            Output.GetComponent<Text>().text = "User creation failed. Error#" + www.text;
        }
    }
    public void VerifyInputs()
    {
        submitButton.interactable = (nameField.text.Length >=3 && passwordField.text.Length >= 5 );
    }
}
```

I James Gammon – 27228 – 8292 – Abbeyfield school

Login.php

```php
<?php

    $con = mysqli_connect('localhost', 'root', 'root', 'unityaccess');

    //check if username exist

     if(mysqli_connect_error())
    {
        echo "1: Connection Failed";
        exit();
    }

    $username = $_POST['username'];
    $password = $_POST['password'];

    $namecheckquery = "SELECT username, hash, salt, score FROM players WHERE username='".$username."';";
    $namecheck=mysqli_query($con,$namecheckquery)or die("2: Name check query did carry out ");//error code #2 name check query failed

     if(mysqli_num_rows($namecheck)!=1)
    {
        echo "5: No username present or there is more than one row with that username";
    exit();
    }

    //get login info from query
    $existinginfo = mysqli_fetch_assoc($namecheck);
    $salt=$existinginfo["salt"];
    $hash=$existinginfo["hash"];

    $loginhash = crypt($password, $salt);

    if($hash != $loginhash)
    {
        echo "6: Incorrect Password";
        exit();
    }

    echo "0\t" . $existinginfo["score"];

?>
```

Prompt

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

public class Prompt : MonoBehaviour {

    public Text playerDisplay;

    private void Start()
    {
        if (DBManager.LoggedIn)
        {
            playerDisplay.text = "Player: " + DBManager.username;
        }
    }
}
```

Stage 4: Choosing game map and game mode

EZSceneSelect:

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

public class EZSceneSelect : MonoBehaviour
{
    public void selectScene(){
        switch (this.gameObject.name) {
        case "Tutorial":
            SceneManager.LoadScene("EZ - tutorial");
            break;
        case "Explore":
            SceneManager.LoadScene("EZ - Explore");
            break;
        case "Time Trial":
            SceneManager.LoadScene("EZ - TimeTrial");
            break;
        }
    }
}
```

DTSceneSelect

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

public class DTSceneSelect : MonoBehaviour
{
    public void selectScene(){
        switch (this.gameObject.name){
        case"Tutorial":
            SceneManager.LoadScene ("DT - Tutorial");
            break;
        case"Time Trial":
            SceneManager.LoadScene ("DT");
            break;
        case"Free Play":
            SceneManager.LoadScene ("DT - Explore");
            break;
        }
    }
}

```

HLScenceSelect

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

public class HLSceneSelect : MonoBehaviour
{
    public void selectScene(){
        switch (this.gameObject.name) {
        case "Tutorial":
            SceneManager.LoadScene ("HL - Tutorial");
            break;
        case "Explore":
            SceneManager.LoadScene("HL - Explore 1");
            break;
        case "TimeTrial":
            SceneManager.LoadScene ("HL - TT");
            break;
        }
    }
}
```

Stage 5: The Game

Finalcarcontroller

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

public class finalcarcontroller : MonoBehaviour
{
    private const string HORIZONTAL = "Horizontal";
    private const string VERTICAL = "Vertical";

    //values for the input keyboard
    private float horizontalInput;
    private float verticalInput;
    private float currentSteeringAngle;
    private float currentBreakingForce;
    private bool isBreaking;

    //[SerializeField] private float motorforce, brakeForce, maxSteerAngle;
    [SerializeField] private float brakeForce, motorforce, maxSteerAngle;
    [SerializeField] private WheelCollider frontLeftWheelCollider, frontRightWheelCollider, backLe
    [SerializeField] private Transform frontLeftWheelTransform, frontRightWheelTransform, backLeft

