OCR GCE A
COMPUTER SCIENCE PROJECT
H446-03

Name:	Jahangir Miah
Candidate Number:	7494
Chingford Foundation School:	13403
Title of Project:	Web Developer Simulator

H446-03 – Project CONTENTS

Table of Contents

A. ANALYSIS

The Problem:

There is no incentive to become web developers or opportunities to develop interests in computer science as a child. Primary schools tend to teach ICT to build on the basic skills of how to use computers but by year 6, many children haven't developed any passion for computer science because exposure to the subject is very low. In addition to this, there aren't many games and other forms of entertainment to attract young students to go into that line of work. One example is Game Dev Tycoon, which can be seen for many too complicated for children to enjoy it. Whilst it does possibly create an interest of computer science and game developing for players, it does not actually teach them how to develop games but is more of a simulation of how to run a business or an abstraction of what game developers actually go through, simplified to be comprehensive to a younger audience. My game is designed to create an interest in the eyes of young teenagers and even younger adolescents about computer science whilst learning how to code in HTML and CSS as well as the basics of computer science for the younger spectrum of my audience. I will teach them in the form that is widely used by self-teaching apps such as solo learn by providing the resources to answer question I set the player so they can use my aid to help them if they need it.

The Stakeholders:

Parents – Parents play a key role in a child's future so it is vital for this game to have any effect in developing computer science as an interest that they also encourage their child to do so and that if the player finds web developing appealing, that they support the child. This also provides a platform for parents to lightly nudge their children into a career path and find out what suits their child. This game will also give the opportunity for parents to give their children a head start in some programming languages and understand the basics of computer science, before they are taught in schools, in an enjoyable learning experience.

Teachers –They also play a key role in the child's or teenager's future, and are very similar to parents as they are influencers to children and teenagers. It is very important that they do not put off children from this path if they do have interests in the subject. They may use the program in classes to make lessons more practical and enjoyable. I will be asking my computer science teachers on their views on the program.

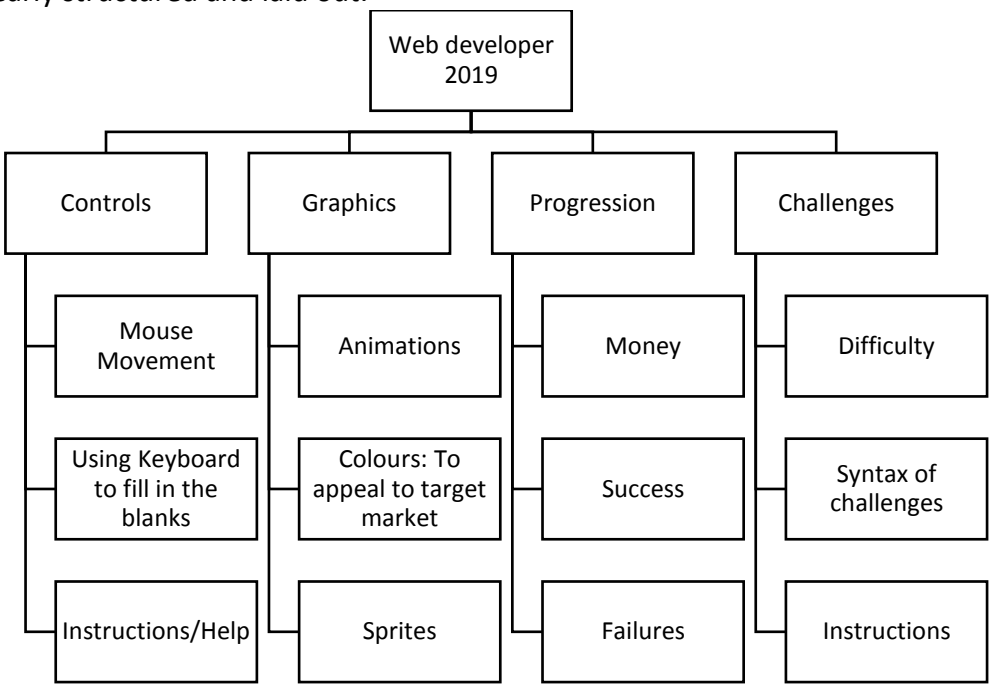
Children (aged 9-17) - The game will primarily be targeted at this age range as I feel this range is old enough to develop a liking to web developing and mature enough to grasp the concept of programming and face the challenges my game will envelop them in. I will primarily be interviewing and gathering information from M.Miah (aged 14) because I will have unlimited access to feedback from her. I will also question 5 students who are in the range of my stakeholders and thus are possible

players of the game. This group includes J.Mikulski, R.Ramsden, J.Coleman, J.Mak and M.Miah

Justifying how my problem can be solved using computational thinking

-Decomposition and thinking procedurally:

I will decompose my problem into many sub problems as it will make solving and understanding the issue much easier. Generally when decomposing an issue, the top down modular design is used this breaks my program into easier solvable issues that are clearly structured and laid out.



I have clearly laid out the key issues when solving this problem. Beginning with the *-controls:*

I need to program the game to allow the user to interact with the simulator mainly using mouse movement and the keyboard itself to answer the challenges that allow the player to progress. The mouse will be used to select options in the main menu and interact with UI elements such as buttons and forms. In the main menu the user will use the mouse to decide whether or not they would like to see the instructions of the game first or to begin straightaway. The mouse will be used to answer multiple choice questions I present to the player. The keyboard will be used to input information onto the forms and will be used by my players/users to answer the harder fill in the blank questions.

-Graphics:

As my target audience ranges from young adolescents to older teenagers, I anticipate that to keep their interest I must make the game aesthetically pleasing by using vibrant colours and interesting sprites to portray the life of a web developer as realistic as I can with the tools available to me whilst also making them colourful so I can immerse them in that environment. I will do

this by designing certain sprites on PowerPoint and asking my stakeholders which they find more appealing.

-*Progression:*

Every good game requires a progression scheme to have the player work toward some objective or goal and thus for my game, the main progression identifier would be money as an abstraction from the real world of being a web developer as when they complete real world tasks, they are paid.

In order to progress I must also have a mechanism to accumulate the score and determine whether a player makes money so I will be using a success/failure scheme where the player loses money for failing to complete a task or submitting invalid syntax, and receives pay cheques for completing the challenges laid out.

As well as using money as a sign of progression, I will also be making use of a "number of lives" count where if the player fails to answer a certain number of challenges, the game will load a game over state and the player will have to start at the beginning.

- *Challenges:*

The key issue I will have to resolve is how to teach the syntax and how difficult to make the game based on the fact it is for ages 9+. I will ask advice from my computer science teacher to obtain insight on what topics, they believe will create a strong foundation for programming in younger children and use that as a guide as to the nature of the challenges I present them. With the target stakeholders range being so great, I anticipate I will need to create different difficulties in my simulator so make the game more challenging on the older stakeholders as they are likely to find the challenges directed at the younger stakeholders to be underwhelming in terms of difficulty.

-***Thinking ahead:***

- Controls will be the use of the entire keyboard to answer the question and challenges I will put the players up against. They will also need to use a mouse to navigate on the screen to choose different options such as choosing to answer the challenge or receive help using to learn the syntax.
- The progression in the game will make use of an accumulating counter money or balance and the number of lives as it gives the player a simple goal to preserve their "number of lives" count and maximise the "balance" or "money" count. I have used balance and number of lives as they are abstractions of what a web developer would need to concern themselves with. Balance being the amount of money a web developer earns and number of lives relating to digital games but also because a web developer can only make so many mistakes before they are fired.
-The GUI will have to be clearly laid out and easy to read so that the user can clearly distinguish certain areas and aspects of the game.
-Challenges will have to difficult enough to challenge them but simple enough to solve and will have to store the answers in variable to compare the user's answers to.

-***Thinking Abstractly:***

The game will have to be abstracted as my target audience may not be able to understand or cope with the complexity of being a web developer due to their age. This includes the taxes that come with working for a company or the severity of making mistakes in real life. Furthermore, the abstraction also makes it simpler for me to program the game whilst keeping the main components of the game to still resemble the role of a web developer and paint that specific environment. I will be abstracting in these ways:

- I will use Icons and sprites to indicate changes in game state and other sections of the game such as a book icon to help the user and a tool box or cogs to change the settings. This makes it easier for my target audience to choose certain options and simplifies the interface as a whole. I will look for some free usage right images for some higher quality sprites possibly for the cogs and book icon.
- The challenges I provide my players with, will need to be abstracted as to put tasks that actual web developers would face will be too difficult potentially. By abstracting here, it allows me to have the leeway on the challenges I put my target audience against as it means it will be easier to solve the issue of the kind of challenges I implement
- The life of a person can be very complex so I will abstract the game to the basics for this game using and/or reputation. By doing so it makes the game easier for me to process, understand and solve
- In reality, a mistake in a code when going public can cost a company thousands upon thousands of pounds which I will abstract into a way of taking away money.

I will also be abstracting when developing my solution in numerous ways. This includes:

- Abstracting the number of questions the game will give. If given the time, I would create a game will many hundreds of challenges of each difficulty but with these time restrictions I will only be developing a few challenges as a demonstration. If I have time after the initial creation of the game, I will add more and more challenges.
- I will also be abstracting the depth in which I go into teaching the syntax to teach the very basics for time's sake.
- The hints I give in the book will also be an abstraction of what the full game will hold as it means I can focus more on the functionality of the game itself. The book will include all the tips required to overcome all the challenges I have time to program into the game.

Input	Process	Output
Select difficulty	Adapt challenge to difficulty	Challenge of selected difficulty
Exit	Terminate Program	Exit game
Start	Begin readying challenge	Challenge
Answer	Checking answer is correct	Well done screen/ Better luck next time screen
Correct or wrong	Reward money or	Changes balance

answer	deplete money	
Wrong answer	Checking Lives: If last life, end game; if not, deplete life	Show game over screen; if not, Show a loss of life
Clicked on Book feature	Load book assets	Book pages feature to help player
Click next page or prev page button	Index for page increases or decreases	Load next page image or prev page image
Click pause icon/Click back	Load pause screen/Load game	Setting Screen/ Display game
Click on instructions button in main menu	Load sprites and components for instruction page	Display instruction screen

-**Thinking Logically:**

To think logically involves having a logical decision to make. For example, deciding whether to have and apple or an orange.

My game will consist of several of these decisions.

- The title screen gives the user the decision to “Play” or to “Exit”
- Many of the challenges are multiple choices and as such a logical decision will have to be made by the user to decide what the correct answer is. If the user answers correctly, the algorithm continue but if answered incorrectly, the user loses points and the game reloads or loops back to that same challenge till the user is correct or the player “loses”
- The user also has the choice to look up the “book” for references on any examples that may help them answer the challenge. If the user is in the book state and exits, the algorithm loops back to the challenge

- The user also has the decision to pause the game at any available stage
- A logical decision will have to be made by my program before giving another challenge to check whether the user has met the max amount of money or has become bankrupt. If the user has reached the max amount of money then the game will display the "Winner" screen and terminate. If the user has become bankrupt or fails too many times in a row, the game will display the "Unlucky, Try again" screen and terminate. However, if the user's state in the game fails to meet any of these requirements, then the game will loop and reload another challenge

I will be using the concept of thinking logically quite frequently in the course of developing the code for the game as I plan to implement parts of the waterfall method and RAD where I will code a section of the program, if it works I continue with the next section but if there are bugs I will loop back to where I believe there is a bug and sought to fix it. Then I will use RAD to design the program and create a prototype, revealing it to some stakeholders and asking for feedback. I will then use the feedback to go back to my program and edit, further implementing the idea of thinking logically as I will have to make the decision of switching between these 2 development methods.

Thinking Concurrently:

For a program to be concurrent, it must be running two or more processes simultaneously in the attempt to increase time efficiency

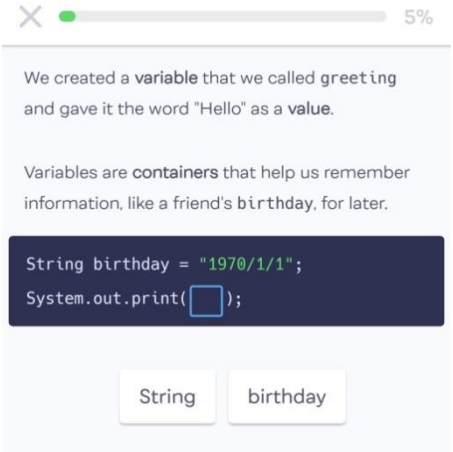
- My game will reward or repossess money from a player for succeeding/failing whilst also preparing the next challenge as to save loading time

In terms of developing my code, I will concurrently think about what the design will be like to appeal to my target audience whilst figuring out how to implement it using unity as I go along. I will also be receiving user feedback whilst developing a solution so I can implement the ideas I receive as I code that aspect of the game.

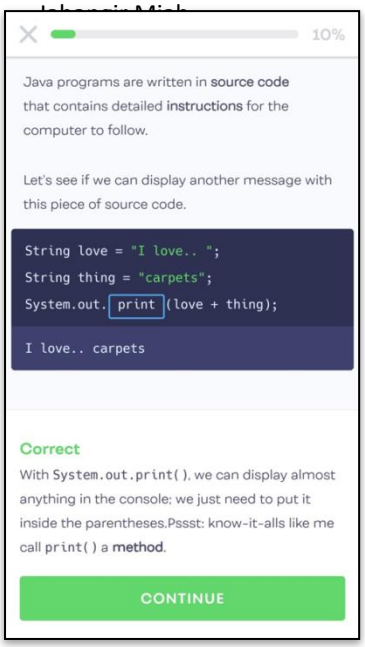
Research

Mimo by Mimohello GmbH Comparison

Mimo is an app created by Mimohello GMBH which was founded in 2016 from Austria. The app is available on app stores that teach users the syntax of various different programming languages such as java, python, HTML and C# are just some of the wide variety they offer. From what I have used of the app, it primarily uses text and reading to teach the syntax and then uses multiple choice questions to test the user's knowledge.



Here we have an example of the ways Mimo educates and tests its users. It is clear that they present information on the syntax above the challenge they offer. They tell the user exactly what it is they are doing by completing the task and develops stronger understanding with the user. They highlight keywords making it easy and pleasant to absorb the information in the users head. They then present the challenge in the form of a multiple choice question to solidify the users understanding of the syntax.



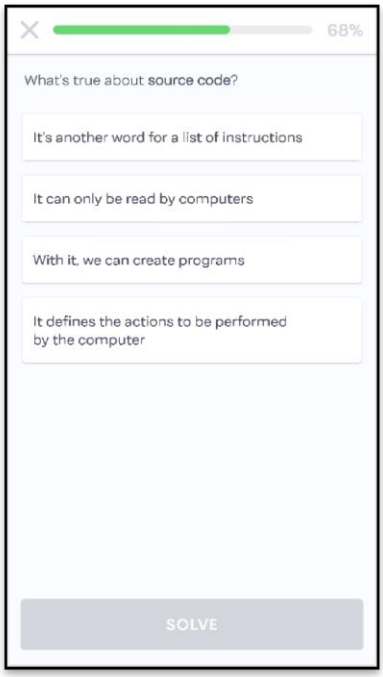
We see here that Mimo explains how the syntax works and also gives background information on the programming language to develop a higher level of familiarity with the syntax the user is working on.

It also has progression concepts. It is evident that the app tells you when you are correct and as a result, you are allowed to progress through the challenges.

On the right side, we can see that the app does not let you progress through the course unless you get the answer right, prompting you to try again guiding you to where you may have got the answer wrong.

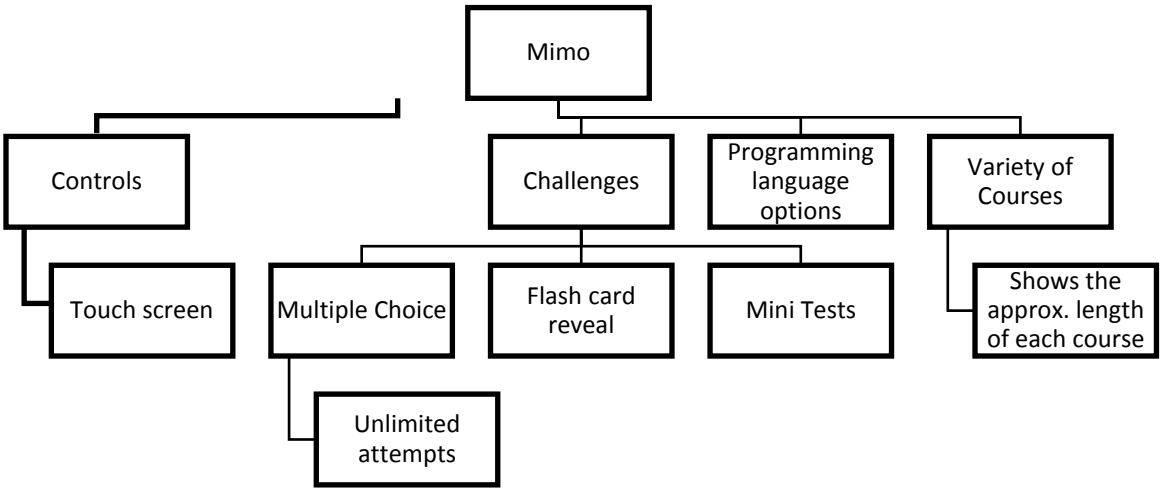
The app gives you little tests of knowledge throughout the progression of the course to hone the skills you are learning in the course with more difficulty as it does not help you until you get it wrong

The app also has some different methods of teaching such as giving you lines of code and asks for the user to read the code and predict the output using a flash-card style of revision.

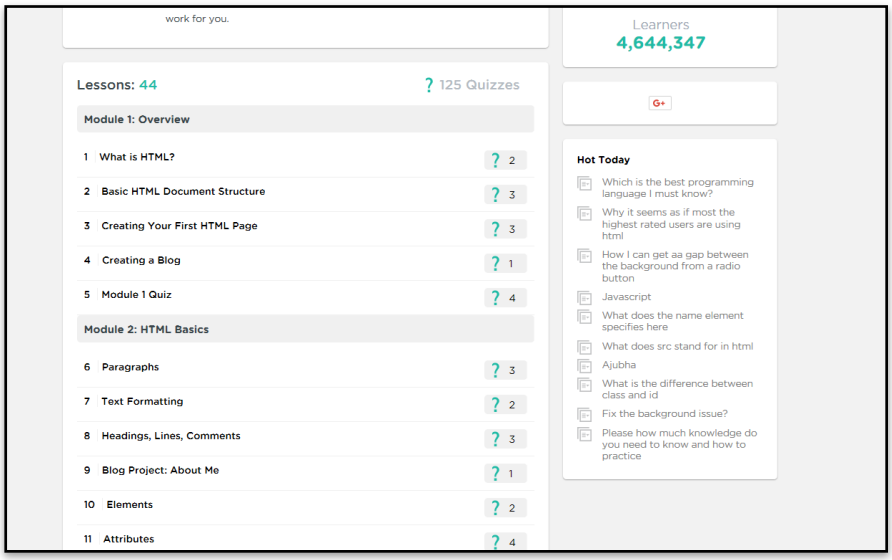


The app also gives tests on general programming knowledge to solidify background knowledge on the subject of programming as a whole

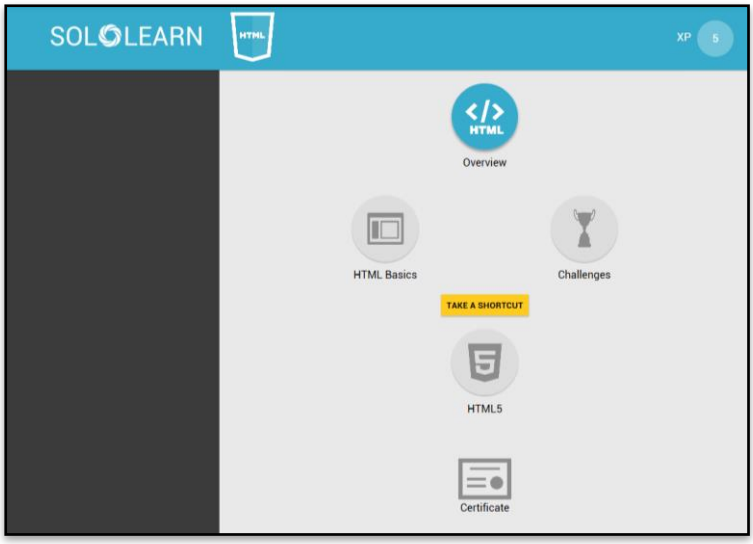
Finally, the limitations of such an app when targeted at my target audience are that, it may not generate an interest in computer science for younger minds. This app as made for people who wanted to learn a specific programming language. Progression and achievement are very limited and the only goal is to learn the programming language which is good for who the app is targeted to but it may not be applicable to my target audience who will probably have little to no exposure to programming.



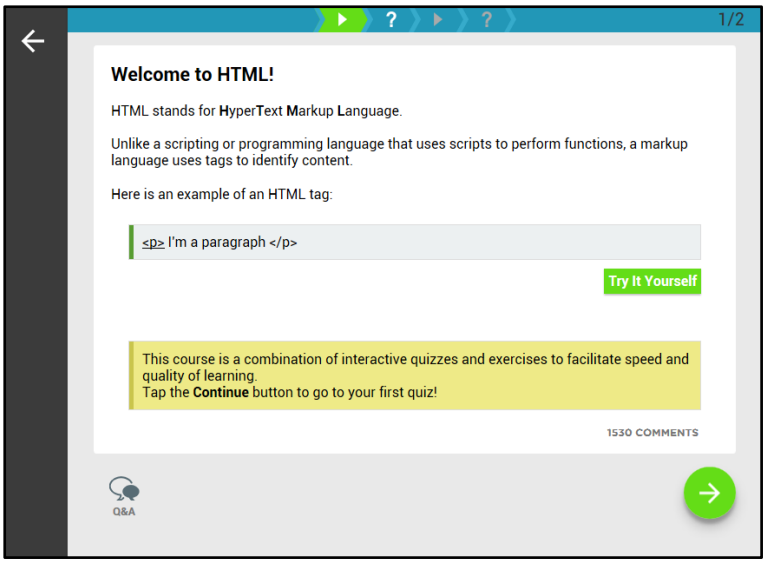
Solo learn: Web page



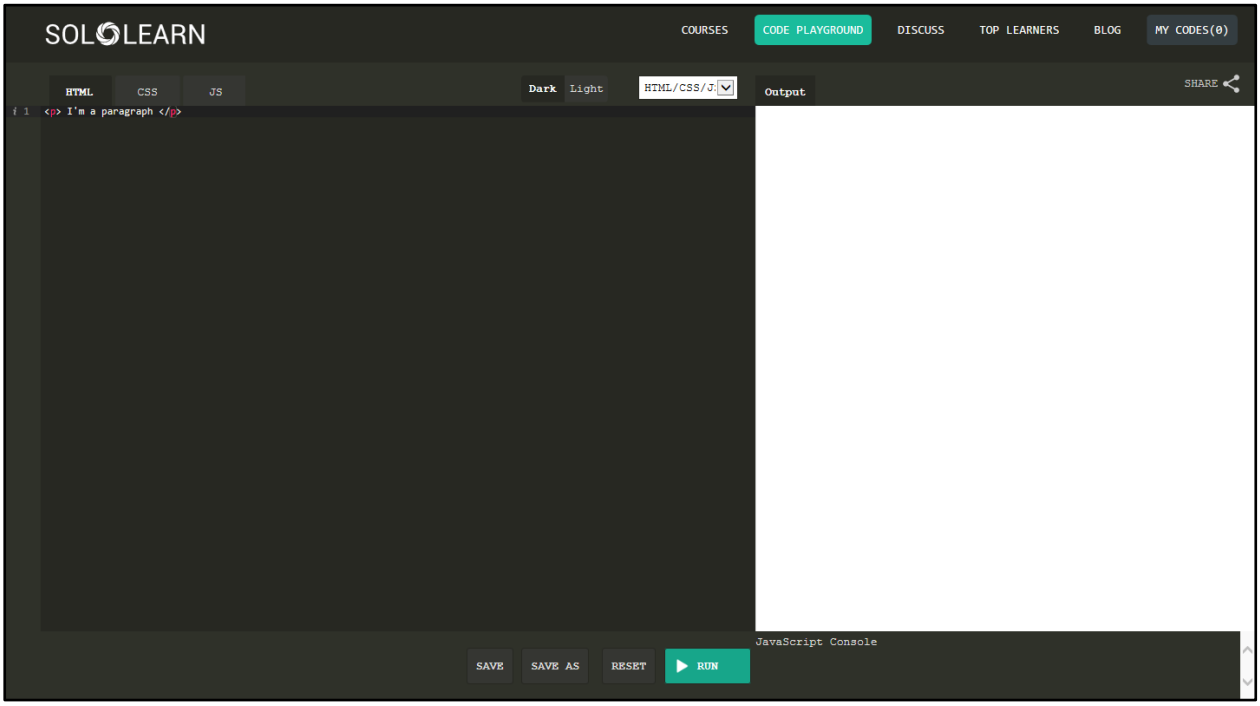
This gives an overview of what the user will be learning by progressing through the course. It's useful as users like to know what they will be learning ahead of time



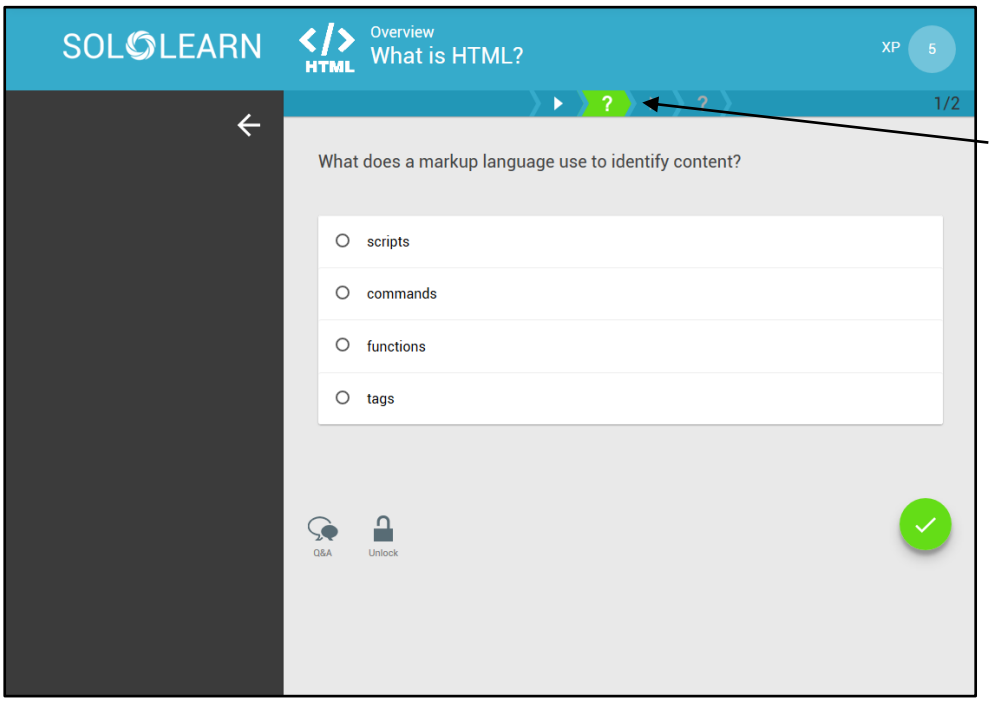
This shows the user the course structure of how the learning will take place. It is shown to be locked until you complete each aspect onto another. The app/website does allow you to take a shortcut however in case you are already skilled at the language and want to hone your current abilities or learn more advanced aspects of the language



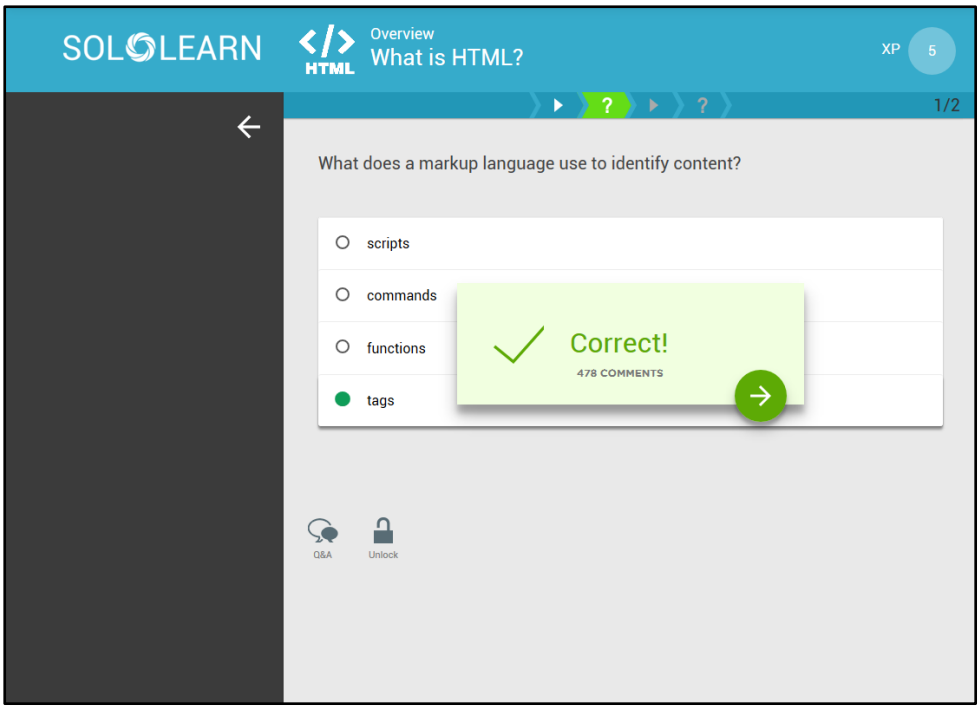
Like Mimo, give the user some information on the language and what the user is currently learning before testing them.



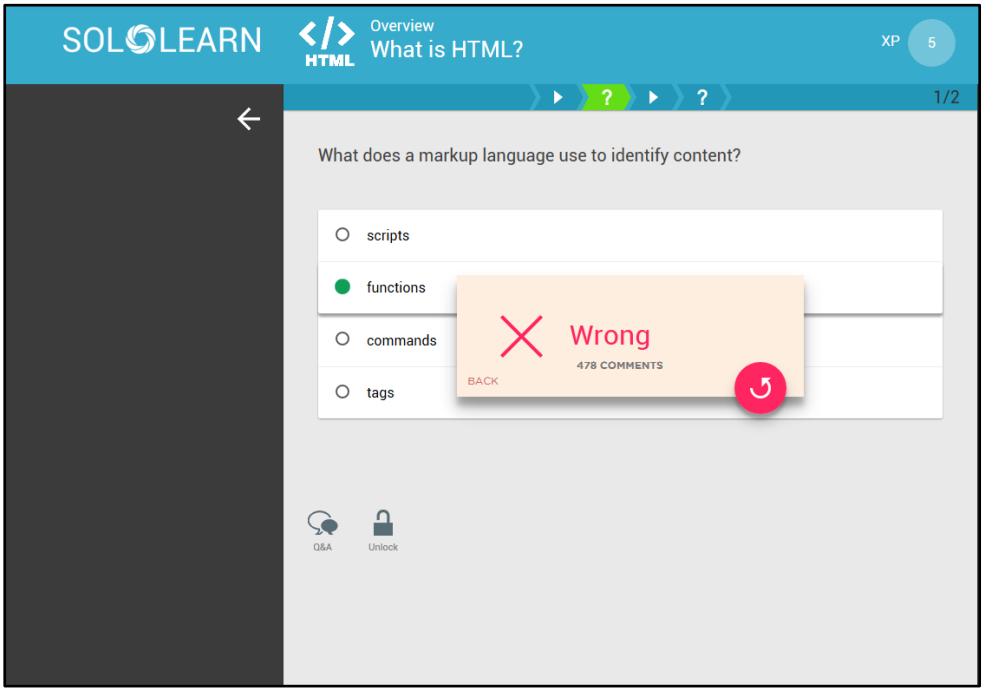
Code Playground: Solo Learn allows the user to test out code themselves in an interface that will run the program giving the user more practical experience which is useful when learning a new language



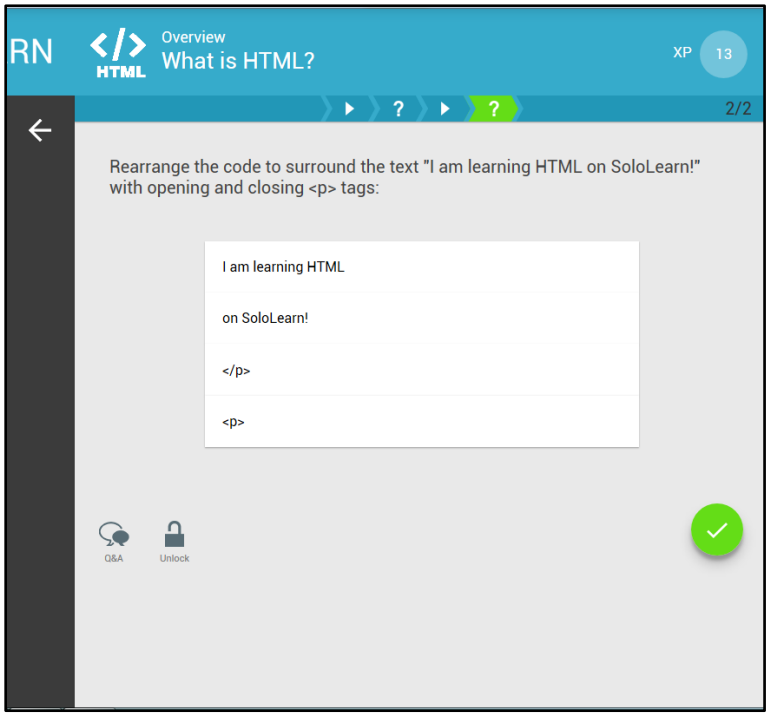
After giving information, Solo Learn asks the user questions on what they learnt from the information that was just given to test what they absorbed. The bar at the top [1] allows the user, if they need to, to go back to the previous screen showing the information they are being tested on. This feature also shows the user how far they are from completing the lesson and indicate what is to come using icons.



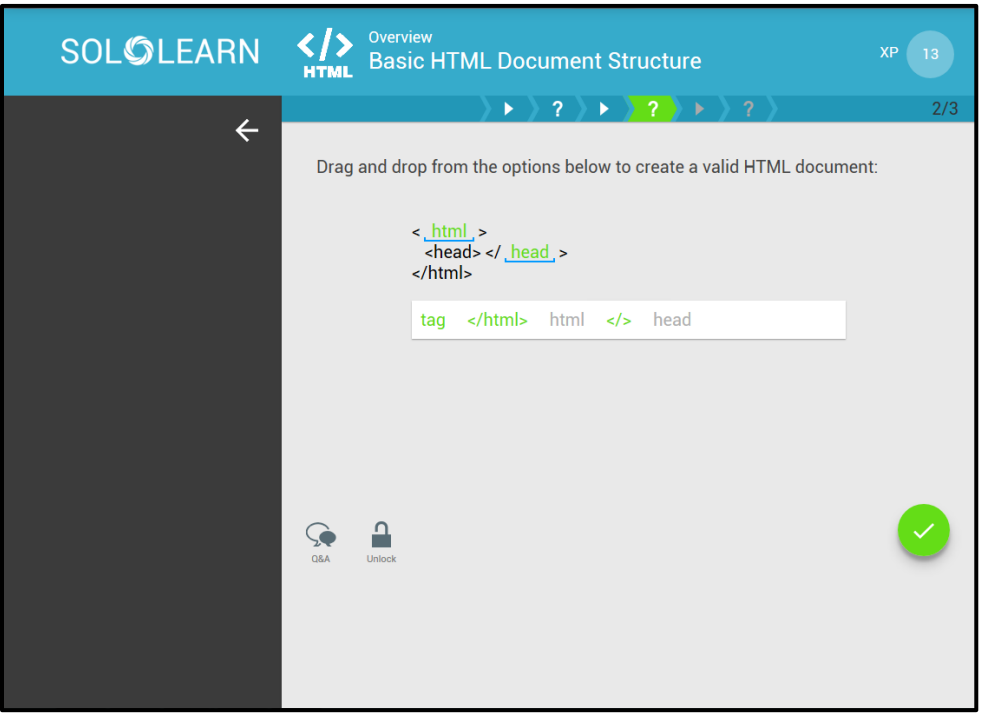
For getting the right answer, the site allows you to progress onto the next part of the challenge



For getting the wrong answer, the site allows you to retry an infinite amount of times until the correct answer is chosen

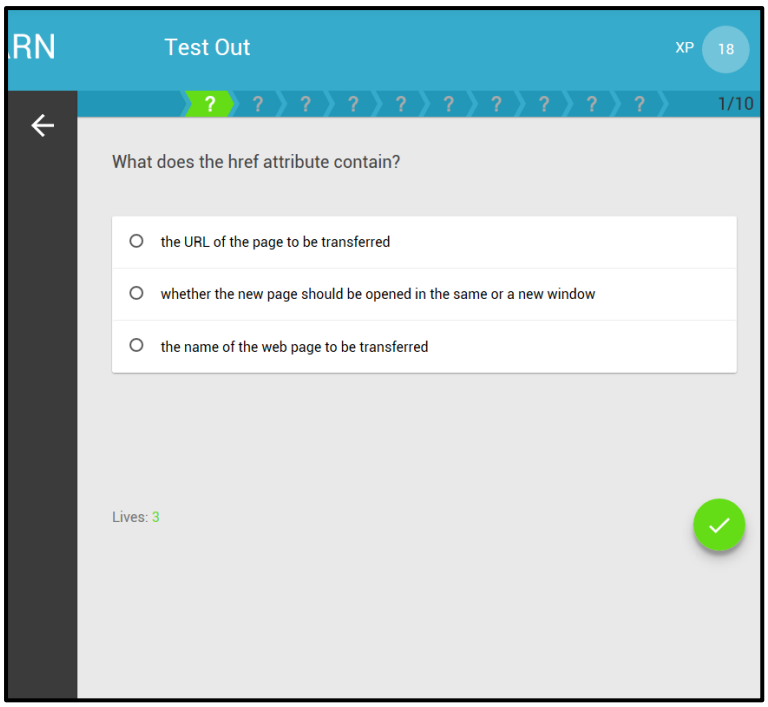


Firstly, it should be noted that for passing onto the next part of the challenge, XP is gained and accumulated and added onto a global leaderboard of the "Top learners". A possible incentive for users to continue using the site. To stay #1.

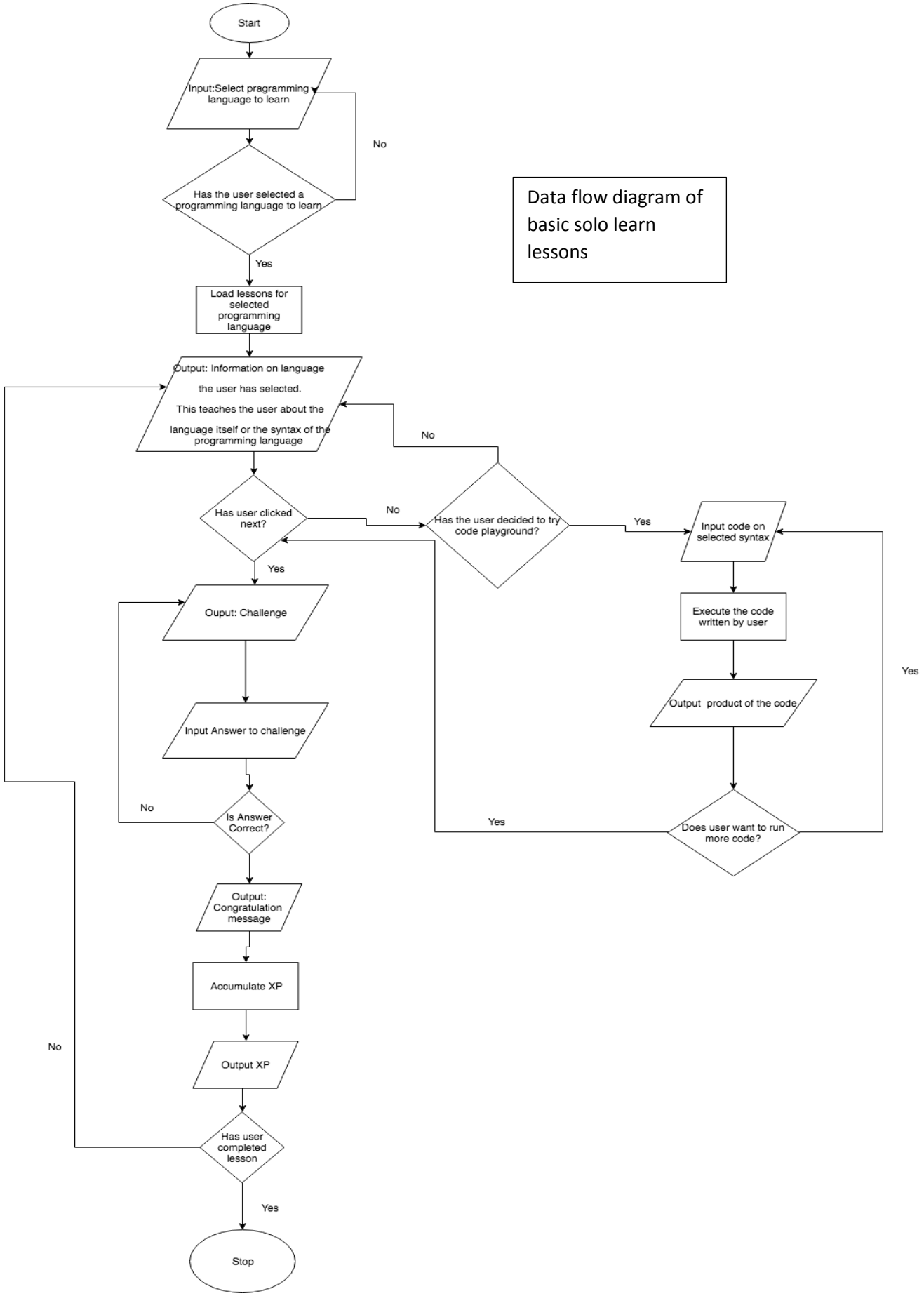


Secondly, This is a challenge Solo learn does to test their users, where they mix up a section of code and ask the user, using the correct syntax, to rearrange them into the correct order. This is a useful feature as it is simple but effective so I may implement this into my final product

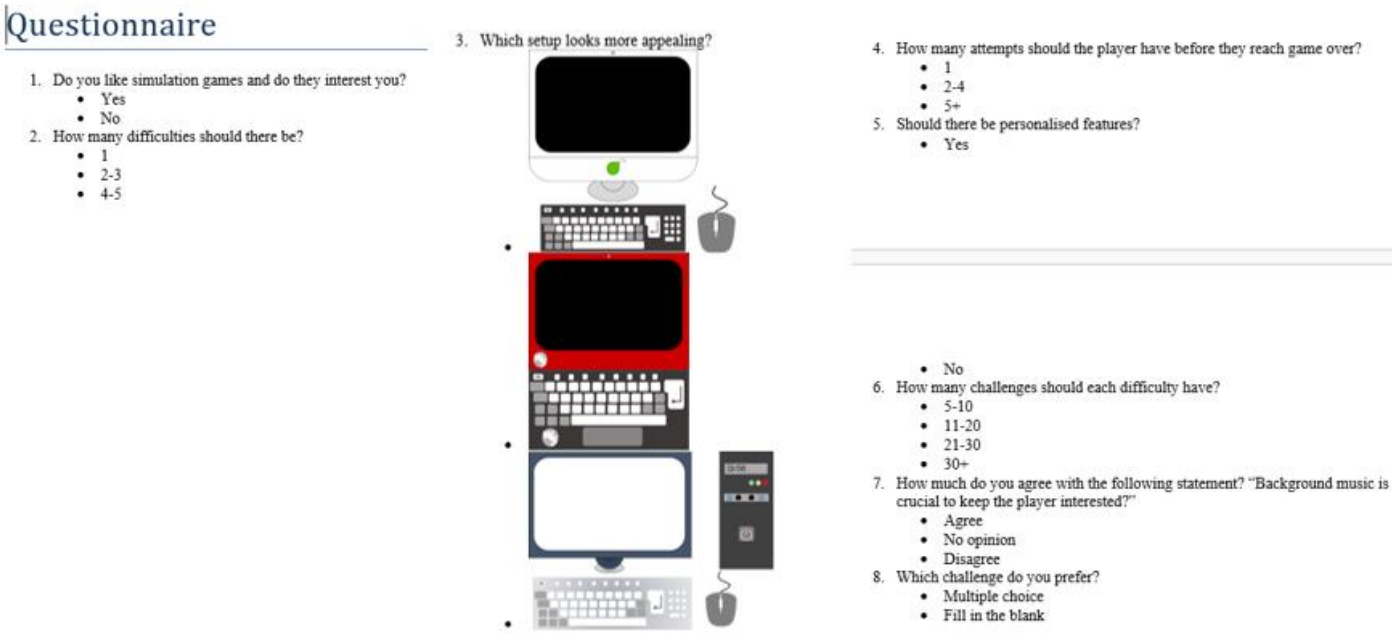
Solo Learn also do use a mixture of fill in the blank and multiple choice challenges by making the user pick and move the answer into the right place. This will be a useful feature for my stakeholders as the game is designed to be simple for children and teens who may find programming entire sections upon sections of code too difficult



This is an example of the site using multiple choice to question the user to test their knowledge


Start
Input:Select pragramming language to learn
Has the user selected a programming language to learn
No
Yes
Load lessons for selected programming language
Output: Information on language the user has selected. This teaches the user about the language itself or the syntax of the programming language
Has user clicked next?
No
Yes
Has the user decided to try code playground?
No
Yes
Input code on selected syntax
Execute the code written by user
Output product of the code
Does user want to run more code?
Yes
Yes
Ouput: Challenge
Input Answer to challenge
Is Answer Correct?
No
Output: Congratulation message
Accumulate XP
Output XP
Has user completed lesson
No
Yes
Stop
Data flow diagram of basic solo learn lessons

Questionnaire



Questionnaire

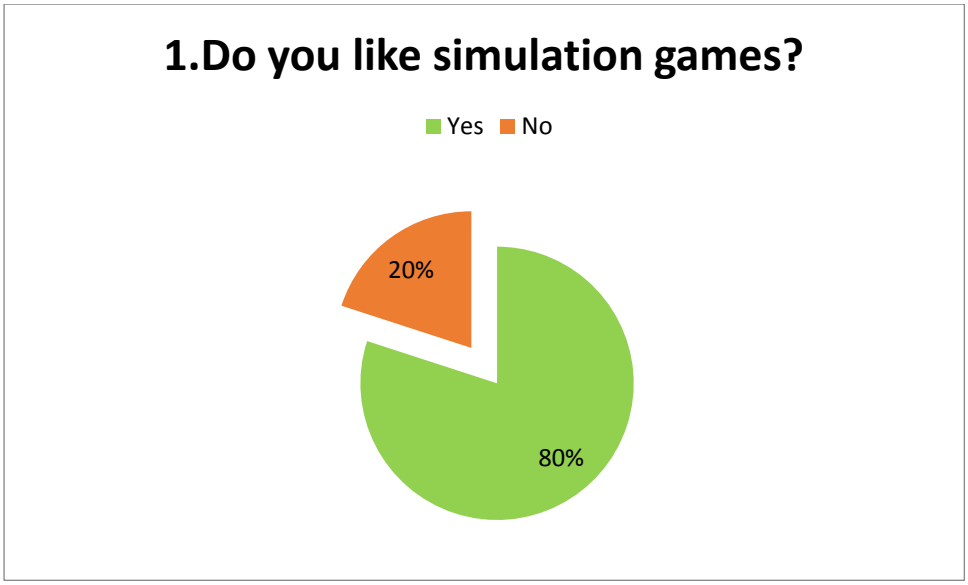
1. Do you like simulation games and do they interest you?
   - Yes
   - No
2. How many difficulties should there be?
   - 1
   - 2-3
   - 4-5
3. Which setup looks more appealing?
4. How many attempts should the player have before they reach game over?
   - 1
   - 2-4
   - 5+
5. Should there be personalised features?
   - Yes
   - No
6. How many challenges should each difficulty have?
   - 5-10
   - 11-20
   - 21-30
   - 30+
7. How much do you agree with the following statement? "Background music is crucial to keep the player interested?"
   - Agree
   - No opinion
   - Disagree
8. Which challenge do you prefer?
   - Multiple choice
   - Fill in the blank

I gave a questionnaire to 5 different people from a range of ages within my stakeholders. Some of the participants did have trouble understanding some of the questions such as "How many challenges should each difficulty have?" because in reality the game would have unlimited questions of each difficulty if I had the time but this question was more to get an understanding of how many challenges the stakeholders wanted to see in the prototype.

Question:

1. Do you like Simulation games?

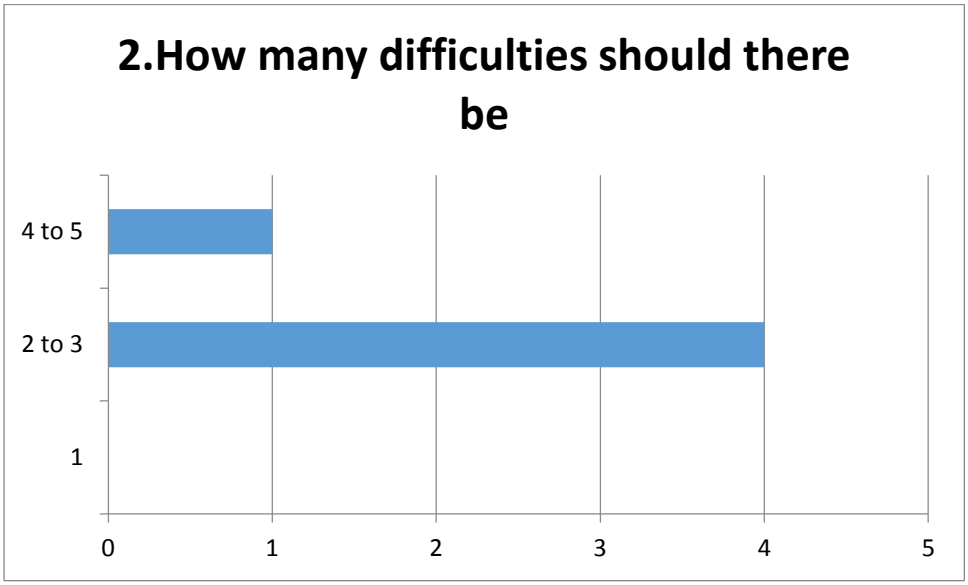
Justification: I have asked my stakeholders if they enjoy simulation games because then I could get an idea of whether my sample, who were mainly chosen randomly within my stakeholders age range to be representative of the general population, like simulation games in general and whether my game would generate interest with my stakeholders.



Summary: This shows me that making a simulation game would generate interest in my stakeholders and that they would be willing to play it whether it's for leisure, revision or to learn a programming language. That said, I will continue to create a simulation game as it would solve my problem that interest in computer scientists need to be created within the youth.

2. How many difficulties should there be?

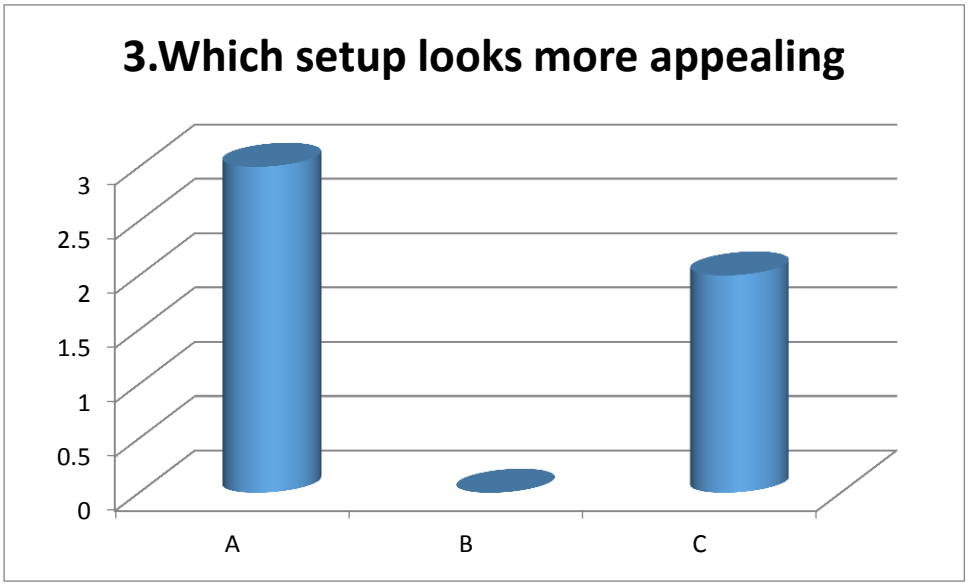
Justification:
I was uncertain about how many difficulties to have in my game ,because of my large age range of stakeholders, or whether or not to have any different difficulties at all so I asked my general stakeholders how many difficulties they thought the game should have.



Summary: This led to the conclusion that I should have 3 difficulties because the feedback I got showed that my stakeholders thought 2 to 3 difficulties would cover what my age range of users should learn.

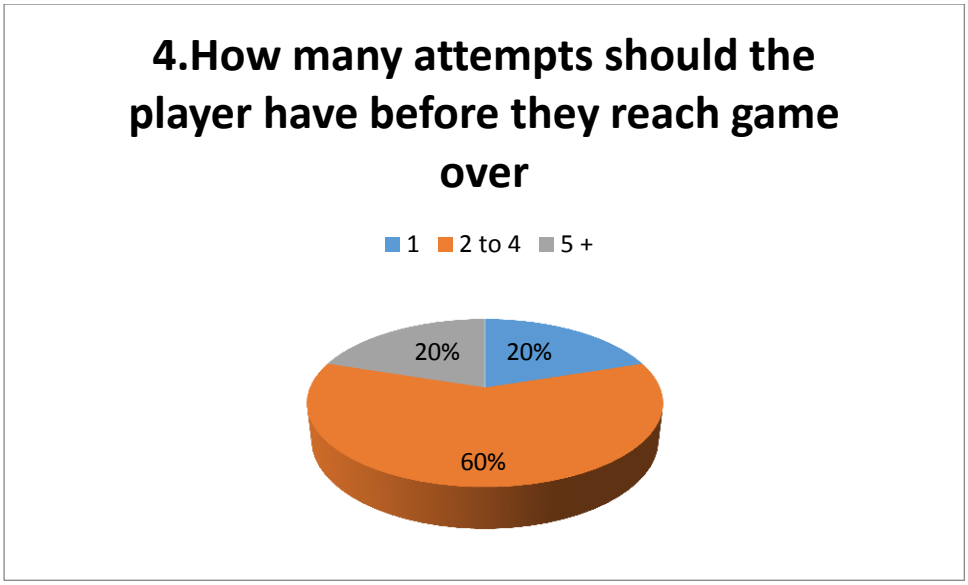
3. Which setup looks more appealing?

   Justification: I asked this question to see if my stakeholders liked the designs I had created and which of the 3 computer designs they would have wanted to see in my game. I asked whether they liked the designs i had created verbally and the general feedback was positive.



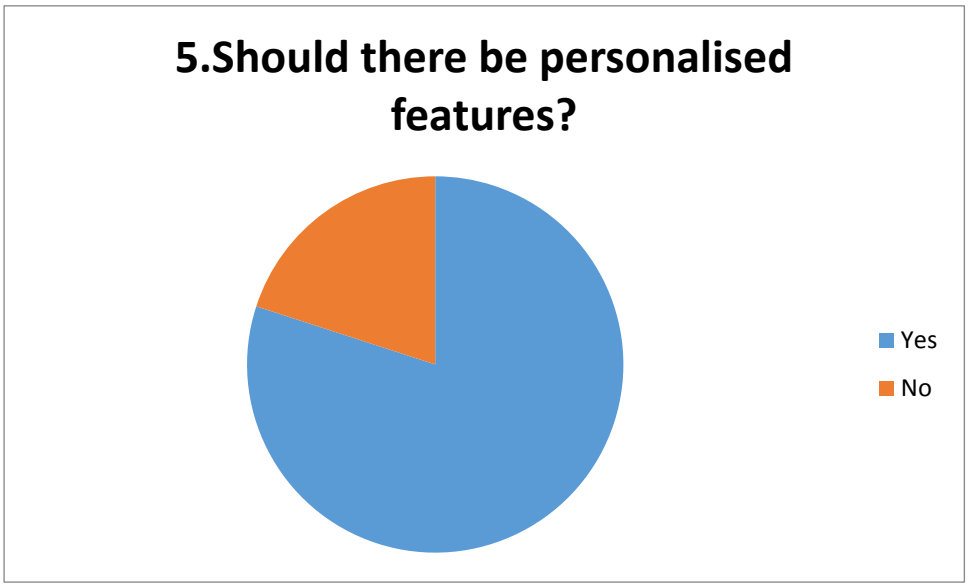
   Summary: It was between option A and Option B so I have decided I am going to use option A because I preferred the design myself.

4. How many attempts should the player have before they reach game over?
   Justification: I asked this question to see what a fair number of lives would be in my game but also to see if the divide of opinion was great enough to make the decision of creating different difficulties in my game.



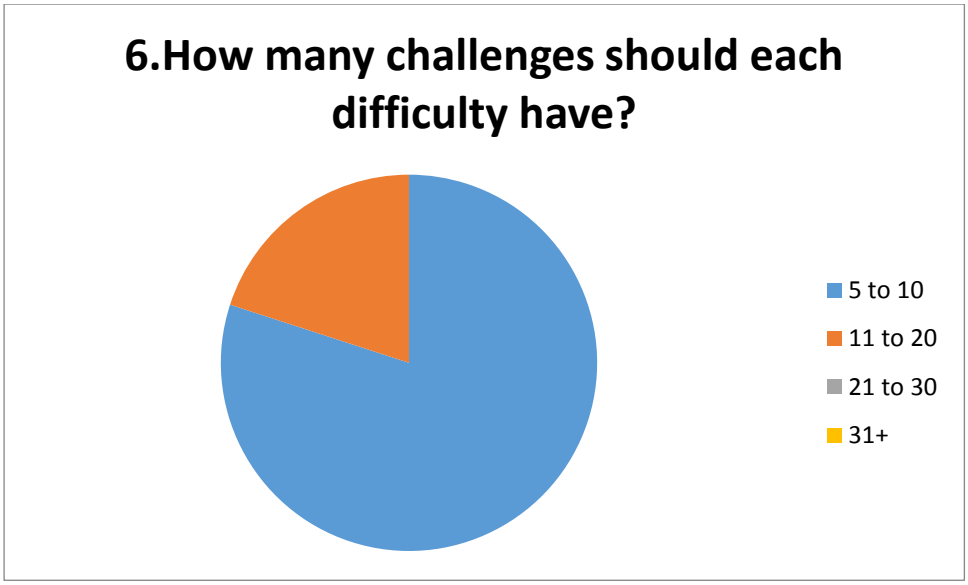
Summary: It was a pretty divided opinion as I anticipated which led me to the conclusion of creating different difficulties.

5. Should there be personalised features?
Justification: This was to see if my stakeholders like the idea of personalisation and whether it would be worth adding this aspect to my game.



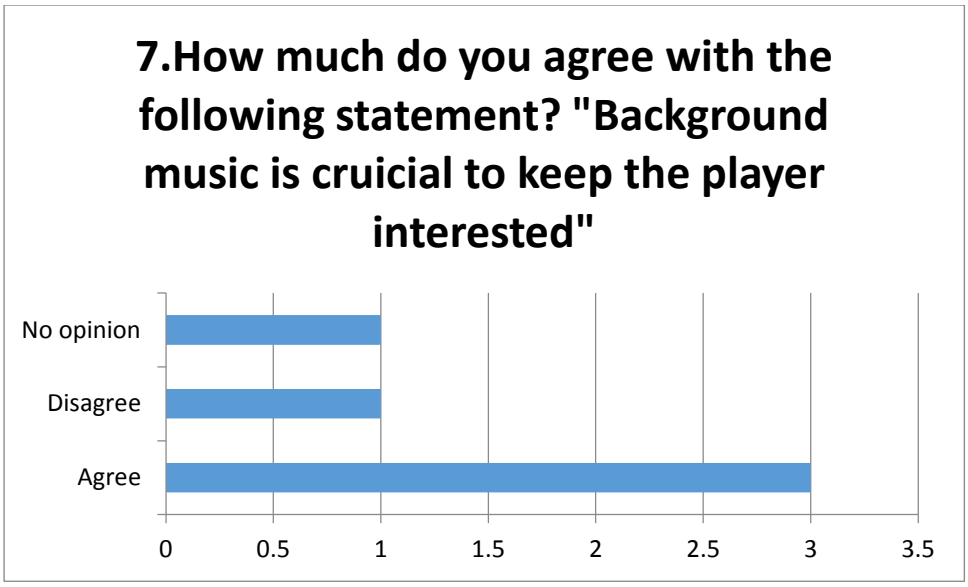
Summary: A majority of my stakeholders clearly wanted personalisation so I have decided that my game will have some form of personalisation because the results show that it is a feature my sample of stakeholders find interesting.

6. How many challenges do you think each difficulty should have?
Justification: I chose to ask this question because after anticipating that my game would need difficulties, I wanted a finite number of how many challenges I would have to put in my prototype for it to be deemed sufficient in the eyes of my stakeholders. In reality, the game would have endless or randomly generate questions using templates.



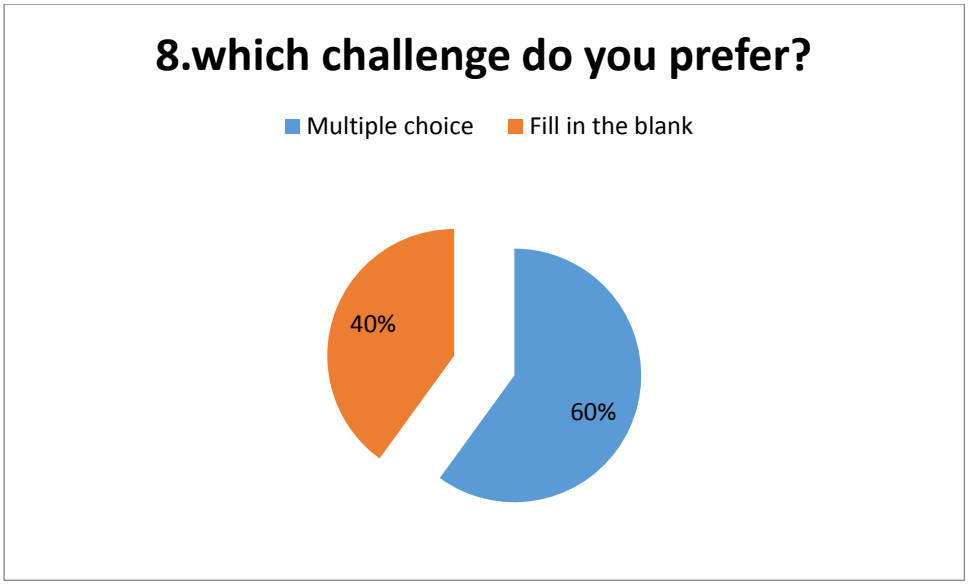
Summary: It was split between 5 to 10 and 11 to 20 so I have decided I will fill each difficulty with at least 10 challenges.

7. How much do you agree with the following statement? “Background music is crucial to keep the player interested”
   Justification: I added this to the questionnaire because I wanted to know whether it would be worth adding a background music element in my game.



Summary: This led to the conclusion that my stakeholders would prefer to have background music rather than not so I have decided to add background music to my game.

8. Which challenge do you prefer?
   Justification: I wanted to get a general idea on which type of challenge my stakeholders like to balance the game in difficulties so the easier difficulties could possibly have the more favoured challenge with the harder difficulties having more of the least favoured as it would be a greater challenge that way.



Summary: My stakeholders prefer multiple choice but it is very close between which one they prefer so I am likely to have an even mix for all difficulties.

Interview:

In the following interview, I asked M.Miah about what her thoughts were in the following questions.

What features do you think would interest the target market?

"Customisation, big achievements that have meaning behind them. There should be more explaining and simpler language used when teaching the basics. It should be colourful but not distracting and it should highlight key words"

A proposed feature of the solution is a "book" feature that the user can click on should they need help. What are your thoughts?

If you do the book feature, it should be separated into sub headings where the player can easily find what they're looking for."

Do you believe adding different difficulties would be useful for this target audience?

Yes.

How simple should the beginner level be?

Based on the target audience, the beginner level should mainly teach them the basic function and develop an understanding of what the language does and what it's used for to let them get familiar with the topic because when I was in year 9, a lot of us would just type thing without knowing what thing did or anything like that.

Should the challenges I put the player through be fixed questions that are pre made by me or randomised by the game itself?

Randomly generated as it would stop the game from getting too repetitive.

Interview review

From this interview, I have consolidated the idea of implementing 3 difficulties as well as adding the book feature that I believed would be a good method to teach my stakeholders how to write in HTML and CSS. In addition, I will add customisation features by allowing the player to enter a nickname in the beginning and customising an avatar before starting the simulation that will appear in the interface along with the questions.

Observations

I also asked her about what she thought of existing solutions so I could emulate the features my stakeholders would find useful and the ones I should avoid creating when developing.

Mimo-

What did you like about Mimo?
It's easy to understand and I like the fact that it was multiple choice. I also found that it was user friendly.
Did you like the learning environment created by the App?
It can be a bit confusing for people who don't know what they're going into because I don't feel like it explained what I was doing when I was using the app.

How would you adapt the app to suit younger audiences?
I would add more pictures and colours to keep them interested and use diagrams and examples for visual learners.

What did you think about the progression of the game and how they don't allow you to proceed further into the questions?
For beginners it's good because it should be easy for them but with others it shouldn't to add the extra level of difficulty

Solo Learn:
First thoughts:
I liked the use of bold keywords and the analogies they used that helps with learning but with the infinite chances, they don't add a note to help you get the question right.
How would you adapt it for a younger audience?
The language they used was quite complex so for younger audiences, they might not understand any of this.

Any final thoughts-
I learned a lot from Solo learn but it's clear it's aimed for Young adults. I think the idea of achievements was great and made me feel like I was working towards something. And let me know that I was doing something right. I felt more engaged with this than Mimo because it was on a monitor rather than a phone.

One issue nowadays is that there is a lack of female developers and a possible issue is that they don't develop an interest to become a programmer or anywhere else in the field of a computer science from an early age, as a part of that demographic, how would you adapt my solution to combat this issue?

I feel personalization is key in trying to develop an interest in not only females but boys too as children like to have something of their own and stand out like this website for maths called mathletics would use

Observation review

The good features from Mimo and solo learn are that they allows the user to go through multiple choice and fill in the blank challenges and user friendly GUI. App gives beginners of the course the ability to retry questions and makes the keywords from learning pages on solo learn, bold to make them stand out to the user.

They lack the graphical features to make the learner experience easier for a younger audience and so I will implement this to grasp the player's enjoyment more.

I will also include personalisation in the attempt to develop a greater interest in the eyes of the younger female demographic to the idea of becoming a web developer or joining the computational software developing field to try to solve the sub problem of the lack in female developers in this field of work.

Interviewing teachers

I decided that the best way to find out how to teach younger children and teenagers how to code would be to ask my teachers what topics to target at what age group.

I told my computer science teacher Mr.Caglar of the different difficulties and what age group they are aimed at so he could give me targeted advice on the topics I should challenge my players on. Knowing that the challenges I will be creating will be limited to 10-20 for each challenge he said I should target:
Easy Mode: 9-12 year olds

- What variables are
- Basic Binary
- And how they correlate to become colours for css.

Medium Mode: 12-14 year olds

- Basic HTML tags
- Binary colouring
- Basic CSS

Hard Mode: 15-17 year olds

- More advanced HTML and application
- More advanced CSS and application
- Possibly JavaScript if I have time

Summary of feedback process

From the questionnaire I was able to take away that there is an interest in simulation games within my general stakeholder's age range. I also found that they believed background music is vital in retaining interest in that demographic and that because the age range of my general stakeholders is so large that I should have 3 difficulties to make it more fair for those who struggle with programming but also challenging for those who

wish to be challenged. I also learned that personalisation would engage my audience more and emerge them further into the simulator making it clearer that they are the web developer.

The interview with M.Miah showed me that my general stakeholders would find searching through the book feature I have implemented useful and further re-enforced that personalisation would aid me to engage my player in the simulation I am to develop.

The observation process with M.Miah gave me insight into how to adapt my game so that it is appealing to younger audiences to which her response was to use simple language when teaching the user how to code, using vibrant colour and implementing personalisation to spread my reach of players interested.

The interview with Mr.Caglar aided me on which topics to target for each difficulty as outlined in the interview above.

Proposed solution:

I will create a game will 3 buttons in the main menu: Start, instructions and quit. The instructions button will take the player to a screen that displays how to play the game. The quit button will terminate the game and the start game will take the player to a screen displaying the difficulty selection screen where they can choose the difficulty of their experience between Easy, Medium and hard. I will also have music playing in the background with a button allowing the player to mute the music.

After the player chooses their difficulty, they will be asked to enter their name or nickname so it can be displayed whilst they play. If they enter a name greater than 12 letters, they will be prompted to enter a name less than 12 letters. After clicking the submit button, they will be presented the opportunity to create and customise a character. The customisation options will include at least:

- 5 hair variants
- 3 or more hair colour tones
- 5 or more skin tone options
- have 3 or more body shapes
- have 3 or more body colours

After clicking the submit button, the player will be sent into the core game. The players balance starts on £100. This will consist of a basic background with a computer image displaying the challenge. The challenge will randomly choose between fill in the blank challenges and multiple choice questions. The fill in the blank questions will have a submit button to enter the answer and will not accept any answer greater than 12 characters. The multiple choice challenges will take the button the player clicks, as the submission of your answer. Every time a player gets a question correct their balance increases by £100 and will decrease by £50 when they get a question wrong. Depending on their selected difficulty, the player will lose a life for getting a question wrong.

- Easy will have unlimited lives
- Medium will have 5 lives
- Hard will have 3 lives

The screen will also have a pause menu to pause the game, mute the music or quit to the main title screen if they wish.
Once a player loses all their lives, a new screen will become active displaying the players score and prompting them to return to the main menu.
In the game itself, in the top left corner will be a book icon which will allow the player to seek hints on how to complete their challenge.

Controls for proposed solution:
The player will use the keyboard to answer questions on “fill in the blank challenges” using both the letters to complete the syntax and numbers for some CSS styling and HTML. The user will also use the mouse to select various answers on the multiple choice challenges and to select various options and select the book.

Features of proposed solution:
This gives an insight of the final features to be added to the game on why they are required and why some are absent.

Feature:	Justification
Book – learning tool	Allows the player to learn and study the syntax before attempting challenge and after failing a challenge.
Multiple Choice	Element of chance gives the users, who is not aware or prepared for the following challenge, an opportunity to still succeed. Too much complexity n the questions may prove too difficult for the target audience so making multiple choice challenges simplifies the problem
Gaining and losing Money	Reward system of the game as it encourages and entices the player to do better and learn from their mistake. Teaches the target audience that there are consequences to creating a faulty system
Losing lives on harder difficulties	This is what creates a game over state in my game. When a player loses all of their lives, they will be shown with a screen with their score a game over message.
Fill in the blank	Gives variety to game as having only one “type” of challenge may prove too easy for the target audience and as the other type of challenge is multiple choice, it is already easy as it is
Pause menu	Allows user to have a break from the screen as they may be interrupted to do something else
10-20 challenges for every difficulty	Due to the limited time, there are restriction on how many challenges there are. Ideally it would be unlimited or a much larger quantity but my skills as a programmer restrict me from generating random challenges
Personalisation	Many stakeholders of that age find personalisation very engaging and that is the goal for my game. This will include allowing my

	player to create their own avatar that will be displayed in the game and also having their name displayed on a name plate to immerse my younger range of stakeholders.
Music	Background music can help keep the player interested but can cause an annoyance so there will also be a mute button to stop it.

Limitations of proposed solution:
The main restrictions for my proposed solution are caused by the time limit and my personal ability in coding as I do not have the time to complete various features that my research proved useful to learn a language such as the code playground on Solo Learn but also, I am not yet skilled enough to program something that complex.

Personalisation is a feature I will try to implement in my game but it may prove too difficult to develop properly in my game. Whilst my game may have fractions of personalisation features, the quality and depth of personalisation are limited to my abilities.
My game will not be able to randomly create questions as I lack the programming ability to do so but would be effective when teaching children how to code.
Due to time restraints, other than being an abstraction of monetary value, the money will not have a purpose such as to purchase different backgrounds or unlock any different personalisation options although, it is an attempt to demonstrate a progression scheme in my game/simulation. However, if I do have more time after completing the development of the core game, I will try to add this feature to my game.

Hardware requirements:

Hardware	Justification
Monitor	**Provides display to player to answer and solve challenges**
Keyboard	**Required for "fill in the blanks" challenges**
Mouse	**Used to answer multiple choice challenges and select various options such as the book option or to quit**
Speaker or headphones	**To output the in-game music and sounds**
1GB RAM	**Many computers have at least 1GB of RAM and the game will not be very big so not a lot of data is stored**
CPU with SSE2 set support: **Dual Core 2.5GHz**	**SSE2 is the instruction set unity games use to store data in registers and perform calculations and make decisions so it is required to run the game**

GPU with DX10 or higher **1GB VRAM**	**GPU required to stores processed information of graphics and images used in my game.** **DX10 is required by the game engine I am using as it has is the oldest model of DirectX with the Shader model 4.0 making it quicker for rendered graphics to be sent to the GPU from the CPU.** **Many computers have a VRAM of 1GB or more but as the graphics I am using are 2D, less may be work fine for the game**
HDD or SSD of at least 2GB	**The game will consist of many images, sounds and animations for the challenges and money counter.**

Software requirement:

Software	Justification
OS: **Windows 7 or Higher**	**I will be making the game for windows only, as this project is time restricted and by making my game for more operating systems further tightens my schedule. The OS provides my target audience with a graphical interface to select and play the game. I have chosen windows as it is the OS my target audience and I are most familiar with as Windows is the most popular OS for computers to have so it is more available to my target market and stakeholders.**
Appropriate drivers for: **Mouse** **Speakers/Headset** **Keyboard** **Internet Dongle**	**These are the basic drivers required to use a computer and play my game as these drivers tell the OS how to communicate with and control the devices.** **The driver for the Internet Dongle is so that the user can go online to download the game if their device doesn't have a built in component that allows them to connect to the internet**

Success Criteria

Success Criteria	Justification
1. **The game will be created in the 16:9 aspect ratio.**	This is the most common aspect ratio to use and working with one aspect ratio makes it easier for me to focus on the more important aspects of my game.

2. Must allow User to choose from different difficulty levels. More specifically, the game must have 3 difficulties. Easy, Medium and Hard	Many children will start on a different level to others. This also allows the player to learn at their own pace. My questionnaire justifies this decision.
3. Must have a form of personalisation e.g. Game must allow player to create a basic character	From my interview and observation process, it was clear that personalisation in my game would keep children interested in the game as children like to personalise characters making them more emerged in the experience. My stakeholder said they wanted a form of customisation and therefore justifies this success criteria.
4. The personalisation should have at least 5 hair variants	I think 5 hair variants gives my user plenty of options to choose from that could appeal to all genders and gives enough opportunity to show I can create a personalisation feature in my game/simulator. My stakeholder said they wanted a form of customisation and therefore justifies this success criteria.
5. The personalisation should have 3 or more hair colour tones	I think 3 hair colour tones gives my user plenty of options to choose from that could appeal to my stakeholders for basic customisation. If I have more time I may add more hair colour tones. My stakeholder said they wanted a form of customisation and therefore justifies this success criteria.
6. The personalisation should have 5 or more skin tone options	5 skin tones will allow me to generalise the spectrum of skin colour into enough different variants that my players can choose from. My stakeholder said they wanted a form of customisation and therefore justifies this success criteria.
7. The personalisation should have 3 or more body shapes	This will give my player the choice to change what shape their body is. My stakeholder said they wanted a form of customisation and therefore justifies this success criteria.
8. The personalisation should have 3 or more body colours	This is to further incentivise younger children to play as it add more imagination and personalisation options to my game. My stakeholder said they wanted a form of customisation and therefore justifies this success criteria.
9. Must have a progression system/scheme	Progression is a very crucial feature in a game as it leads the player to believe they are working towards something. I will be using money as a form of progression as my game is an

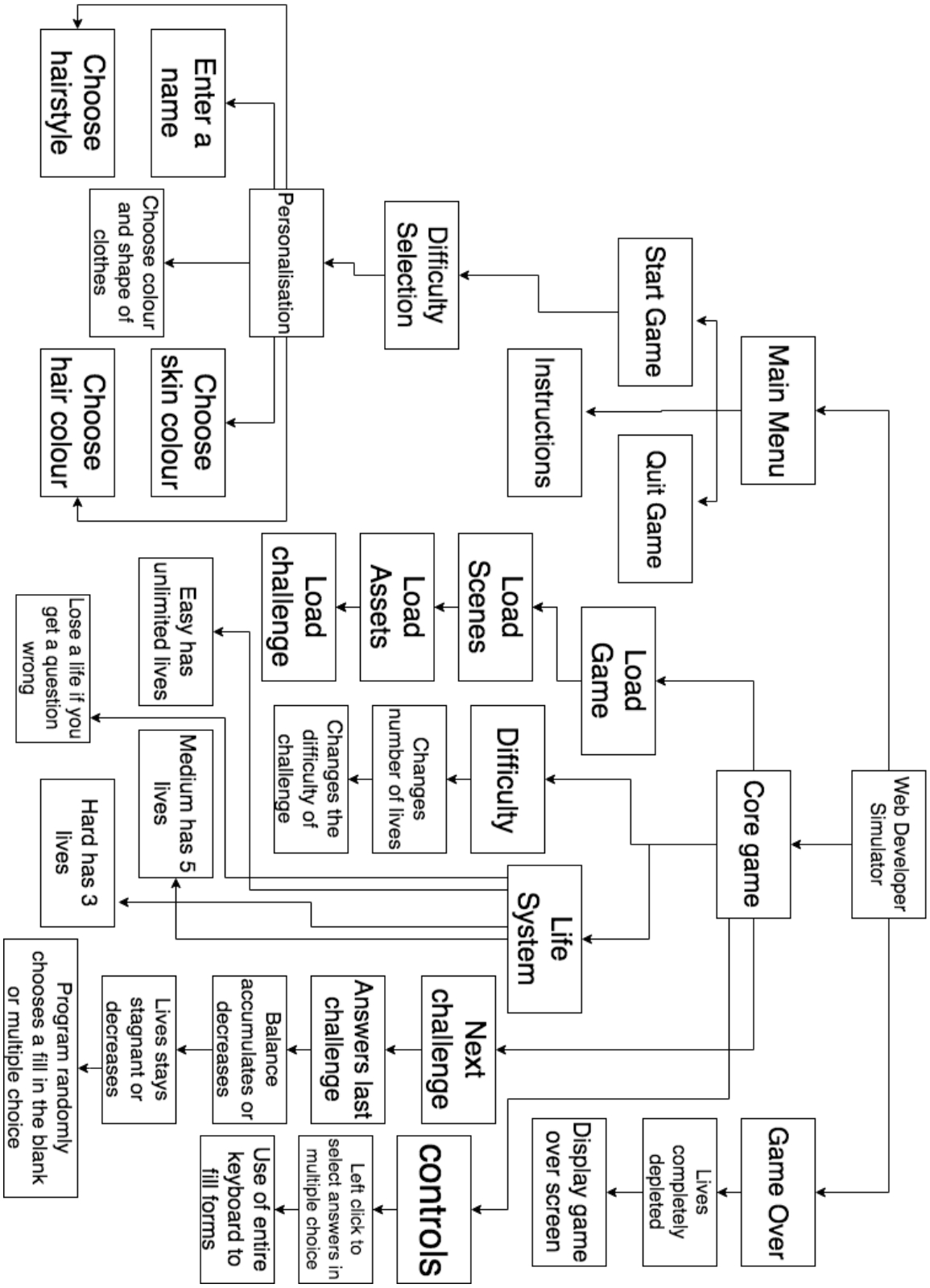
	abstraction of a web developers life and in the real world, wealth can show progression of sorts. The observation process justifies this.
10. If a player gets a question correct, their "balance" should increment by £100	This is an abstraction of getting paid in the real world is like. If a real web developer does their job correctly, they get paid and so if the player completes a challenge then they too shall be paid in the form of in game currency. This will be the form of progression.
11. If a player gets a question wrong, their "Balance" must decrement by £50	This is an abstraction of consequences in my game of what may happen to a web developer if they create an error in code for the company they work for. This will be the form of progression.
12. Must have background music	Background Music will not only keep the player interested but may also increase their levels of concentration. My interviewee told me that it is important that a game has background music especially to keep the children wanting to play the game. This is backed by my questionnaire. It is the lowest priority of my success criteria as the base of the game must first be completed to ensure I complete the coursework within the time frame.
13. The book feature must be working and provide useful tips that would help the player in the challenges	The book feature is one of the main features I am adding in the game as it is an abstraction of researching that web developers may do to learn how to code a certain feature into their website. Justified by my research with Mimo and Sololearn as it provides the user with a new learning experience that may prove more effective.
14. The book feature must have 2 buttons that allow the player to change pages back and forth to their liking.	This is a simple design that will allow my user to easily skim over the book to find the information they are looking for demonstrating the computational method of abstraction.
15. Game must adapt challenges to the selected difficulties	From my interview process, it was clear that people from the variety of age range that my stakeholders are, that different difficulties are a requirement as children will all start from different stages. Some more gifted than others. Thus the game must have challenges of different difficulties to care to that.
16. Game must allow user to pause and have a pause menu	As my stakeholders are children between 9-14 years old, there may be a need to pause the

	game for example to go answer the door. This will also allow the user to pause the game from timed challenges on harder difficulties. This is justified by my choice of stakeholders when beginning this project.
17. Game must deplete lives of user if they get question wrong on harder difficulties	Should the player choose a more difficult level, my game will become harder with the risk of losing from getting too many wrong in a row. This is justified by the questionnaire to make more difficulties.
18. Game must allow user to attempt challenges infinitely on easy difficulties	Should the player choose an easier difficulty, they will be given infinite chances to answer as they are more likely to be younger. This is justified by my questionnaire as my stakeholders outlined the importance of difficulties.
19. Medium level must have 5 lives to begin with.	To make the game more challenging for my player, giving them 5 lives as opposed to the infinite amount easy mode has means that the player is more cautious about the mistakes they make and those who like the challenge will find the game more enjoyable. This is justified by my questionnaire as my stakeholders outlined the importance of difficulties.
20. Hard Level must have 3 lives to begin with.	This makes the level more difficult than medium mode and will make it more challenging specifically for the older age range of stakeholders. As someone apart of that age range, I think that 3 lives is enough to be challenging on players/users but is fair enough based on the difficulty of the questions. This is justified by my questionnaire as my stakeholders outlined the importance of difficulties.
21. Game must have working multiple choice challenges	This means that the player should be able to choose and submit an answer with the game responding accordingly. This is a good choice for challenges as my stakeholders are quite young and I do not expect them to be able to code without aid. This is justified by my research on existing current designs with Mimo and Sololearn both using this method to teach the user.
22. Game must have working input fields to fill in the blank challenges	Very similar to the multiple choice; This means that the player should be able to choose and submit an answer with the game responding accordingly. This is a good choice for challenges as my stakeholders are quite young and I do not

	expect them to be able to code without aid. This is justified by my questionnaire as my stakeholders outlined the importance of difficulties. It's also a simpler version of solo learns code playground I found in my research to be effective so I have decided I will implement it.
23. To validate the input fields, I will allow my user to enter no more than 12 characters of the English character set	This is because there will be no need for my user to enter values or answers of over 12 characters but also will require my user to use certain character that are not alphanumeric such as "<" and ">" to answer specific questions.
24. Game will teach syntax of HTML and CSS	The game is an abstraction of a web developer's job with which they work with HTML and CSS. They also work with JavaScript but I believe my stakeholders will be unable to grasp the concept of JSS and thus I will not be teaching them this syntax unless I have more time than I anticipate.
25. Game must have working Start screen allowing user to begin or quit	This is a requirement in most games to terminate the game itself if in full screen mode
26. Game must send player to start screen if they run out of lives	This is to stop the player progressing for making too many mistakes. By doing this for the more difficult levels, it increases the difficulty and makes the game more satisfying for completing challenges without failing.
27. Game must allow user to write their name and display their name in-game	This is to add more and more personalisation features and engage my stakeholders into the game more. This is justified by questionnaire and interview with M.Miah
28. Game must displays the players character in the game	This is to add more and more personalisation features and engage my stakeholders into the game more. This is justified by questionnaire and interview with M.Miah
29. The game must validate the length of the players name so that it only allows a name no longer than 12 letters.	The average US first name is 6 letters long so my game will allow players 12 characters to write names so that it is short enough to fit in the screen but also long enough to allow any longer names if needed.

B. DESIGN

Systems diagram



Justification of systems design

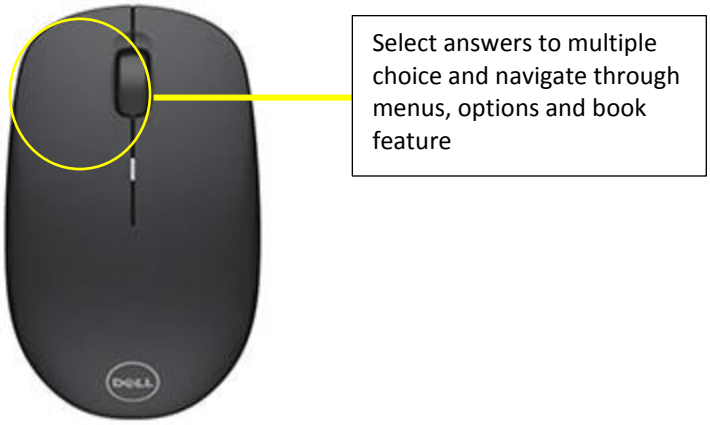
Main Menu: This will consist of the menu screen and loading the game. The menu screen will include the start button that begins the game, the exit button to terminate and the instructions page to guide the user on the components that are used in my game. The start button will then lead the player to make a choice of which difficulty they wish to play in. Once decided, the game will begin its load process where it will retain assets such as the games music and play on a continuous loop, the components of the game such as the customization of the character and allow the player to enter a name of up to 12 characters as well as the graphics of the program itself. It will also load up the other main elements of my game such as the lives and money of the player. The lives of the player are affected by the difficulty. Easy mode will allow for unlimited lives, medium will have 5 lives and hard will have 3.

> Personalisation *in depth*- Will consist of player's name that they add in beginning of the game which will be displayed whilst players complete challenges. They player will also customize an avatar before beginning the challenges which will also display on the screen. This is to combat the issue of making the game more interesting by allowing more customization to emerge the player in the games environment. The player will be able to choose between a selection of hair styles, hair colours, skin colours and colour of clothing.

Game over: This occurs when the player either quits or all of their lives are depleted. If all of their lives are depleted then the game will portray a game over sign.

The core game- Split into 5 sub-systems:

> Load Game: This is the process of loading the challenge, music, features such as the book and the score to 0 so when the game loads, the player can play.
>
> Difficulty: The game will have 3 difficulties to give the player a variety of questions. The game will be split into Easy, Medium and Hard modes which are easy to distinguish for my young target market as well as the older range of children my stakeholders consist of. My game will change certain settings depending on the difficulty selected. For easy mode, the player will only receive questions from a specific list containing questions for easy mode and will have unlimited attempts to answer them. Medium will be challenged using the easy and medium mode questions and will have 5 lives to answer them. Hard will have access to all the questions from the various difficulties but will only have 3 lives.
>
> Life system: Player's lives are allocated based on their difficulty. They must lose all lives in a row to reach game over state e.g. on hard difficulty, if a player plays on "hard mode" then they will have to get 3 wrong in a row to get "game over".
>
> Next challenge: This will only occur once the question has been answered. Once answered, money is accumulated in the player's score. The game will then check what difficulty the player chose in the beginning and load a question based on their difficulty.
>
> Controls: This will consist of all the alphanumeric characters on the keyboard as the player will have to fill in forms to answer "fill in the blank" questions. The player will also have to use the mouse's left click to select answers for multiple choice questions.



Structure of the solution

The player enters a menu screen upon opening the program. The menu screen will consist of 3 options; Start game, instructions and Quit. Quit will leave the game and terminate the program. The instructions page will provide basic information on the games interactive features, the controls, how to advance in the game and the difference in difficulties. By pressing on the start button, the player is prompted to select a difficulty. Then they will personalise an avatar.

Once the game begins, the user is shown the main scene and will be asked a question (the player will have availability to questions of their chosen difficulty). The screen will consist of an object the challenge is portrayed onto, an object displaying the money the user has gained, the number of lives the user has (based on difficulty) and the “Book feature”.

By answering the question, the user will move onto the next challenge. By answering correctly, their money will accumulate. By getting an answer wrong, the player loses a life and loses some money if the player’s difficulty is set above easy.

The player will answer questions by clicking on them in multiple choice questions or by filling out mini forms to type an answer in fill in the blank questions.

The book feature is how a player learns to answer question. It is meant as an abstraction of researching that web developers may need to go through to help them solve problems. By clicking on the book icon, the player can identify the information that will help guide them to the answer of a question they are stuck on.

The game will end when the player either; clicks on the menu button in-game and clicks return to title screen or the player is out of lives. If the player is out of lives then a game over screen will overlap the interface where the player will be prompted to return to the title screen.

Algorithms

Note: All variables and functions/procedure names are placeholders and subject to change throughout the development of the game

Main Menu – includes the algorithm for selecting a difficulty

```
Procedure mainMenu()
        display (menuScreen)
        // displays the main menu screen with all the buttons
        startGame = 1
        instructions = 2
        // This is to portray the options the user has in the start menu
        choice = input("Which option ?")
        // In development, these will be displayed as buttons to choose
        choiceSelected = false
        while choiceSelected == false then
                if choice = 1 then
                        difficulty = 0
                        while difficulty != 1 or difficulty != 2 or difficulty != 3
                                difficulty = input ("Select a difficulty ?")
                                //The game itself will have buttons for the user to decide on which
        difficulty
                                If difficulty = 1 then
                                        difficulty = easy
                                elseif difficulty = 2 then
                                        difficulty = medium
                                elseif difficulty = 3 then
                                        difficulty = hard
                                endif
                        endwhile

                        customiseCharacter()
                        choiceSelected = true
                if choice = 2 then
                        instructions = true
                        //This is to begin scene of instruction pages
                        while instruction == true
                                display(instructionPages)
                                // displays the instruction pages
                                pageTurner = false
                                pageTurner = input(turn page?)
                                        if pageTurner = yes then
                                                turnPage()
                                exiting = input ("would you like to exit instruction")
                                if exiting = yes then
                                        instructions = false
                                endif
```

```
                else then


                endif
        endwhile
endproceedure
```

loadGame – Also includes the elements of difficulty and lives

```
//The challenges are stored in arrays based on their difficulty for the sake of representation
// as I will not be coding the loading of the game itself, the function/ procedure load(x) will be used as a
placeholder

procedure loadGame(difficulty)
        money = (100)
        load(music)
        //This must play on loop while the game is running. An idea of how it will play is using a while loop
such as
        // while x then
        //      play (music)
        load(gameGraphics)
        if difficulty = easy then
                lives = -1
                load (challenges1)
                //the variable challenges N Indicated what list of challenges are loaded. As the selected
difficulty
                //is easy, the player will only be able to access level 1 questions whereas difficulties medium
and
                //hard have a greater range of challenges
        if difficulty = medium then
                lives = 5
                load(challenges2)
        else then
                lives = 3
                load(challenges3)

        endif
        load(money)
        load(book)
endprocedure
```

Sound and music-

```
//The sounds and music will be stored as music files .mp4 possibly

Procedure playMusic(musicfile)
        x = TRUE
        while x then
                play (musicfile)
                //play(x) is the function of playing the music in the background
                // while x is used as the game should play the music in the background on a loop constantly
as
                // game is being played
        Endwhile
```

```
Endprocedure
```

Life system – Some elements are taken from other sections in the game

```
Procedure lifeSystem(answer, difficulty, money) // by reference//
    If difficulty = easy then
        MaxLives = -1
        lives  = -1
        //lives is a variable as it will change
    If difficulty = medium then
        MaxLives = 5
        lives  = 5
    Else then
        MaxLives = 3
        lives = 3
    endif

    if answer = True then
    // answer = true means that the answer passed in the arguments was correct and vice versa for false
        money = money + 100
        display (nextChallenge)
    else then
        //The player has gotten the question wrong
        lives = lives-1
        if lives = 0 then
            money = money – 100
            // decreases money for getting answer wrong
            endgame()
            //This function will show game over
        else
            money = money – 100
        display current challenge
        endif
    endif
endprocedure
    // Note this only highlights the life system aspect of the game and does not show how the challenge
    itself is replayed
```

END GAME-

```
Function Endgame():
    Display(gameover screen)
    If back to menu button is clicked then
        Load main menu scene
```

Next Challenge-

```
function nextChallenge(difficulty)
    if difficulty = easy then
        currentChallenge = selects challenge from easyArray
        remove currentChallenge from easyArray
        return currentChallenge
    if difficulty = medium then
        currentChallenge = selects challenge from mediumArray
        // mediumArray will have questions from medium difficulty
        remove currentChallenge from mediumArray
```

```
            return currentChallenge
        if difficulty = hard then
            currentChallenge = selects challenge from hardArray
            // hardArray will have questions from  hard difficulty
            remove currentChallenge from hardArray
            return currentChallenge
        endif
endfunction
```

Controls – N/A

N/A as the controls are mouseclicks and Alphanumeric characters

customise Character-

```
function nextChallenge(name) //by reference
    while name.length >13 then
        name = input("input a name under 12 characters")
    endwhile
```

Character creator –

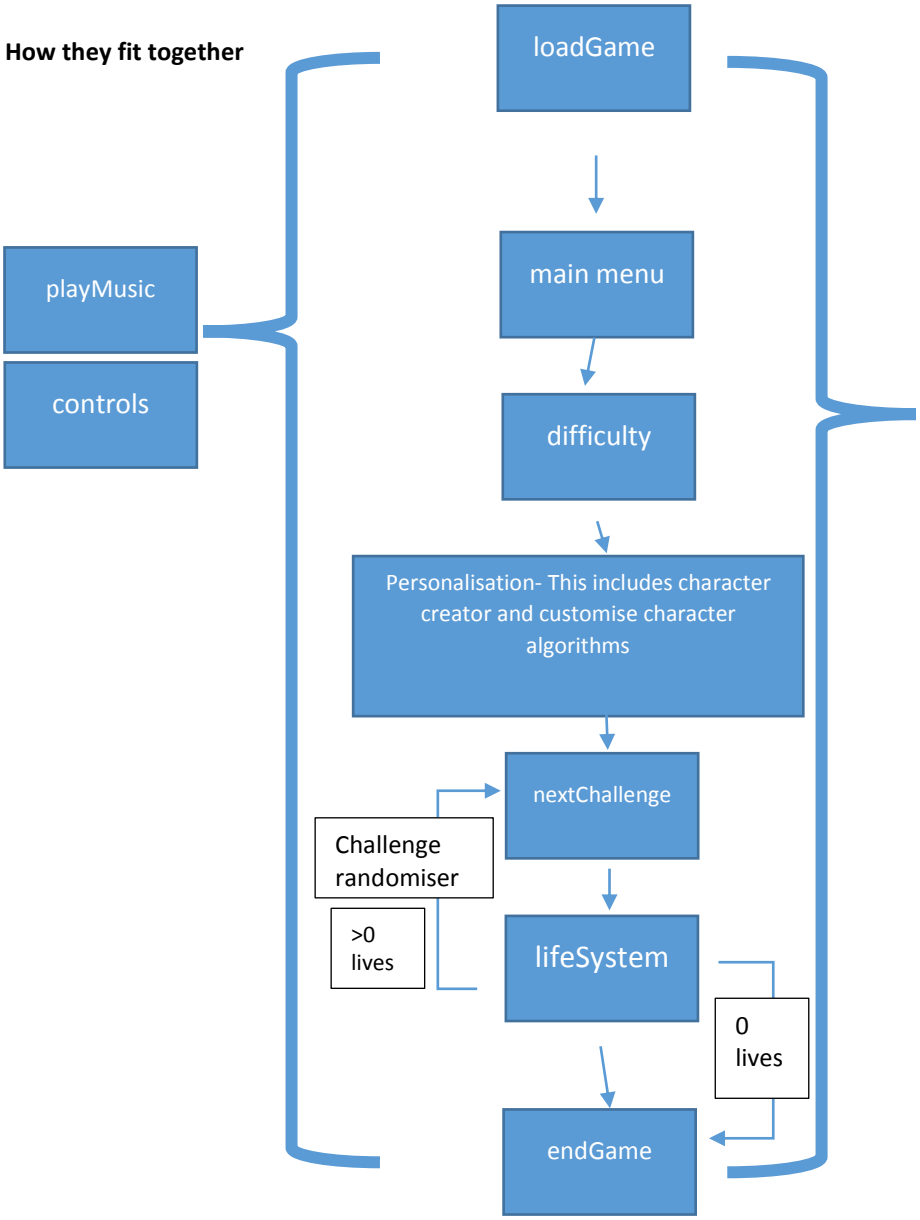
```
//Parameters will be passed for the array such as the hairstyles array or the hair colour array
procedure selector(array)
    pointer = 0
    display (array[pointer])
    customisationscreen = true //this means the user is on the customisation screen
    while customisationscreen = true then
        if nextbutton is pressed and pointer != array.length then
            pointer ++
            display (array[pointer])
        if prevbutton is pressed and pointer != 0then
            pointer --
            display (array[pointer])
        endif
        if submit is pressed then
            customisationscreeen = false
        endif
    endwhile
endprocedure
```

RANDOMISE CHALLENGE -

//*Randomises challenge between multiple choice and fill in the blank*

```
Function randomisechallenge():
    challengeSelector = random number between 0 and 1
    if challengeSelector is equal to 1 then
        display(multiple choice question)
    else then
        display(fill in the blank challenge)
Endfunction
```

How they fit together

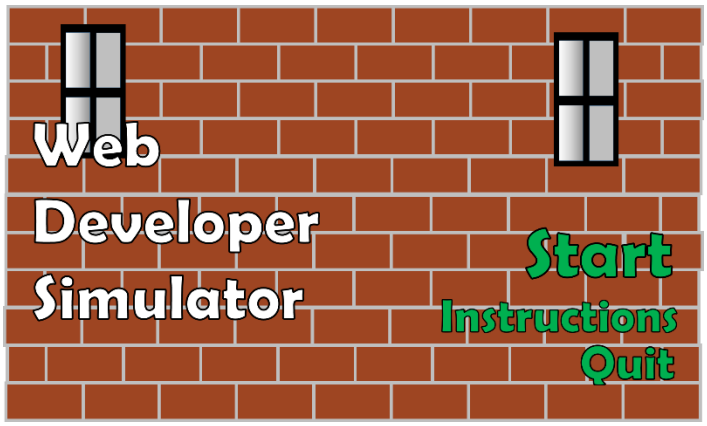


Structure of development

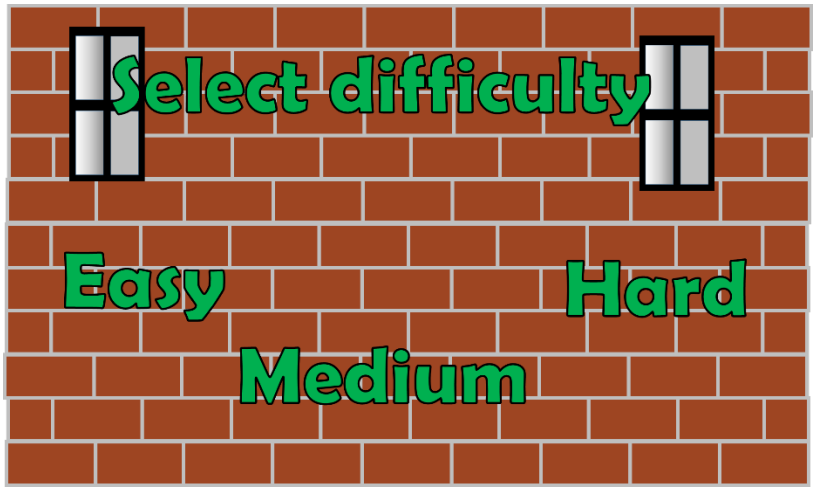
This is the order in which I will develop my game to ensure that I have a functioning game within the time constraints of the projects deadline.