```

```
22
23      private void GetInput()
24      {
25          horizontalInput = Input.GetAxis(HORIZONTAL);
26          verticalInput = Input.GetAxis(VERTICAL);
27          isBreaking = Input.GetKey(KeyCode.Space);
28
29      }
30
31      private void FixedUpdate()
32      {
33          //calling all subroutines
34          GetInput();
35          HandleMotor();
36          HandleSteering();
37          UpdateWheels();
38
39      }
40
```

```
41      private void HandleMotor()
42      {
43          frontLeftWheelCollider.motorTorque = verticalInput * motorforce;
44          frontRightWheelCollider.motorTorque = verticalInput * motorforce;
45
46          currentBreakingForce = isBreaking ? brakeForce : 0f;
47
48          ApplyBreaking();
49      }
50
51
52      private void ApplyBreaking()
53      {
54          //applying breaking force
55          frontLeftWheelCollider.brakeTorque = currentBreakingForce;
56          frontRightWheelCollider.brakeTorque = currentBreakingForce;
57          backLeftWheelCollider.brakeTorque = currentBreakingForce;
58          backRightWheelCollider.brakeTorque = currentBreakingForce;
59      }
60
61      private void HandleSteering()
62      {
63          frontLeftWheelCollider.steerAngle = 30 * horizontalInput;
64          frontRightWheelCollider.steerAngle = 30 * horizontalInput;
65      }
66
```

```
76
77      private void UpdateSingleWheel(WheelCollider wheelCollider, Transform wheelTransform)
78      {
79          Quaternion rot;
80          Vector3 pos;
81                                                                              ;
82          wheelCollider.GetWorldPose(out pos,out rot);
83          wheelTransform.rotation = rot;
84          wheelTransform.position = pos;
85      }
86  }
87
```

I James Gammon – 27228 – 8292 – Abbeyfield school

AntiRoll

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

public class AntiRollBar : MonoBehaviour
{
    //wheel colliders
    public WheelCollider backWheelLeftCollider;
    public WheelCollider backWheelRightCollider;


    //antiroll force
    public float AntiRoll = 5000.0f;

    private Rigidbody Player;

    void Start()
    {
        //rigidbody from the player
        Player = GetComponent<Rigidbody>();
    }

    void FixedUpdate()
    {
        WheelHit hit;
        float travelL = 1.0f;
        float travelR = 1.0f;

        //checking if left wheel is grouned
        bool groundedL = backWheelLeftCollider.GetGroundHit(out hit);
        if (groundedL)
        {
            //applying the transform change from the wheel collider - the radius to get the difference

            travelL = (-backWheelLeftCollider.transform.InverseTransformPoint(hit.point).y - backWheelLeftCollider.radius) / backWheelLeftCollider.suspensionDistance;
        }

        //checking if right wheel is grouned
        bool groundedR = backWheelRightCollider.GetGroundHit(out hit);
        if (groundedR)
        {
            travelR = (-backWheelRightCollider.transform.InverseTransformPoint(hit.point).y - backWheelRightCollider.radius) / backWheelRightCollider.suspensionDistance;
        }

        //applying anti-roll to the difference in the wheel distance
        float antiRollForce = (travelL - travelR) * AntiRoll;

        if (groundedL)
            Player.AddForceAtPosition(backWheelLeftCollider.transform.up * -antiRollForce, backWheelLeftCollider.transform.position);

        if (groundedR)
            Player.AddForceAtPosition(backWheelRightCollider.transform.up * antiRollForce, backWheelRightCollider.transform.position);
    }
}
```

Checkpoint system

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

public class CheckpointsSystems : MonoBehaviour
{
    public GameObject CheckpointEnter;

    void OnTriggerEnter()
    {
        CheckpointEnter.SetActive(false);
        Debug.Log("Checkpoint Entered");
    }
}
```

Countdowncontroller

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

public class CountDownController : MonoBehaviour
{
    public int countdownTime;
    public Text countdownDisplay;

    private void Start()
    {
        StartCoroutine(CountDownToStart());
    }

    IEnumerator CountDownToStart()
    {
        while(countdownTime > 0)
        {
            countdownDisplay.text=countdownTime.ToString();

            yield return new WaitForSeconds(1f);

             //decreases count down time by one
            countdownTime--;