-Main Menu
-Fill in the blank questions
-Multiple choice questions
-Pause Menu
-Book feature
-Instructions
-Character customisation
-Music

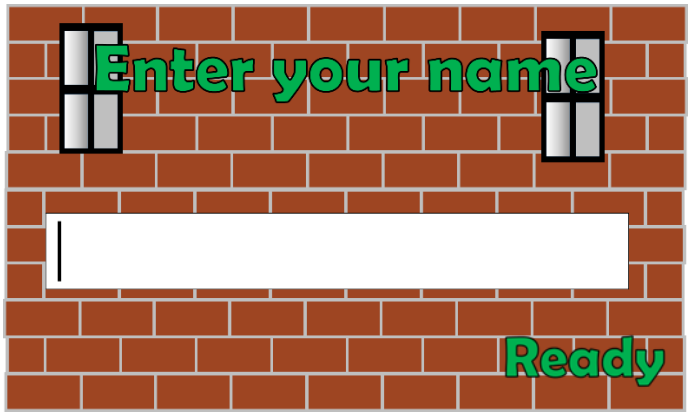
Usability features & graphics



Start screen – Very simple for stakeholders to select an option
Clicking instructions leads to page where basics of how to answer questions are displayed and how to receive help using the book/research feature
And finally, quit will terminate the application.
The graphics are more colourful than real life because it will be more appealing to my stakeholders



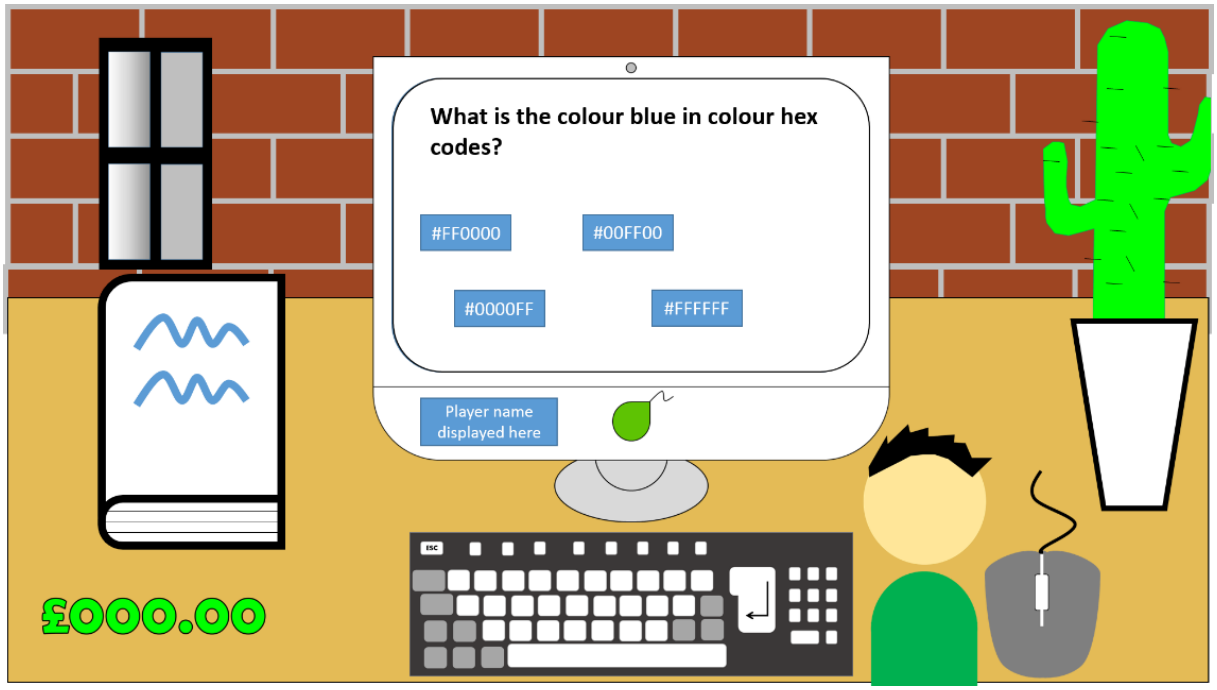
Selecting difficulty – Simple interface for stakeholders to make it easier to navigate before entering the game.



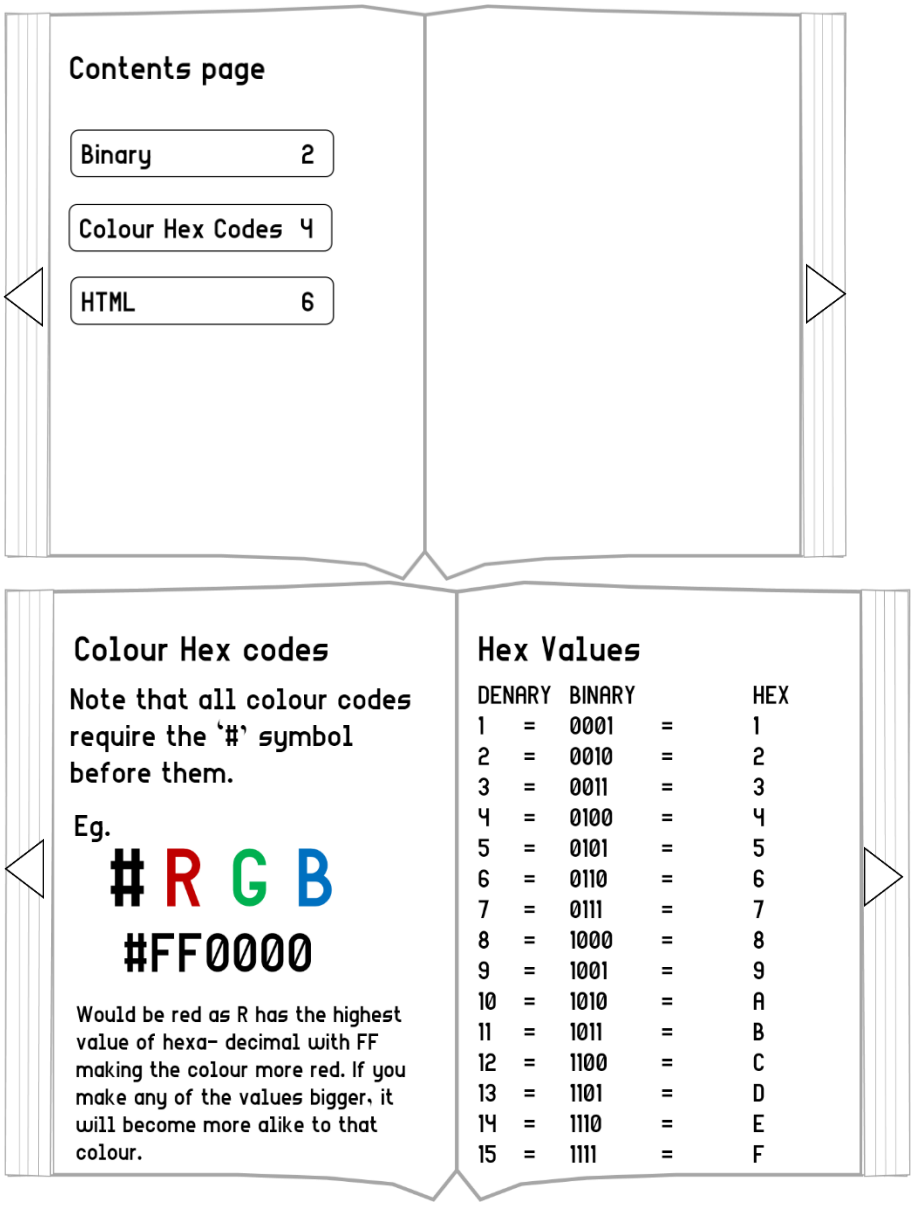
After selecting a difficulty the game will prompt to ask of the player's name which will be displayed in game. The ready button will submit the name storing the value in the variable

Then the player will have the chance to create a character or avatar which will also appear in game. The user can then click ready to begin the game. I have used consistent fonts to make it looks more organised. The white button on the brown background will make it stand out and is understandably the mechanism on how to change certain features.

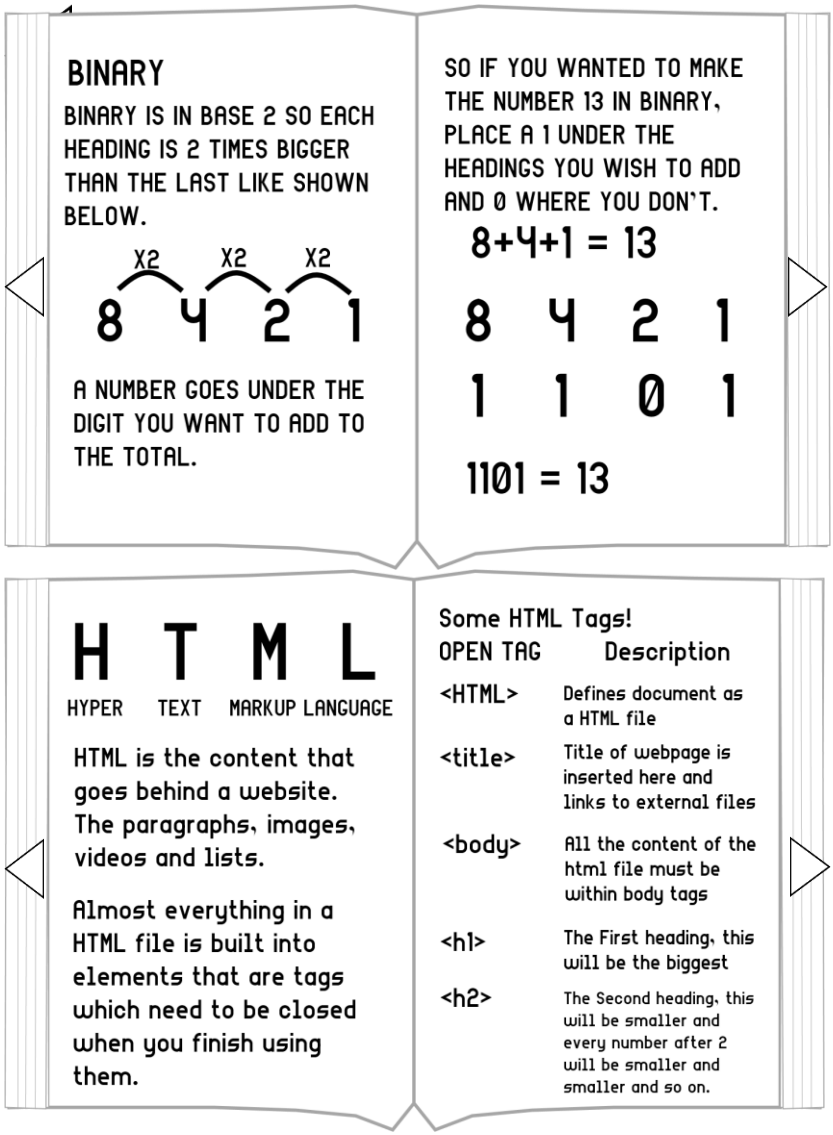
Simple triangles will be used to navigate between options allowing the user to create an avatar with ease as my stakeholders do contain a young demographic. Black outline with a white fill means it will contrast to the background colours I have used



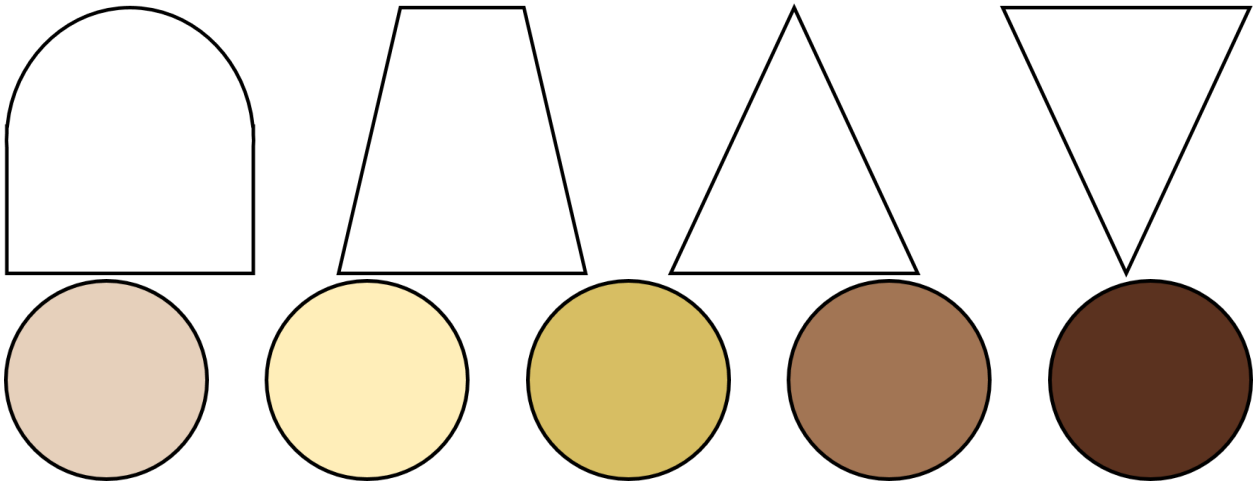
Colourful vibrant background and objects to keep stakeholders attention. Includes book feature to help the player using a comprehensive icon so the player understands what it is. Shows the player and screen displaying the challenge. Additionally, the main game screen displays player name and bank balance. This is a classic case of a multiple choice question. The player will select the answer to the first one and then the next one. If they believe the answer they put for the first question was wrong then they click on their answer and it will go back into the selection of answers.

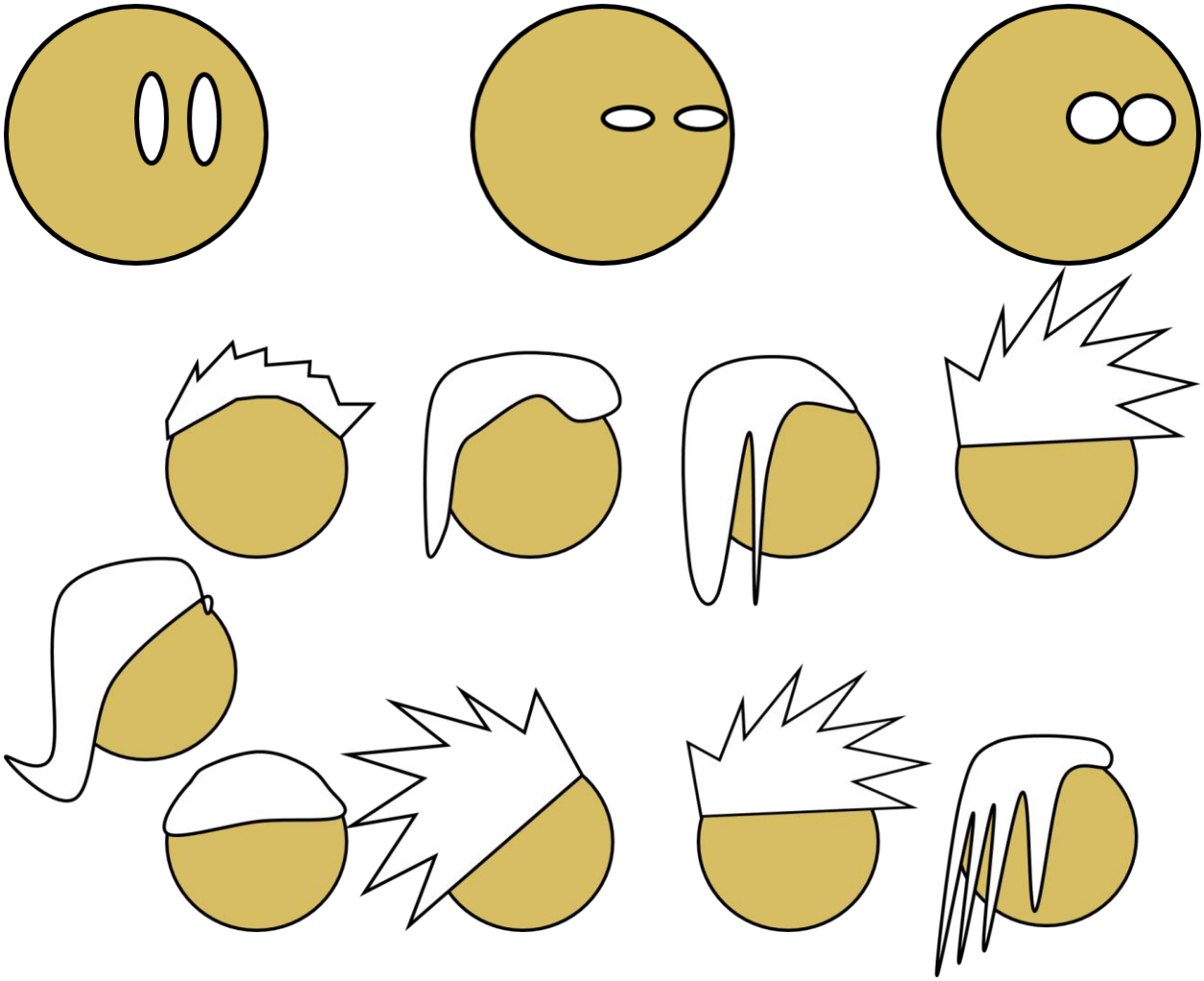


This is a very simplistic design of what will occur when you click on the book feature. If given enough time towards the end of developing the game, I would add a contents page upon opening the book to make it quicker for the player to find the information they need. There has to be medium sized spacing to make the information easy to read and digest to make learning a better experience for my stakeholders. The directional buttons are a simple design allowing the player, my stakeholders, to change pages.

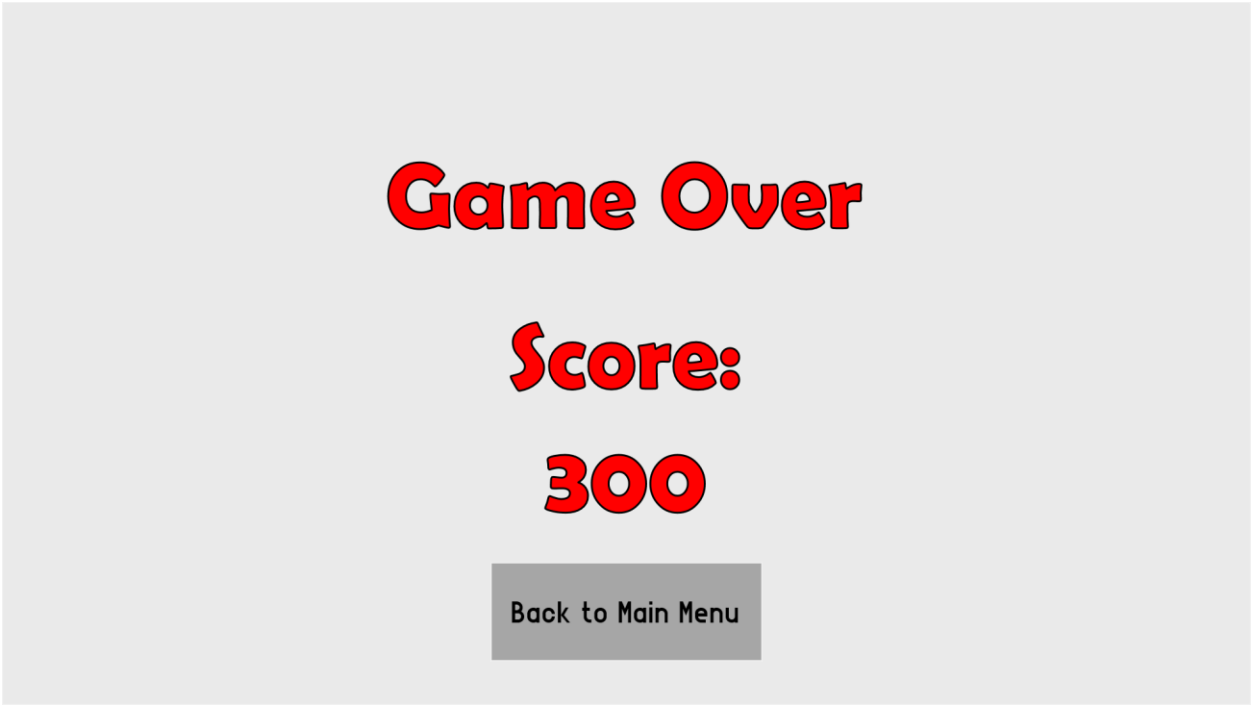


I have created 3 pages for the time being as it will cover the topics I will be asking about in the game.





Here shown are the eye shapes, complexion, hair styles and body shapes that I will use in my personalisation screen so that my user can create this character. They are cartoony to appeal to my younger stakeholders but not to childish to make my older stakeholders not want to play.



These are the gameover and paused screens. The red is to represent failure which I have added to both to try to convince my user to continue to keep going rather than quitting. They will be transparent so the background will be partially visible behind these.

I will not have an instruction screen till after the development of my core game as I will screenshot the game to create instructions making it easier to learn how to play the game.

Easy Mode:

The questions I will ask in this difficulty based on my interview with Mr.Caglar will be split into 5 multiple choice questions and 5 fill in the blank questions.

MC questions ~

1. What is the binary equivalent to 26
2. What is 00100100 in denary
3. What colour is #00ff00
4. What does CSS stand for
5. What does the L in HTML stand for

FITB questions ~

1. What is the binary equivalent to 17
2. What is the binary equivalent to 21
3. What is the denary equivalent to 00010000
4. What is the denary equivalent to 00010110
5. What colour is the hexcode #00ff00

Medium mode~

MC questions ~

1. What tag is used to hold the content of the website
2. What tag is used to define the website
3. What tag is used to hold paragraph information
4. What colour is #00ffff
5. What colour is #ffffff

FITB questions ~

1. What does the M in HTML stand for
2. What is the binary equivalent to 123
3. What is the denary equivalent to 11001110
4. What is the denary equivalent to 01010101
5. What colour is #000000

Hard mode~

MC questions ~

1. What tag is used for hyperlinks
2. What does the H in HTML stand for
3. What tag is used for the second biggest heading
4. What tag is used for the third biggest heading
5. What property do you change to edit the font size

FITB questions ~

1. What tag is used for ordered lists

2. What tag is used for unordered lists
3. What tag is used for list items
4. What do you close list tags with
5. What tag is used to close the “body” of the HTML document

Justification of graphics and usabilty features

Graphics: I have decided to go for a cartoony and unrealistic approach to my graphics because I want to make the player interested in the vibrant colours. Realistic colours would be too bland for my players as some will be younger and thus they are more likely to be engaged if they like the look of the game. Had I chose to have realistic graphics then shading of the assets and 3D models may have to been used but that would have taken too long for me to make in the time frame of the project and is too difficult for me as I lack the skills to create 3D assets so I created 2D assets instead.

Usability features:

Personalisation: Simple personalisation allows my user to create a character out of a few options the user can choose by clicking the indicative arrows to shift between options making it easy system to understand for my younger stakeholders.

Multiple choice questions: Because my stakeholders consist of children and teenagers, multiple choice questions is a middle ground for challenges as the question can be challenging but the players will very easily be able to answer the question for my younger stakeholders.

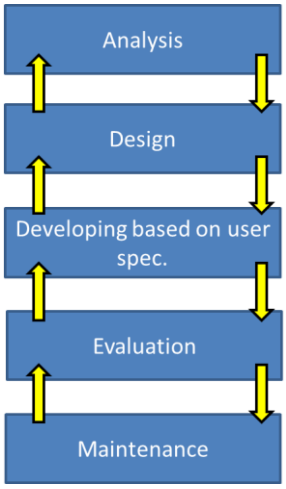
Fill in the blank questions: This will be for the useful for my older stakeholders and children looking for a challenge as the answer isn’t given in the options. It will make the user research the answer via the book feature thus encouraging independent research rather than just guessing.

Book feature: This will allow the user to search through a book of information that can guide the user to the answer. The book will not have the direct answers to the question but will give them useful information on what the answer could be. This feature is an abstraction of how a real programmer or web developer might go on the internet to find the solution to a feature of their program.

Key variables and structures

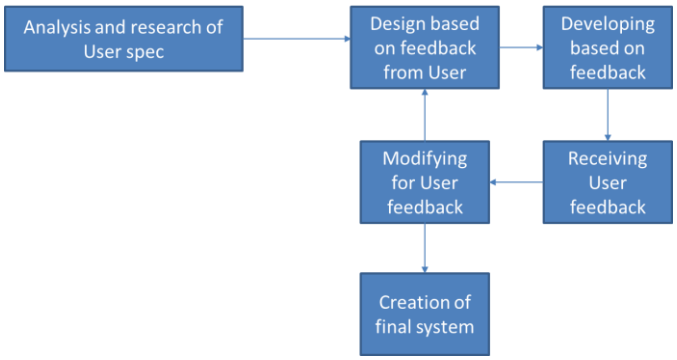
Development process:

There are 2 development processes that are suitable for this project. The waterfall method is suitable as this is a fairly small project and risks are not a major issue. However, in the waterfall method, the user only has input in the design and analysis when researching what the user would want meaning that the end product may not portray the users specification as the developer, me, may not have completely understood what exactly the user was wanting. This will not suffice for this project as the goal of this project is to make a program

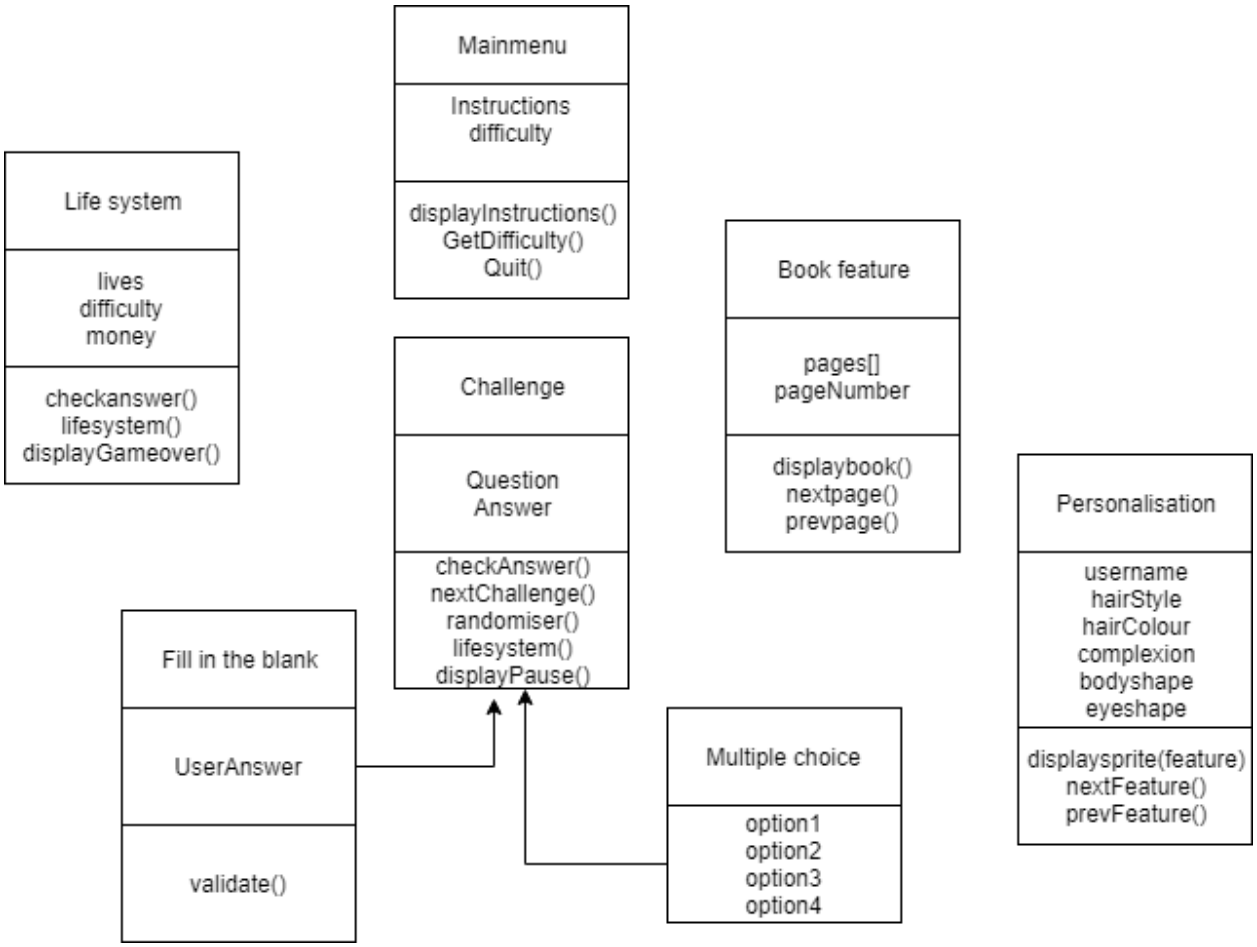


based on user specification and having a lot of user input in the development stages is key to creating a program that the players want. As the development team for this project only consists of me, this process is suitable as all the roles are based around me and thus my responsibility as the only programmer and designer are clear. Some features of the waterfall method will be implemented such as how the process allows the developer to go back and forth through the stages to debug if the testing shows any syntax or logic errors or to make modifications to the program if I have enough time to make the game as a whole, a better experience for the players.

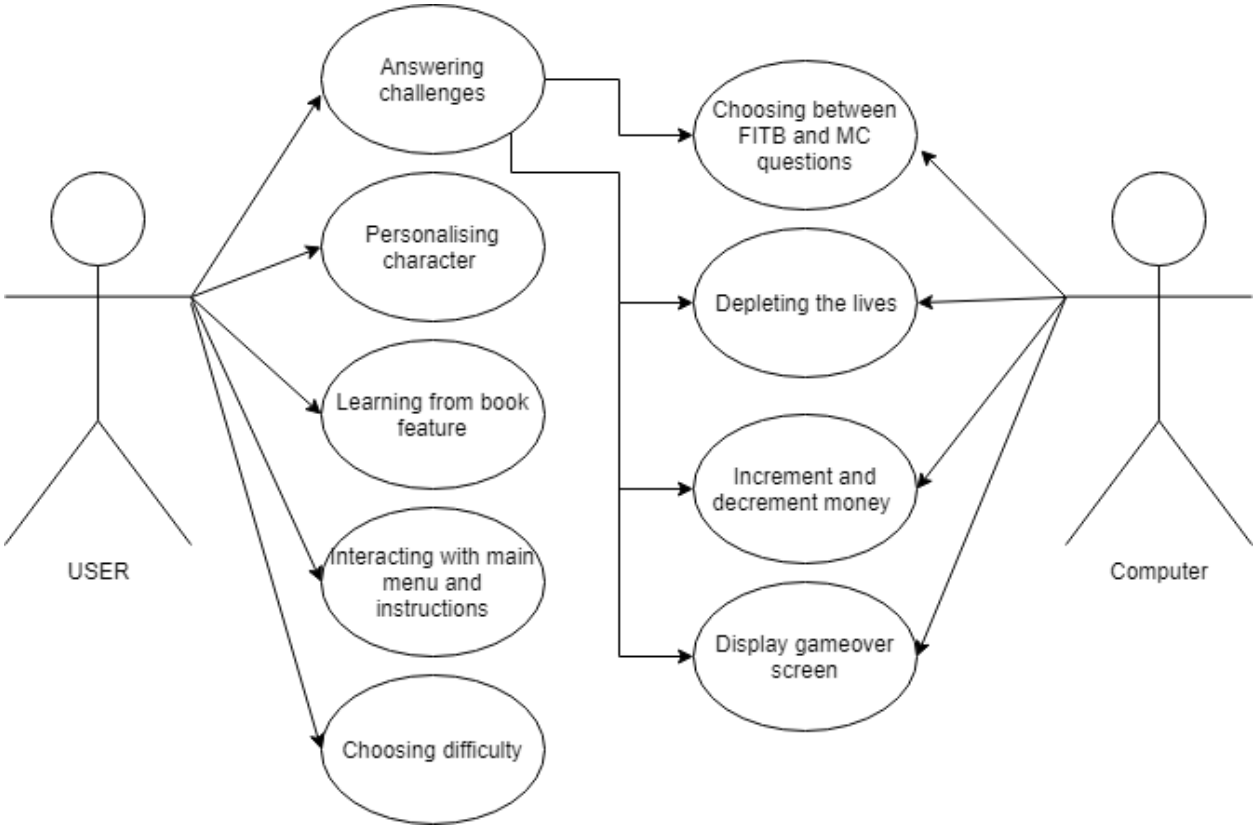
The other methodology suitable is the agile method where the development team design a program based a simple user specification. This is then implemented into a program and is shown to the stakeholders of the program where they give feedback on what to keep, remove and add. Then a new design is created based on the feedback I receive from my stakeholders and the new features and modifications are made. This is then shown once again to the user and this cycle continues until my user is happy with the prototype shown. Lastly a final product is created based on all the previous feedback given. This method is very suitable to my project as it makes use of high amounts of user feedback to create a program that the user wants and that is one of the goals of this project. I will be implementing all the features of the agile methodology into my development as it is the best suited to the goal of this project as I will be creating a prototype for my users so they can give me feedback.



CLASS DIAGRAM



USE CASE DIAGRAM



Key Variables

NOTE: Subject to change through development as I cannot completely predict which variables I will be using.

Main menu

Stored label	Data type	Actual meaning	Justification
instructions	image	This image will display instructions	This is to ensure that my users understand how to play the game
difficulty	string	This will hold the value of the players selected difficulty	to ensure that the correct array of questions are loaded

Personalisation

Stored label	Data type	Actual meaning	Justification
userName	string	Holds the users chosen name	To be displayed in game as a form of personalisation.
HairStyle	sprite	The selected hair style by the player	To be displayed in game as a form of personalisation
HairColour	sprite	The selected hair colour by the player	To be displayed in game as a form of personalisation
Complexion	sprite	The selected complexion by the player	To be displayed in game as a form of personalisation
Eyeshape	sprite	The selected eye shape by the player	To be displayed in game as a form of personalisation
Bodyshape	sprite	The selected body shape by the player	To be displayed in game as a form of personalisation
BodyColour	sprite	The selected body colour by the player	To be displayed in game as a form of personalisation

Music

Stored label	Data type	Actual meaning	Justification
backgroundMusic	Audio file	Holds the background music	To keep the player interested throughout the game.

Book feature

Stored label	Data type	Actual meaning	Justification
Pages[]	List	Hold the list of all the pages that will be displayed	So the player can traverse through the book.

Life system

Stored label	Data type	Actual meaning	Justification
StartLives	Int	Holds the number of lives the user should begin to have	To ensure the user begins with the right amount of lives
Lives	Int	Holds the number of lives the user has	To compare with gameover threshold – being 0
Money	Int	Hold the amount of money the user has accumulated	To be displayed as a form of progression

Challenges

Stored label	Data type	Actual meaning	Justification
easyQuestions[]	List	Hold the list of all easy questions	To hold and randomly select from when summoning new question when selected difficulty is easy
mediumQuestions[]	List	Hold the list of all medium questions	To hold and randomly select from when summoning new question when selected difficulty is medium
hardQuestions[]	List	Hold the list of all hard questions	To hold and randomly select from when

			summoning new question when selected difficulty is hard
currentQuestion	string	Holds the question displayed	Takes a question from selected difficulty and displays it
answer	string	Holds value of answer to question	To be compared with user's answer
userAnswer	string	Holds value of users answer	To be compared with answer
Option1	string	Holds value of one of the options the user can make in the multiplechoice questions	Multiple choice questions require multiple options to choose from
Option2	string	Holds value of one of the options the user can make in the multiplechoice questions	Multiple choice questions require multiple options to choose from
Option3	string	Holds value of one of the options the user can make in the multiplechoice questions	Multiple choice questions require multiple options to choose from
Option4	string	Holds value of one of the options the user can make in the multiplechoice questions	Multiple choice questions require multiple options to choose from

Test data for Development testing

Note: LC and RC are short of Left click and Right click respectively

Test data is valid unless stated otherwise

What is being	Test data to input	What is the expected result

tested		
A. Main menu options	1. LC Start game button 2. LC Instruction 3. LC Exit 4. RC (invalid) 5. Any other key(invalid)	1. The game moves onto the difficulty selection 2. The game displays an image of how to navigate through main gameplay 3. The game terminates 4. Null 5. Null
B. **Instructions**	1. LC back button 2. RC back button (invalid)	1. Goes back to title screen 2. Null
C. Difficulty section	1. LC easy 2. LC medium 3. LC hard 4. RC (invalid) 5. Any other key(invalid)	1. The game should only show easy questions. It should not display any lives. It should not decrease bank balance when getting a question wrong 2. The game should show medium questions. It should display 5 lives on screen Should decrease 1 life and £50 when getting a question wrong and should gain £100 when getting an answer correct. 3. The game should show hard questions. It should display 3 lives on screen Should decrease 1 life and £50 when getting a question wrong and should gain £100 when getting an answer correct. 4. Null 5. Null
D. Name	1. John 2. Jonnothanoth(borderline) 3. Bob Dylan bobby (invalid)	1. Game allows name 2. Game accepts name 3. Game prompts player to pick a name less than 12 characters
E. Book feature and its components	1.LC book 2.LC arrow pointing east 3. LC arrow pointing west 4.LC X at top of book 5. LC arrow pointing west on 0 pages(invalid) 6. LC arrow pointing east on last	1. Displays the opening pages of the book 2. Turns page forward by 1 page 3. Turns to previous page 4. Exits book 5. No page turn 6. No page turn

	page(invalid) 7. RC(invalid) 8. p (invalid)	7. N/A 8. Null
F. Life system	1. Getting question correct on maxLives 2. Getting to 0 lives	1. N/A No increment in lives 2. GameOver state is displayed.
G. Avatar customisation	1.LC left arrow on hair option 2. LC right arrow on hair option 3. LC left arrow on complexion option 4. LC right arrow on complexion option 5. LC left arrow on clothes option 6. LC right arrow on clothes option 7. RC (borderline) 8. LC Click next	1. Displays previous hair option 2. Displays next hair option 3. Displays previous complexion option 4. Displays next complexion option 5. Displays previous clothing option 6. Displays next clothing option 7. N/A 8. Goes into the main game
H. FITB challenges	1. Inputting character i.e. q 2. LC confirm 3. RC (invalid) 4. Inputting correct answer 5. Inputting incorrect answer 6. Giving answer of 12 characters i.e 000000000000 (borderline) 7. Giving answer greater than 12 characters i.e 0000000000000(invalid)	1. The game prints character onto form 2. Game prints whether answer is correct 3. N/A 4. increments bank balance by £100 5. Game takes away 1 extra life if possible and decrements bank balance by £50 6. Accepts answer 7. Rejects answer and prompts user to enter answer less than 13 digits
I. Multiple choice Challenges	1. LC correct option 2. LC incorrect option 3. RC option (invalid)	1. increments bank balance 2. Game takes away 1 extra life if possible and decrements bank balance by £50 3. Null
j. Music	I. Start game	1. Background music file should play on loop

Testing for beta testing

This is post development testing done by my stakeholder M.Miah.

Test number	What is being tested	Input data	Expected outcome
1.	Aspect ratio	N/A	16:9

2.	Difficulties	1.Choosing Easy 2.Choosing Medium 3.Choosing Hard 4. Click back on selection screen	1.Easy questions loaded 2.Medium Questions loaded 3.Hard questions loaded 4. back to main menu
3.	Instructions	1.LC instructions button (valid) 2.RC (invalid) 3. Go back to menu	1.Instructions load 2.Nothing 3. Go back to menu
4.	Quit	1.LC quit 2.RC	1. Quit is responded on console 2.Quitted
5.	Music	1. Begin game	1. Music is played
6.	Personalisation	1. Press next button 2. Press prev button 3. Enter name with 11 characters(valid) 4. Enter name with 12 characters (borderline) 5. Enter name with 13 characters (invalid) 6. RC all buttons 7. Click ready	1. Next personalisation sprite is displayed 2. Previous personalisation sprite is displayed 3. Accepts name 4. Accepts name 5. Denies name 6. Null 7. Name is displayed on nameplate and character is displayed with game
7.	Life system 1.Money increases by 100 2. Money decreases by 50 3.Lives decrease by 1 4.Gameover at 0 lives 5.No gameover for easy mode	1.Getting answer correct 2.Getting answer wrong 3.Getting answer wrong 4.Losing all lives 5.N/A	1.Money increases by 100 2. Money decreases by 50 3.Lives decrease by 1 4.Gameover at 0 lives 5.No gameover for easy mode
8	Fill in the blank	1.A(valid)	1.Accepts answer

	challenges	2.000000000000 (borderline) 3.0000000000000 (invalid)	2.Accepts answer 3.Denies answer
9	Multiple choice challenges	1.clicking answer 2.Rc answer(invalid)	1.Accepts answer 2.Denies answer
10	Book feature	1.click book button 2. Click next page 3.Click prev page 4. Click X 5. Click back on book 6.RC on buttons (invalid)	1. Book appears 2.Next page is shown 3. Previous page is shown 4.Book closes 5.Book opens on first page 6. Null
11	Pause Menu	1.click pause 2. Click Back to main menu 3. RC buttons	1. Pause menu opens 2.Takes player to main menu 3. Null

FEEDBACK FROM USER

After showing my stakeholder the graphics of game and the usability features, I asked them for feedback on the game itself from what they had seen.

MAHIMA

The game: the game looks fun, really colourful. It seems pretty simple to learn how to play and the customisation make me feel like I add some sort of my persona into the game but it. The games screen looks a bit packed with detail. It seems a bit clustered. I like the idea of a book teaching me how to answer the questions and guiding me because it's like real life how a programmer will have to do research and look for answers instead of just being told them.

She also suggested adding in a back to game button on the pause menu which I agreed to add as she represents my stakeholders and I thought it was essential that the player can return to the game.

This also meant that before implementing I would change the customisation screen layout to make it less clustered. Positive taken from this were that she liked the screen designs for the book feature and thought the number of questions were sufficient.

I then showed her my success criteria and we discussed how I would be unable to add achievements due to time restraints but how I would add them if I had more time after development. We also discussed how I would be unable to create randomly generated questions due to my lack of skillset using C# to implement this feature.

She agreed the rest of my success criteria was sufficient.

Signed: **22/11/18**

Development Story

Main Menu Week starting: 26/11/18

White Box testing

White box testing is I look at my code and traces the inputs and outputs making sure they and stored, processed and loaded when requested by the program.

Throughout the development of my code, I will be testing the inner workings of my code to ensure the flow of input processing and output work as they should and below I have outlined what sort of inputs and outputs I will be tracing throughout my development process and justifying the need to test these certain inputs though the code.

Inputs	Expected Outcome
Main menu buttons	Goes to correct screen
Submitting name	Name appears in game
Money	Value changes during game
Users Answers for FITB questions	Compared with correct answer of question
Buttons clicked multiple choice question	Compared with correct answer of question

Black box Testing

Black box testing is when you test a programs functionality without needing to know the workings of the code. By doing this, I will get an idea of the experience my stakeholders will have when using my

simulation/game. It shows what the End user experience looks like. It's very similar to encapsulation where the user doesn't need to know how it works but rather that it does.

Detailed below are the key functions of my code that I will be testing to ensure my user has a high quality end product and a justification of why I have decided to test these certain components.

Component	Expected Outcome
Main menu	Choosing Start takes user to difficulty selection
	Choosing instructions lets user view instructions screen
	Choosing quit lets the user quit
Difficulty selection	Questions in game corresponding to difficulty selected
Submitting name	Name appears on name plate
Customisation and buttons changing variants	Selection changes when buttons are clicked
Answering Multiple choice questions correctly	Money increases by £100
	Game randomly chooses between FITB and MC questions
Answering Multiple choice questions incorrectly	Money decreases
	1 life is lost
	Game randomly chooses between FITB and MC questions
Answering Fill in the blank questions correctly	Money increases
	Game randomly chooses between FITB and MC questions
Answering Fill in the blank questions incorrectly	Money decreases
	1 life is lost
	Game randomly chooses between FITB and MC questions
Game Over	Game over screen appears
Book feature	Book should open and buttons should allow user to sift through pages of information

Alpha Testing

This is the process of me, the developer, testing my game to ensure it does what I have wanted it to do based on my success criteria that was created around my analysis and design. This will include me going through all the pathways to ensure there are no bugs, logical or syntax errors. This will only give me a limited view on whether the final product is good enough as it is my stakeholder who ultimately know what they want from the development of this game so whilst this is an effective method of testing I will use, I must also apply another form of testing that will ensure my game/simulation is a product that my stakeholders are pleased with.

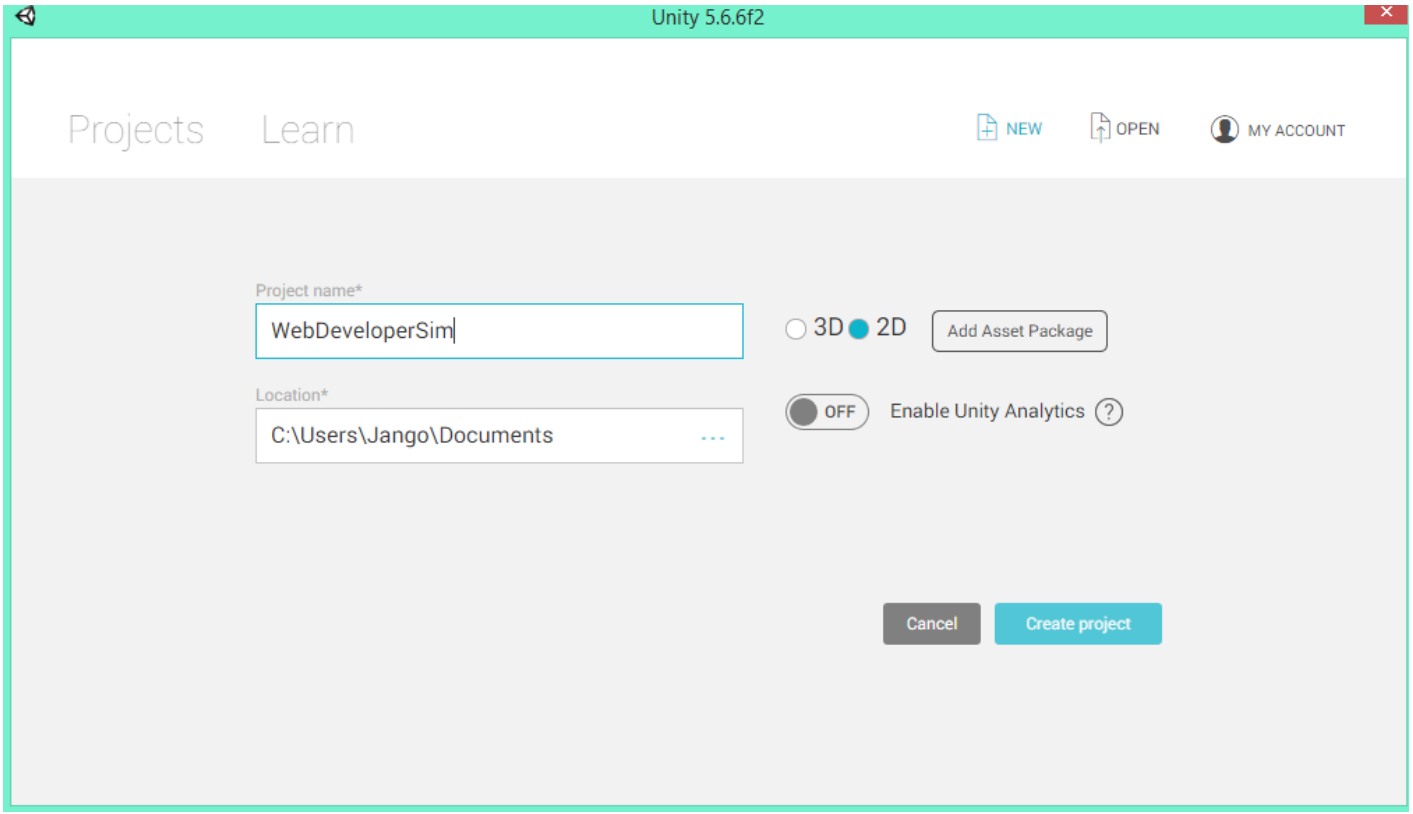
Beta Testing

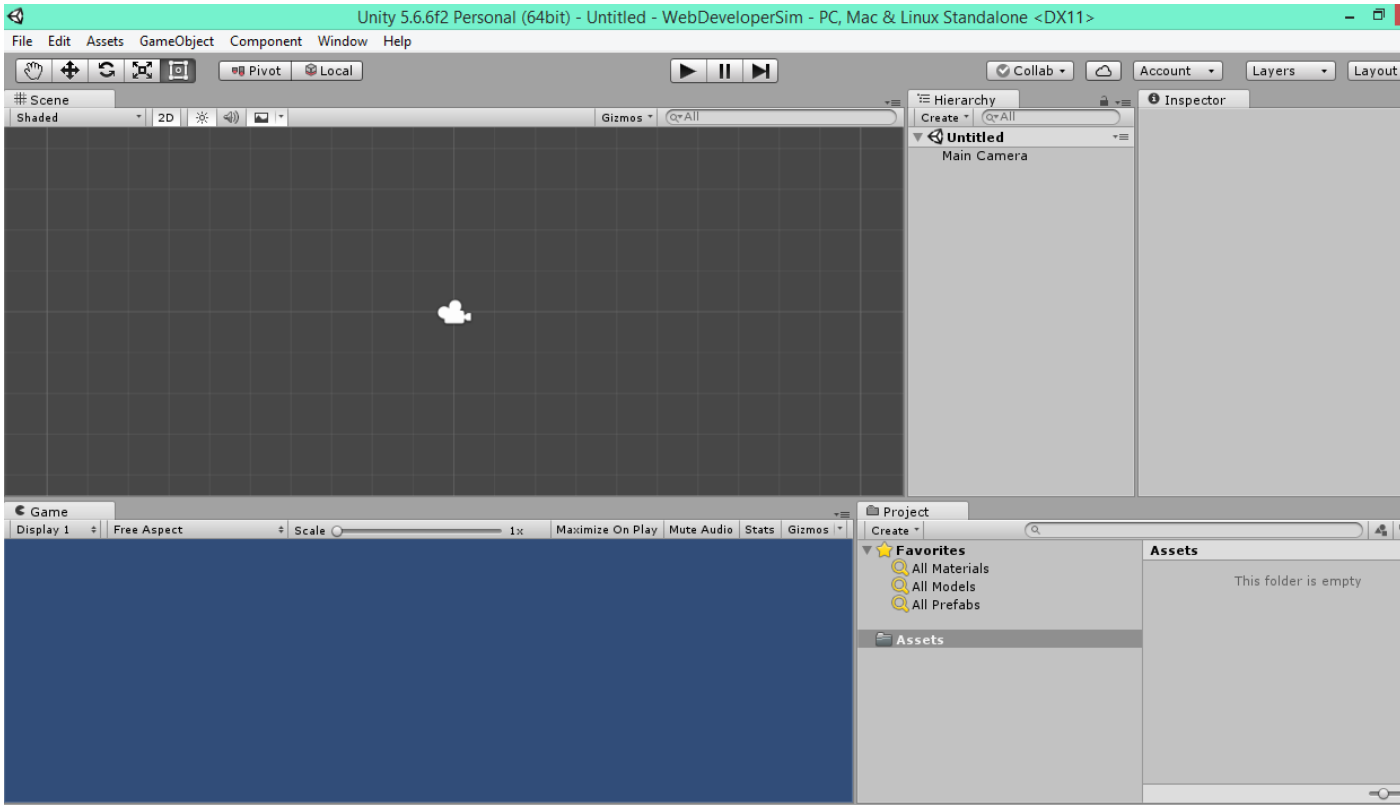
This is the key form of testing that my program will undergo as it will help me create an end product that my stakeholders are happy with. This form of testing is when I get a sample of my stakeholder and get them to test my game. In this case I will be asking M.Miah on their opinions on my game as I will be able to get quick feedback in abundance throughout the development of this course. I have chosen M.Miah as I know I will be able to communicate with her a lot and because they represent the age range of my stakeholder. This form of testing means that it is crucial to have constant feedback from stakeholders which as I have discussed, I will be able to, so they can laisse with me to make the game better and better quality for my stakeholder. This method is very effective in creating a product that follows the specification of what my stakeholders want which for me is a priority. I will conduct this form of testing by creating prototypes using the agile method and getting feedback from every component of game before moving onto the next stage making sure the component of the game is as high to the standards my stakeholders would like it. I am also weary of time however and I will analyse the importance of certain components to ensure the most important features are completed.

Prototype 1

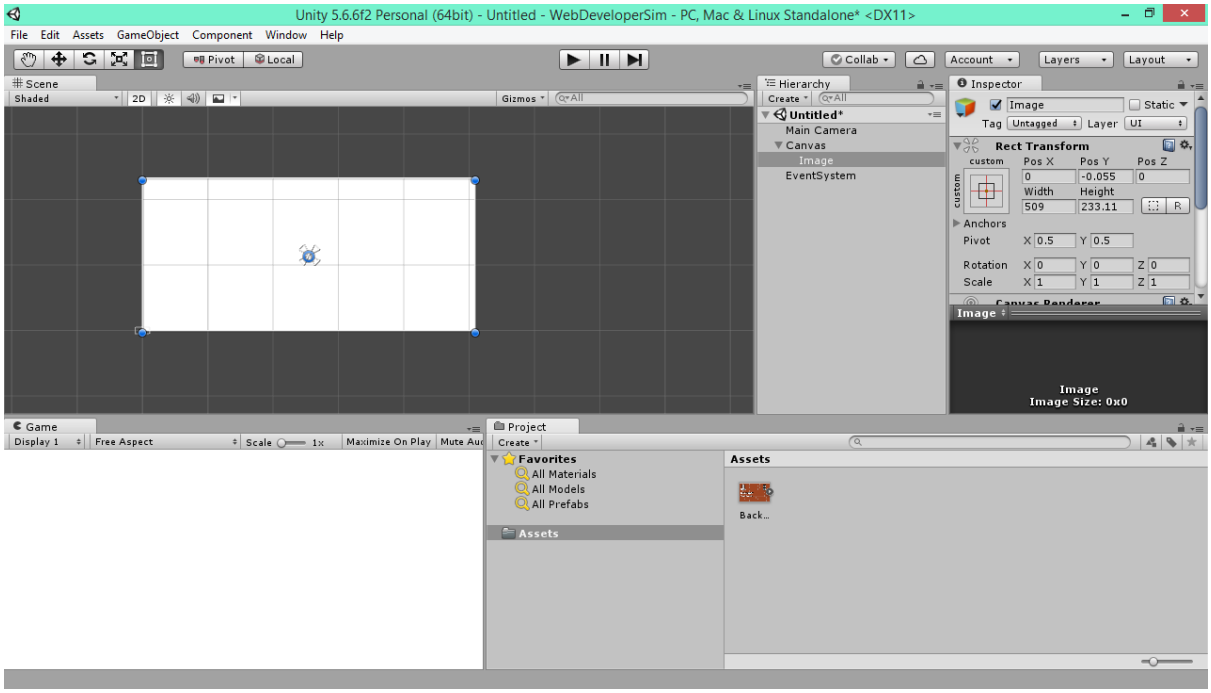
To begin with, I started developing my main menu as it will be the screen my stakeholders first see when they open the application and will let them navigate through the screens to get to the main game.

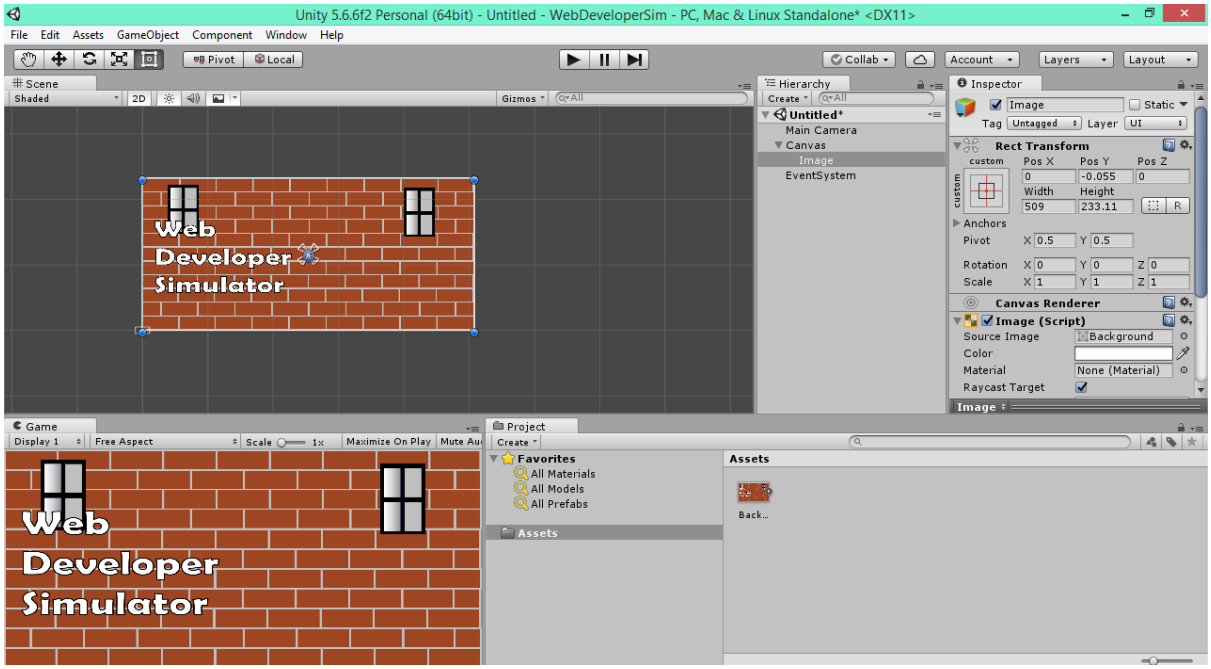
I opened unity and created the project as a 2D game as it will only consist of image assets to represent real world objects such as the computer and keyboard. This is a form of abstraction that I had to perform as I would have found it difficult to create 3D models of these objects. However, this is justified because my younger target audience of 9 – 13 will enjoy the colourful aesthetic and those who are on the older spectrum of my audience will primarily play it for the challenge and revision purposes.



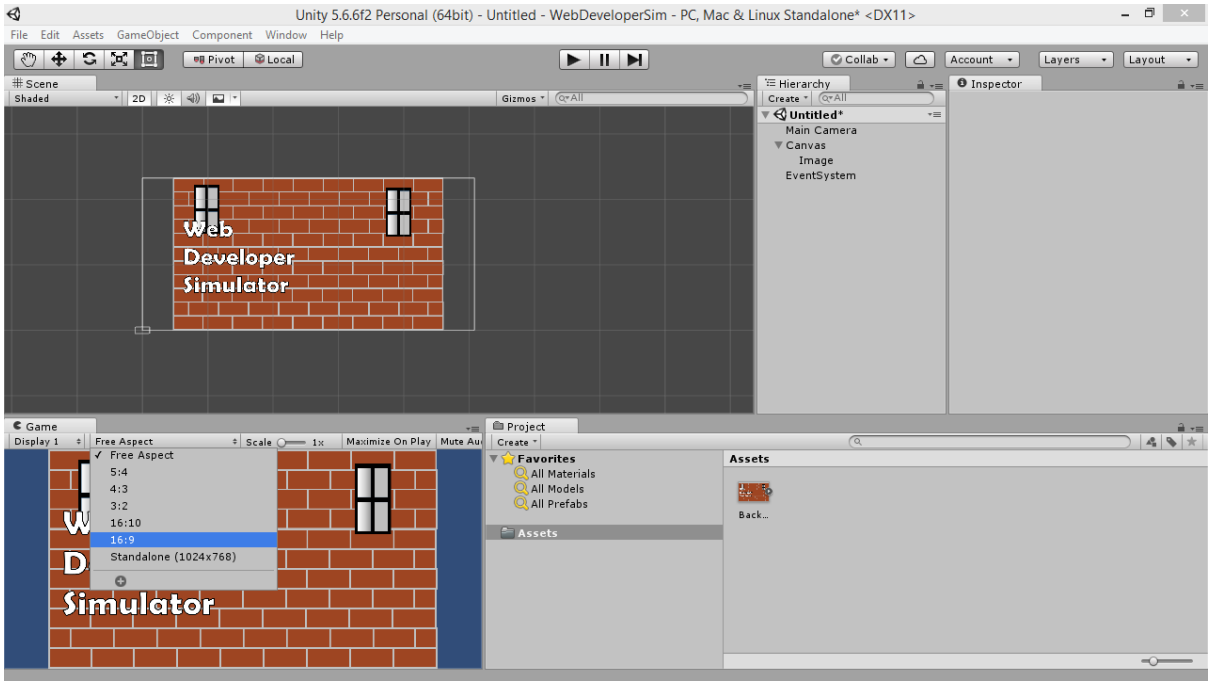


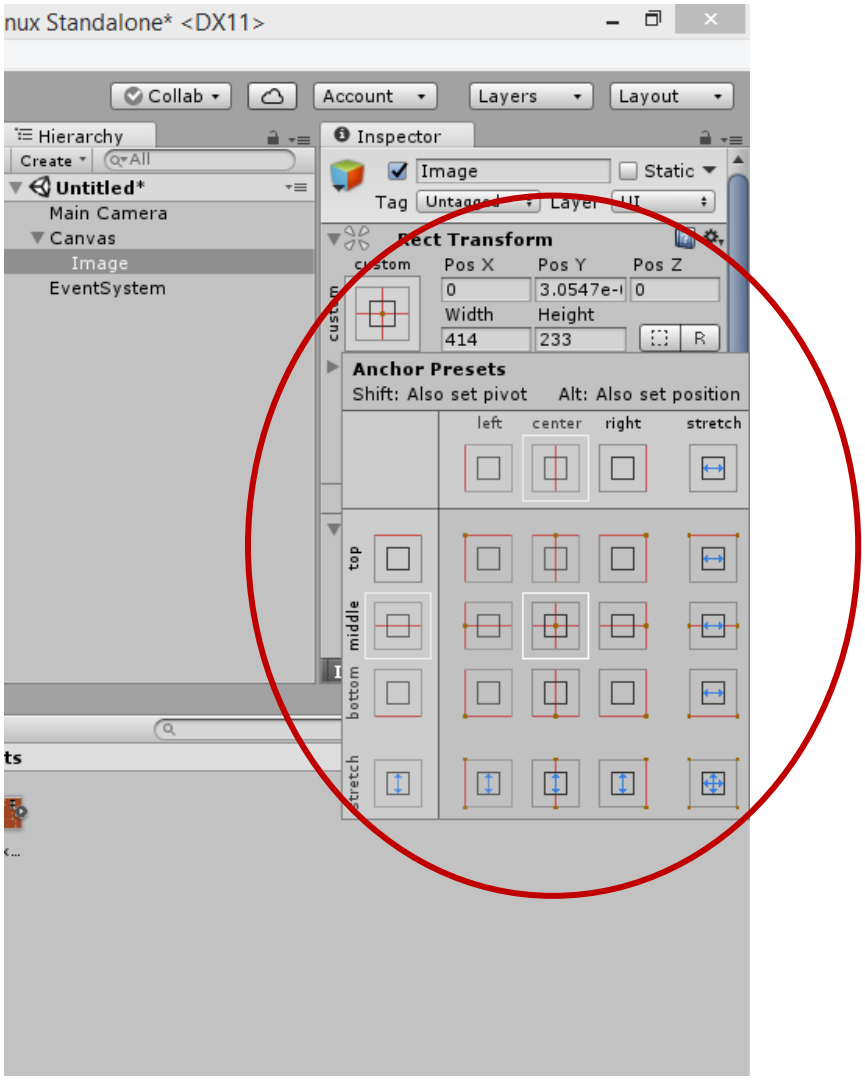
I began by adding the background that I had previously created in the camera frame. I saved the background image I had created in powerpoint as a .png file. I then draged the background image from my folder into the assets folder in the unity editor.



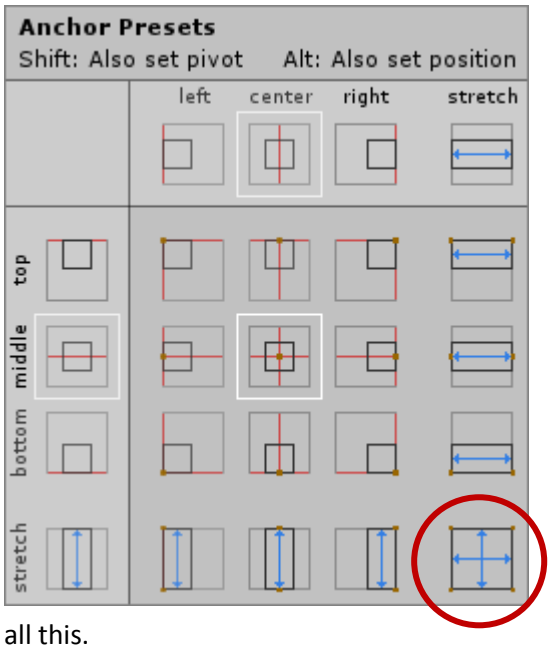


As shown, the image was stretch beyond original thoughts so I changed the aspect ratio of the camera to my desire of 16:9 as most monitors are either 16:9 or 16:10.

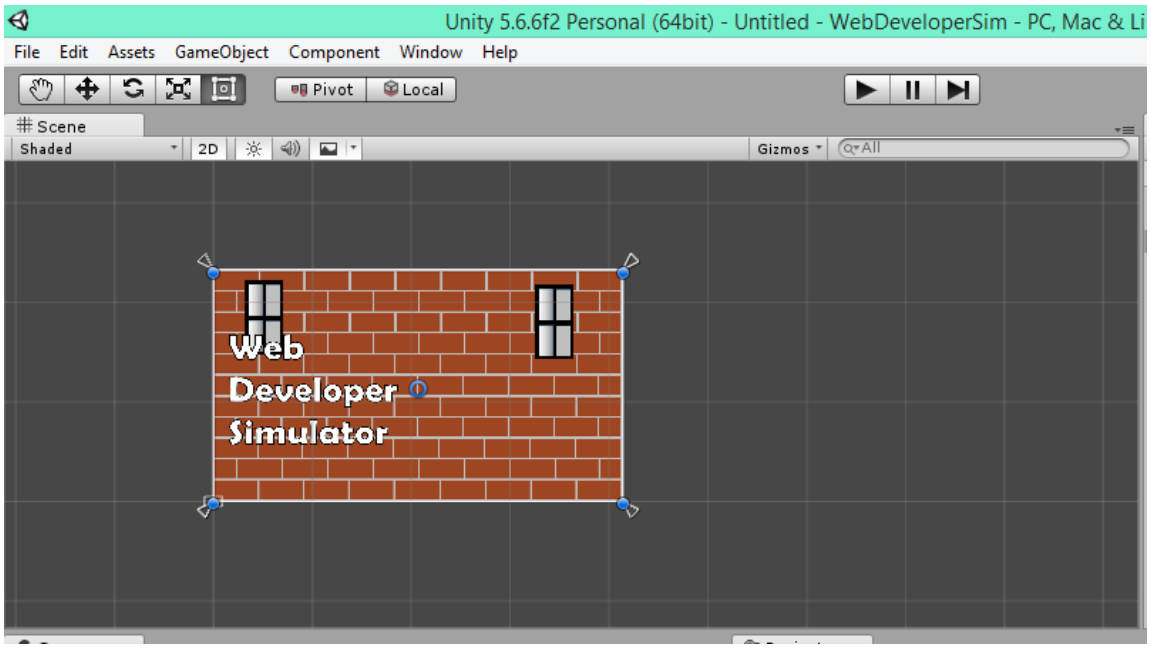

Unity 5.6.6f2 Personal (64bit) - Untitled - WebDeveloperSim - PC, Mac & Linux Standalone* <DX11>
File Edit Assets GameObject Component Window Help
Pivot Local
Collab Account Layers Layout
Scene
Shaded 2D Gizmos
Hierarchy
Create
Untitled*
Main Camera
Canvas
Image
EventSystem
Inspector
Web Developer Simulator
Game
Display 1 Free Aspect Scale 1x Maximize On Play Mute Au
Free Aspect
5:4
4:3
3:2
16:10
16:9
Standalone (1024x768)
Project
Create
Favorites
All Materials
All Models
All Prefabs
Assets

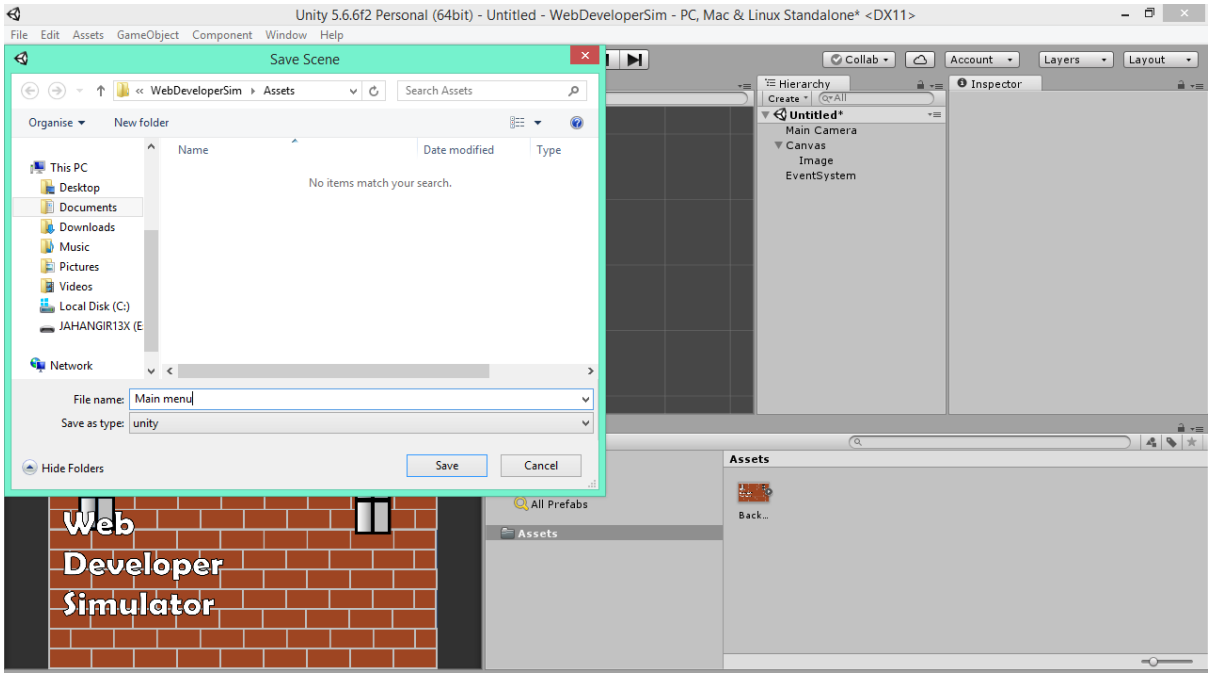

nux Standalone* <DX11>
Collab
Account
Layers
Layout
Hierarchy
Create
Untitled*
Main Camera
Canvas
Image
EventSystem
Inspector
Image
Static
Tag
Rect Transform
Pos X
Pos Y
Pos Z
Width
Height
414
233
Anchor Presets
Shift: Also set pivot
Alt: Also set position
left
center
right
stretch
top
middle
bottom

This function on Unity means that I am able to make the background grow and shrink and stretch accordingly with the screen.



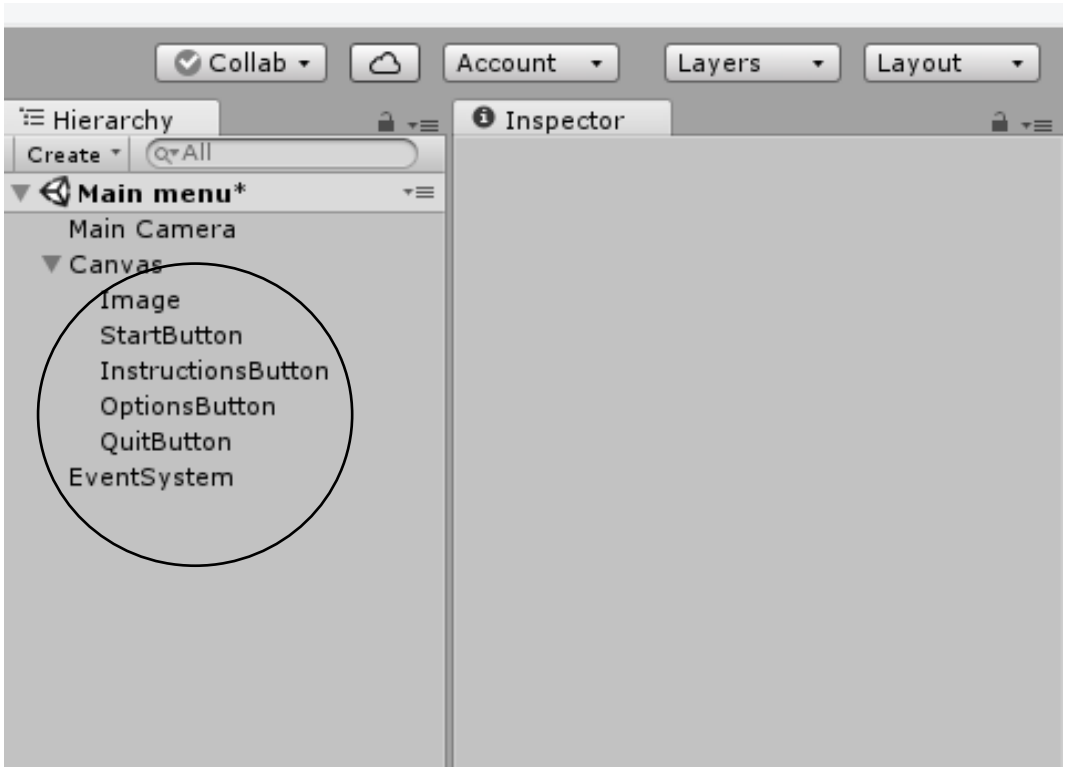
This is the specific anchor preset that allows me to do all this.



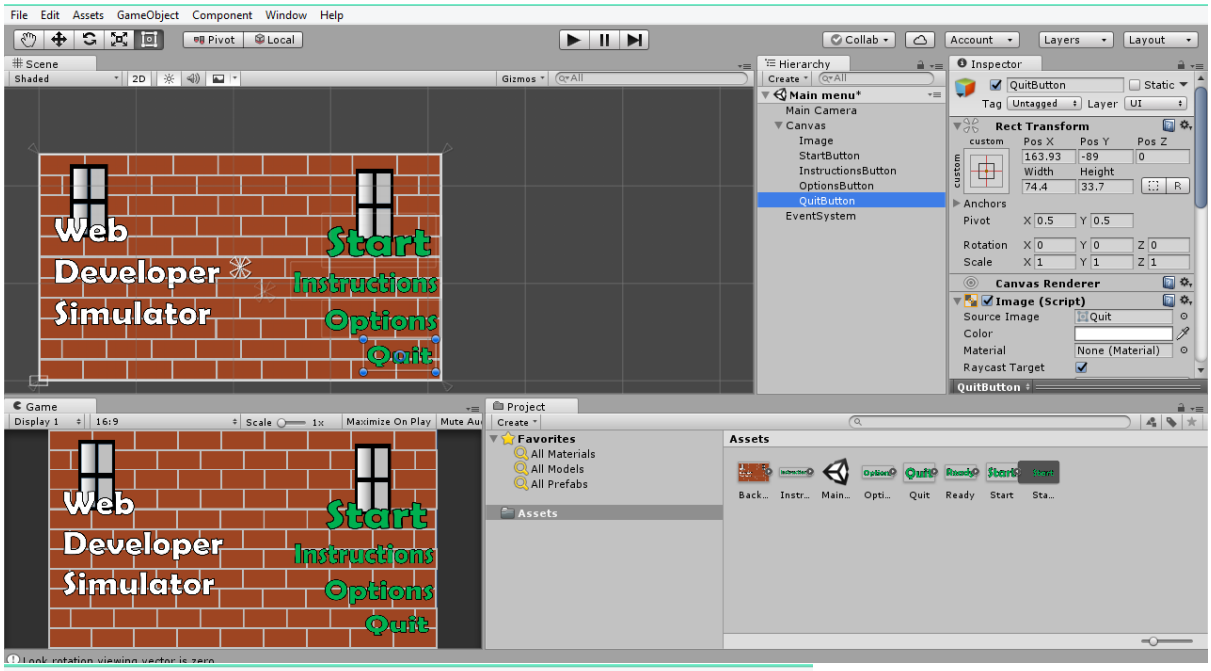


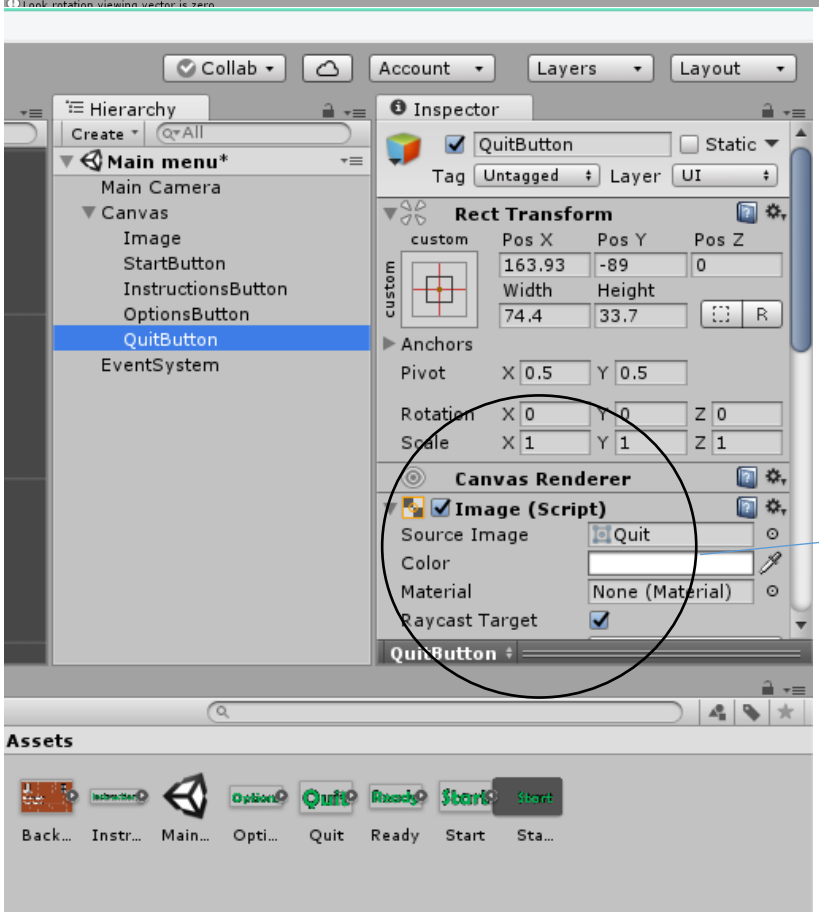
It is very important that I don't lose any data so I regularly save my project and scenes.

I then created 4 UI buttons named appropriately

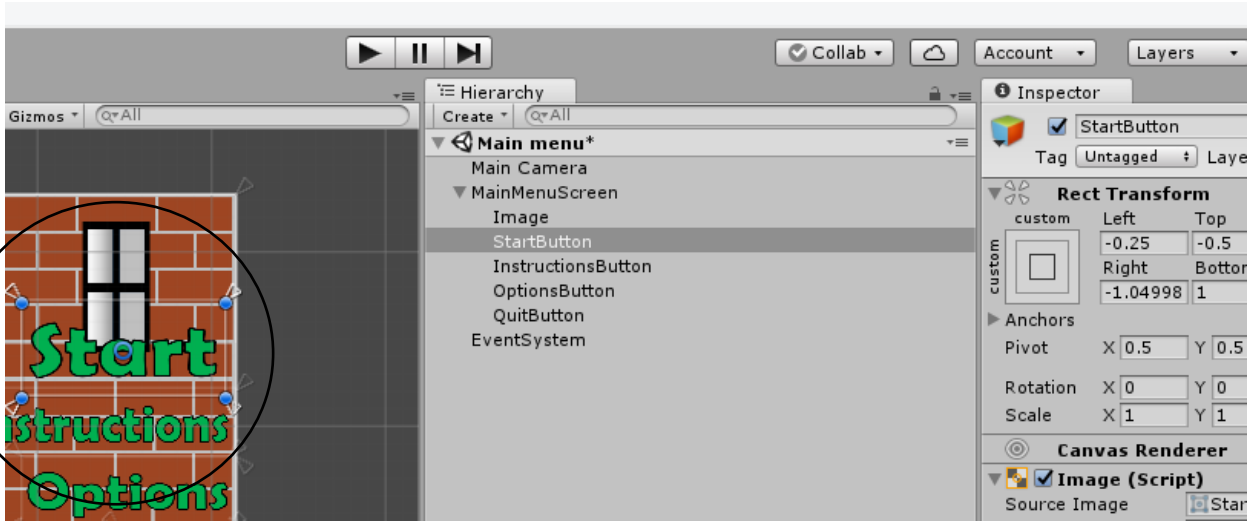


Next, I started to add the sprites I created in my design and implemented them into the buttons and scaled the images to fit the space. I did so using the image options that come with the unity engine that allow me to allocate an image to a button rather than just text.

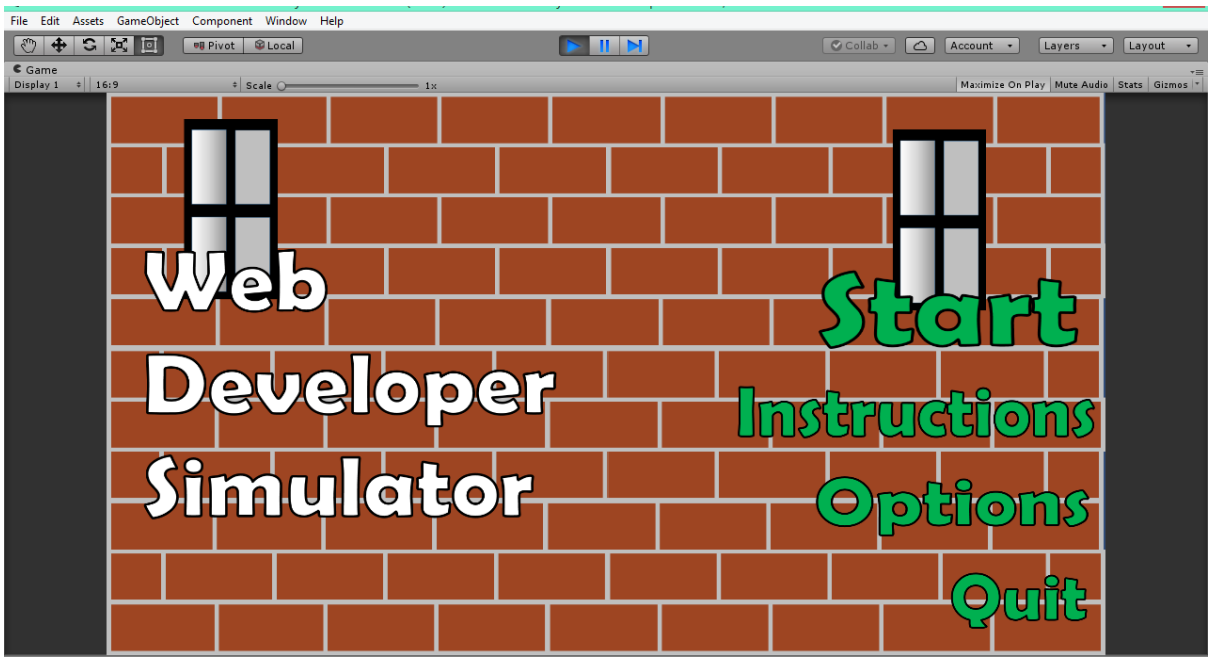




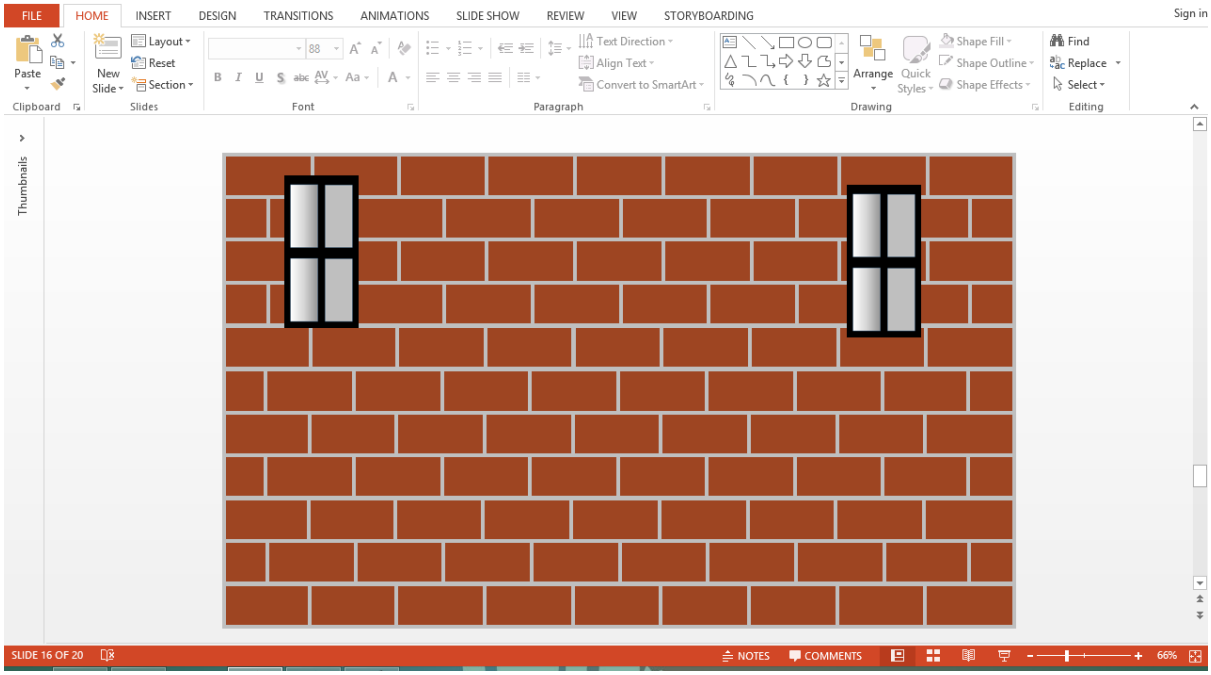
This feature allows me to edit the button using a pre-created image.



I then anchored all the buttons accordingly to scale correctly with the screen leading to the result:



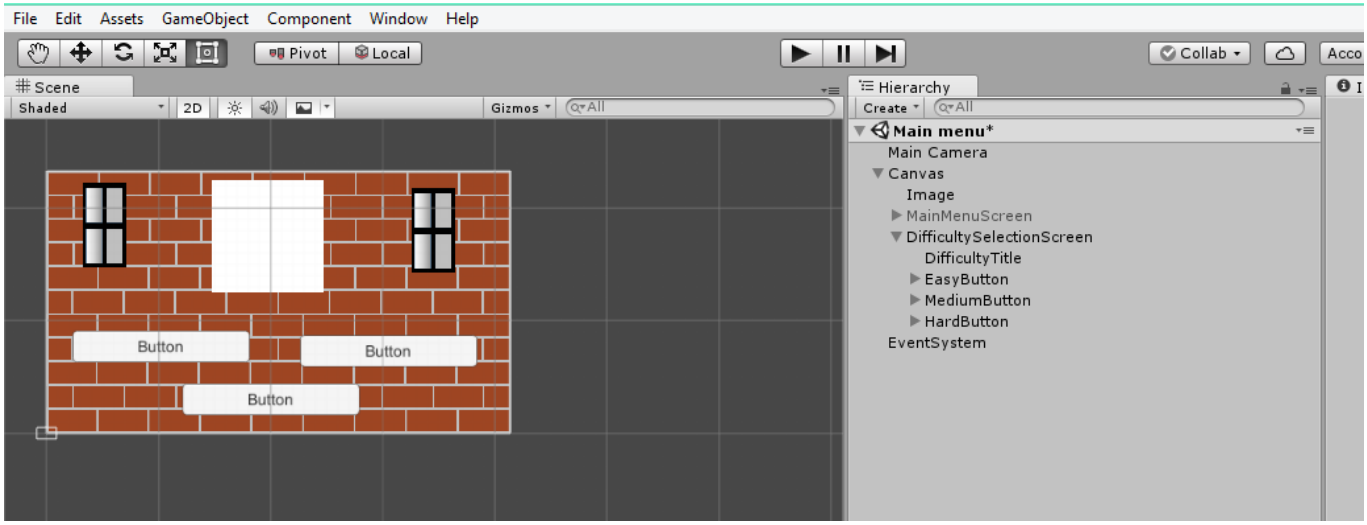
I went into powerpoint where I had created these sprites and edited the background so that the image was split into the backdrop and the logo separately. I did this so that I could keep the backdrop in my difficulty selection and personalisation screen without the logo being there as well.

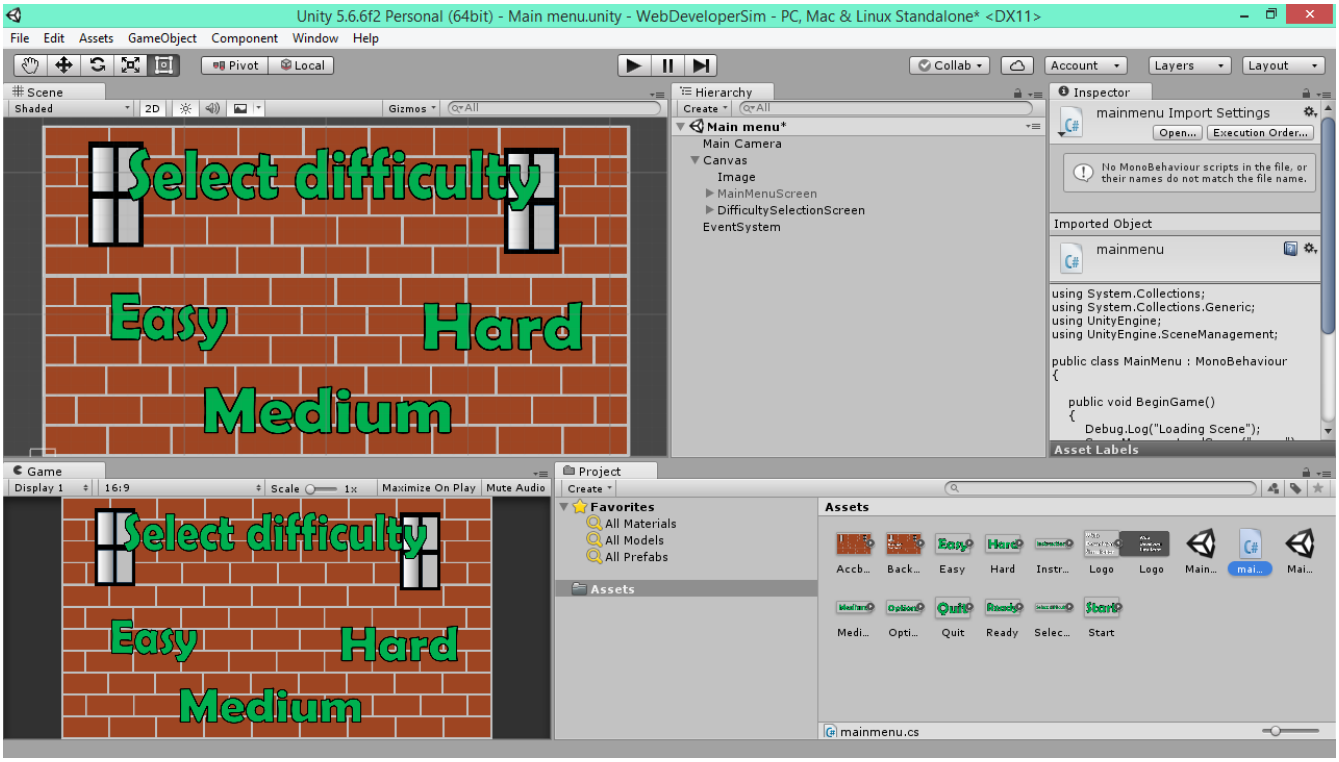


I decided to then create the difficulty selection screen so that my user can choose their chosen difficulty. I did this before creating the scripts so I could test that the buttons are fully functional.

To do so I would have to move everything I had created so far into an empty object so that I could switch between displaying the screens. I copied the gameobject now titled “mainMenuScreen” and titled the new one “difficultySelectionScreen”.

I would have to disable the mainmenuscreen from showing, to edit the difficulty selection and I created 4 elements in the gameobject; 3 buttons for the difficulty and 1 image to hold the title of the screen.

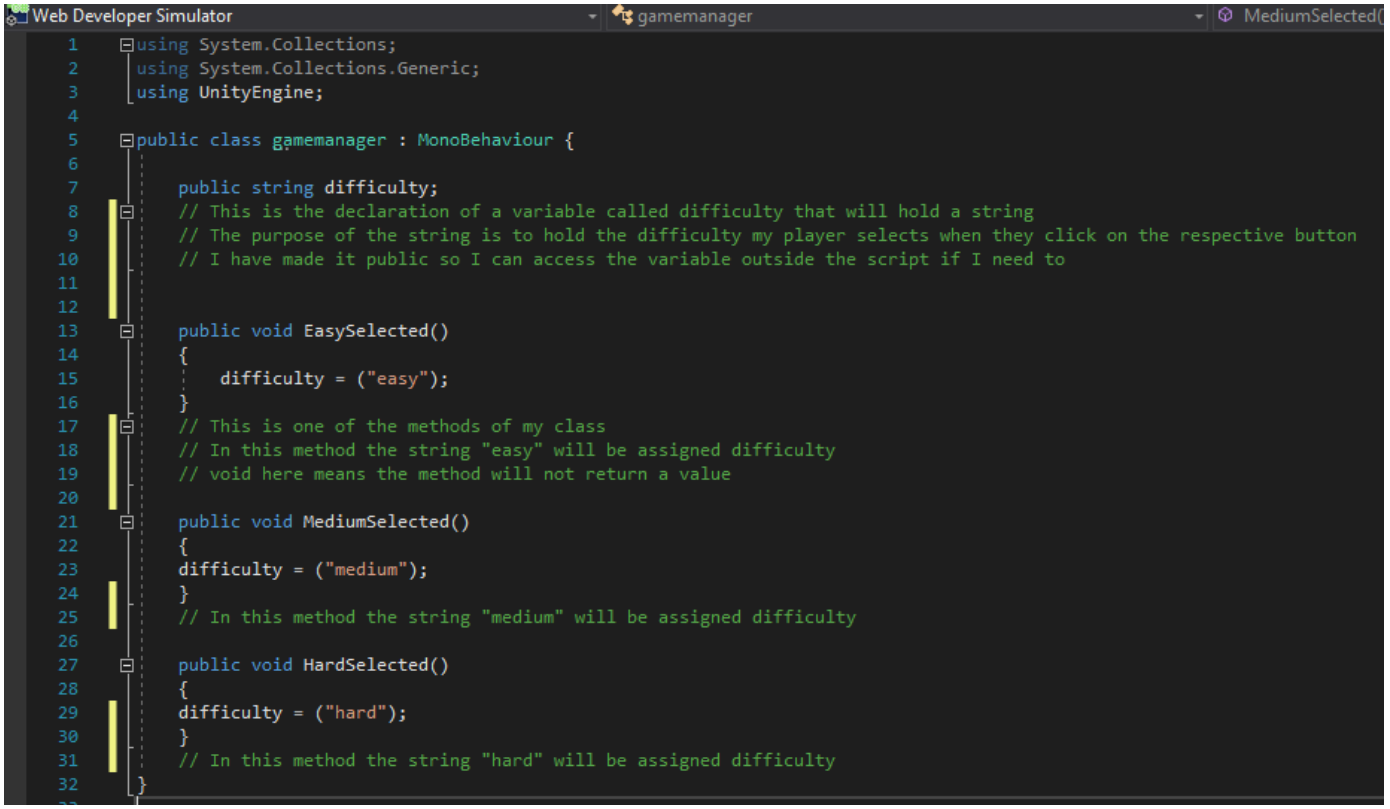




I then replaced all the image sources of my buttons and created scripts to assign values to a difficulty variable to be used when the game is played.

To do so, I created an empty game object called GameManager and attached a script called gamemanager.

I then created the script below after deleting the void and start functions Visual studios opened the script with. The script contained 3 methods.



I have noted what my code does in the screenshot for further maintenance . I then used the on click functionality of Unity to assign these values to difficulty so when they click the specific button, the corresponding value will be assigned to difficulty.

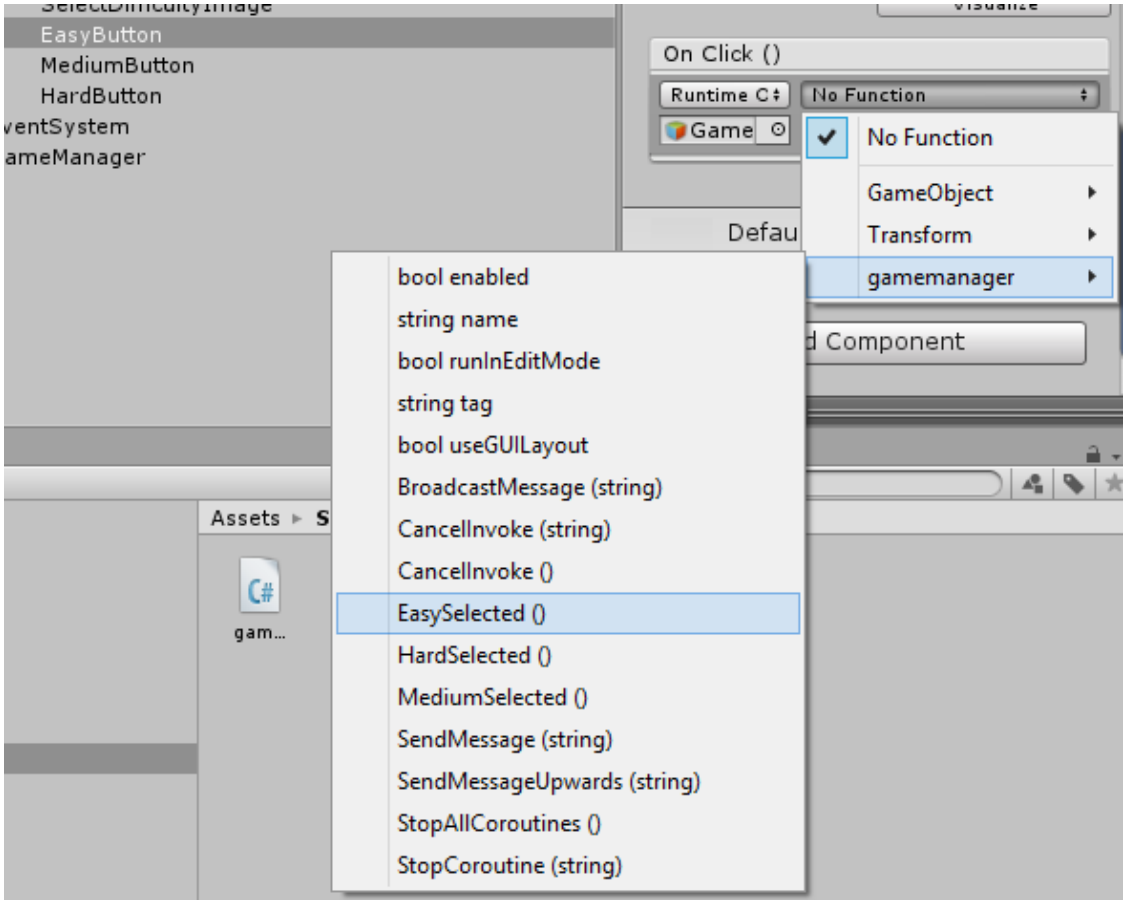
The first 3 lines, I declare the namespaces I will be using in this class. A namespace is a collection of classes and I will use these 3 namespaces to aid me in my development.