             //changes the text to red
             if(countdownTime>=3.5f){countdownDisplay.color = Color.white;}
             if(countdownTime<3.5f){countdownDisplay.color = Color.red;}
        }

       //starts the timer once the timer has reached GO!
        countdownDisplay.text = "GO!";
        yield return new WaitForSeconds(1f);
        countdownDisplay.gameObject.SetActive(false);
        ScoreTimeManager.instance.BeginTimer();
    }
}
```

The countdown controller has been duplicated for the tutorial display

```
        //sets active tutorial GUI
        tutorialbuttons.gameObject.SetActive(true);
        countdownDisplay.gameObject.SetActive(false);
    }
}
```

I James Gammon – 27228 – 8292 – Abbeyfield school

Loggedinplayer (for username display)

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

public class loggedinplayer : MonoBehaviour
{
    public Text PlayerDisplayText;

    private void Start()
    {
        if (DBManager.LoggedIn)
        {
            PlayerDisplayText.text = "Player: " + DBManager.username;
        }
    }
}
```

Movingtomenus

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

public class movingtomenus : MonoBehaviour
{
    // EZ time trial
    public void EZTimeTrial()
    {
        SceneManager.LoadScene("EZ - TimeTrial");
    }

    // DT Tutorial
    public void DTTutorial()
    {
        SceneManager.LoadScene("DT - Tutorial");
    }

    // DT TimeTrial
    public void DTTimeTrial()
    {
        SceneManager.LoadScene("DT");
    }

    // Hard Level Tutorial
    public void HLTutorial()
    {
        SceneManager.LoadScene("HL - Tutorial");
    }

    //Hard level TimeTrial
    public void HLTimeTrial()
    {
        SceneManager.LoadScene("HL - TT");
    }
}
```

Stage 6: The Game – Time Trial

DBManger

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

public static class DBManager {

    public static string username;
    public static int score;
    public static string scoreeee;
    public static bool LoggedIn { get { return username != null;}}

    public static void LogOut() //username changed to null to log out
    {
        username = null;
    }
}
```

Game

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

public class game : MonoBehaviour
{
    public Text playerDisplay;
    public Text scoreDisplay;

    private void Awake()
    {
        //player not logged in is returned to the main menu
        if (DBManager.username == null)
        {
            SceneManager.LoadScene("MainMenu");
        }

        playerDisplay.GetComponent<Text>().text = "" + DBManager.username;
        scoreDisplay.GetComponent<Text>().text =  "" +  DBManager.scoreeee;
    }
}
```

GameDB

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

public class gameDB : MonoBehaviour
{
    public Text playerDisplay;
    public Text scoreDisplay;

    //logs out player
    private void Awake()
    {
        if (DBManager.username == null)
        {
            SceneManager.LoadScene("MainMenu");
        }

        playerDisplay.GetComponent<Text>().text = "" + DBManager.username;
        scoreDisplay.GetComponent<Text>().text = "" + DBManager.score;
    }
```

```
    //saves player data
    public void CallSaveData()
    {
        StartCoroutine(SavePlayerData());
    }

    IEnumerator SavePlayerData()
    {
        WWWForm form = new WWWForm();
        form.AddField("username", DBManager.username);
        form.AddField("score", DBManager.score);

        //uploads data to the flatfile DB
        WWW www = new WWW("http://localhost/sqlconnect/savedata.php", form);
        yield return www;
        if (www.text == "0")
        {
            Debug.Log("Game is Saved. ");
        }
        else
        {
            Debug.Log("Save Failed. Error no: " + www.text);
        }
    }
}
```

HalfwayPointTrigger1

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

public class HalfPointTrigger1 : MonoBehaviour
{

    public GameObject LapCompleteTrig;
    public GameObject HalfLapTrig;

    void OnTriggerEnter ()
    {
        LapCompleteTrig.SetActive(true);
        HalfLapTrig.SetActive(false);
        Debug.Log("Halfway point entered");
    }
}
```

Scoretimemanager

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

public class ScoreTimeManager : MonoBehaviour
//public class TimeManager : MonoBehaviour
{
    public static ScoreTimeManager instance;
    //public static TimeManager instance;

    public Text timeCounter;
    public Text BesttimeCounter;
    public TimeSpan timePlaying;
    public static bool timerGoing;
    public static float elapsedTime;
    public int score;

    private void Awake()
    {
        instance = this; //allows to call functions from outside of the class
    }

```

```
25      // Start is called before the first frame update
26      public void Start()
27      {
28          timeCounter.text = "Time: 00:00.00";
29          BesttimeCounter.text = "00:00.00";
30          timerGoing = false;
31          Debug.Log("Timer Reset");
32
33      }
34
35      public void BeginTimer()
36      {
37          timerGoing = true;
38          elapsedTime = 0f;
39
40          StartCoroutine(UpdateTimer());
41          Debug.Log("Timer began");
42      }
43
44      public void EndTimer()
45      {
46          timerGoing = false;
47      }
48
```

```
48
49      private IEnumerator UpdateTimer()//counts the timer for the box
50      {
51          while (timerGoing)
52          {
53              elapsedTime += Time.deltaTime;
54              timePlaying = TimeSpan.FromSeconds(elapsedTime);
55              string currenttimePlayingStr = "Time: " + timePlaying.ToString("mm':'ss'.'ff");
56              timeCounter.text = currenttimePlayingStr;
57              yield return null;
58              //return to this point on the next frame checks the while loop again
59          }
60      }
61  }
62
```

I James Gammon – 27228 – 8292 – Abbeyfield school

ScoreTimeManagerLapComplete:

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

public class ScoreTimeManagerLapComplete : MonoBehaviour
{

    //publicDisplay
    public Text BestTimeCounter;

    //sorting bestTime
    public static TimeSpan mintimePlayed;
    public static List<float> bestTimes = new List<float>();

    //laptriggers
    public GameObject LapCompleteTrig;
    public GameObject HalfLapTrig;

    //laps counter
    public GameObject LapCounter;
    public static int LapsDone;

    //defined for the pauseMenu
    public GameObject ScoreMenu;
    public GameObject UserNamePanelUI;
    public GameObject TimertextUI;
    public GameObject LapLabelUI;
    public GameObject BestTimePanel;
    public GameObject MoveMap;
    public Text playerDisplay;
    public Text scoreDisplay;

```

```
    void Start()
    {
        BestTimeCounter.text = "00:00.00";
    }

    void OnTriggerEnter()
    {
        //increasing the amount of laps
        LapsDone += 1;

        //entering the trigger
        Debug.Log("LapCompleteTrigger Entered");

        //adding time to the array
        if (ScoreTimeManager.elapsedTime != 0)
        {
            bestTimes.Add(ScoreTimeManager.elapsedTime);
            Debug.Log("Time added to array");
        }

        //upload laps to laps done
        if (bestTimes.Count > 0)
        {
            bestTimes.Sort();

            mintimePlayed = TimeSpan.FromSeconds(bestTimes[0]);
            DBManager.scoreee = "" + mintimePlayed.ToString("mm':'ss'.'ff");
            BestTimeCounter.text = DBManager.scoreee;

            Debug.Log("The fastest score was " + DBManager.scoreee + " Seconds... Well Done");

        }
```

```

        //if the laps completed
        if (LapsDone == 1)
        {
            // things being activdxdxrfr1ated
            scoreDisplay.GetComponent<Text>().text = "" + DBManager.scoreee;
            playerDisplay.GetComponent<Text>().text = "" + DBManager.username;
            ScoreMenu.SetActive(true);

            //things being deactivated
            TimertextUI.SetActive(false);
            UserNamePanelUI.SetActive(false);
            ScoreTimeManager.timerGoing = false;
            LapLabelUI.SetActive(false);
            BestTimePanel.SetActive(false);
            MoveMap.SetActive(false);

            //Debug.Log
            Debug.Log("Laps Finished");
        }

        //set timer back to 0
        ScoreTimeManager.elapsedTime = 0;