Line 5. I have declared the a class called gamemanager that derives from monobehavior which most classes in c# derive from.

Line 7, I have declared a variable that holds a string value called difficulty.

Line13-16 I have declared a function which does not return a value as shown by the return type stated as void. The function is called EasySelected() which changes the value of difficulty to easy. This is so that when the difficulty buttons are clicked, the value of the difficulty changes.

Lines 21-24 and lines 27-30 are similar to the EasySelected() but are named MediumSelected() and HardSelected() respectivly and change the values of difficulty to their corresponding difficulties.



First I needed to drag the GameManager into the slot that would allow me to choose the function. I did this first for EasyButton and selected the EasySelected function so when the easy button is pressed, the difficulty will change.

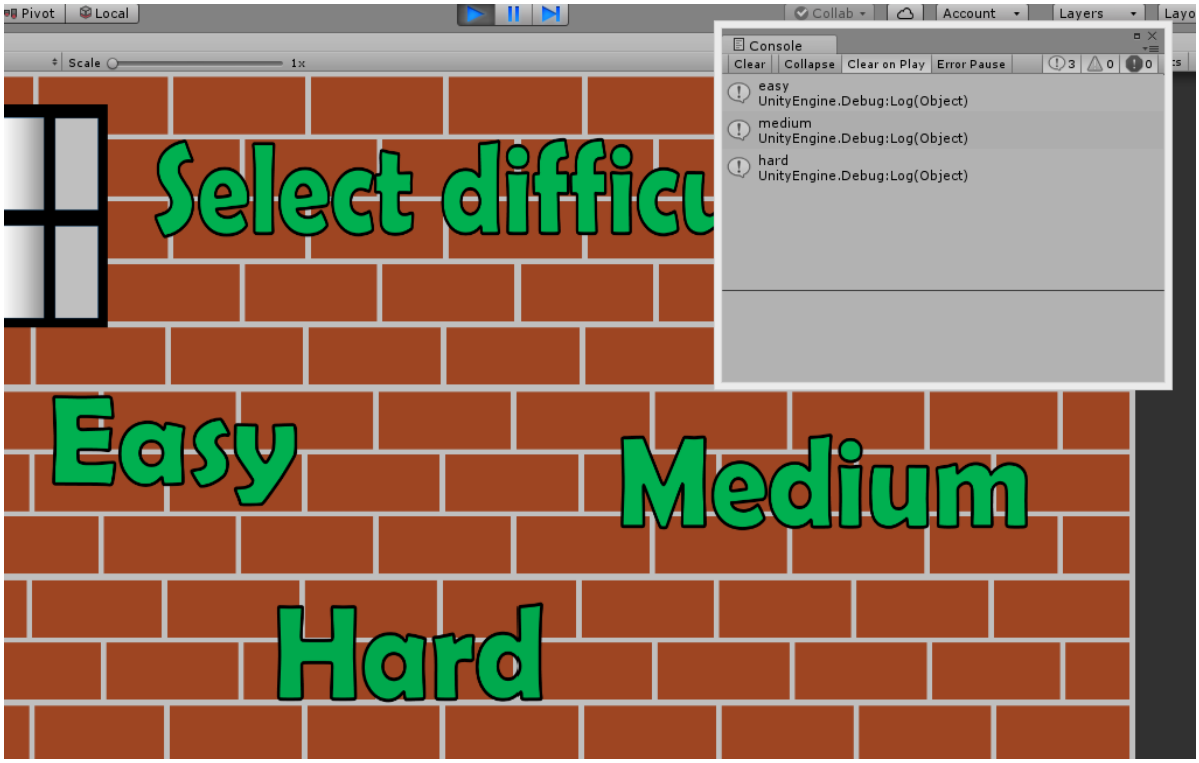
I then did this for Medium button and Hard button so they had similar functionality to the easy button.

To test that the value of difficulty will change, I added debug.log after all the functions to print the difficulty to the console.

```
public string difficulty;
// This is the declaration of a variable called difficulty that will hold a string
// The purpose of the string is to hold the difficulty my player selects when they click on the respective button
// I have made it public so I can access the variable outside the script if I need to


public void EasySelected()
{
    difficulty = ("easy");
    Debug.Log(difficulty);
}
// This is one of the methods of my class
// In this method the string "easy" will be assigned difficulty
// void here means the method will not return a value

public void MediumSelected()
{
    difficulty = ("medium");
    Debug.Log(difficulty);
}
// In this method the string "medium" will be assigned difficulty

public void HardSelected()
{
    difficulty = ("hard");
    Debug.Log(difficulty);
}
// In this method the string "hard" will be assigned difficulty
```

I have added a Debug.Log(difficulty) line to the end of all of the functions so when they are clicked the console will tell me the value has changed.

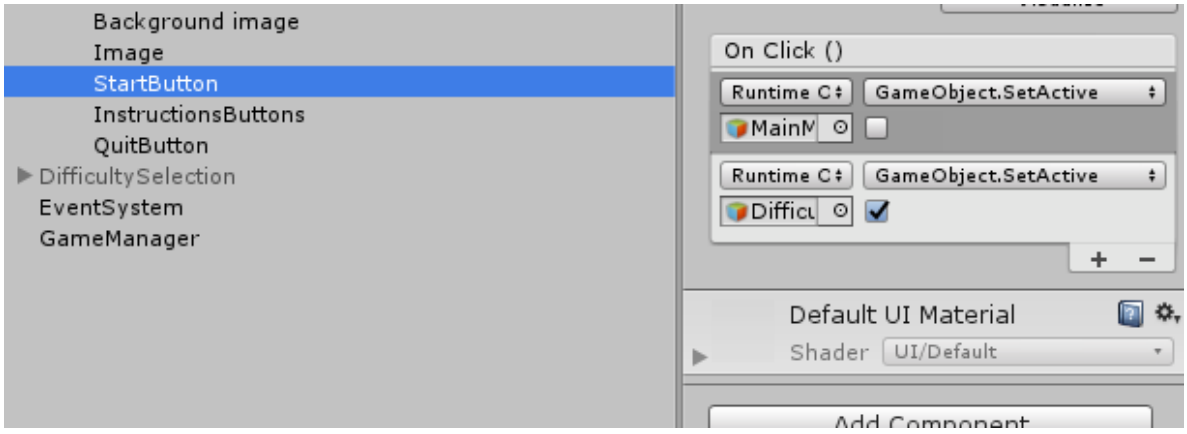
When I tested if the value changed, the results were :



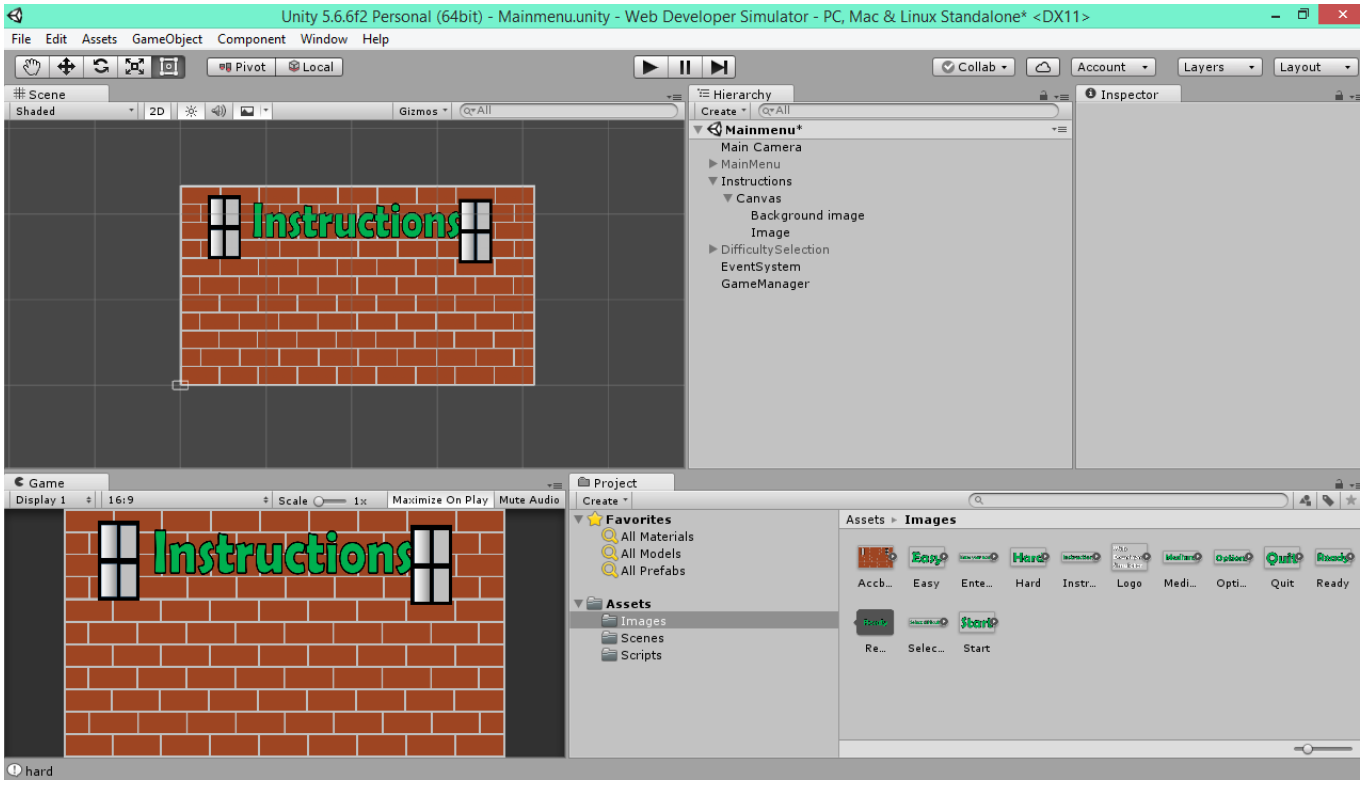
SC 1.1

Now that I knew the buttons were functional I went back to my main menu to create functionality for the instructions and quit button.

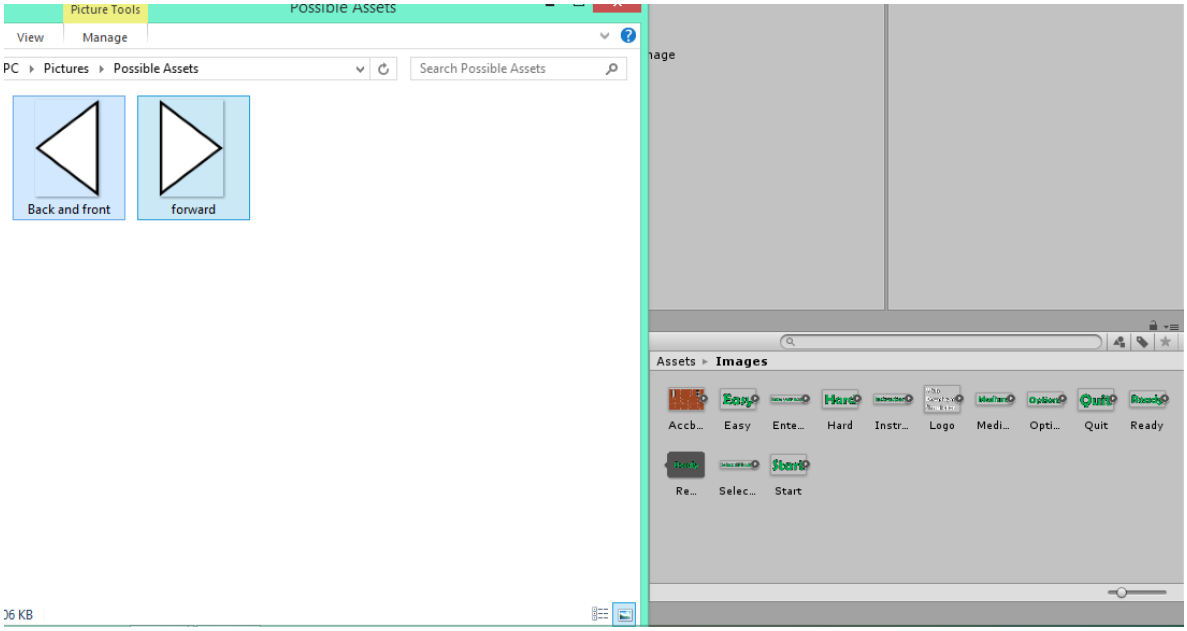
First I added functionality to the start button by using the on click feature of unity and draging the MainMenu object and the difficulty selection object to the on slot and used the method GameObject.SetActive() for both and set the MainMenu object to false and DifficultySelection to True so that when the Start button was clicked, it would stop showing the main menu screen and only show the difficulty selection screen. SetActive returns a boolean on whether the game object is True as in it shows on display or False where it is inactive.



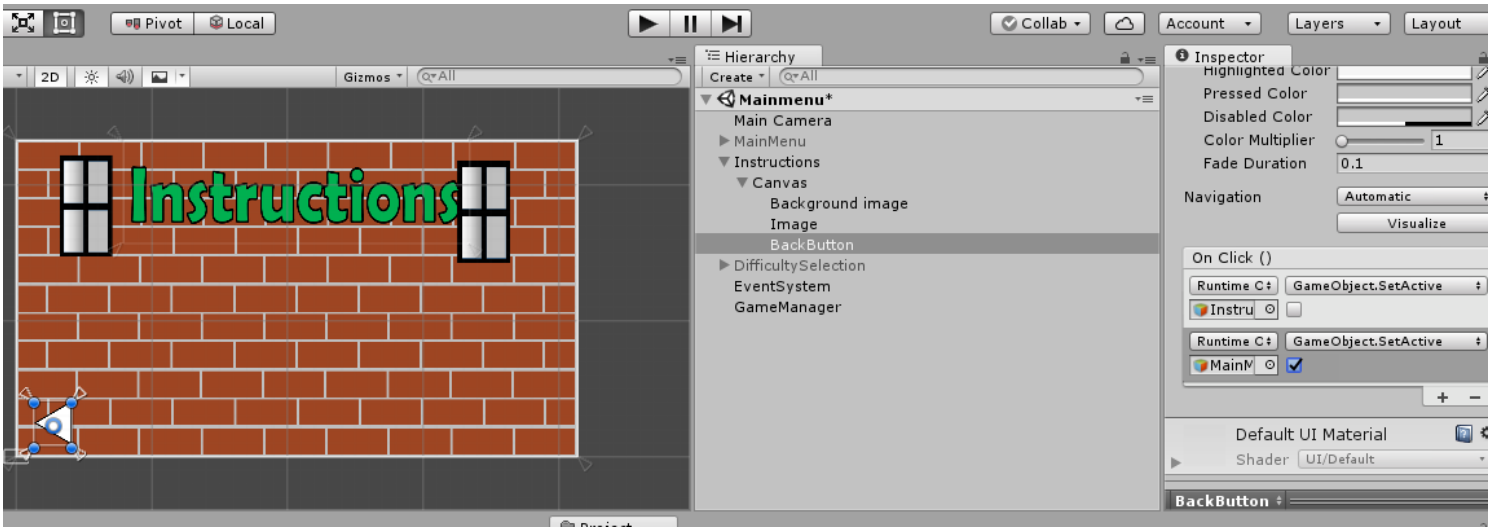
Next I created a new GameObject called instructions by copying and pasting the MainMenu object and started creating a Layout but I would leave the actual instructions until after I finshed creating the layout of the actual game because I may change the layout of my original design.



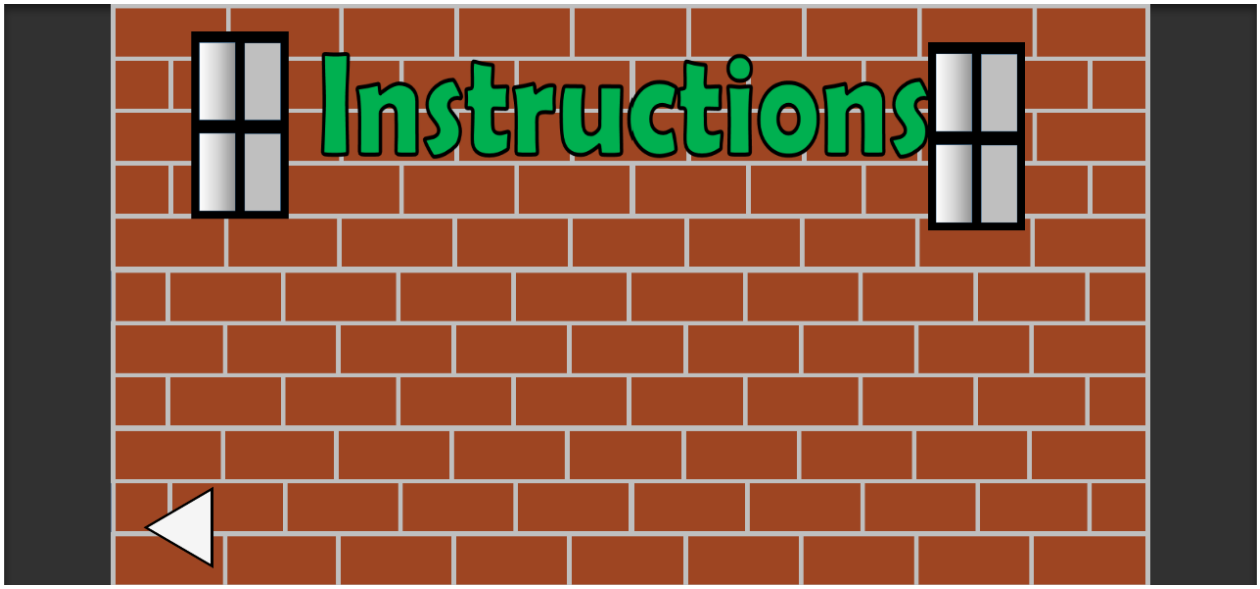
I imported a back and forwards button into my images folder in unity so I could use the back button to go back into the main menu once the player knew how to play the game.



I created a button called back button, edited it's source image and added the functionality to go back to the menu screen using the OnClick() feature with the instructions object and MainMenu

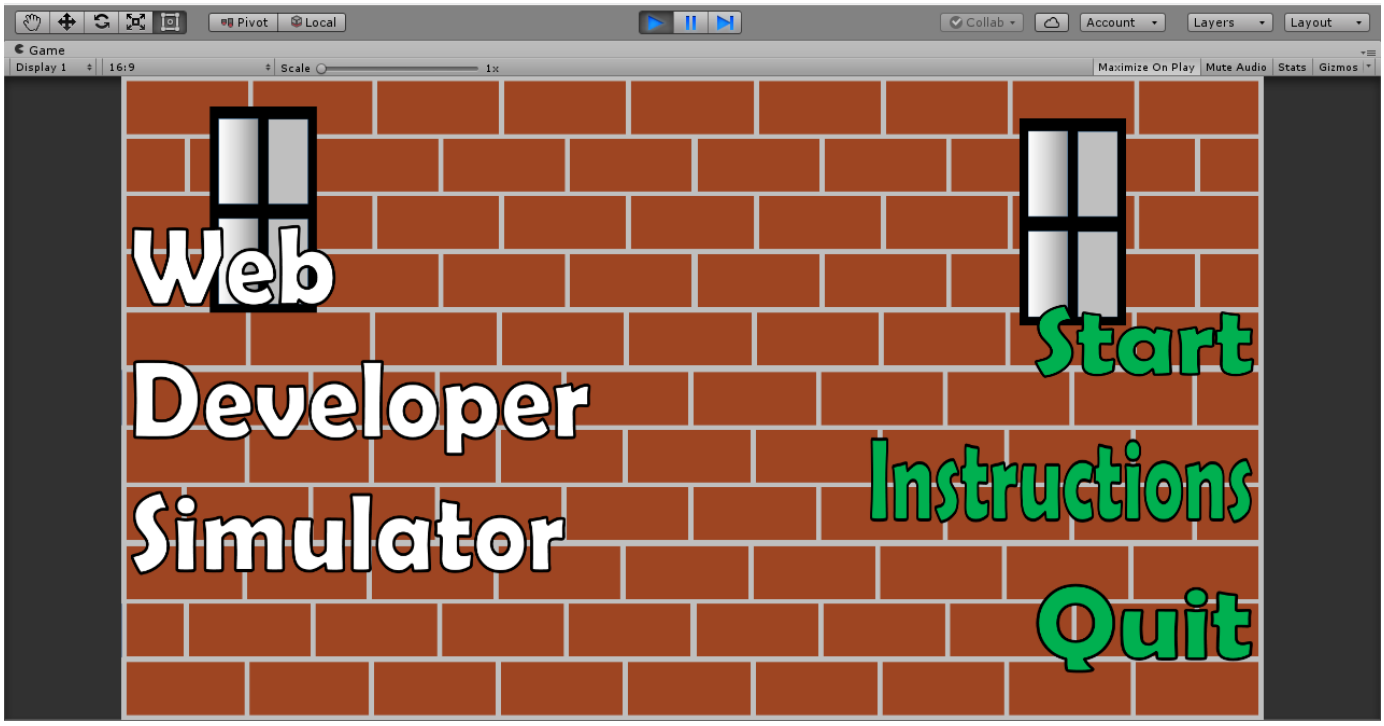


object.



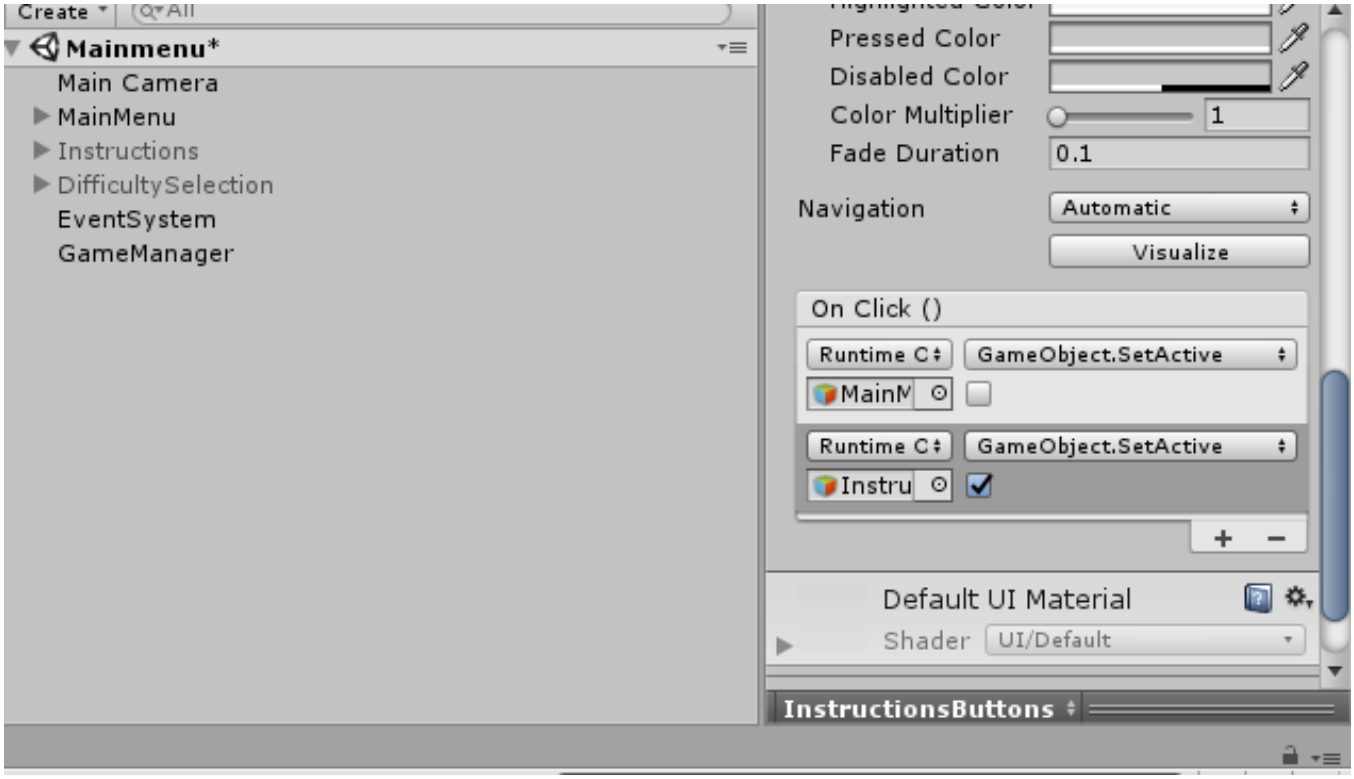
I tested the functionality of the back button and the result was:

SC 1.2.1



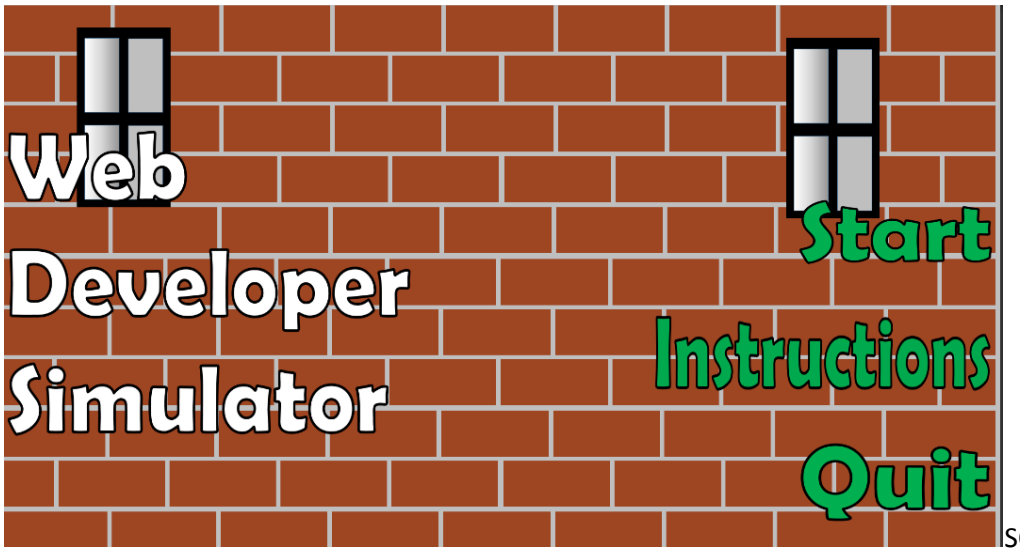
SC 1.2.2

This shows that it works. To access the instructions however, I would have to edit the onClick of the instructions button to display that screen. So I put the instruction object and Mainmenu object into the field and set the Booleans to the settings below to allow my user to access the instruction screen. Also note that I removed the options button as my game will not have a need for options until my game implements music but I do not yet have Music and will implement music only if I have enough time so until I implement music, I do not have a need for options in my game as it is not in my user specification to implement it.



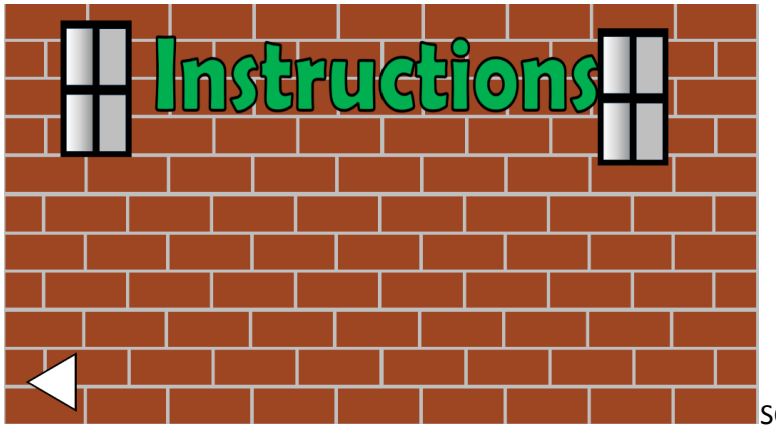
I tested the button and the result was:

Before click:



SC 1.3.1

After Click:

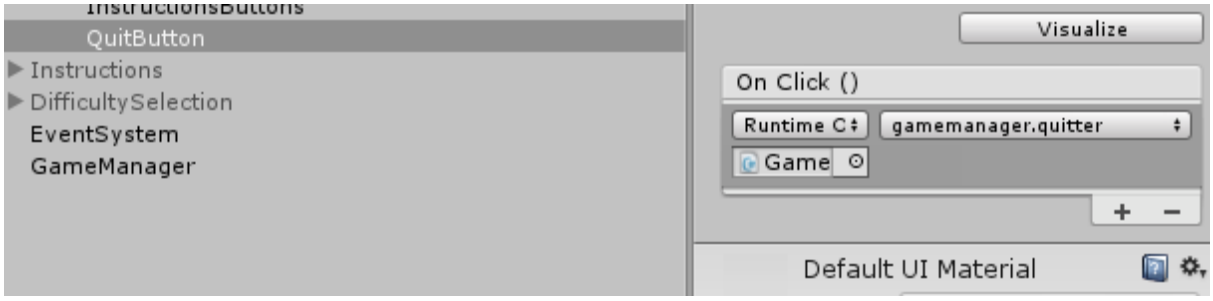


SC 1.3.2

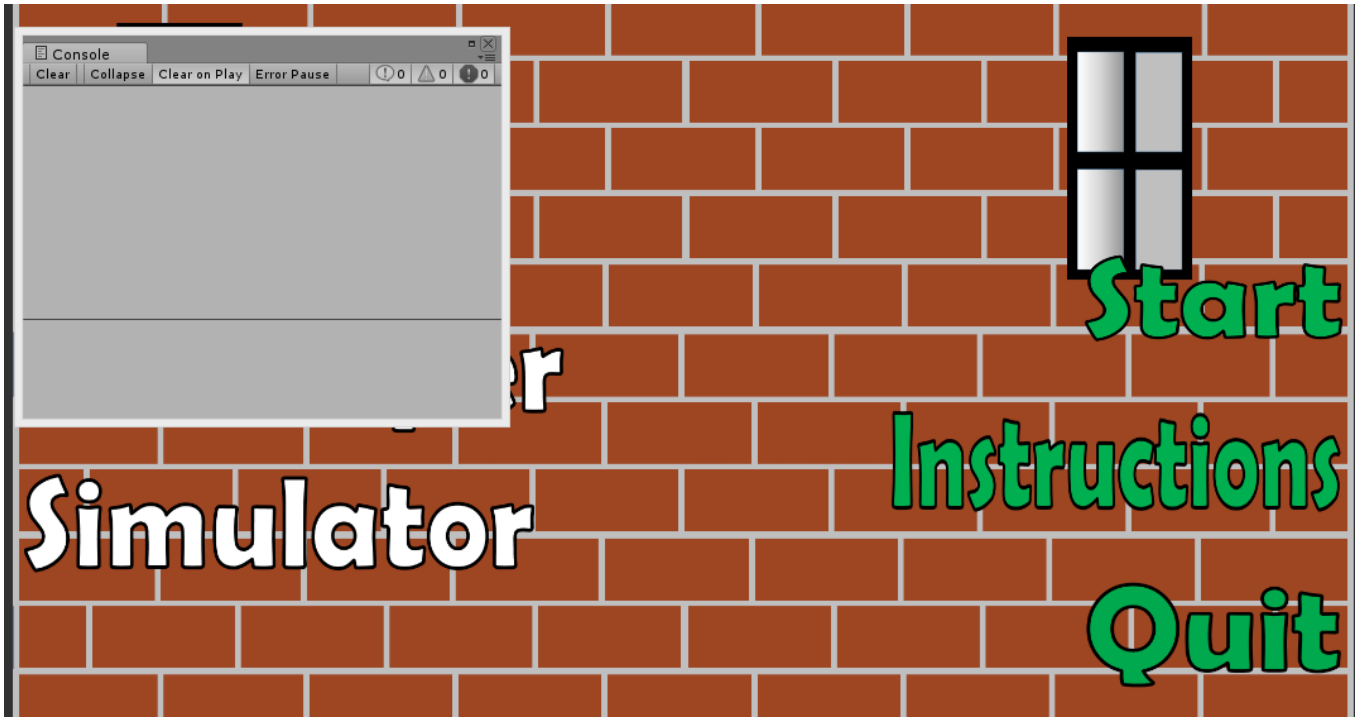
Lastly, for the main menu, I created another method on the gamemanager script to terminate the game using the function Application.Quit() and using Debug.Log to make sure my game terminated when the quit button was clicked because the editor itself does not quit the game.

```
// In this method the string "hard" will

public void quitter()
{
    Application.Quit();
    Debug.Log("Quitted");
}
}
```

I set the OnClick() for the quit button to the quitter() method.



I tested the button and the outcome was:



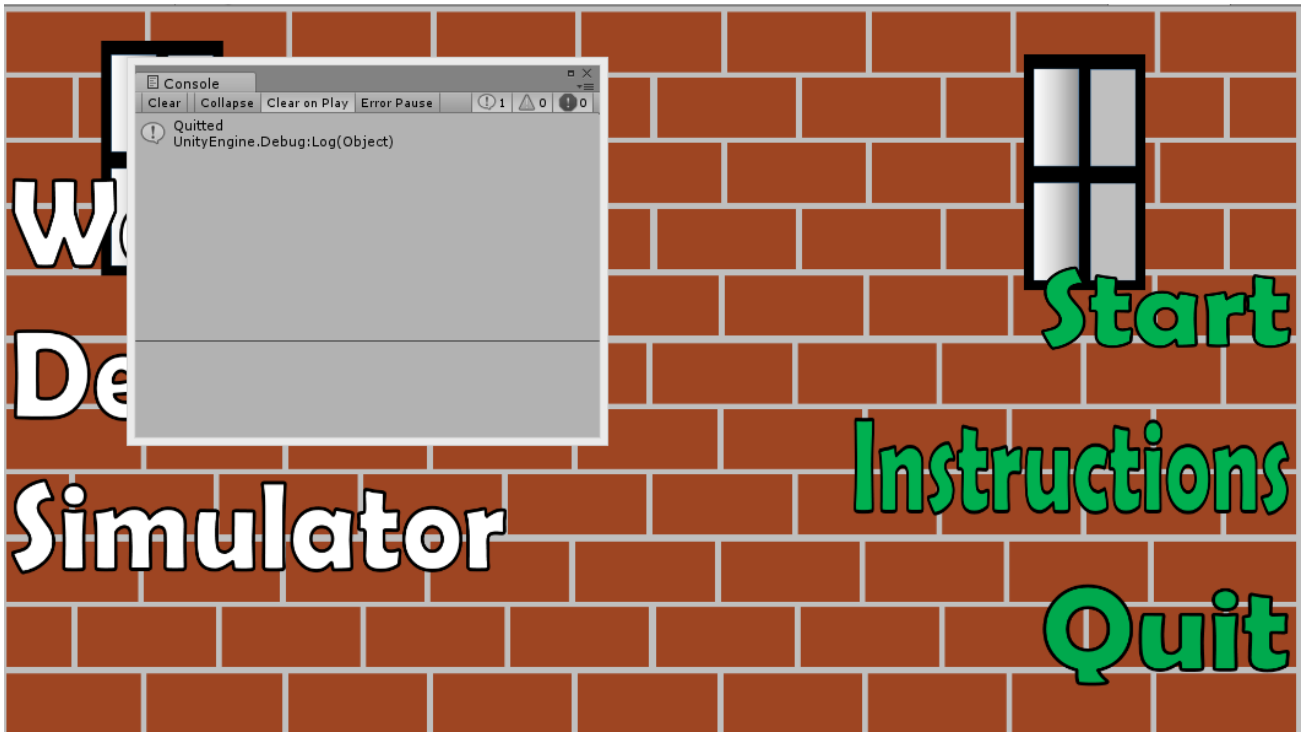
SC 1.4.1

No output. I thought this was because I coded the script to debug.Log after quitting the application so I swapped the order of the lines of code.

```
    public void EasySelected()
    {
        difficulty = ("easy");
        Debug.Log(difficulty);
    }
    // This is one of the methods of my class
    // In this method the string "easy" will be assigned difficulty
    // void here means the method will not return a value

    public void MediumSelected()
    {
        difficulty = ("medium");
        Debug.Log(difficulty);
    }
    // In this method the string "medium" will be assigned difficulty

    public void HardSelected()
    {
        difficulty = ("hard");
        Debug.Log(difficulty);
    }
    // In this method the string "hard" will be assigned difficulty

    public void quitter()
    {
        Debug.Log("Quitted");
        Application.Quit();

    }
}
```

What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
Main menu options	6. LC Start game button 7. LC Instruction 8. LC Exit 9. RC (invalid) 10. Any other key(invalid)	6. The game moves onto the difficulty selection 7. The game displays instruction screen 8. The game terminates 9. Null 10. Null	1. To be tested 2. SC1.3.1 to SC1.3.2 3. SC1.4.1 to SC1.4.2 4. To be tested 5. To be tested	1. N/A 2. As expected 3. As expected 4. N/A 5. N/A	3. Moving debug line to before the game is terminated otherwise, the game terminates before debugging and therefore does not display in the console.
Instructions	3. LC back button 4. RC back button (invalid)	3. Goes back to title screen 4. Null	1. SC1.2.1 to 1.2.2 2. To be tested	1. As expected	
Difficulty section	6. LC easy 7. LC medium 8. LC hard 9. RC (invalid) 10. Any other key(invalid)	6. Easy appears on console 7. Medium appears on console 8. Hard appears on console 9. Null 10. Null	1 2 and 3 tested in one screenshot SC1.1 4.To be tested 5.To be tested	1. As expected	

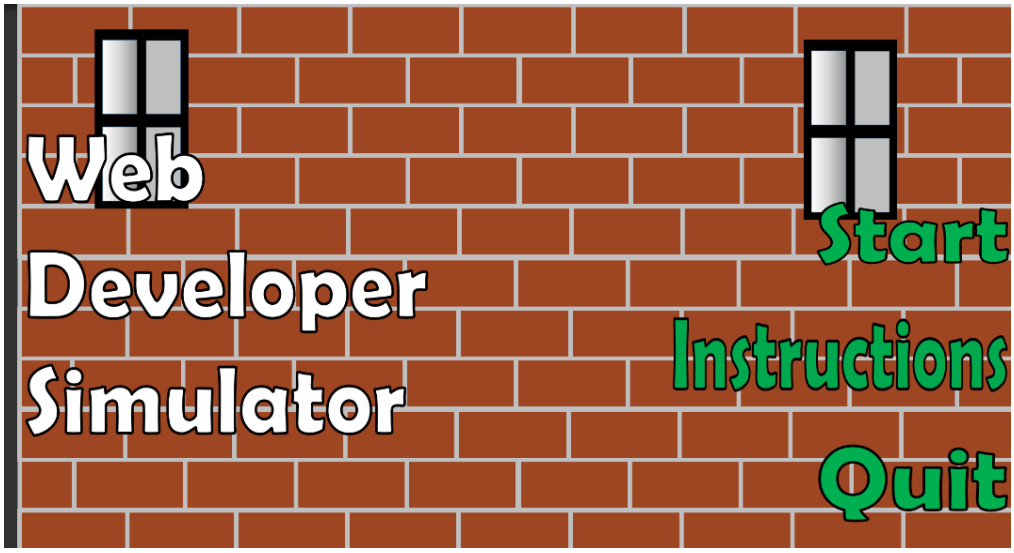
The outcome when tested was:



1.4.2

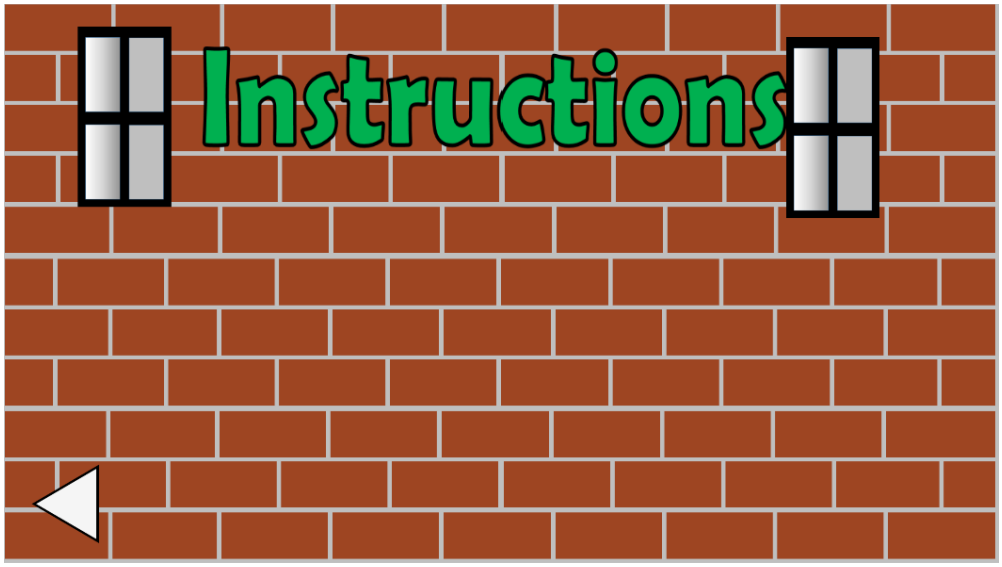
Now I test the tests that I have not completed yet.

After RC all of the buttons, nothing happened as expected. The same outcome was shared for pressing the button B which led to no affect as expected.

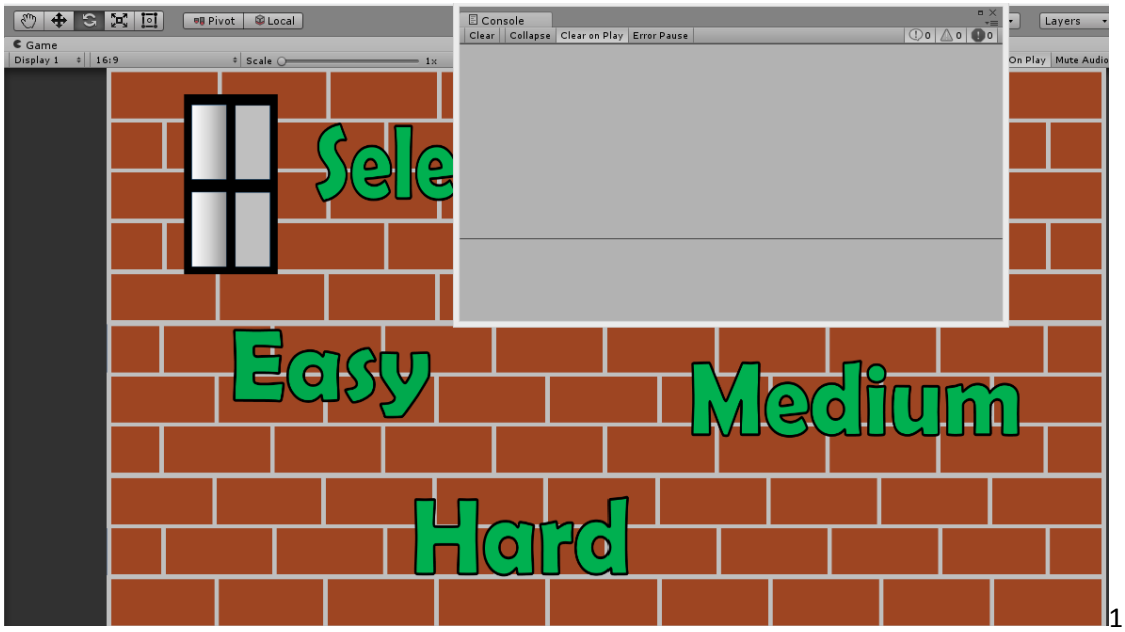


1.5.1

When RC the back button on the instructions page and had no effect as expected because it is meant to be an invalid test



1.5.2

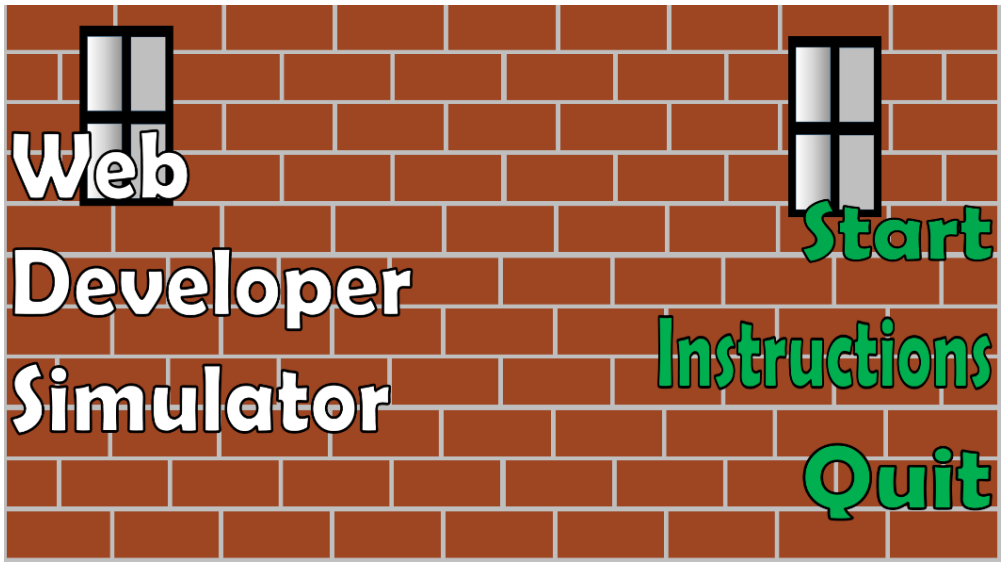


1.5.3

As shown above, RC any of the buttons or pressing any keyboard button had no effect and did not change value of difficulty, as expected in my test plan.

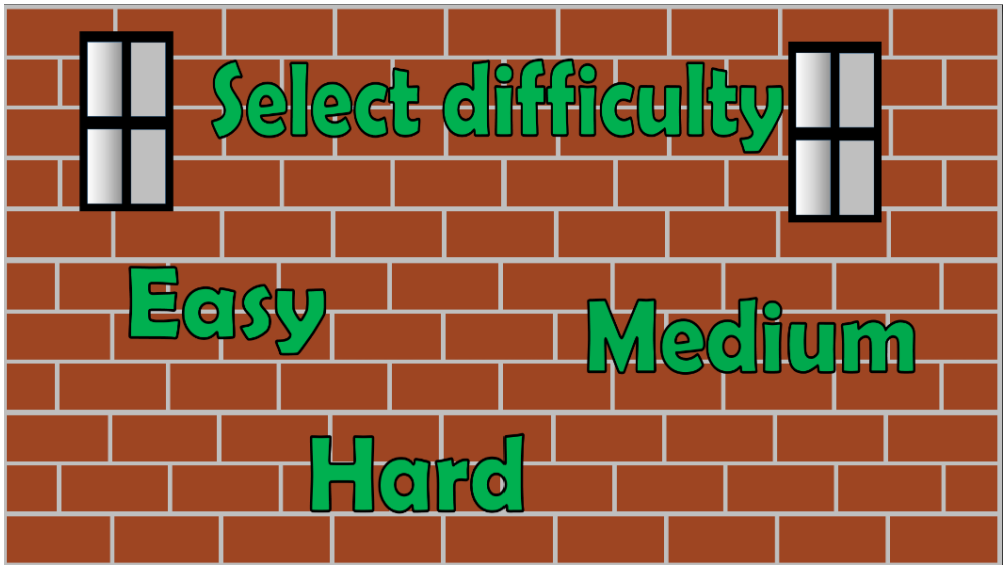
Testing if LC start led to difficulty selection

Before click



1.6.1

After click



1.6.2

What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
Main Menu	1.LC Start game button 4.RC (invalid) 5.Any other key(invalid)	1.The game moves onto the difficulty selection 4.Null 5.Null	1)1.6.1 to 1.6.2 4)1.5.1 5)1.5.1	As expected As expected As expected	

Instructions	RC back button (invalid)	Null	1.5.2	As expected
Difficulty selection	4.RC (invalid)	Null	1.5.3	As expected
	5.Any other key(invalid)	Null		

Stakeholder feedback Main menu: Week Starting 3/12/18

After testing my main menu, I believe my first prototype is complete and ready to be shown to my stakeholder for feedback.

Summary of M.Miah's feedback:

Success criteria

Success Criteria	Justification
Must allow User to choose from different difficulty levels. More specifically, the game must have 3 difficulties. Easy, Medium and Hard	Many children will start on a different level to others. This also allows the player to learn at their own pace. My questionnaire justifies this decision.
Game must have working Start screen allowing user to begin or quit	This is a requirement in most games to terminate the game itself if in full screen mode

And evidently from my development testing I have met both of these criteria related to my main menu. I showed my stakeholder, M.Miah, my first prototype and she agreed that both of these criteria have been met.

What went well ~

She said that the she liked the design and layout of the main menu as she did when she reviewed the screen designs in the usability features section of my design work.

We discussed how I would add instructions based on the screenshots of the actual finished game rather than any approximate I would make in design section and justified this by saying that it would make the instructions easy to learn as it would have the exact layout of the full game and she thought this was a sufficient reason to wait to add instructions.

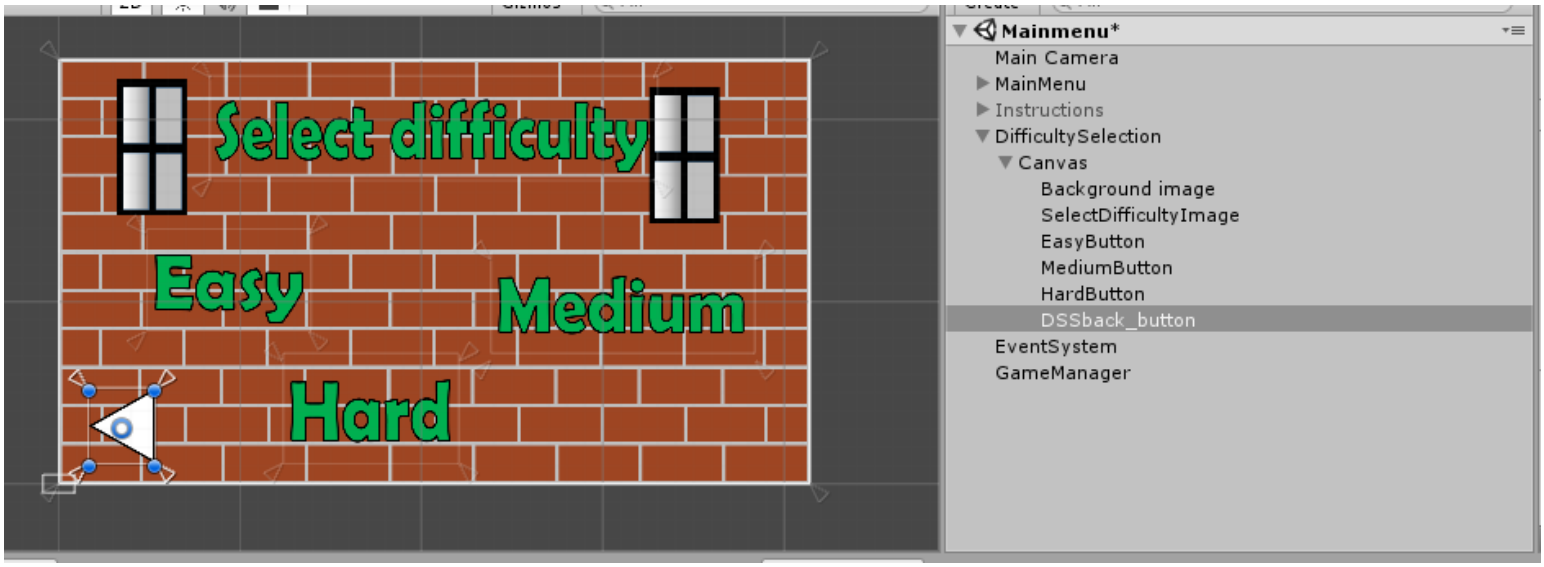
Improvements ~

To add instructions once I have the final design of the game and add a back button to the difficulty selection screen as she believed this would be a useful feature in case the user wants to review the instructions once more before playing so I added these to the success criteria.

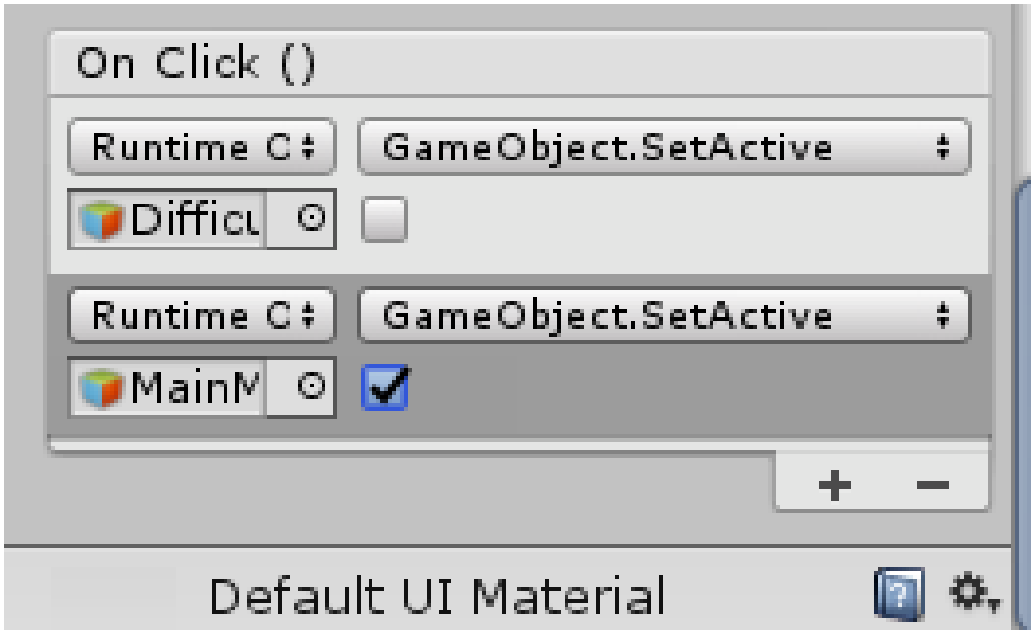
Success Criteria	Justification
31. Instructions must use screenshots of the actual design of the game.	This means that the instructions will be clearer and correlate to the final design of the game.
32. Back button on difficulty selection screen allows user to return to title menu	My stakeholder M.Miah believed this would be a useful feature in case the user wanted to review the instructions once more before playing.

Improvements on main menu: Week Starting 3/12/18

To meet my new success criteria marked no.32 I added a button on the difficulty selection screen

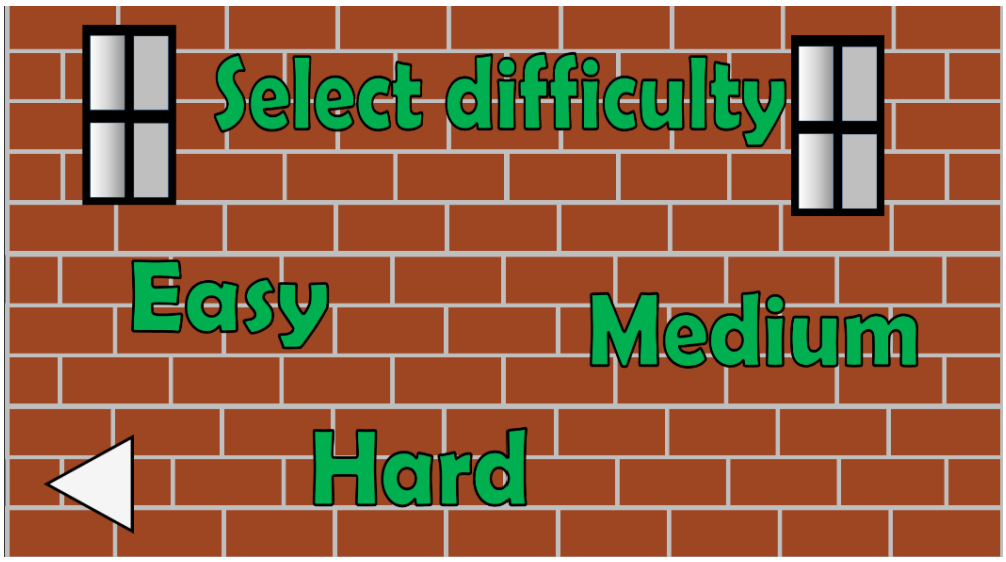


named DSSback_button and mapped the onClick() to go back to the main menu.



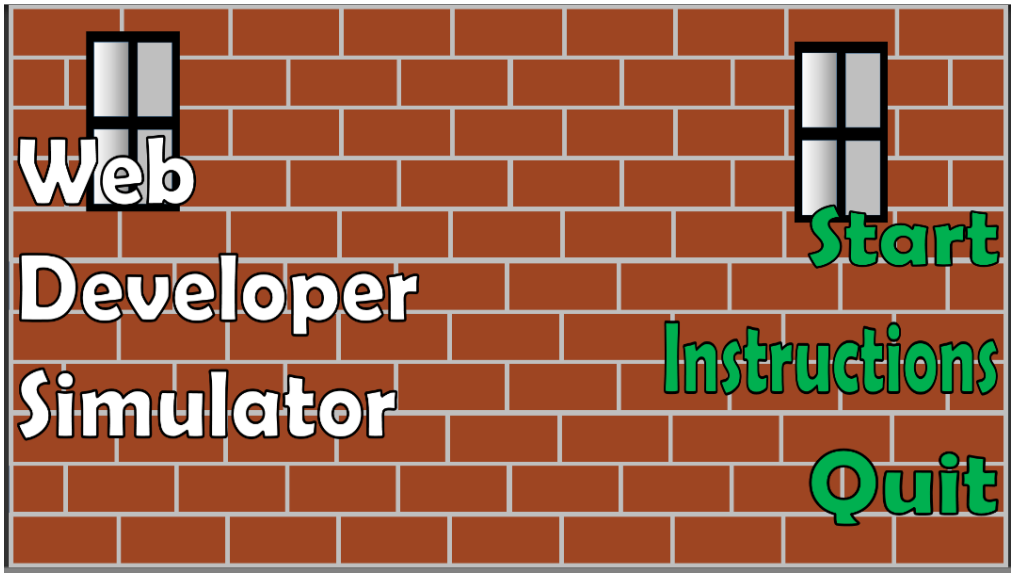
I then tested this to ensure the button did return the user to the main menu.

Before click



1.6.3

After click



1.6.4

I will no longer need to RC as a test as the onClick() mapping only works for Left mouse click so to test would be redundant.

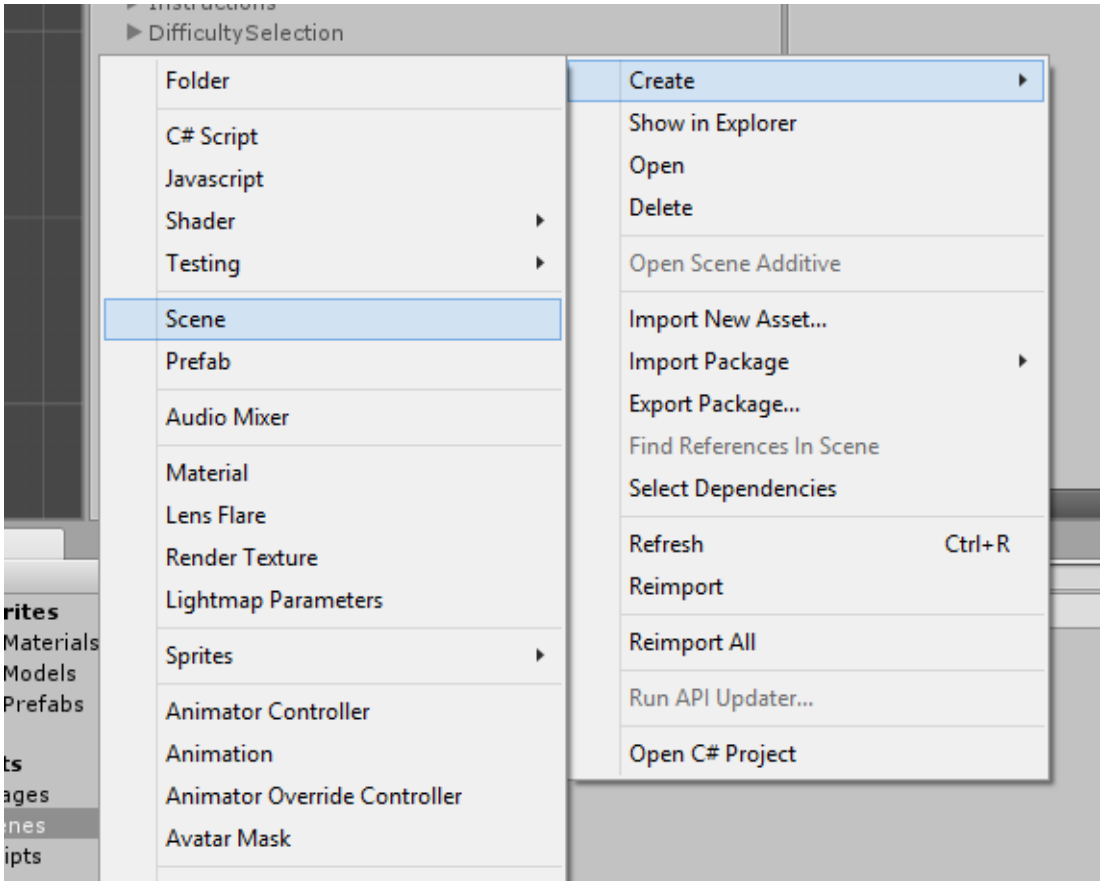
What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
Difficulty selection Back button	6.LC back button	6. Returns user to main menu	1.6.3 to 1.6.4	As expected	

Upon showing my stakeholder, M.Miah, my new main menu, she agreed this section of my game was as complete as it could be till I have screenshots of my final design in my game as I have met success criteria no.32.

Prototype 2

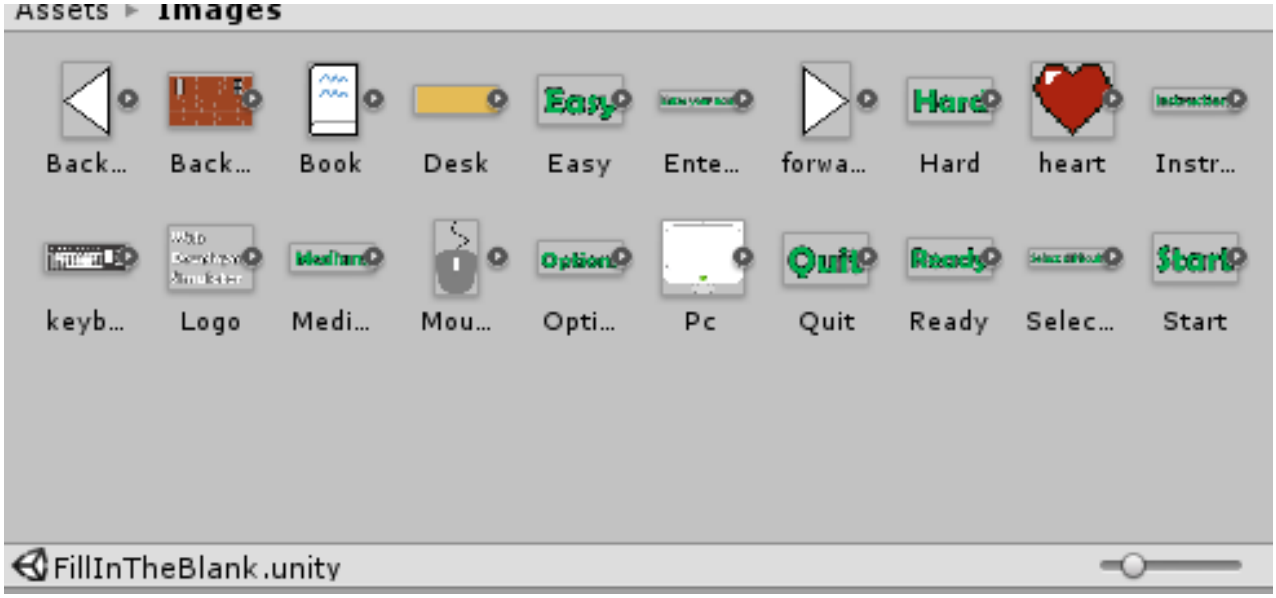
Fill in the blank challenges: Week Starting 10/12/18

I decided to start creating the UI for the fill in the blank screen to ensure that I would have a usable simulator within my time frame. I began by creating a new scene called FillInTheBlank. I felt this was an appropriate name as it would contain the UI for fill in the blank challenges and is understandable for when I need to do further maintenance.


DifficultySelection
Folder
C# Script
Javascript
Shader
Testing
Scene
Prefab
Audio Mixer
Material
Lens Flare
Render Texture
Lightmap Parameters
Sprites
Animator Controller
Animation
Animator Override Controller
Avatar Mask
Create
Show in Explorer
Open
Delete
Open Scene Additive
Import New Asset...
Import Package
Export Package...
Find References In Scene
Select Dependencies
Refresh Ctrl+R
Reimport
Reimport All
Run API Updater...
Open C# Project


Inspector
FillInTheBlank
Asset Labels
Assets ▸ Scenes
FillI...
Mai...

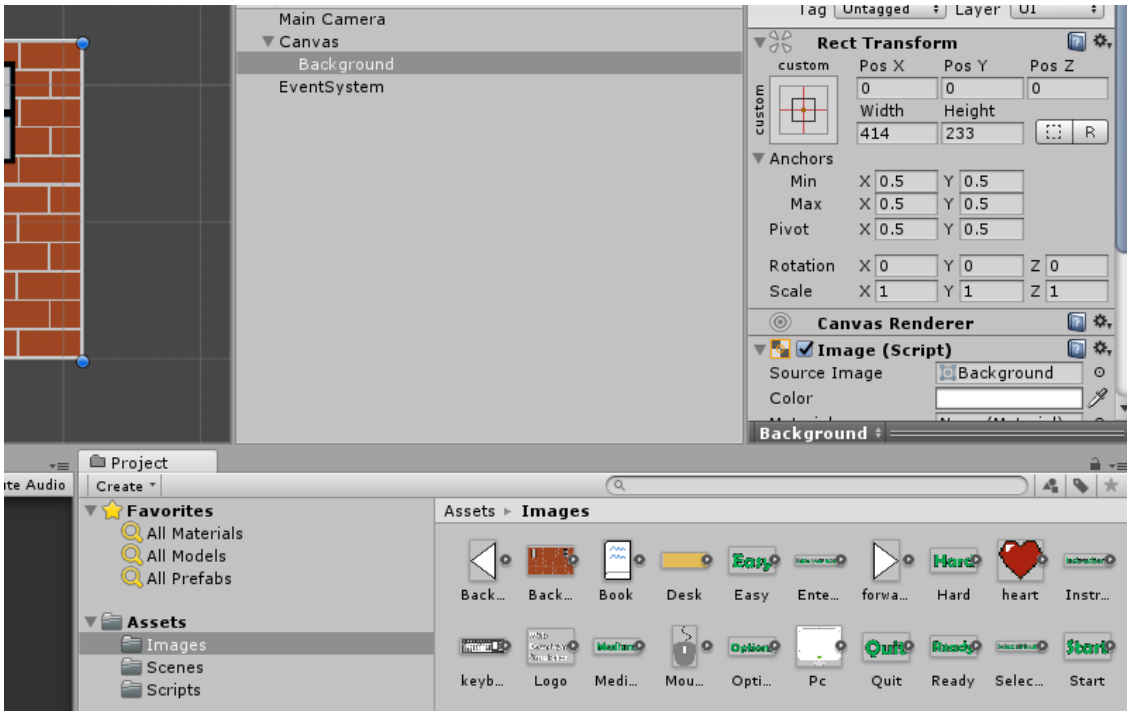
I started by importing the background, desk, pc, keyboard, mouse, book and heart images into my images folder in unity. I stored these images in a folder called “prep” to help me keep track with which assets I need to import.

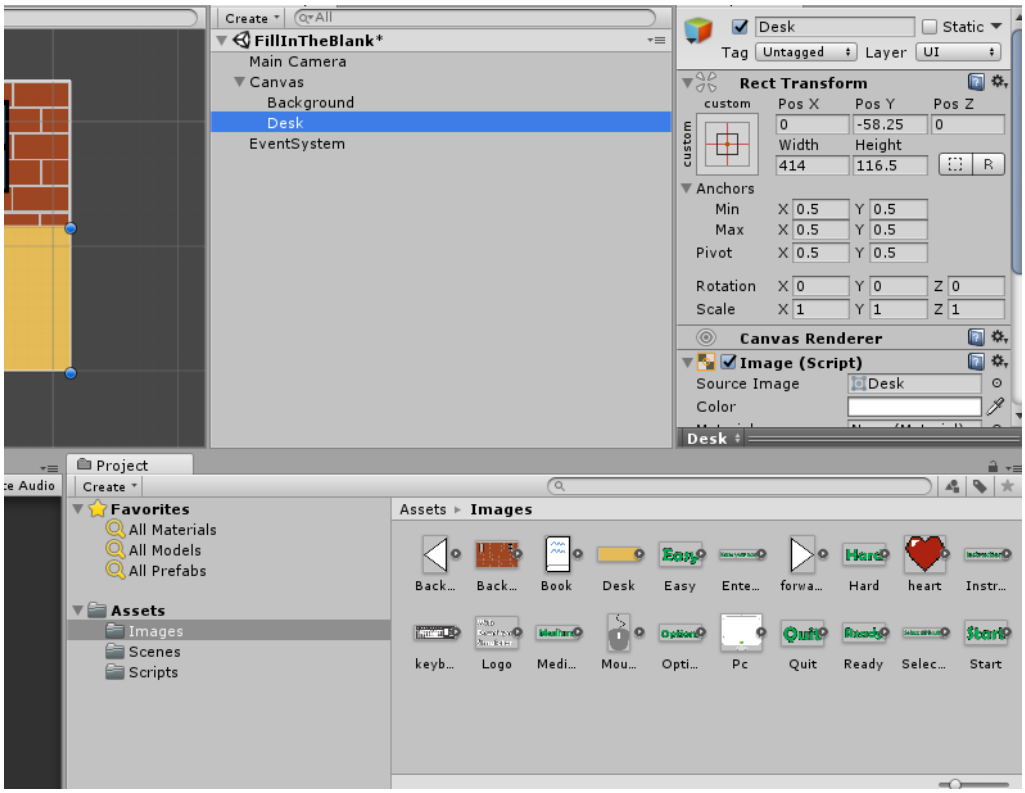


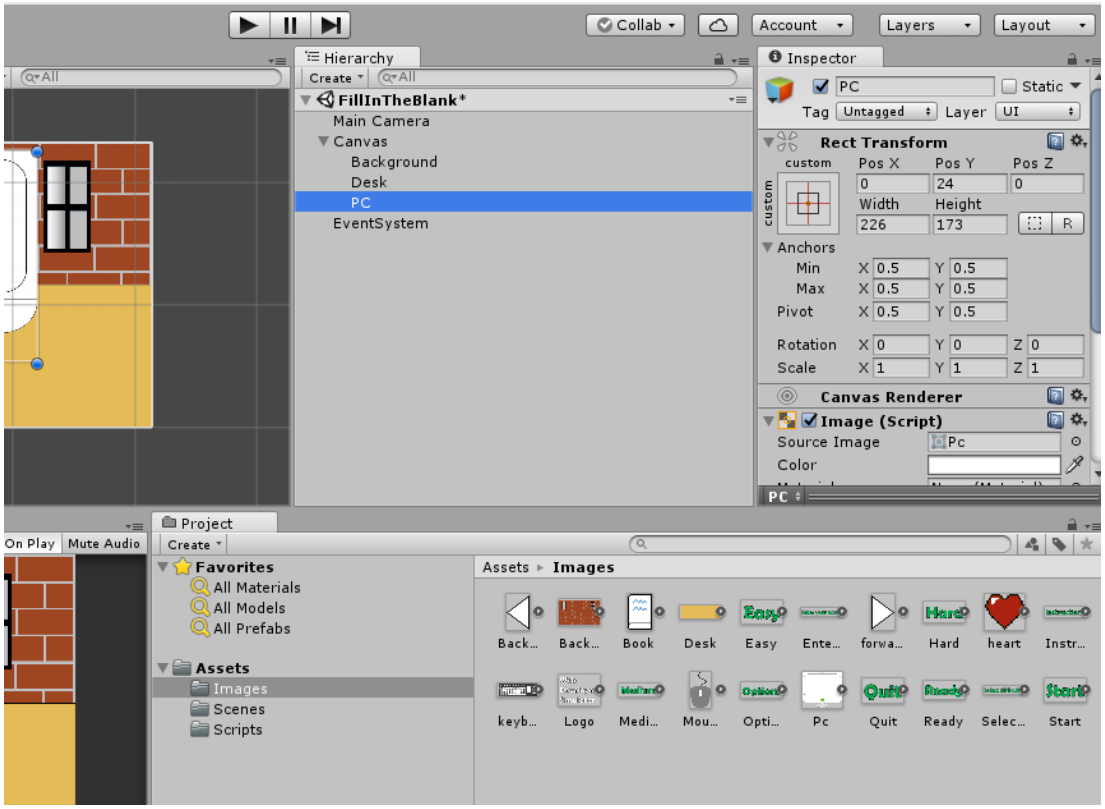
I then created multiple images and changed the source of the image to their matching asset.

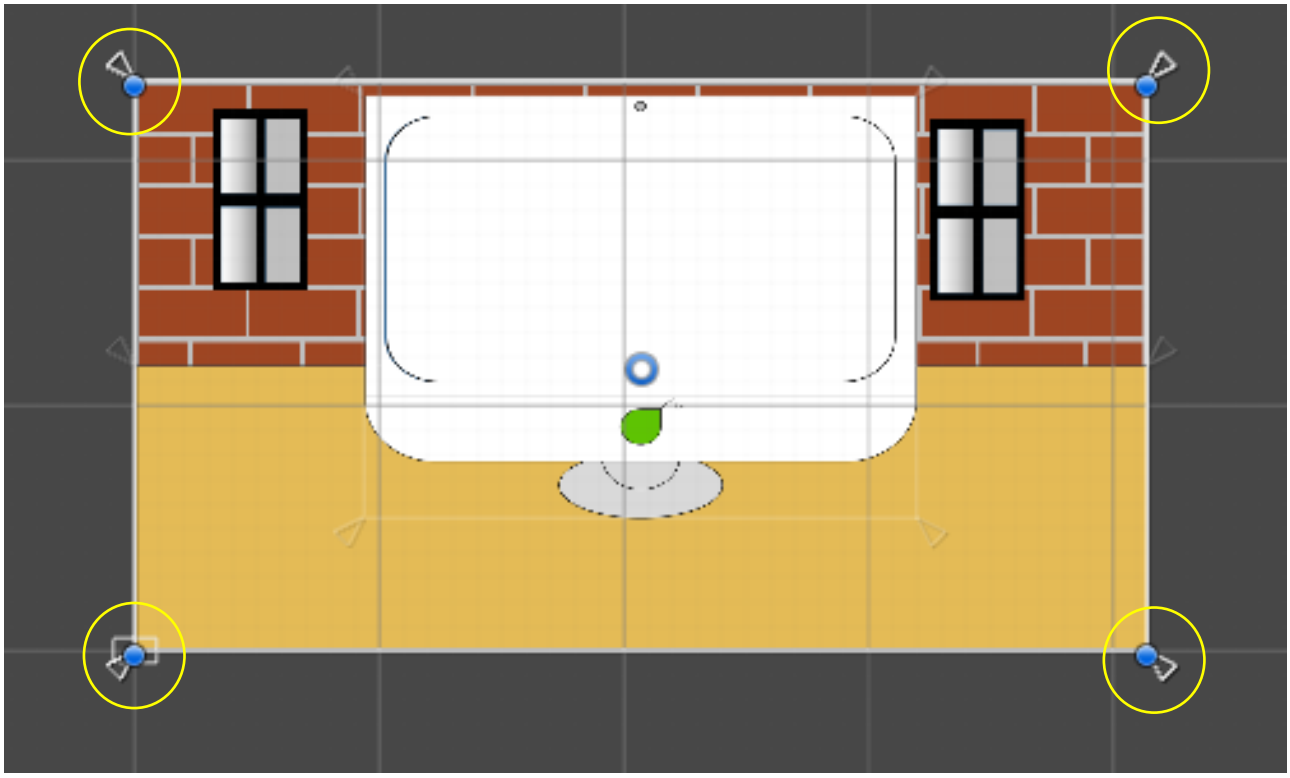
First I started by renaming the respective assets to what image they would hold, so the image with the background image as its source would be renamed background and so on.

I did this for the background image, then the desk image and finally the Pc image before anchoring them into place.

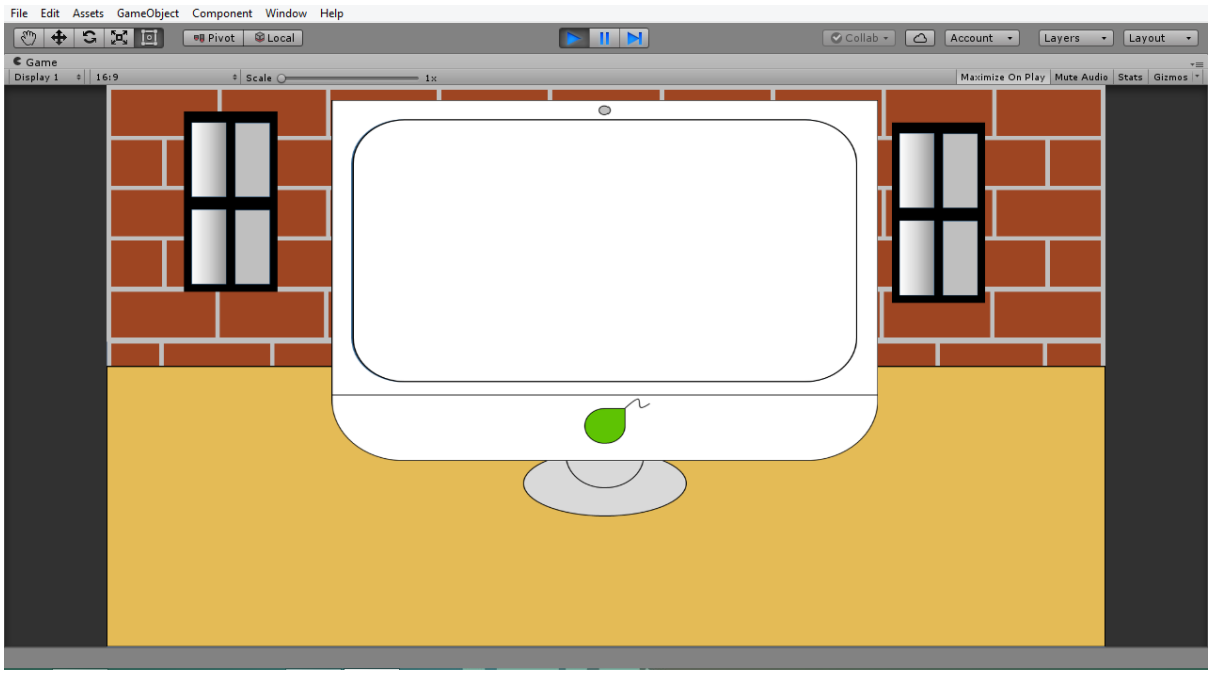



Create
FillInTheBlank*
Main Camera
Canvas
Background
Desk
EventSystem
Desk
Static
Tag Untagged
Layer UI
Rect Transform
custom
Pos X 0
Pos Y -58.25
Pos Z 0
Width 414
Height 116.5
Anchors
Min X 0.5 Y 0.5
Max X 0.5 Y 0.5
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Canvas Renderer
Image (Script)
Source Image Desk
Color
Project
Create
Favorites
All Materials
All Models
All Prefabs
Assets
Images
Scenes
Scripts
Assets ▸ Images
Back...
Back...
Book
Desk
Easy
Ente...
forwa...
Hard
heart
Instr...
keyb...
Logo
Medi...
Mou...
Opti...
Pc
Quit
Ready
Selec...
Start


Collab
Account
Layers
Layout
Hierarchy
Create
FillInTheBlank*
Main Camera
Canvas
Background
Desk
PC
EventSystem
Inspector
PC
Static
Tag Untagged
Layer UI
Rect Transform
custom
Pos X 0
Pos Y 24
Pos Z 0
Width 226
Height 173
Anchors
Min X 0.5 Y 0.5
Max X 0.5 Y 0.5
Pivot X 0.5 Y 0.5
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Canvas Renderer
Image (Script)
Source Image Pc
Color
Project
On Play
Mute Audio
Create
Favorites
All Materials
All Models
All Prefabs
Assets
Images
Scenes
Scripts
Assets ▸ Images
Back...
Back...
Book
Desk
Easy
Ente...
forwa...
Hard
heart
Instr...
keyb...
Logo
Medi...
Mou...
Opti...
Pc
Quit
Ready
Selec...
Start



And they were anchored properly as shown in when testing.

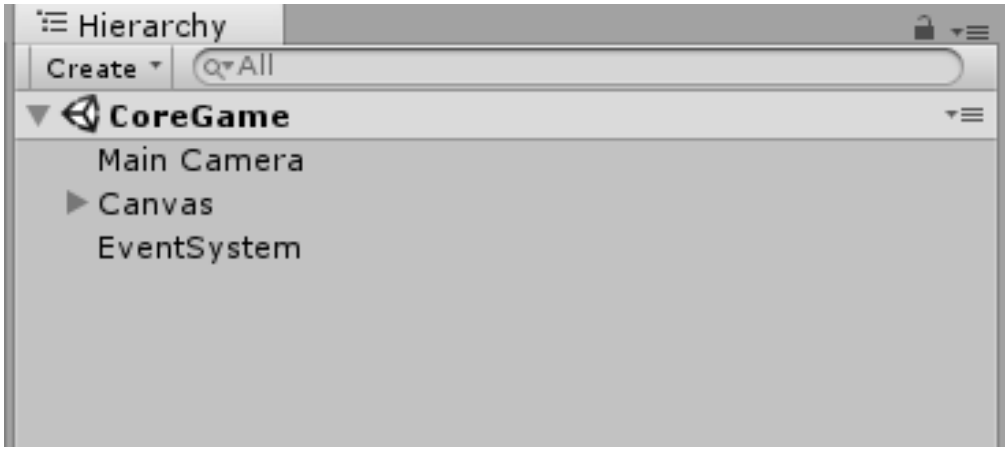


Upon reviewing my code for the main menu again, I went back to the quitter method to annotate it so it would be comprehensible to anyone who does further maintenance on the system. I made the following change:

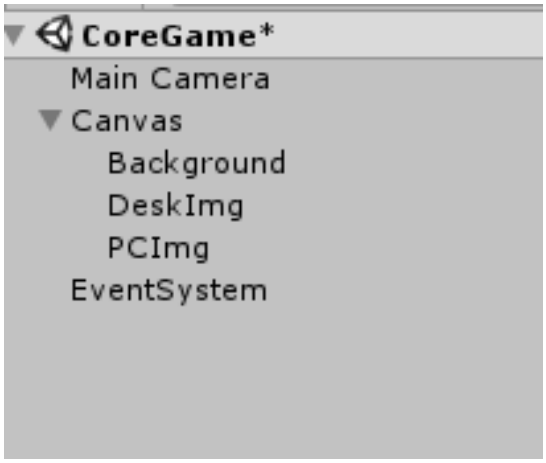
```
public void quitter()
{
    Debug.Log("Quitted");
    Application.Quit();
    // This method will terminate the game. This will be linked to the quit buttons OnClick()
    //"Application.Quit();" This is the built-in function that allows the program to terminate
}
```

Then I went back to creating the layout of the fill in the blank challenges.

I decided to rename the scene from FillInTheBlank to CoreGame as it will include the fill in the blank questions and multiple choice questions both in one scene rather than separately as I am able to randomly choose between which gameobject is active using the setActive() function and randomly generating a number between 1 and 2 and assigning Multiple choice and fill in the blank questions to one of the numbers so which ever number is generated, the game will display that gameobject.

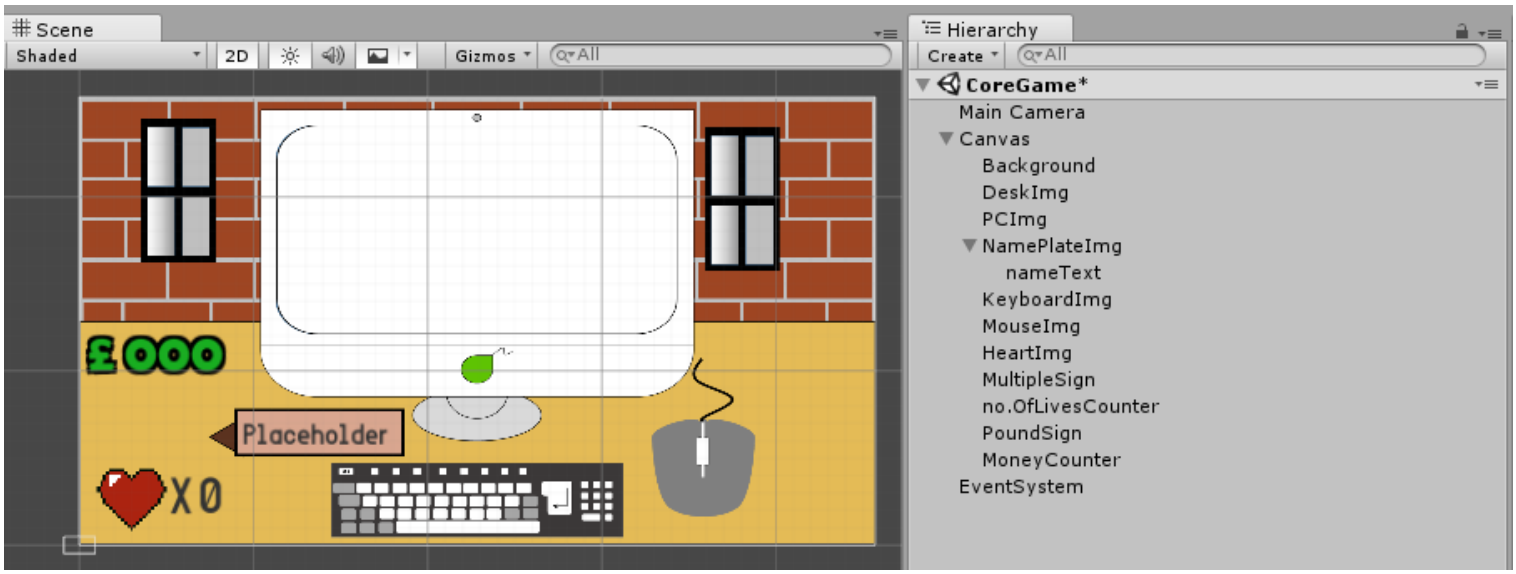


I then renamed a few of the image elements to be more representative of what they are.

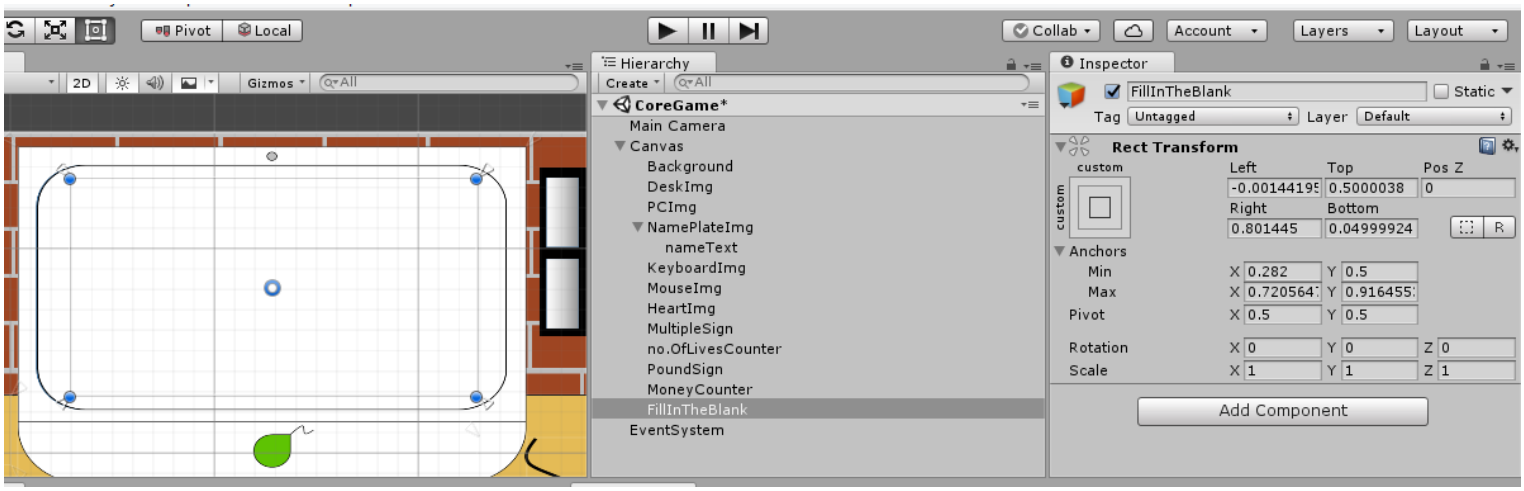


And then finished off adding the other features to match my screen design. I also edited the screen design to display the number of lives which I left out in my screen designs and was requested by my stakeholder in the proposal sign off.

First I imported the sprites and fonts and then added the elements into the game. Then I added the



source image of the buttons and images to their respective elements.

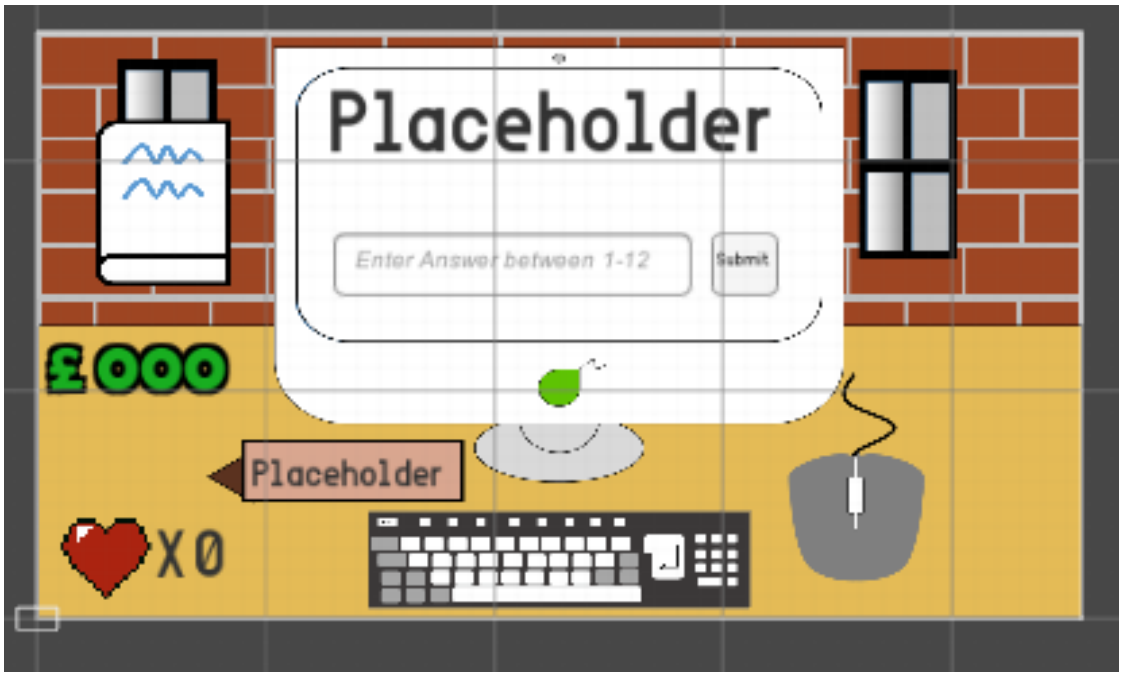


I then created a game object called FillInTheBlank and added the elements that will be used in Fill in the blank challenges and added the rect transformer function so the challenges scale with the screen.

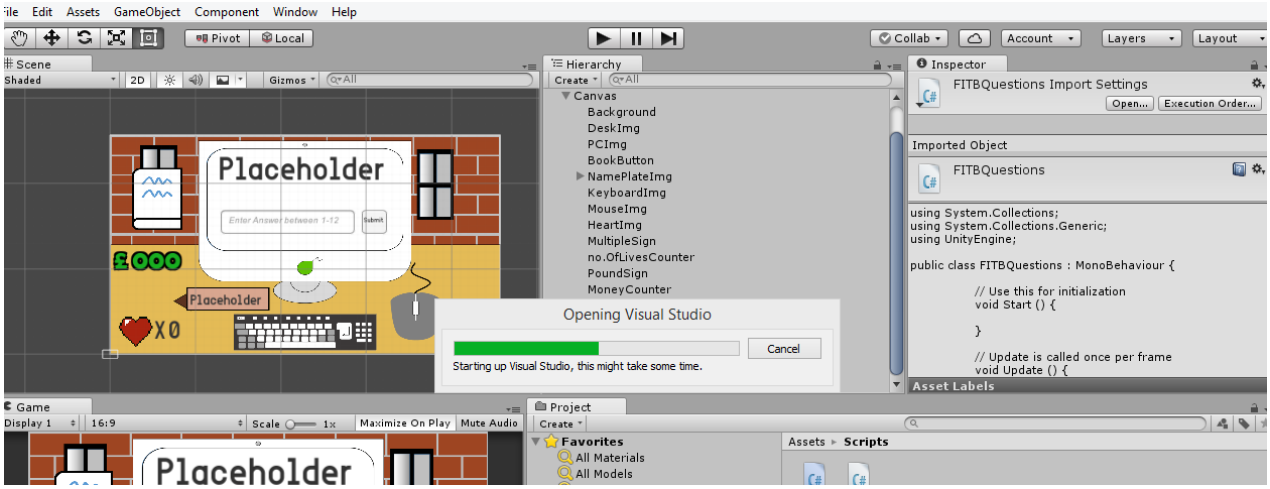
Then I added the elements and named them appropriately and edited all of their rect transformers

to scale with the screen.

Then I added the book feature button as part of the core games layout.



Next I created a new script called FITBQuestions that will hold the class definition of FITB questions including the questions and the answer so I can create lists of these FITBQuestions objects for each of the difficulties.



I then created the following script:

```
1
2   [System.Serializable]
3   public class FITBQuestions
4   {
5
6       public string theQuestion;
7       public string FITBanswer;
8
9   }
10
11
```

I first got rid of the “using” lines of code as I would not be using any of the given ones when opening a c# script in unity. Then I added the line2 [system. Serializable] so that when I create the lists of different questions, I can input what I want the questions and answers to be in the inspector.

Line 3 declares the class, FITBQuestions appropriately named as it defines what the object, FITBQuestions is defining.

Line 6 and Line 7 I have declared public strings theQuestion and FITBanswers that will hold the question string displayed when playing the game and the string the users answer will be compared to when they submit their answer, checking if the user’s response is correct. They are public to declare their accessibility type and as I will be changing their values in different scripts, it is effective to make them public.

I then annotated for further maintenance purposes:

```
1   //Allows dev to edit questions in unity inspector
2   [System.Serializable]
3   public class FITBQuestions
4   {
5
6       public string theQuestion;
7       //will hold value of FITB question
8       public string FITBanswer;
9       //holds value for corresponding answer
10
11  }
12
13
```

I renamed the variable FITBanswer to fitbAnswer which makes use of camel casing and is a form of best practice.

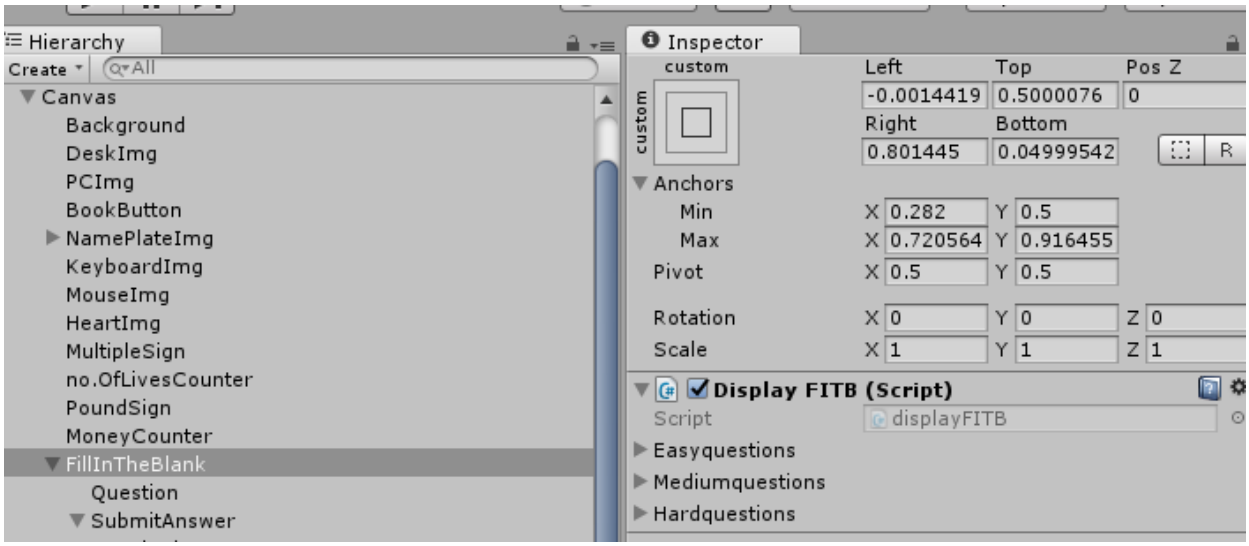
```
1   //Allows dev to edit questions in unity inspector
2   [System.Serializable]
3   public class FITBQuestions
4   {
5
6       public string theQuestion;
7       //will hold value of FITB question
8       public string fitbAnswer;
9       //holds value for corresponding answer
10
11  }
12
13
```

Next I created a new script called displayFITB that would change the text on the computer to the questions.