        //set half lap trigger
        HalfLapTrig.SetActive(true);
        // set lap complete trigger
        LapCompleteTrig.SetActive(false);
        LapCounter.GetComponent<Text>().text = "" + LapsDone;
    }
}
```

ScoreTimeManagerLapCompleteDB

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

public class ScoreTimeManagerLapCompleteDB : MonoBehaviour
{
    public Text BestTimeCounter;
    //public static Text timePlayedStr;
    public static TimeSpan mintimePlayed;
    public static List<float> bestTimes = new List<float>();
    public GameObject LapCompleteTrig;
    public GameObject HalfLapTrig;
    public GameObject LapCounter;
    public static int LapsDone;
    public GameObject ScoreMenuUI;
    public GameObject UserNamePanelUI;
    public GameObject TimertextUI;
    public GameObject LapLabelUI;
    public GameObject BestTimePanel;
    public Text playerDisplay;
    public Text scoreDisplay;

    // Start is called before the first frame update
    void Start()
    {
        BestTimeCounter.text = "00:00.00";
    }

    void OnTriggerEnter()
    {
```

```
    void OnTriggerEnter()
    {
        //entering the trigger
        Debug.Log("LapCompleteTrigger Entered");
        LapsDone += 1;


        //adding the score to the aray
        if (ScoreTimeManager.elapsedTime != 0)
        {
            bestTimes.Add(ScoreTimeManager.elapsedTime);
            Debug.Log("time added to array");
        }
```

```
44
45          //upload laps to laps done
46          //printing the best time
47          if (bestTimes.Count > 0)
48          {
49              bestTimes.Sort();
50              mintimePlayed = TimeSpan.FromSeconds(bestTimes[0]);
51              string timePlayedStr = "" + mintimePlayed.ToString("mm':'ss'.'ff");
52              BestTimeCounter.text = timePlayedStr;
53              DBManager.score = int.Parse(mintimePlayed.ToString("mmssff"));
54
55
56              //assigning the best player score to DBManager to be uploaded
57              Debug.Log("The fastest score was " + DBManager.score + " Seconds... Well Done");
58          }
59
60          //if the laps are completed
61          if (LapsDone == 3)
62          {
63              TimertextUI.SetActive(false);
64              UserNamePanelUI.SetActive(false);
65              ScoreTimeManager.timerGoing = false;
66              LapLabelUI.SetActive(false);
67              BestTimePanel.SetActive(false);
68              Debug.Log("Laps Finished");
69              ScoreMenuUI.SetActive(true);
70
71              scoreDisplay.GetComponent<Text>().text = "" + DBManager.score;
72              playerDisplay.GetComponent<Text>().text = "" + DBManager.username;
73          }
74
75
76          //set timer back to 0
77          ScoreTimeManager.elapsedTime = 0;
78
```

```
78
79              //set half lap trigger
80              HalfLapTrig.SetActive(true);
81              // set lap complete trigger
82              LapCompleteTrig.SetActive(false);
83              LapCounter.GetComponent<Text>().text = "" + LapsDone;
84          }
85      }
```

Stage 7: The Game – Tutorial

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

public class PanelOpeningScript : MonoBehaviour
{
    public GameObject TutorialPanel;
    public GameObject MapPanel;
    public GameObject GamePanel;

    public void OpenTurorialPanel()
    {
        if (TutorialPanel != null)
        {
            bool isActive = TutorialPanel.activeSelf;
            TutorialPanel.SetActive(!isActive);
            Debug.Log("Opening Tutorial Panel...");
        }
    }

    public void OpenGamePanel()
    {
        if (GamePanel != null)
        {
            bool isActive = GamePanel.activeSelf;
            GamePanel.SetActive(!isActive);
            Debug.Log("Opening Game Panel...");
        }
    }


    public void OpenMapPanel()
    {
        if (MapPanel != null)
        {
            bool isActive = MapPanel.activeSelf;
            MapPanel.SetActive(!isActive);
            Debug.Log("Opening Map Panel...");
        }
    }
}
```

Stage 8: The Game – Free Play

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

public class MinimapScript : MonoBehaviour
{

    public Transform player;

    //update end of every second
    void LateUpdate()
    {
        //vector3 = car position
        Vector3 newPosition = player.position;
        // causes the y position to stay the same so camera does not move up and down
        newPosition.y = transform.position.y;
        transform.position = newPosition;

        transform.rotation = Quaternion.Euler(90f, player.eulerAngles.y, 0f);
    }

}
```