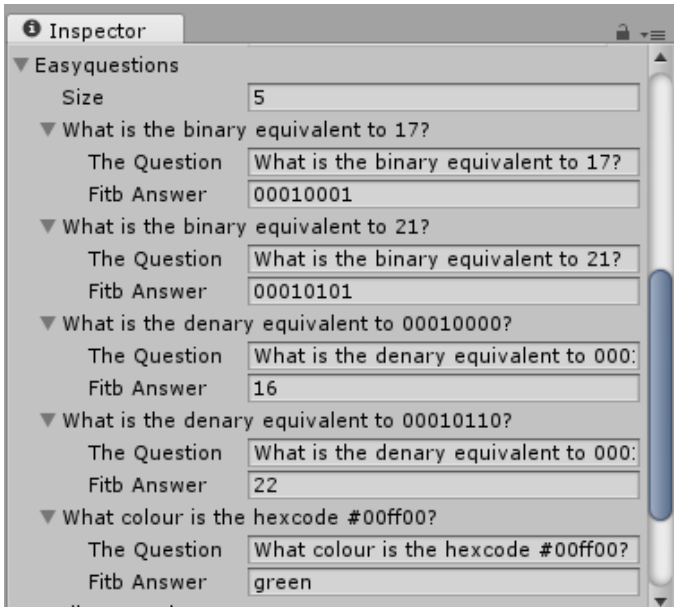
```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4
5   public class displayFITB : MonoBehaviour {
6
7       public FITBQuestions[] easyquestions;
8       public FITBQuestions[] mediumquestions;
9       public FITBQuestions[] hardquestions;
10      //holds list of questions and answers for FITB questions
11
```

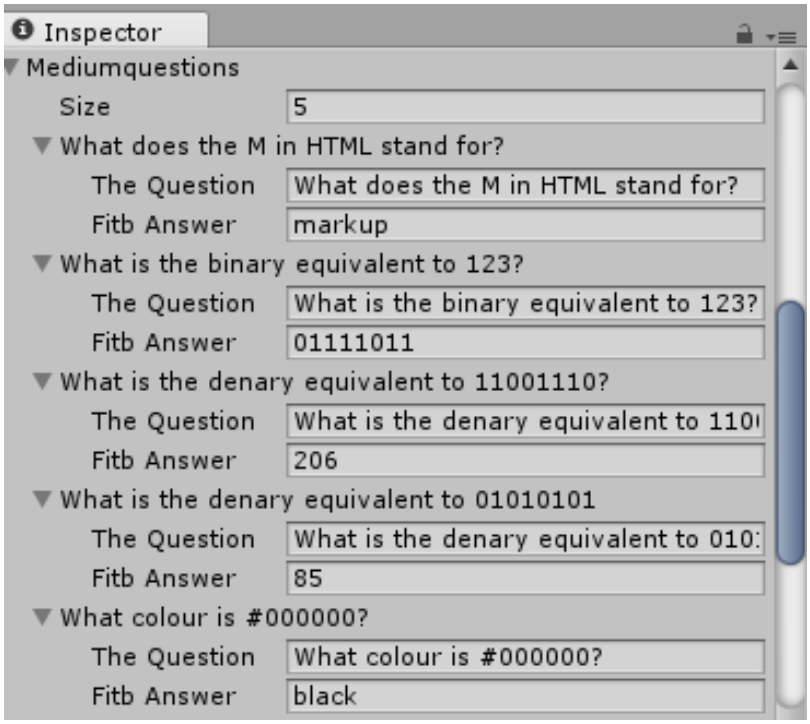
Lines 7-9 here declare a list of objects named to represent the difficulty of questions they hold.

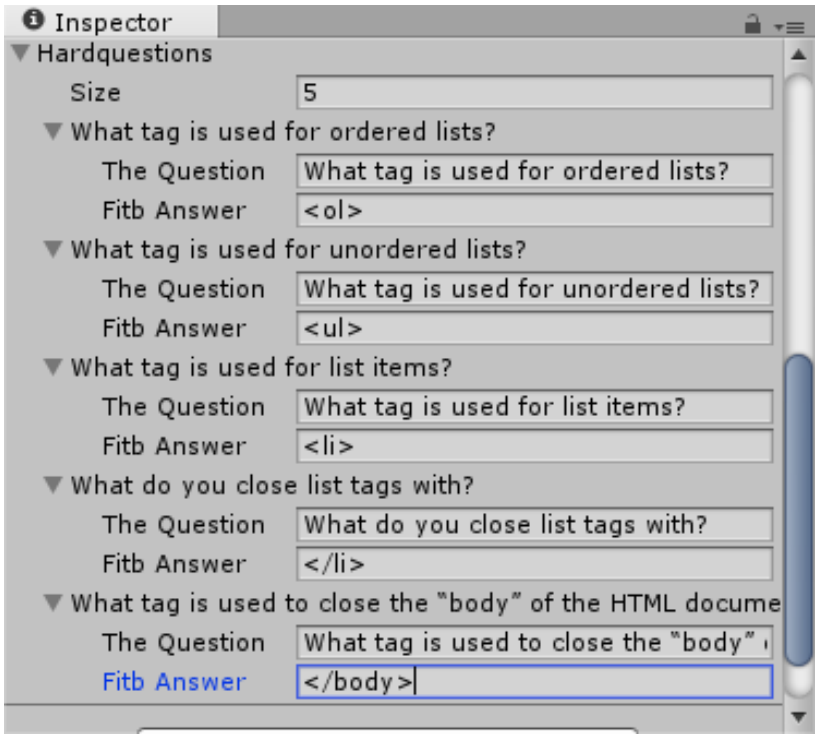
And back in the unity editor, because I made the class serializable, I am now able to edit the contents of the arrays within the inspector after I added the displayFITB script to the FillInTheBlank gameobject as a component.



I then filled the arrays with the challenges I planned to from my design section.







Then I continued to edit the script to serialize certain elements such as the question text, the submit button and the input field that the user would enter their answer into. To do so, I had to add a line in the “using” namespace section that would allow me to link my script to the elements in my game. Using the line “using UnityEngine.UI”.

Allows the developer to configure certain UI elements via scripting

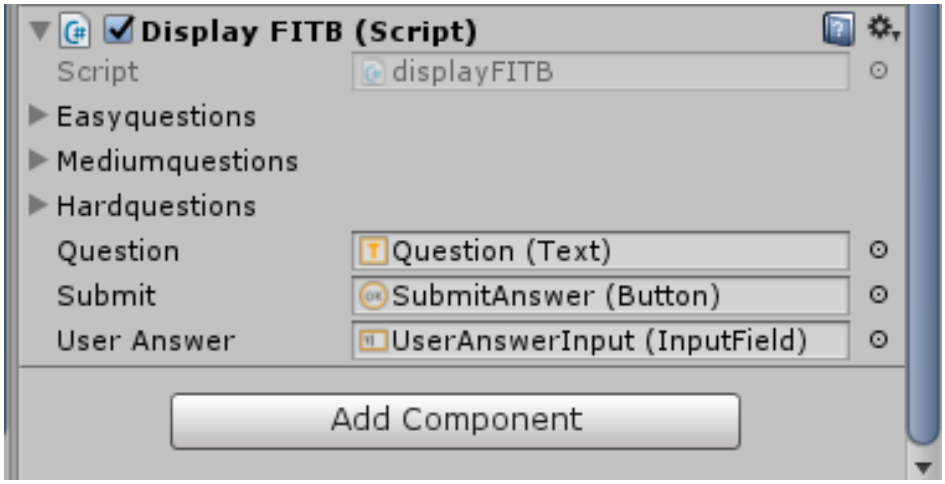
```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.UI;
```

Now I can serialize elements to edit them in my script.

```
6  public class displayFITB : MonoBehaviour {
7
8      public FITBQuestions[] easyquestions;
9      public FITBQuestions[] mediumquestions;
10     public FITBQuestions[] hardquestions;
11     //holds list of questions and answers for FITB questions
12
13     [SerializeField]
14     private Text question;
15     //links the text from the core game to the script
16     [SerializeField]
17     private Button submit;
18     //links the submit button from the core game to the script
19     [SerializeField]
20     private InputField userAnswer;
21     //links the answer input field from core game to script
22
23
```

Lines 13 to 20 allow me to link the UI elements in my game to my script so I can change their values within my code and add functionality to my buttons.

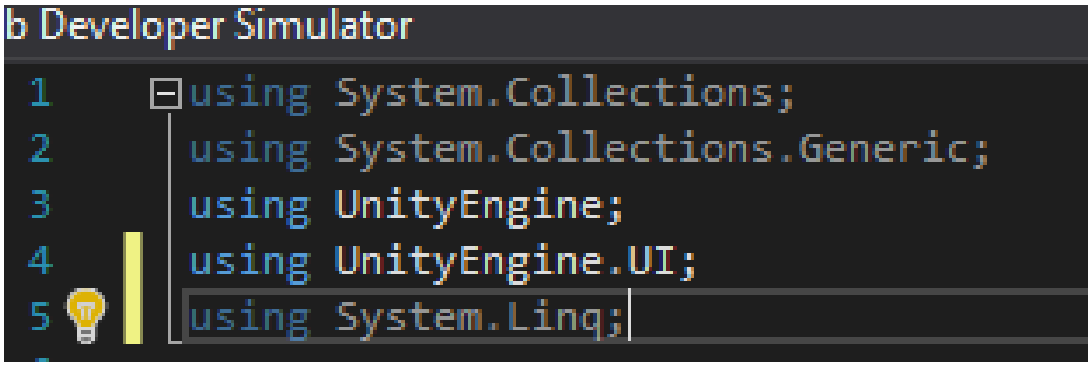
I then dragged and dropped them into their respective boxes in the inspector.



I then changed the name of the Question UI element to FITBquestion as I realised I would soon be adding the MCquestion text.

I then created a new empty gameobject called gameMechanics that will control the money counter and the number of lives for example and then added a new script to it called gamemechanics.

To display a single question randomly, I would need to create a new list that would have all the questions that were yet to be answered so I created a private static list called unansweredQuestions and a private FITBQuestion object called displayedQuestion so my code can randomly choose a question from the unansweredQuestions list and make the value of the displayedQuestion, the randomly chosen question from the unansweredQuestions list. For me to do this, I would have to add Using system.linq which would allow me to convert the array of questions into a list for the unansweredQuestions.



```
10      public FITBQuestions[] mediumquestions;
11      public FITBQuestions[] hardquestions;
12      //holds list of questions and answers for FITB questions
13
14      [SerializeField]
15      private Text question;
16      //links the text from the core game to the script
17      [SerializeField]
18      private Button submit;
19      //links the submit button from the core game to the script
20      [SerializeField]
21      private InputField userAnswer;
22      //links the answer input field from core game to script
23
24      private static List<FITBQuestions> unansweredQuestions;
25      // creates a list of FITB objects
26
27      private FITBQuestions displayedQuestion;
28      //Will be the displayed question in-game
29
30      // Use this for initialization
31      void Start () {
32
33      }
34
35      // Update is called once per frame
36      void Update () {
37
38      }
39  }
40
```

Line 24 declares a list of the FITBQuestions objects that are private so they can't be accessed accidentally as I will have unansweredQuestions for Multiplechoice questions. It is static so it can persist between scenes because I plan to reload the scene every time the question is answered and therefore, I must keep the list of unansweredquestions persistent to ensure the data isn't lost every time the scene reloads.

Next, I began to edit the start function so that whenever the scene reloads, depending on the selected difficulty, the array of questions from the respective difficulty is added into the list of unansweredQuestions but only if it is empty.

```
30      // Use this for initialization
31      void Start () {
32
33          if (unansweredQuestions == null )
34          {
35
36              if ((gamemanager.difficulty) == ("easy"))
37              {
38                  unansweredQuestions = easyquestions.ToList<FITBQuestions>();
39              }
40
41              if ((gamemanager.difficulty) == ("medium"))
42              {
43                  unansweredQuestions = mediumquestions.ToList<FITBQuestions>();
44              }
45              if ((gamemanager.difficulty) == ("hard"))
46              {
47                  unansweredQuest
48              }
49
50
```

(field) string gamemanager.difficulty

An object reference is required for the non-static field, method, or property 'gamemanager.difficulty'

This would create a compile error as it has been underlined in red because I am trying to reference an object without any being instantiated so I edited the gamemanager script to make the variable difficulty static, so I wouldn't need to instantiate an object to reference it.

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

public class gamemanager : MonoBehaviour {

    public static string difficulty;
    // This is the declaration of a variable called difficulty that will hold a string
    // The purpose of the string is to hold the difficulty my player selects when they click on the respective button
    // I have made it public so I can access the variable outside the script if I need to
    //static so it can be referenced without the need to instantiate
```

```
    //links the answer input field from core game to script

    private static List<FITBQuestions> unansweredQuestions;
    // creates a list of FITB objects

    private FITBQuestions displayedQuestion;
    //Will be the displayed question in-game

    // Use this for initialization
    void Start () {

        if (unansweredQuestions == null )
        {

            if ((gamemanager.difficulty) == ("easy"))
            {
                unansweredQuestions = easyquestions.ToList<FITBQuestions>();
            }

            if ((gamemanager.difficulty) == ("medium"))
            {
                unansweredQuestions = mediumquestions.ToList<FITBQuestions>();
            }
            if ((gamemanager.difficulty) == ("hard"))
            {
                unansweredQuestions = hardquestions.ToList<FITBQuestions>();
            }
```

This rid the script of the red lines so there should be no compiling error.

I then added another condition to the if statement stating that it would also load the questions back into unanswered questions if all the questions had been answered. This change is justified as I believe that by completing the same question again and again, it will help my stakeholder get better at the task as repetition is a valid method of teaching such as revising. In addition, this will also make it so that the game can continue even after all the questions have been answered. This relates back to my analysis of the initial problem of teaching my stakeholders as both Mimo and Solo learn allowed their user to go back to previous tasks to revise the topic to help them improve in their skill so I added this condition.

```
25      // creates a list of FITB objects
26
27      private FITBQuestions displayedQuestion;
28      //Will be the displayed question in-game
29
30      // Use this for initialization
31      void Start () {
32
33          if (unansweredQuestions == null || unansweredQuestions.Count == 0)
34          {
35
36              if ((gamemanager.difficulty) == ("easy"))
37              {
38                  unansweredQuestions = easyquestions.ToList<FITBQuestions>();
39              }
40
41              if ((gamemanager.difficulty) == ("medium"))
42              {
43                  unansweredQuestions = mediumquestions.ToList<FITBQuestions>();
44              }
45              if ((gamemanager.difficulty) == ("hard"))
46              {
47                  unansweredQuestions = hardquestions.ToList<FITBQuestions>();
48              }
49
50
51          }
52      }
```

Line 33 is a condition that will only allow the lists to be loaded if the unansweredQuestions list is empty or if all the questions have been answered.

Lines 33 to 49 take the value of the difficulty from the game manager script that is chosen by the player in the main menu and then decides, based on the choice, which difficulty of questions to input into unansweredquestions. In the if statements, the arrays of FITBQuestions are turned to lists and then update the elements inside the unansweredQuestions list.

I would then need to create a new method apart of the script that would randomly choose an element from the unansweredQuestions list and make that element into the value for displayedQuestion so it could be displayed. After loading this element into displayedQuestion, I would have to remove this question so it would not reappear to the user till all FITB questions were answered.

```
54      void SetDisplayedQuestion()
55      {
56          int randomQuestionIndex = Random.Range(0, unansweredQuestions.Count);
57          // generates a random question between range of 0 to the number of elements in the list
58          displayedQuestion = unansweredQuestions[randomQuestionIndex];
59
60
61          question.text = displayedQuestion.theQuestion;
62
63          unansweredQuestions.RemoveAt(randomQuestionIndex);
64          //removes the question from the list
65      }
66
```

Line 54 defines a procedure that does not return a value, hence being defined as void, called setDisplayedQuestion() and is not public as it will not interact with a button.

Line 56 declares a integer variable called randomQuestionIndex that randomly picks a number between 0 , the first element, and the length of unansweredQuestion, which will be different

whenever the procedure is called as questions are continuously added and removed to and from unansweredQuestions. This is done by using unansweredQuestion.Count that counts the number of variables.

Line 58 then sets the value of displayedQuestion to the element that lies in the unansweredQuestions list in the randomQuestionIndex element.

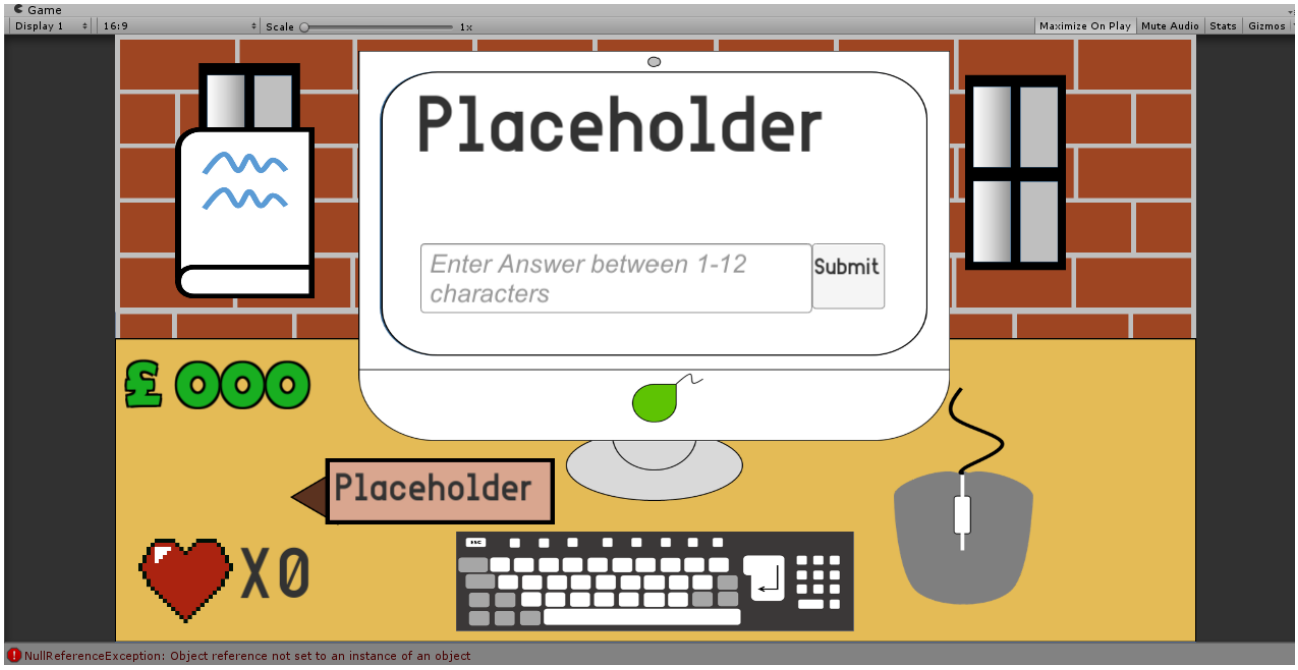
Line 61 then changes the value of the Question text in-game to the theQuestion value of the displayQuestions object using displayQuestions.theQuestion .

Lastly at line 63, the question that was displayed is now removed from unansweredQuestion so it will no longer appear until all the FITBquestions have been answered. .RemoveAt() allows me to delete from the unansweredQuestion list wherever I require by passing an index as a parameter.

Next, I needed to add a line of code that would set a new displayed question every time the scene reloaded by adding it to the start function.

```
25      // creates a list of FITB objects
26
27      private FITBQuestions displayedQuestion;
28      //Will be the displayed question in-game
29
30      // Use this for initialization
31      void Start () {
32
33          if (unansweredQuestions == null || unansweredQuestions.Count == 0)
34          {
35
36              if ((gamemanager.difficulty) == ("easy"))
37              {
38                  unansweredQuestions = easyquestions.ToList<FITBQuestions>();
39              }
40
41              if ((gamemanager.difficulty) == ("medium"))
42              {
43                  unansweredQuestions = mediumquestions.ToList<FITBQuestions>();
44              }
45              if ((gamemanager.difficulty) == ("hard"))
46              {
47                  unansweredQuestions = hardquestions.ToList<FITBQuestions>();
48              }
49
50
51          }
52          SetDisplayedQuestion();
53      }
54
```

I then tested this to see if this would work and the results were:



2.1

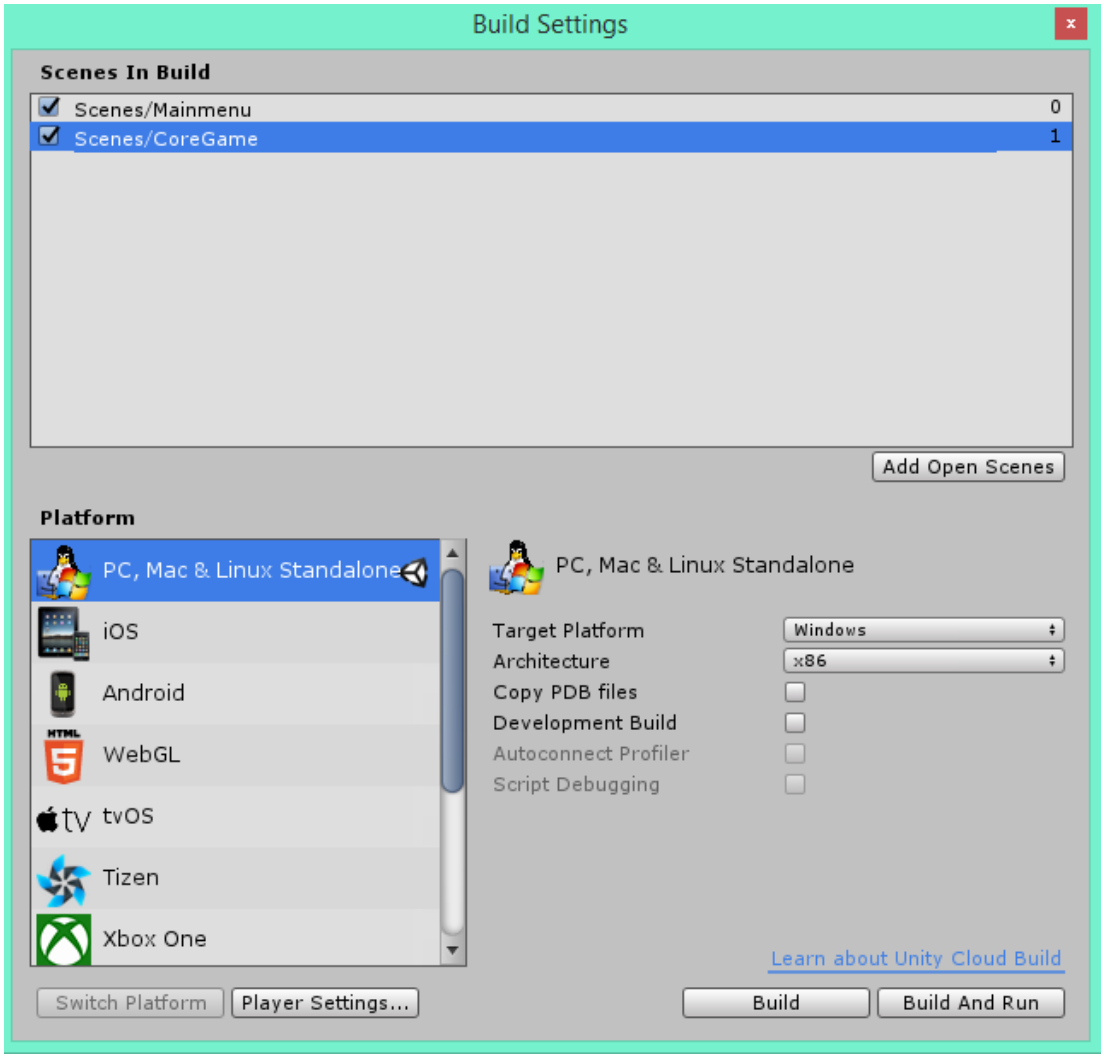
What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
Question loads randomly	Press play	Random question	2.1	Unexpected error.	

Upon testing, I realised that I was yet to choose a difficulty to load the questions in but I hadn't linked the 2 scenes together to change scenes when a difficulty was selected so I made the following modifications to the main menu and game manager script.

Firstly I added another namespace to the gamemanager script called unityEngine.scenemangement that would allow me to traverse through scenes.

```
1  using System.Collections;
2  using System.Collections.Generic;
3  using UnityEngine;
4  using UnityEngine.SceneManagement;
```

Next I went into the build settings of Unity and added both scenes to the build.

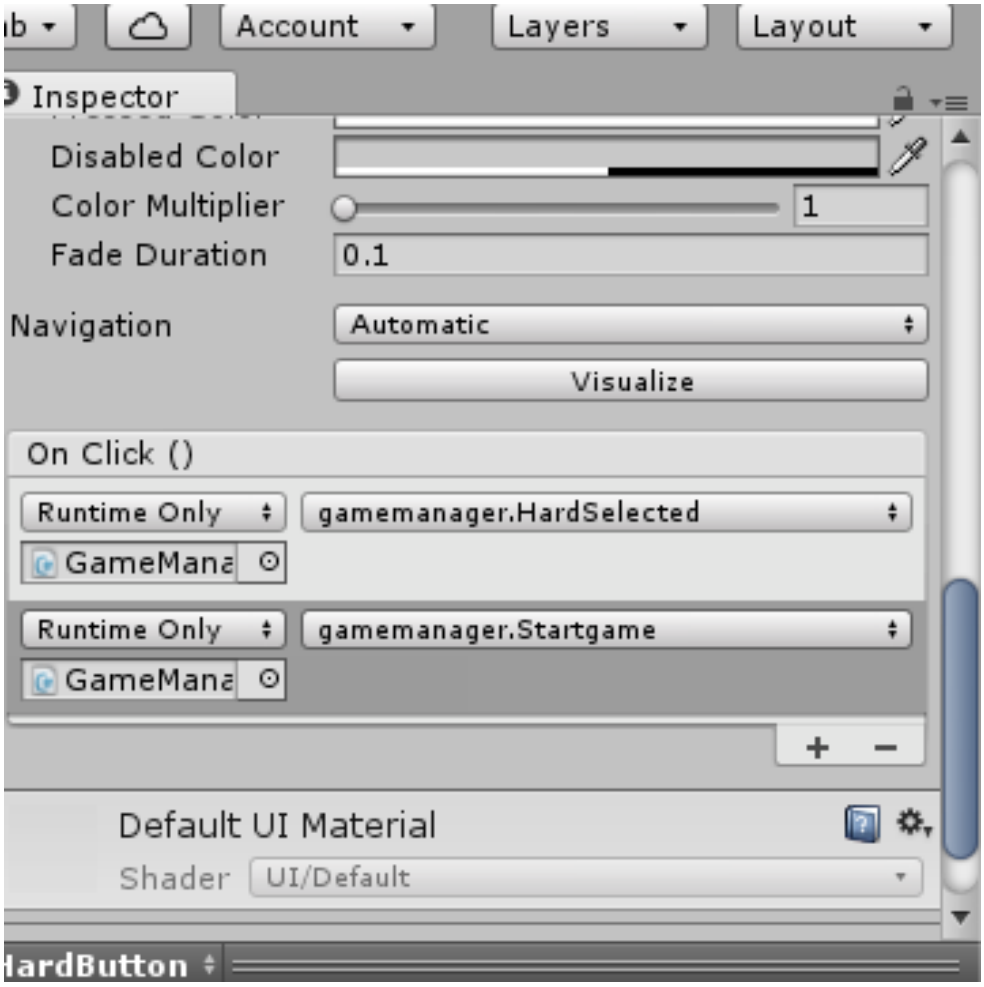


I then created a function that would run if any of the difficulty buttons are clicked and added them to the onClick() feature for those buttons.

```
36
37      public void Startgame()
38      //begins game
39      {
40          SceneManager.LoadScene("CoreGame");
41          //Loads main game
42      }
```

Line 37 declares the public procedure named StartGame appropriately as it starts the game.

Line 40 loads the CoreGame scene using the scenemanager namespace.

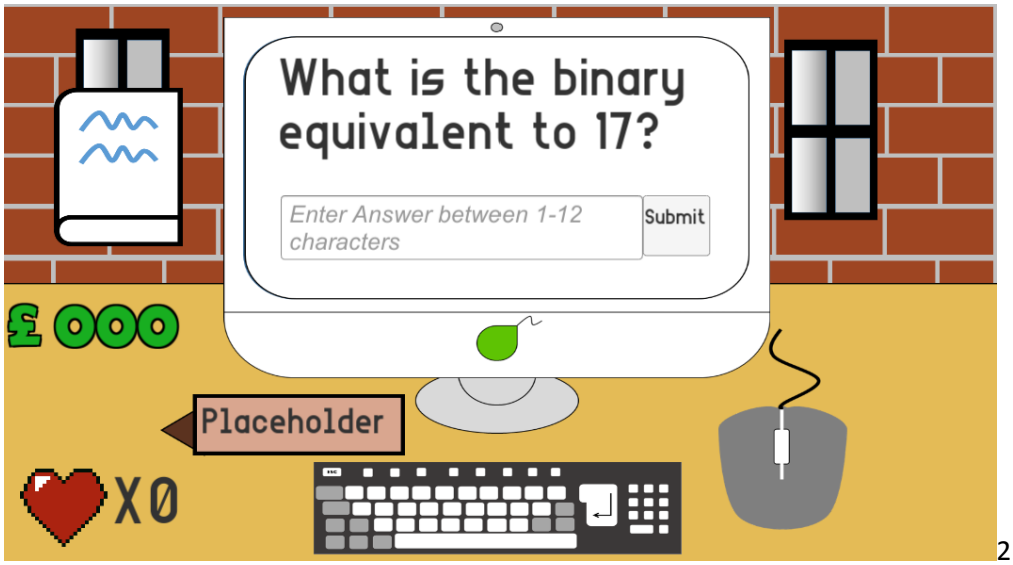


I then tested to see if these would load the game and then randomly display a question.

Before click

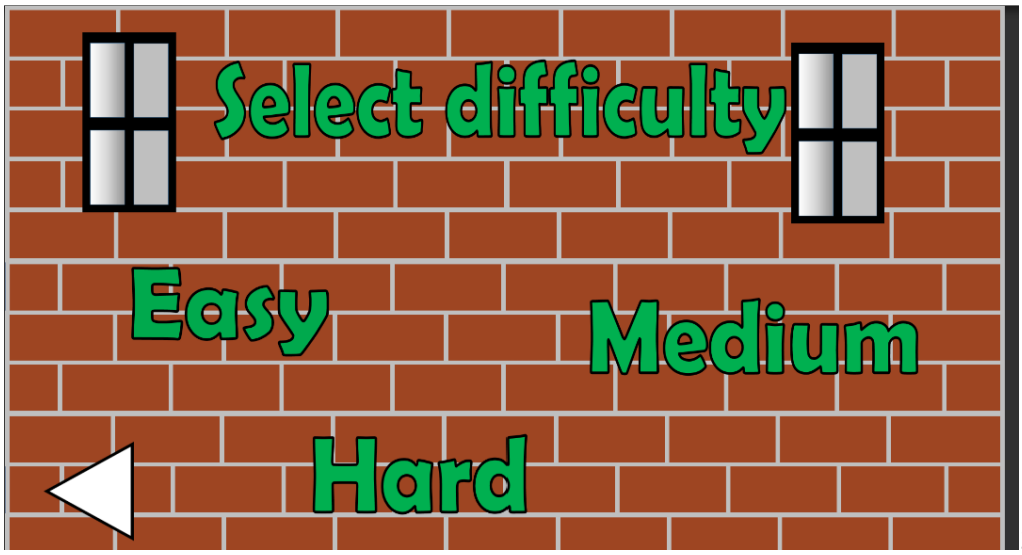
2.2.1

After click



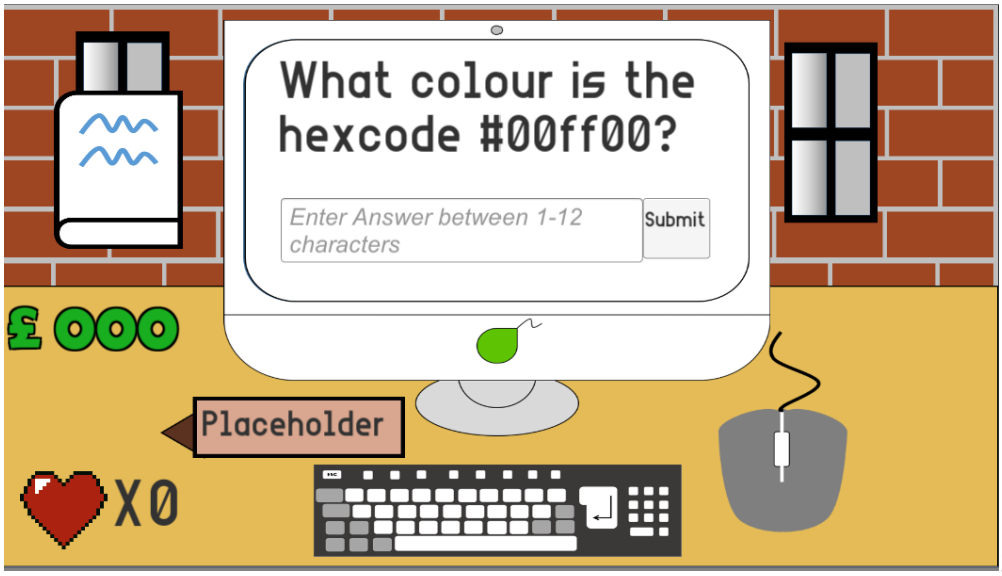
2.2.2

Before click



2.2.3

After click



2.2.4

What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
Loads game	Click easy	Loads coreGame scene	2.2.1 to 2.2.2 And 2.2.3 to 2.2.4	As expected	Added scene managing function to all difficulty buttons
Question loads randomly	Press easy	Random question	2.2.1 to 2.2.2 And 2.2.3 to 2.2.4	As expected	Added scene managing function to all difficulty buttons and then choosing a difficulty.

I then created a function that would run whenever the submit button was pressed. This would then compare the answer and change values of the number of lives and the money as well as the checking if the answer is less than 13 characters.

```
74      public void submitter()
75      {
76          if ((userAnswer.text).ToLower() == (displayedQuestion.fitbAnswer))
77          {
78              Debug.Log("Correct");
79
80          }
81          else
82          {
83              Debug.Log("Wrong");
84
85
86          }
87
88
89
90      }
91
```

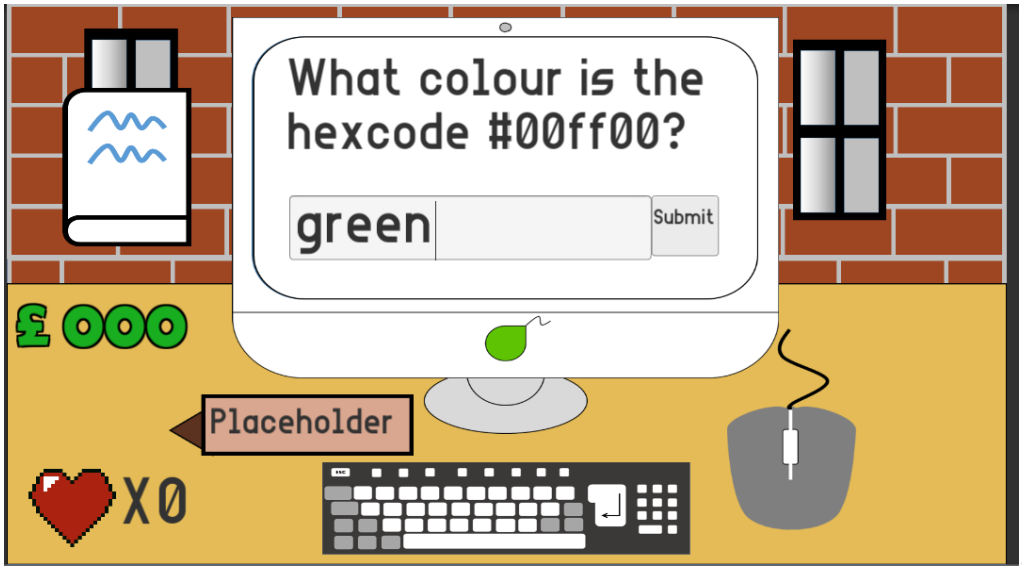
I named the procedure submitter as it runs whenever the submit button is clicked.

Line 76 changes whatever the user has written to a lowercase version and compares it to the answer of the questions using .ToLower()

I then altered the onClick() so that whenever the submit button was clicked, it would run the proceedure

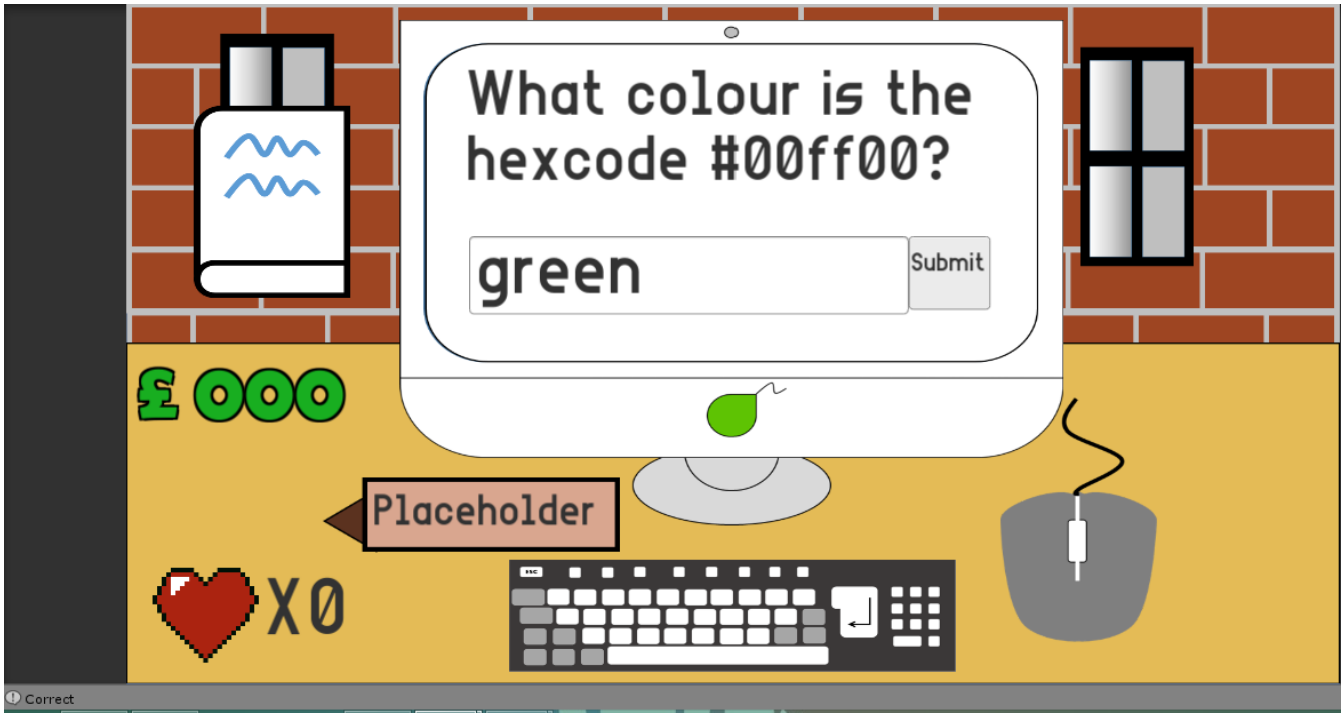
When I tested this:

Before click



2.3.1

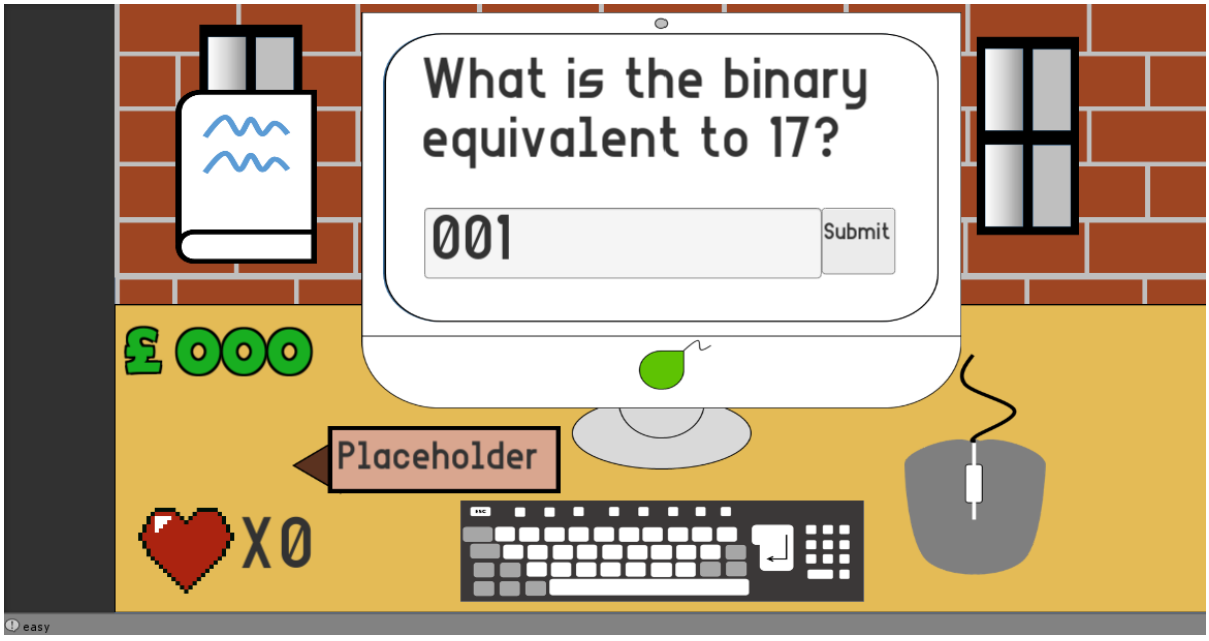
After click



2.3.2

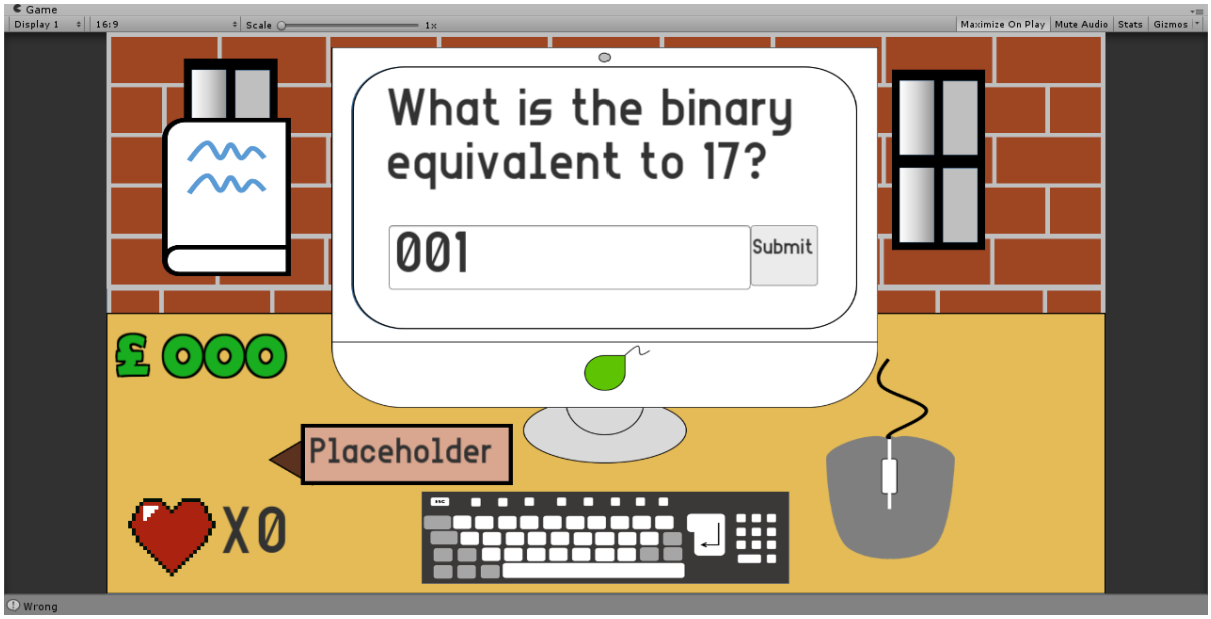
2.3.3

Before click



2.3.4

After click



2.3.5

2.3.6

This is also demonstration of my whitebox testing as I have traced the value of the answer.

What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
Ability to compare answer and determine answer as correct	Enter correct answer	Console replies correct	2.3.1, 2.3.2, 2.3.3	As expected Success	
Ability to compare answer and determine answer as incorrect	Enter incorrect answer	Console replies incorrect	2.3.4, 2.3.5, 2.3.6	As expected Success	

Successful tests.

Now I would need a function that reloads the scene but I would also need a delay for the player to acknowledge that they got the question correct so I would need a coroutine. Coroutines delay the execution of a program until yield instructions are complete so my yield instruction will be to wait for 1 second after answering before reloading the scene. My coroutine will be named transition, appropriately as when it is performed, it handles the transition of my game from scene to reloaded scene.

First I added a float variable that is serializable so in case I thought 1 second was too quick in the future, it is much easier to edit in the inspector for further maintenance purposes.

```
23
24      [SerializeField]
25      private float delay = 1f;
26      //Delay between reloading scenes
27
```

This is a private so the value cannot be changed accidentally. It is a float incase would like fractional time delays in the future. 1f is the value of the float.

I then added the Coroutine to my script.

```
95
96      IEnumerator Transition()
97      {
98          unansweredQuestions.Remove(displayedQuestion);
99          //removes the question from the list
100         yield return new WaitForSeconds(delay);
101
102
103         SceneManager.LoadScene("CoreGame");
104
105     }
106
107
```

The name 'SceneManager' does not exist in the current context

Show potential fixes (Alt+Enter or Ctrl+.)

```
96      IEnumerator Transition()
97      {
98          unansweredQuestions.Remove(displayedQuestion);
99          //removes the question from the list
100         yield return new WaitForSeconds(delay);
101
102
103         SceneManager.LoadScene("CoreGame");
104
105     }
106
107
108
```

The name 'SceneManager' does not exist in the current context

Show potential fixes (Alt+Enter or Ctrl+.)

I got this error in code as I have not yet included the using Unity.scenemanagment line in the namespace block.

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.UI;
5   using UnityEngine.SceneManagement;
6   using System.Linq;
```

```
97      IEnumerator Transition()
98      {
99          unansweredQuestions.Remove(displayedQuestion);
100         //removes the question from the list
101         yield return new WaitForSeconds(delay);
102
103         SceneManager.LoadScene("CoreGame");
104
105     }
106
```

I then removed the lines of code that removed the question from the unansweredquestion list from the setDisplayquestion function and moved it here so that the question is only removed once the question is answered rather than being removed beforehand.

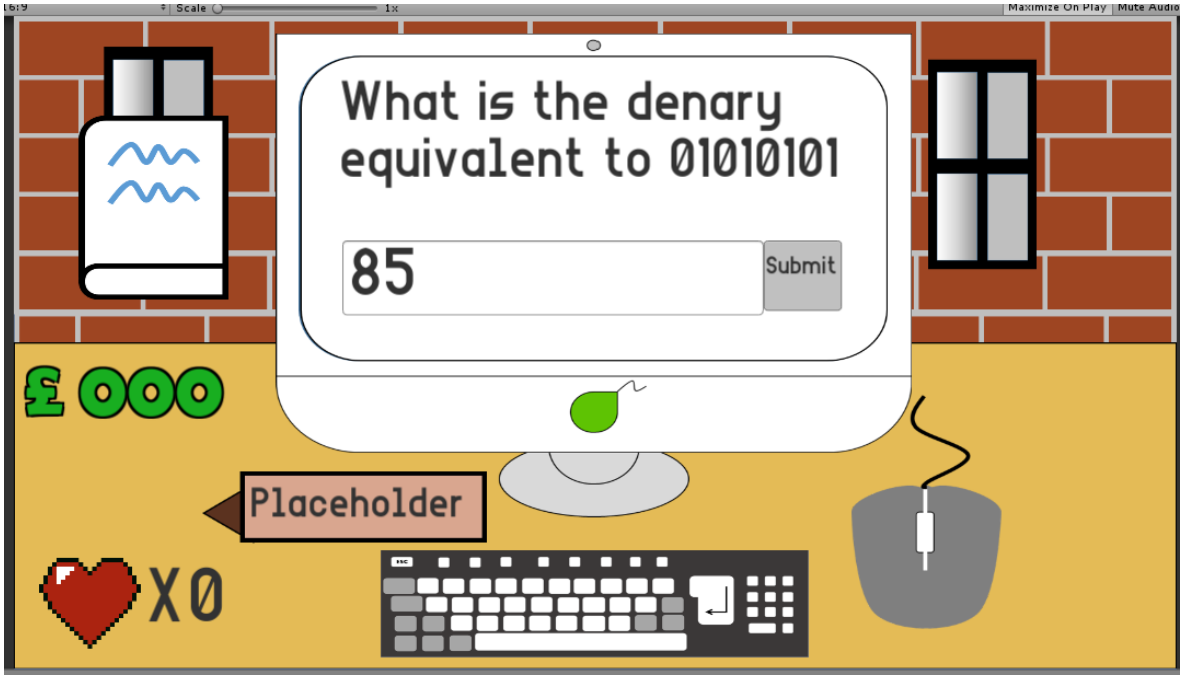
Line 101 is the yield instruction the coroutine must wait for where it returns the new instantiated object that is waitforseconds with delay as the parameter so it waits that length of time. It then reloads the scene with what should be a different question. Coroutines return an IEnumerator value so the return value I have set is an IEnumerator.