Stage 9: Completing the game and time display

LeaderboardMenu

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

public class leaderboardMenu : MonoBehaviour
{
    public void Back2Menu()
    {
        SceneManager.LoadScene("MainMenu");
    }

    public void Back2Leaderboard()
    {
        SceneManager.LoadScene("LeaderBoard");
    }
}

```

Savedata.php

```
<?php
    $con = mysqli_connect('localhost', 'root', 'root', 'unityaccess');

    //check if username exist
    if(mysqli_connect_error())
    {
        echo "1: Connection Failed";
        exit();
    }

    $username = $_POST["username"];
    $newscore = $_POST["score"];

    //double check only one user in the game
    $namecheckquery = "SELECT username FROM players WHERE username='".$username."';";

    $namecheck = mysqli_query($con, $namecheckquery)or die("2: Name check query failed"); //error code #2 name check

    if(mysqli_num_rows($namecheck)!=1) {
        echo "5: Either no user with that name or more than 1";
        exit();
    }

    $updatequery = "UPDATE players SET score = ". $newscore ." WHERE username = '" .$username ."';";
    mysqli_query($con, $updatequery) or die("7: Save Query Failed"); //error code no.7 update query failed

    echo"0";
?>

```

Stage 10: Pause Menu

PauseMenuRacing

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

public class PauseMenuRacing : MonoBehaviour
{
    public static bool GameIsPaused = false;
    public GameObject pauseMenuUI;
    public GameObject UserNamePanelUI;
    public GameObject LapsCounter;
    public GameObject TimeDisplay;
    public GameObject BestTimeCounter;
    public GameObject CountDownDisplay;



    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (GameIsPaused)
            {
                Resume();
            }

            else
            {
                Pause();
            }
        }
    }

    public void Resume()
    {
        //elements being disabled
        pauseMenuUI.SetActive(false);

        Time.timeScale = 1f;
        GameIsPaused = false;
        Debug.Log("Game Resumed");

        //elements being enabled
        UserNamePanelUI.SetActive(true);
        LapsCounter.SetActive(true);
        TimeDisplay.SetActive(true);
        BestTimeCounter.SetActive(true);
        CountDownDisplay.SetActive(true);
    }

    void Pause()
    {
        // elements being enabled
        pauseMenuUI.SetActive(true);

        Time.timeScale = 0f;
        GameIsPaused = true;
        Debug.Log("Game Paused");

        // elements being disabled
        UserNamePanelUI.SetActive(false);
        LapsCounter.SetActive(false);
        TimeDisplay.SetActive(false);
        BestTimeCounter.SetActive(false);
        CountDownDisplay.SetActive(false);


    }

    public void LoadMenu()
    {
        Time.timeScale = 1f;
        Debug.Log("Loading Menu...");
        SceneManager.LoadScene("MainMenu");
    }

    public void QuitGame()
    {
        Debug.Log("Quitting Game...");
        Application.Quit();
    }

}
```

I James Gammon – 27228 – 8292 – Abbeyfield school

PauseMenuExplore:

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

public class PauseMenuExplore : MonoBehaviour
{
    public static bool GameIsPaused = false;
    public GameObject pauseMenuUI;
    public GameObject UserNamePanelUI;
    public GameObject MiniMap;
    public GameObject CountDownDisplay;



    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (GameIsPaused)
            {
                Resume();
            }

            else
            {
                Pause();
            }
        }
    }

    public void Resume()
    {
        pauseMenuUI.SetActive(false);
        Time.timeScale = 1f;
        GameIsPaused = false;
        Debug.Log("Game Resumed");

        // elements being enabled
        UserNamePanelUI.SetActive(true);
        MiniMap.SetActive(true);
        CountDownDisplay.SetActive(true);
    }

    void Pause()
    {
        // elements being enabled
        pauseMenuUI.SetActive(true);

        Time.timeScale = 0f;
        GameIsPaused = true;
        Debug.Log("Game Paused");

        //elements being disabled
        UserNamePanelUI.SetActive(false);
        MiniMap.SetActive(false);
        CountDownDisplay.SetActive(false);

    }
```

```
    public void LoadMenu()
    {
        Time.timeScale = 1f;
        Debug.Log("Loading Menu...");
        SceneManager.LoadScene("MainMenu");
    }

    public void QuitGame()
    {
        Debug.Log("Quitting Game...");
        Application.Quit();
    }

}
```

I James Gammon – 27228 – 8292 – Abbeyfield school

PauseMenuTutorial

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

public class PauseMenuTutorial : MonoBehaviour
{
    public static bool GameIsPaused = false;
    public GameObject pauseMenuUI;
    public GameObject UserNamePanelUI;
    public GameObject tutorialbuttons;
    public GameObject CountDownDisplay;
    public GameObject GamePanel;
    public GameObject MapPanel;
    public GameObject TutorialPanel;



    void Update()
    {
        if (Input.GetKeyDown(KeyCode.Escape))
        {
            if (GameIsPaused)
            {
                Resume();
            }

            else
            {
                Pause();
            }
        }
    }

    public void Resume()
    {
        // elements being disabled
        pauseMenuUI.SetActive(false);

        Time.timeScale = 1f;
        GameIsPaused = false;
        Debug.Log("Game Resumed");

        // elements being enabled
        UserNamePanelUI.SetActive(true);
        tutorialbuttons.SetActive(true);
        CountDownDisplay.SetActive(true);
        GamePanel.SetActive(false);
        MapPanel.SetActive(false);
        TutorialPanel.SetActive(false);

    }
```

```
    void Pause()
    {
        // elements being enabled
        pauseMenuUI.SetActive(true);

        Time.timeScale = 0f;
        GameIsPaused = true;
        Debug.Log("Game Paused");

        //elements being disabled
        UserNamePanelUI.SetActive(false);
        tutorialbuttons.SetActive(false);
        CountDownDisplay.SetActive(false);
        GamePanel.SetActive(false);
        MapPanel.SetActive(false);
        TutorialPanel.SetActive(false);
    }

    public void LoadMenu()
    {
        Time.timeScale = 1f;
        Debug.Log("Loading Menu...");
        SceneManager.LoadScene("MainMenu");
    }

    public void QuitGame()
    {
        Debug.Log("Quitting Game...");
        Application.Quit();
    }

}
```

Stage 11: Bubble sort for leaderboard

leaderboard

```
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using UnityEngine;
using UnityEngine.Networking;
using UnityEngine.UI;

public class LeaderBoard : MonoBehaviour
{
    // creates variables to upload into unity
    string a = string.Empty;
    string b = string.Empty;

    public Text Score;
    public Text Username;

    // creation of an array of usernames from the leaderboard
    string[] array2  = new string[1000];

    // array creation of for the scores inside of the leaderboard as initgers
    int[] array1 = new int[1000];


    List<int> list = new List<int>();
    string temp;


    // immediately after opening the leaderboar script the resutls are imported and updated
    void Awake()
    {
        StartCoroutine(BubbleSort());
    }


    IEnumerator BubbleSort()
    {
        // creates a form that will retrieve the current scores and inputs from the leaderboard and input them
        // into the array

        WWW www = new WWW("http://localhost/sqlconnect/leaderboard.php");

        yield return www;

        // splits the usernames from a single line of string into differnet lines
        string[] result = www.text.Split('\t');

        //takes the first element in th results and while is less than total results
        // then adds 2 to get to the next username

        // score array
        for (int i = 1; i < result.Length; i += 2)
        {
            // changes the position of score to an intiger
            // passing the value of the leaderboard by value
            int Num = int.Parse(result[i]);

            // array elements are now value numbers
            array1[i] = Num;
        }
```

```

        int temp = 0;

        // for every element in the array
        for (int write = 0; write < array1.Length; write++)
        {
            // Last i elements are already in place
            for (int sort = 0; sort < array1.Length - 1; sort++)
            {
                // checks the position of the score, if it is smaller than than the one above

                // if (array1[sort] < array1[sort + 1])
                if (array1[sort] > array1[sort + 1])
                {

```

```
                    // temp is the current element
                        temp = array1[sort + 1];
                        // moves onto the next element
                        array1[sort + 1] = array1[sort];
                        // makes the current element to the adjacenet one
                        array1[sort] = temp;
                    }
                }
            }
            //code to implement into the unity leaderboard

            // for each element in the score array
            for (int j = 0; j < array1.Length; j++)
            {
                // if the element is larger than the first one
                if (array1[j] > 1)
                {
                    // a = string variable inputting the score into
                    // a now becomes the current score in the array1[j] the current second best
                    a = a + (array1[j].ToString() + "\n");

                    // creating a form to submit to the database
                    WWWForm form = new WWWForm();
                    // incramenting the score variable to have the sorted score

                    // posts the score and variable to the form
                    form.AddField("score", array1[j]);

                    // php script which assigns the score variable to the username stored in the database
                    WWW www2 = new WWW("http://localhost/sqlconnect/RetrieveUsername.php", form);

                    // returns of the usernames from the php script from the database
                    yield return www2;

                    // array which stores the string of usernames with associated scores
                    string[] result2 = www2.text.Split('\t');
                    // partitioins the values returned from the query

                    for (int q = 0; q < result2.Length; q++)
                    {
                        // b (the usernames) are converted to string and assinged to the b array
                        b = b + (result2[q].ToString() + "\n");
                    }
                }
            }

            // assings these to game objects inside of unity
            Score.text = (a);
            Username.text = (b);



        }


}
```

I James Gammon – 27228 – 8292 – Abbeyfield school

Leaderboard.php

```php
<?php

  $con = mysqli_connect('localhost', 'root', 'root', 'unityaccess');

  if (mysqli_connect_errno())
    {
      echo "1: conection failed"; // error code #1 = connection failed
      exit();
    }

    $sql = "SELECT username, score FROM players";

    $result = $con->query($sql);

    if ($result->num_rows > 1) {
     // output data of each row
     while($row = $result->fetch_assoc()) {

      echo $row["username"]."\t".$row["score"]. "\t";
     }

    }

   $con->close();

?>
```

RetrieveUsername

```php
<?php

  $con = mysqli_connect('localhost', 'root', 'root', 'unityaccess');

   //check that conection happened
   if (mysqli_connect_errno())
   {
      echo "1: conection failed"; // error code #1 = connection failed
      exit();
   }



   // takes score input from the c++ script and creates a variable in php
   // usernames with this score are then printed back to the c++ script in order


   $score = $_POST["score"];
   $int = (int)$score;


   // sql statement which connects to the database to find associative scores
   $sql = "SELECT username FROM players WHERE score = $int";


   // sent as a query , using the $con variable to find the relational database inside of UnityAccess where the players table is
   $result = $con->query($sql);


   // if there is more than 0 rows then there are users in the database with this score (input from the SaveData courutine)
   if ($result->num_rows > 0) {
    // output data of each row
    while($row = $result->fetch_assoc()) {

      // returns the row with username as the primary key
      echo $row["username"]."\t";
    }


    // else there are no scores of this value inside of teh database
  } else {
    echo "No results in the database";
  }

  $con->close();

?>
```

I James Gammon – 27228 – 8292 – Abbeyfield school

leaderboardMenu

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

public class leaderboardMenu : MonoBehaviour
{
    public void Back2Menu()
    {
        SceneManager.LoadScene("MainMenu");
    }

    public void Back2Leaderboard()
    {
        SceneManager.LoadScene("LeaderBoard");
    }
}

```