Then I called the transition() using StartCoroutine after comparing the answer so when submit is clicked, the coroutine begins.

```
91
92          StartCoroutine(Transition());
93
94      }
```

I then tested this:

Before click

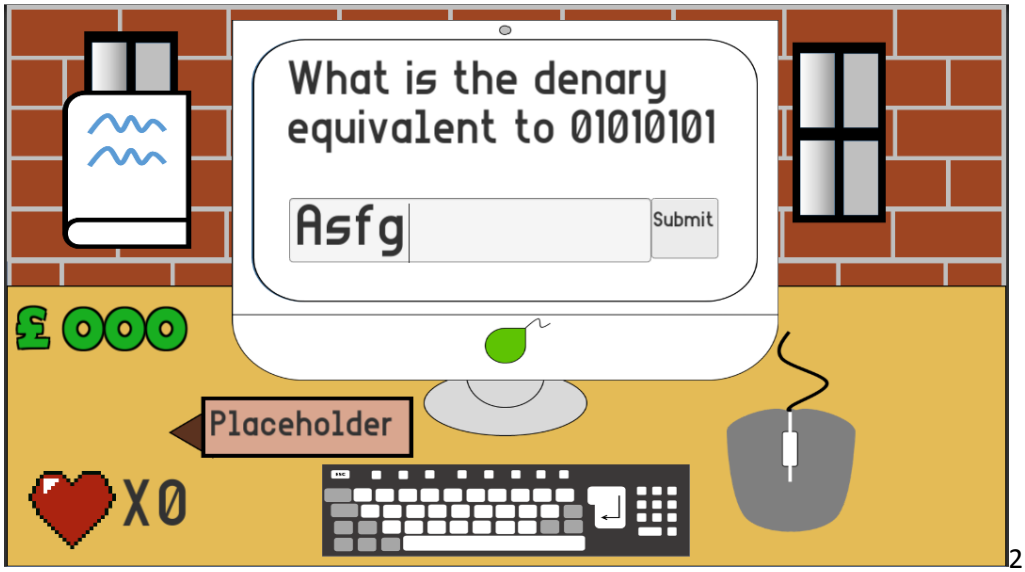


2.4.1

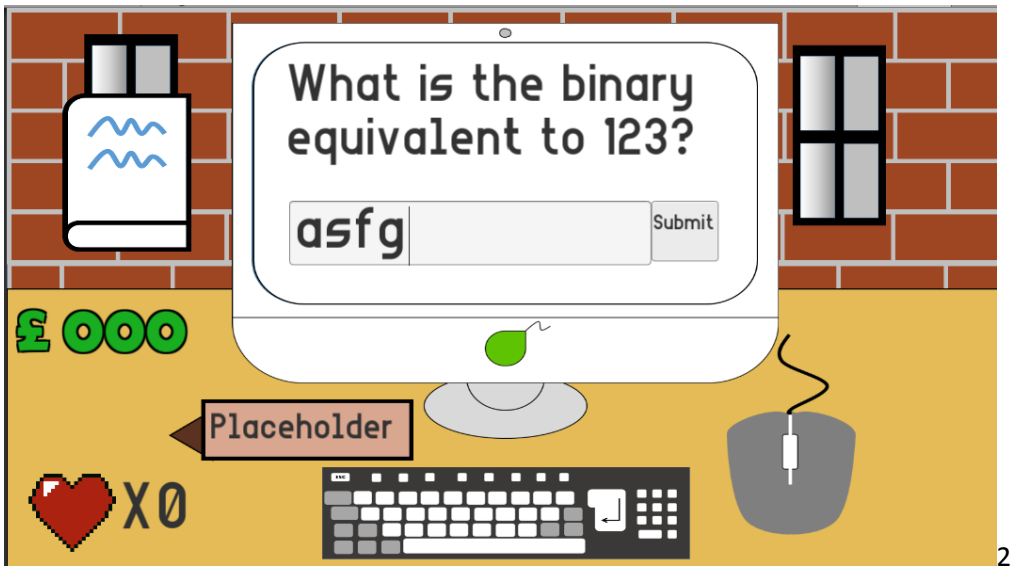
After click

2.4.2

I then tested if questions would reappear more than once before all questions were asked and the result was:


What is the denary equivalent to 01010101
Asfg
Submit
£ 000
Placeholder
X 0

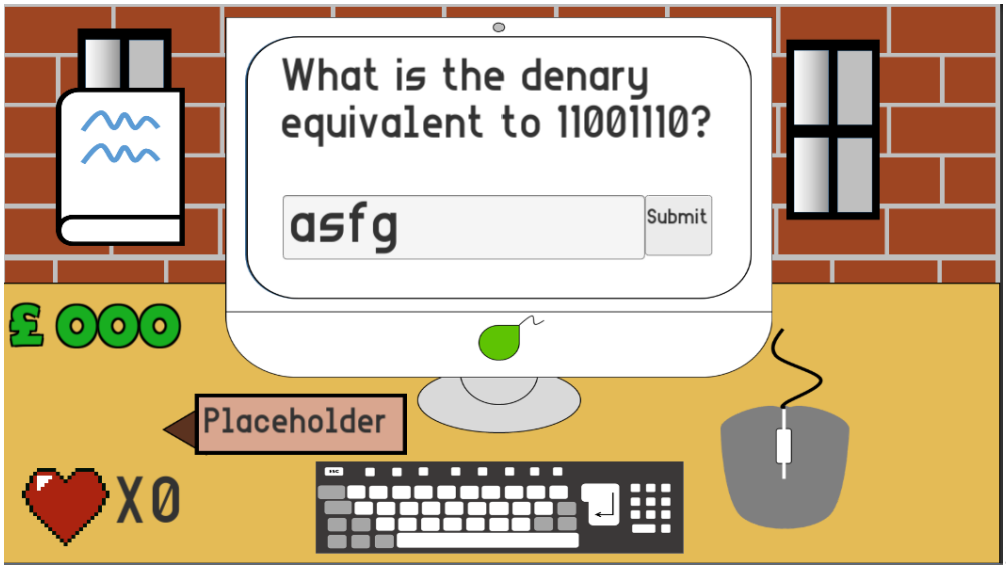
2.5.1


What is the binary equivalent to 123?
asfg
Submit
£ 000
Placeholder
X 0

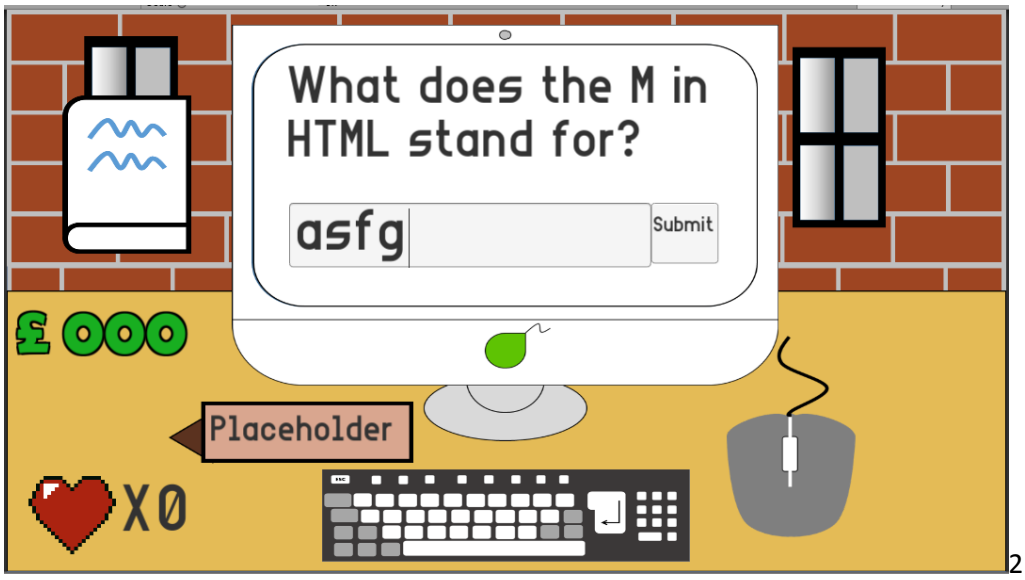
2.5.2


What colour is #000000?
asfg
Submit
£ 000
Placeholder
X 0

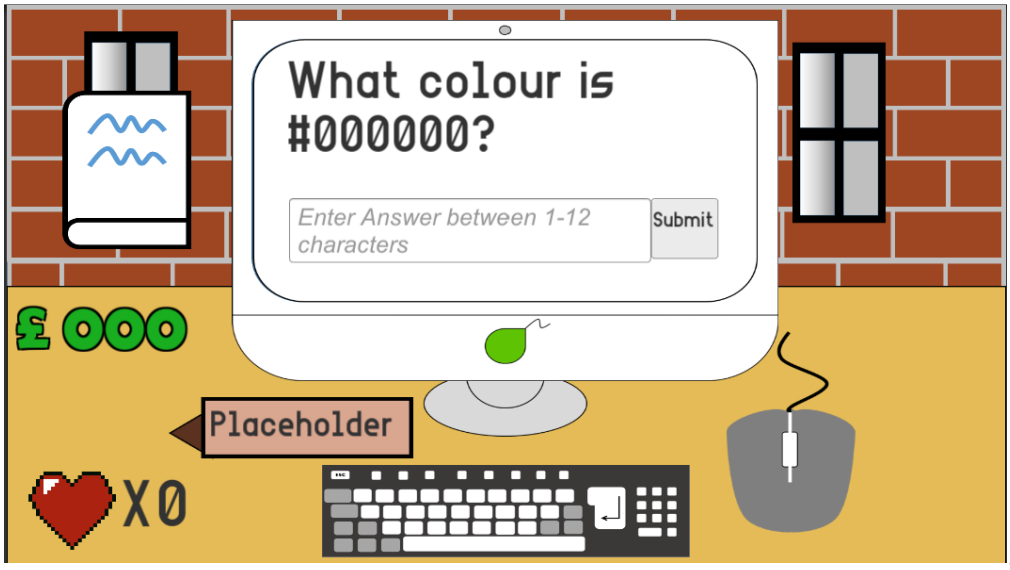
2.5.3


What is the denary equivalent to 11001110?
asfg
Submit
£ 000
Placeholder
X 0

2.5.4


What does the M in HTML stand for?
asfg
Submit
£ 000
Placeholder
X 0

2.5.5


What colour is #000000?
Enter Answer between 1-12 characters
Submit
£ 000
Placeholder
X 0

2.5.6

What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
Load another question	asdf	New question is displayed after 1 second	2.4.1, 2.4.2	As expected Success	
Goes through all questions once before repeating another question	Asfg Asfg Asfg Asfg Asfg Asfg (5 inputs to ensure all 5 are answered first)	No question is repeated before all questions have been asked	2.5.1, 2.5.2, 2.5.3, 2.5.4, 2.5.5, 2.5.6	As expected Success	

I then went back to annotate the coroutine so if someone else were to edit the code they would know what the coroutine is there for and would make it easier to find if further maintenance is done on the system.

Next I opened the gameMechanic script and created variables holding the value of the number of lives and money counter.

Web Developer Simulator

```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.UI;
5
6   public class gamemechanics : MonoBehaviour {
7
8       [SerializeField]
9       public static int lives;
10      //holds number of lives
11
12      [SerializeField]
13      public static int money;
14      //holds accumulated money
15
16
```

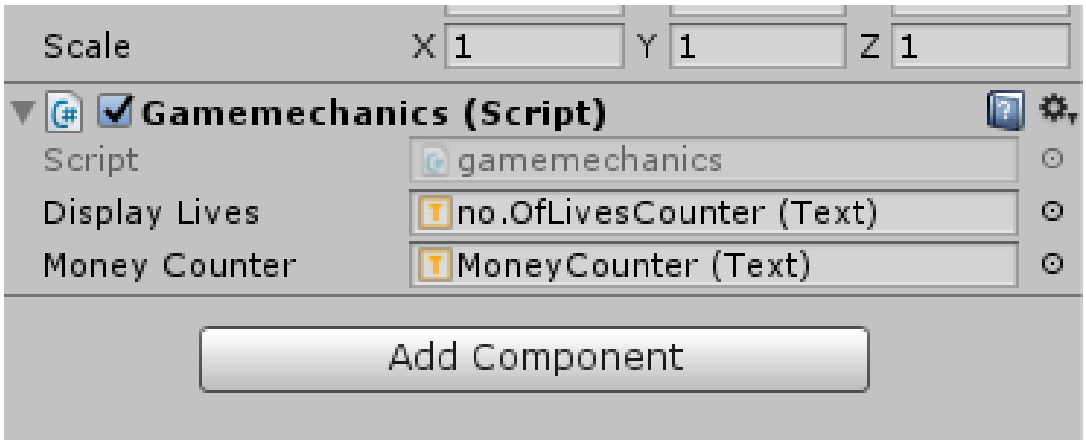
Line 4 includes UnityEngine.UI in the namespace as I will be using it to alter the money counter and number of lives displayed in game.

Lines 8 to 13 declare 2 variables that are serialized for the purpose of easier further maintenance and are static so they will not reset when the scene reloads and can be changed from different scripts as they will need to be.

Then I added 2 UI elements, both text so I can alter the values in game via the game mechanics script and then update them via update functions within the game mechanics script.

```
6   public class gamemechanics : MonoBehaviour {
7
8       [SerializeField]
9       public static int lives;
10      //holds number of lives
11
12      [SerializeField]
13      public static int money;
14      //holds accumulated money
15
16      [SerializeField]
17      private Text displayLives;
18      //the lives displayed
19
20      [SerializeField]
21      private Text moneyCounter;
22      //money displayed
23
24      // Use this for initialization
25      void Start () {
26
```

Both are serialized so that I can alter the elements via the scripts.



Then I changed the update procedure so that every frame, the value of number of lives and money counter are reviewed and updated.

```
29      // Update is called once per frame
30      void Update () {
31          moneyCounter.text = (money.ToString());
32          displayLives.text = (lives.ToString());
33      }
34  }
35
```

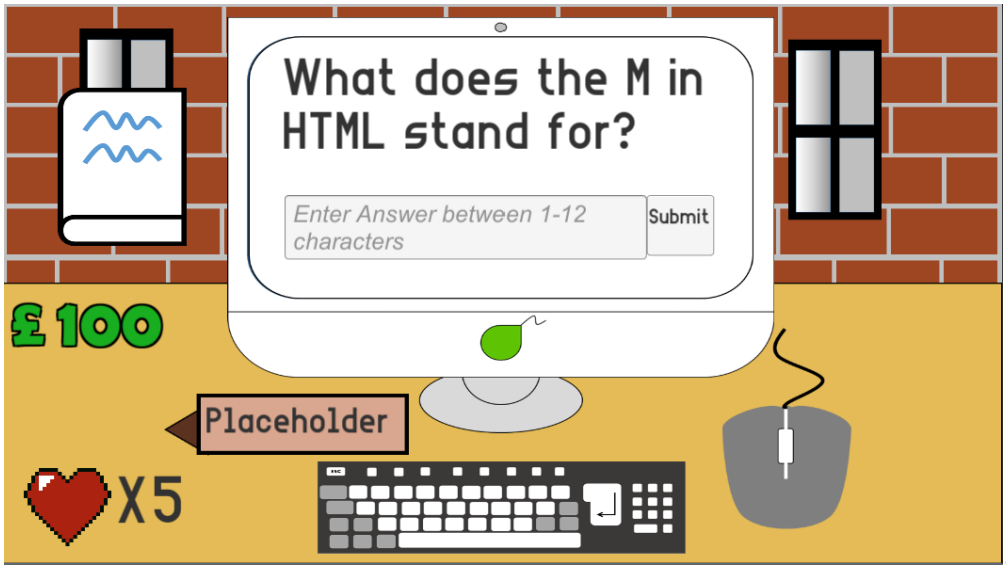
Line 31 to 32 changes the value of the text of the UI elements that were put in their place in the inspector to the value of money and Lives when they are converted as the UI elements are text so the values that come in place of them must be strings.

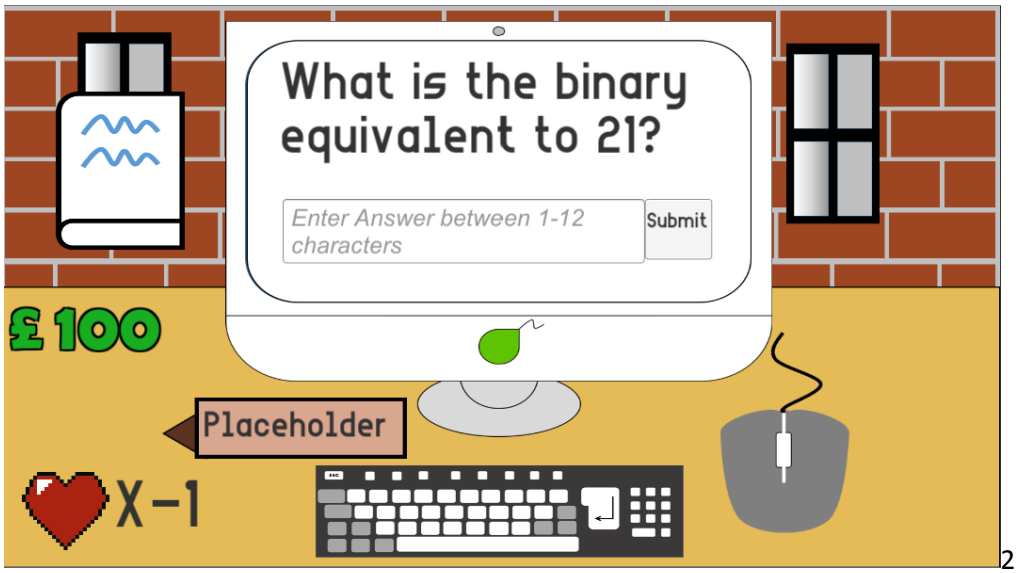
I then created if statements so that if a certain difficulty is chosen, then the value of displayLives is different as for easy mode, they have infinite amount of lives, Medium has 5 lives and hard has 3 lives.

```
27      // Use this for initialization
28      void Start () {
29          if ((gamemanager.difficulty) == ("easy"))
30          {
31              lives = -1;
32          }
33          if ((gamemanager.difficulty) == ("medium"))
34          {
35              lives = 5;
36          }
37          if ((gamemanager.difficulty) == ("hard"))
38          {
39              lives = 3;
40          }
41      }
42
43      // Update is called once per frame
44      void Update () {
45          moneyCounter.text = (money.ToString());
46
47
48          displayLives.text = (lives.ToString());
49      }
50  }
51
```

Line 29 to 41 take the difficulty chosen from the main menu and assigns a set number of lives to them depending on their difficulty that was specified in my design section and is a part of my success criteria. The number of lives for easy mode is so that by reducing it will not reach 0 lives as an attempt to give players on easy mode, unlimited tries because I will implement a script where game over is displayed when the number of lives is = to 0.

I then tested this code:

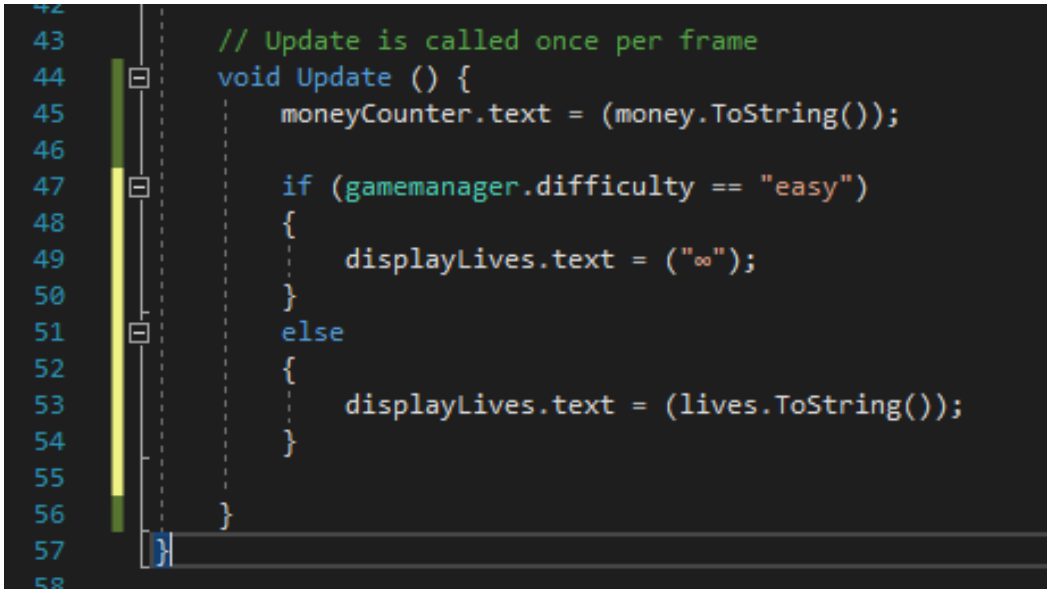

What does the M in HTML stand for?
Enter Answer between 1-12 characters
Submit
£100
Placeholder
X5
2.6.1


What is the binary equivalent to 21?
Enter Answer between 1-12 characters
Submit
£100
Placeholder
X-1
2.6.2


What do you close list tags with?
Enter Answer between 1-12 characters
Submit
£100
Placeholder
X3
2.6.3

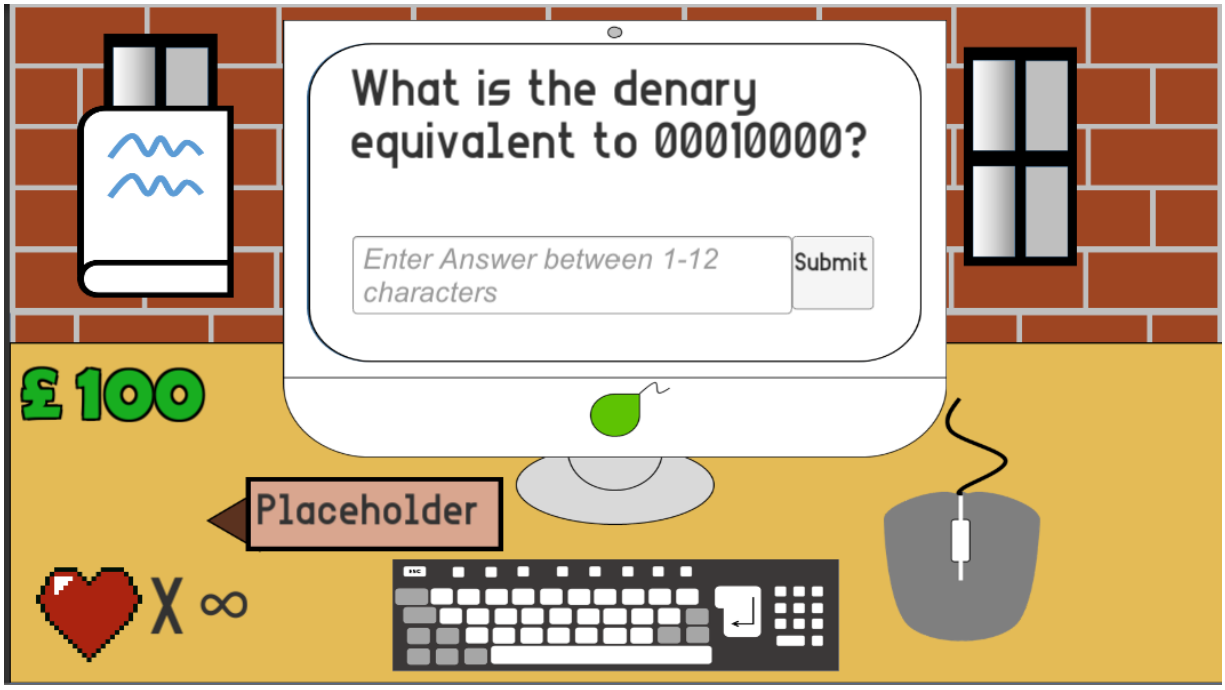
What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
Loads lives on easy mode	Choosing easy mode	Displays -1 lives	2.6.2	As expected Success	
Loads lives on medium mode	Choosing medium mode	Displays 5 lives	2.6.1	As expected Success	
Loads lives on hard mode	Choosing hard mode	Displays 3 lives	2.6.3	As expected Success	

Now this is expected but I then added a few if statements to make sure that the infinity sign is displayed when easy mode is chosen:

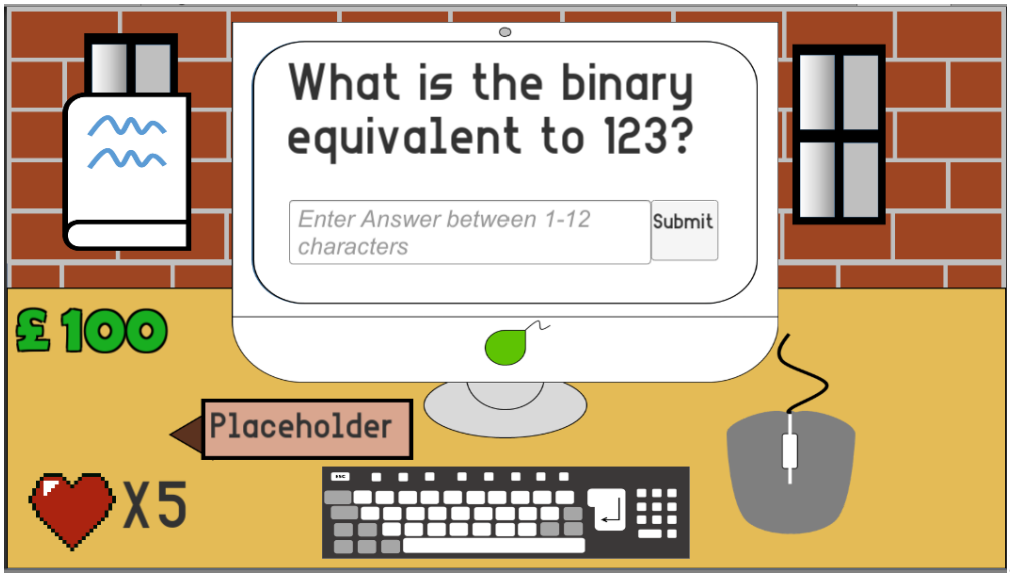


Lines 47 to 54 are a conditional statement where if the difficulty selected in the main menu is easy then the displayed lives will be infinity rather than -1, otherwise, it will display the users actual value for lives.

I then tested this and found:


What is the denary equivalent to 00010000?
Enter Answer between 1-12 characters
Submit
£100
Placeholder
X ∞

2.6.4


What is the binary equivalent to 123?
Enter Answer between 1-12 characters
Submit
£100
Placeholder
X5

2.6.5


What tag is used for ordered lists?
Enter Answer between 1-12 characters
Submit
£100
Placeholder
X3

2.6.6

What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
Displays number of lives as infinite on easy mode	Choosing easy mode	Displays infinity lives	2.6.4	As expected Success	If statement that means that if easy is chosen, infinity sign will display as lives
Displays number of lives as 5 on medium mode	Choosing medium mode	Displays 5 lives	2.6.5	As expected Success	
Displays number of lives as 3 on hard mode	Choosing hard mode	Displays 3 lives	2.6.6	As expected Success	

The tests were a success.

I then needed to update the scripts that change the value of the money counter and number of lives.

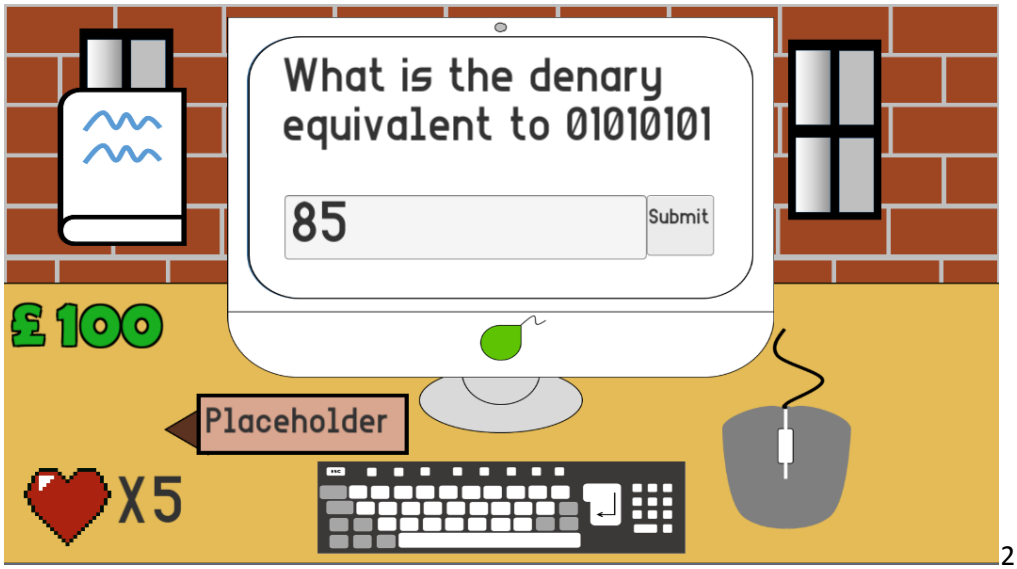
To do so, I would have to edit my submit function so that when comparing, if correct, the value of money increases by £100 as stated by my success criteria and decreases by £50 and lose a life if incorrect as stated by my success criteria.

```
78      public void submitter()
79      {
80          if ((userAnswer.text).ToLower() == (displayedQuestion.fitbAnswer))
81          {
82              Debug.Log("Correct");
83              gamemechanics.money += 100;
84
85          }
86          else
87          {
88              Debug.Log("Wrong");
89              gamemechanics.money -= 50;
90              gamemechanics.lives -= 1;
91
92          }
93
94          StartCoroutine(Transition());
95
96      }
97
```

Lines 83 and 89 to 90 increment the money value apart of the gamemechanics gameobject by £100 and decreases the value if incorrect as well the number of lives.

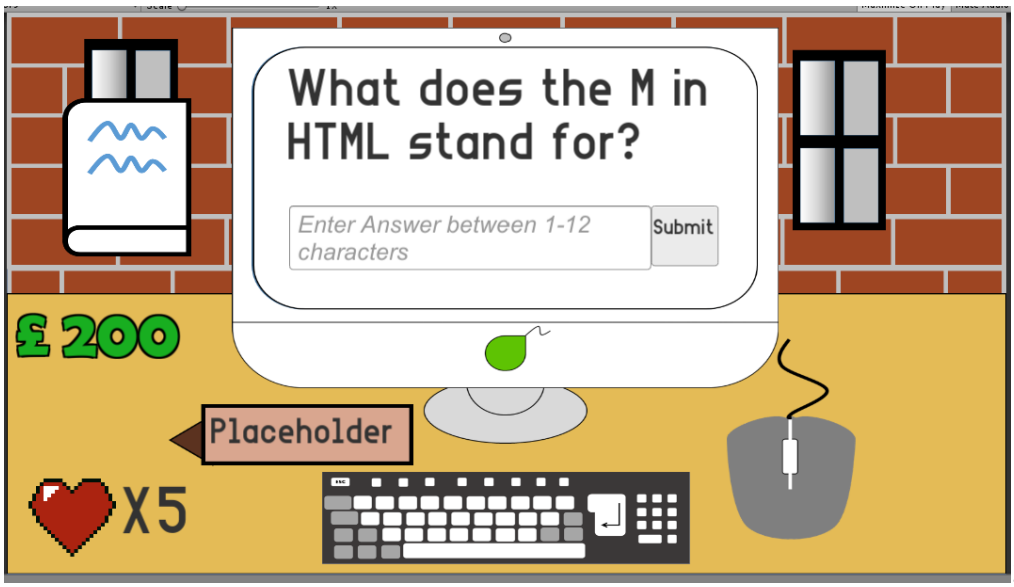
I then tested to see if the values would change:

Before click



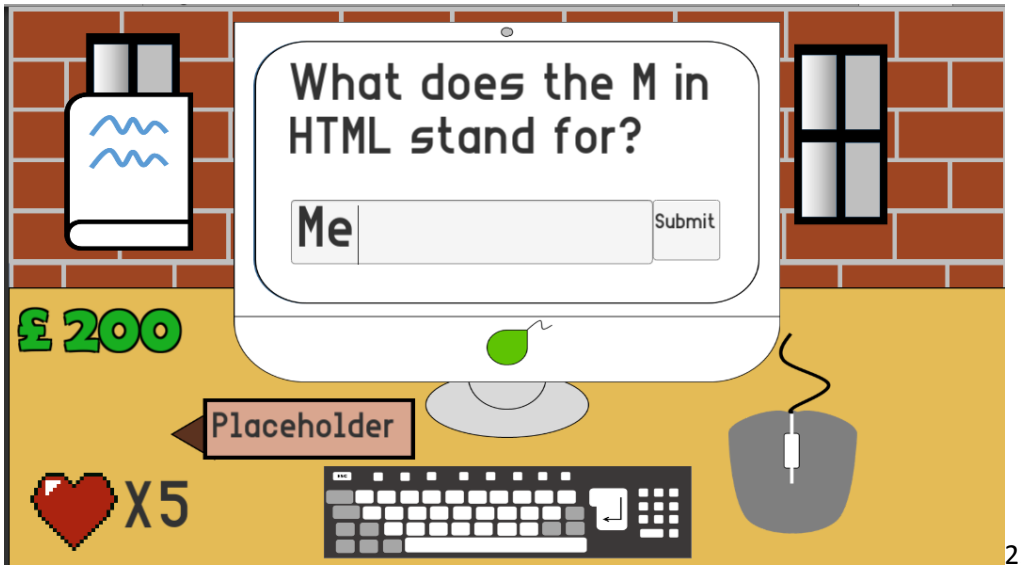
2.7.1

After click



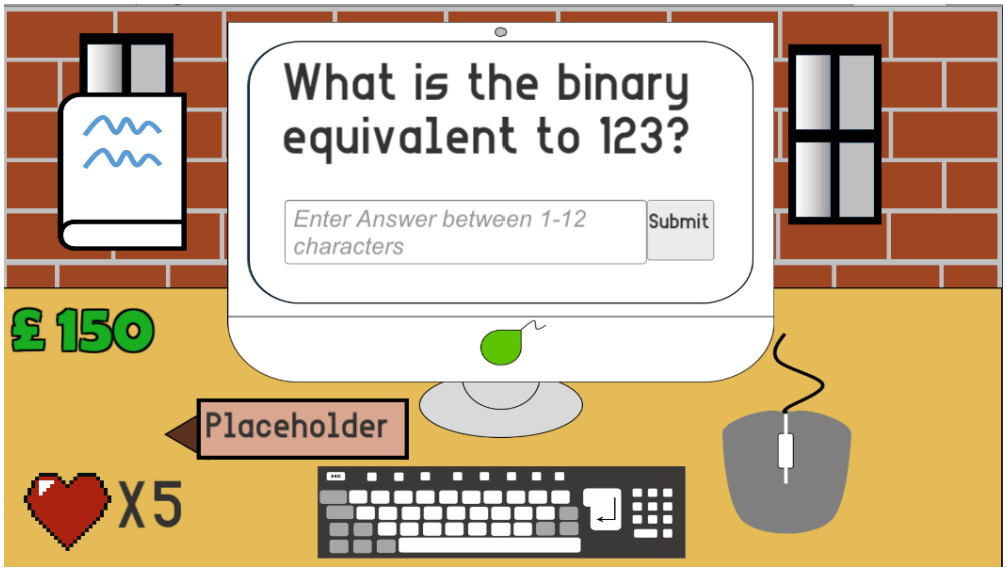
2.7.2

Before click



2.7.3

After click



2.7.4

What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
money increases by 100	85	Money increases by 100	2.7.1 to 2.7.2	As expected Success	
money decreases by 50 and lives decrease by 1	Me	Money decreases by 50 and lives decrease by 1	2.7.3 to 2.7.4	Money decreases and lives decreases but went back to original value	

As shown, the value of lives went down but after the co routine, the value went back up to the original value.

After reviewing the code, I found this issue was caused by setting the values of lives in the start function and every time the scene is reloaded, the values are reset back to their original.

To resolve this, I created a static counter that increments whenever the scene is loaded and will only reset the lives whenever the counter = 0.

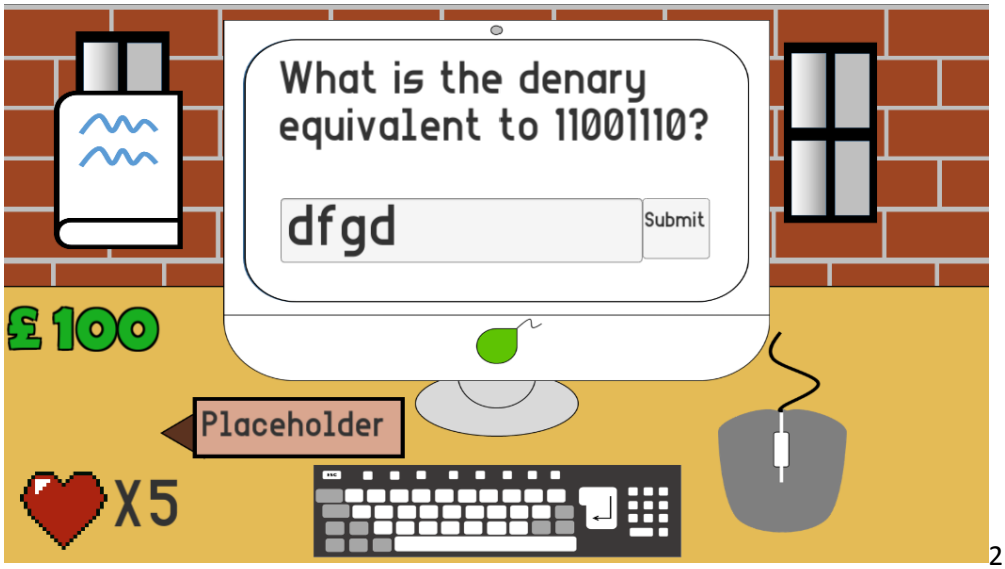
```
27      // Use this for initialization
28      void Start () {
29          if (counter == 0)
30          {
31
32
33              if ((gamemanager.difficulty) == ("easy"))
34              {
35                  lives = -1;
36              }
37              if ((gamemanager.difficulty) == ("medium"))
38              {
39                  lives = 5;
40              }
41              if ((gamemanager.difficulty) == ("hard"))
42              {
43                  lives = 3;
44              }
45          }
46          counter++;
47      }
```

Line 29 makes it so that the values will only reset if counter is equal to 0.

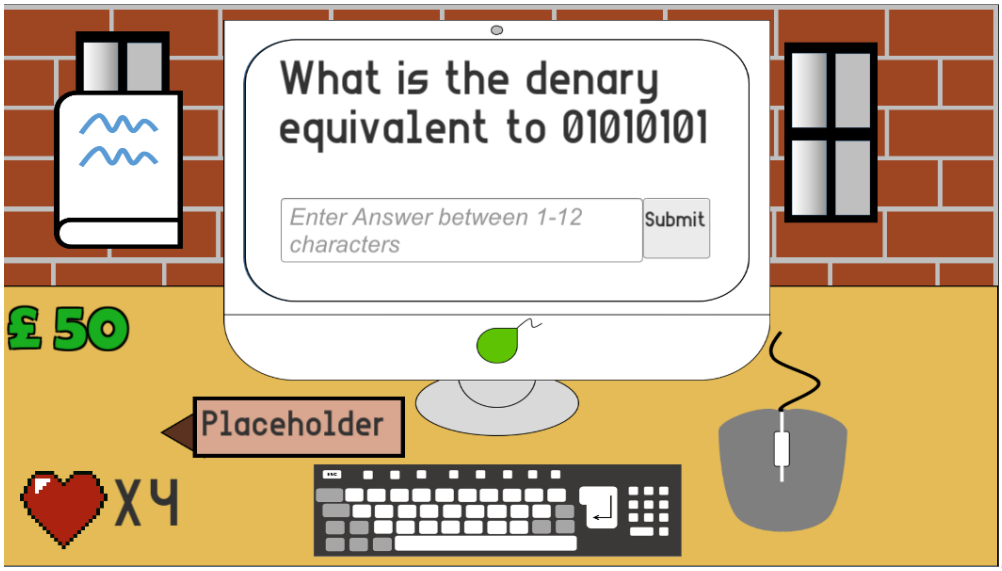
Line 46 increments the value of counter every time the scene is reloaded so that it only sets the lives values once.

When tested:

Before click



2.8



2.8.2

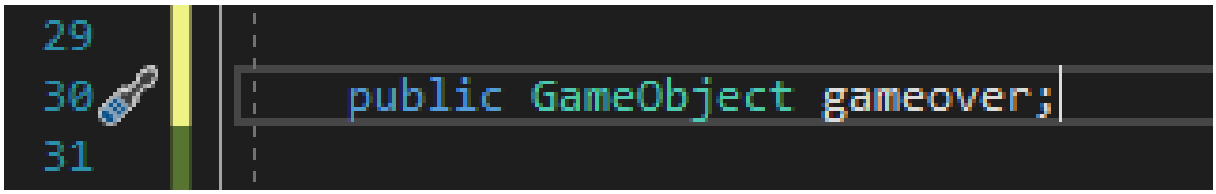
What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
money decreases by 50 and lives decrease by 1	dfgd	Money decreases by 50 and lives decrease by 1	2.8.1 to 2.8.2	Money decreases and lives decreases as expected	Adding a counter that increments whenever the scene is reloaded.

Now I will alter the update functions to have a statement that if lives reach 0, to then portray the game over screen I created in my screen designs. I placed the UI elements in the editor to my liking and made these edits to the gamemechanics script.

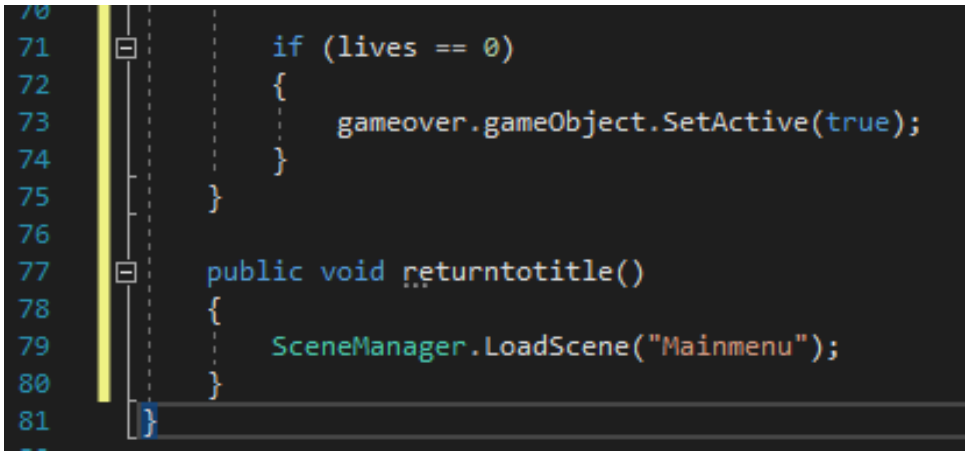
```
26      [SerializeField]
27      private Text gameoverscore;
28      //money displayed
29
```

```
// Update is called once per frame
void Update () {
    moneyCounter.text = (money.ToString());
    gameoverscore.text = (money.ToString());
```

These lines of code take the score the player has and stores them in another text UI element that will be displayed when the game over is reached.

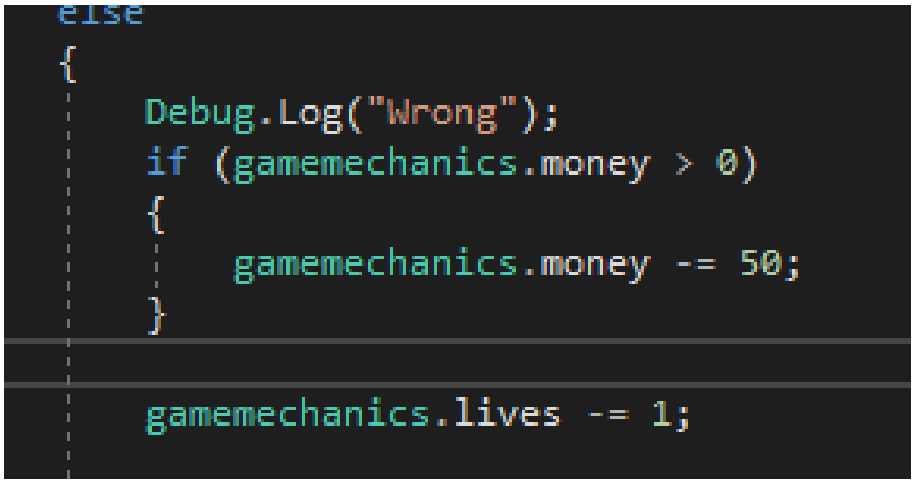


This will hold the game object of gameover that I have created and will become active when 0 lives is reached using .setactive(true) . First I added the namespace of scenemanagement so I could create a button that takes the user back to the title screen.



Line 71 is a part of the update function where if lives = 0, gameover screen will become active so I added functionality to the backtomainmenu button.

In addition, I added script that doesn't allow the money feature to go below £0. This is representative of real life as you cannot have negative money so this is a valid abstraction.



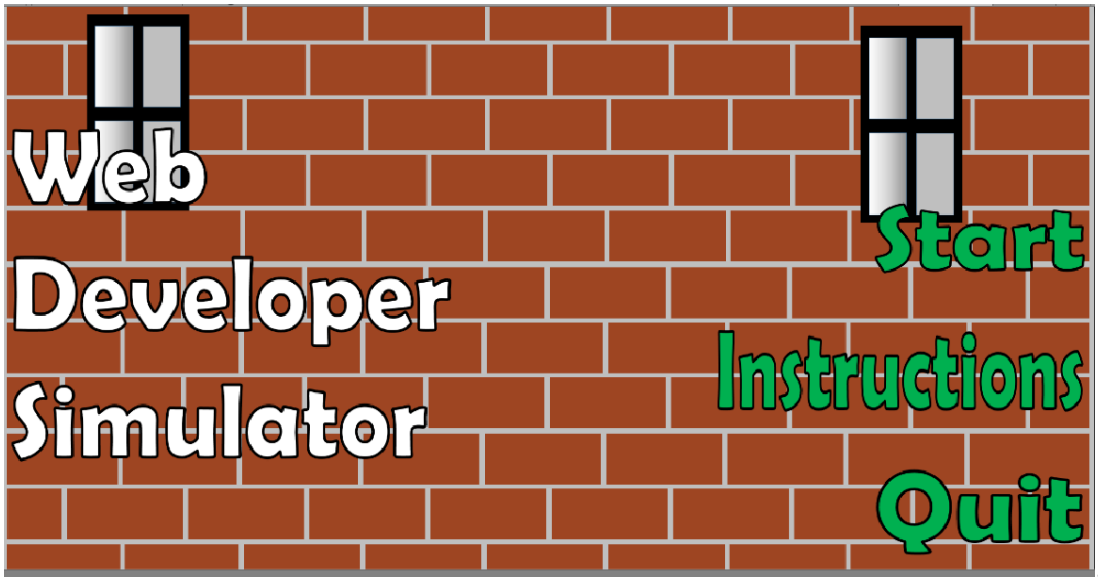
I tested it:

Before click

2.9.1

After click

2.9.2



2.9.3

What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
Gameover appears	Submit button	Gameover appears	2.9.1 to 2.9.2	Gameover appears	
Back to main menu screen	Click back to Mainmenu screen	Goes back to menu	2.9.3	As expected	

I then added validation to my input field by adding script that did not allow any answers above 12 characters by adding :

```
public GameObject failedvalidation;
//holds text prompting user to enter answer with less than 13 characters
```

```
84      {
85          if (userAnswer.text.Length < 13)
86          {
87
88
89              if ((userAnswer.text).ToLower() == (displayedQuestion.fitbAnswer))
90              {
91                  Debug.Log("Correct");
92                  gamemechanics.money += 100;
93
94              }
95              else
96              {
97                  Debug.Log("Wrong");
98                  if (gamemechanics.money > 0)
99                  {
100                     gamemechanics.money -= 50;
101                 }
102
103                 gamemechanics.lives -= 1;
104
105             }
106
107             StartCoroutine(Transition());
108
109
110         }
111         else
112         {
113             failedvalidation.gameObject.SetActive(true);
114         }
```

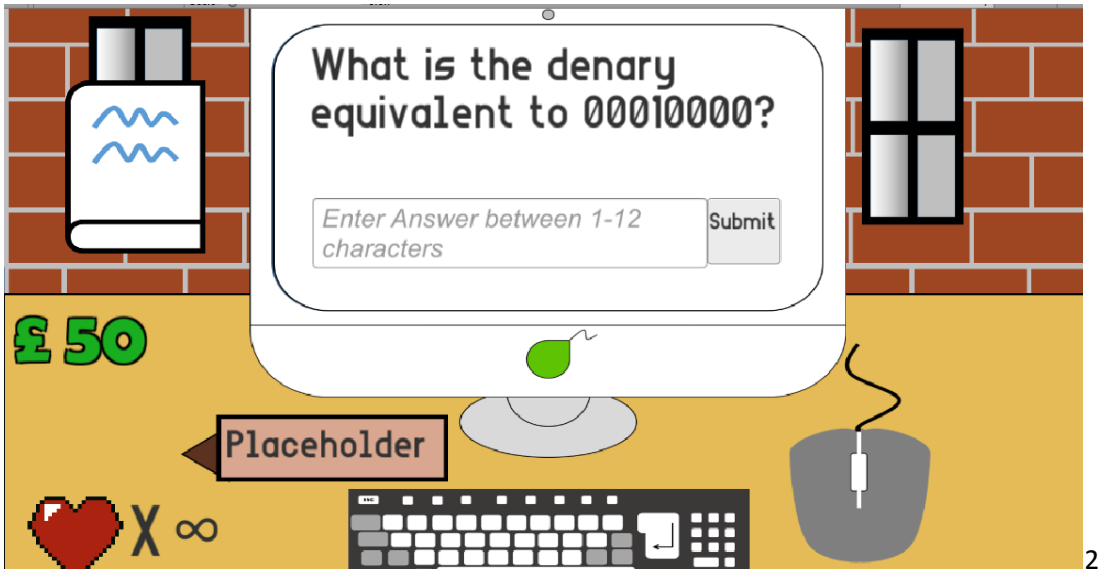
Line 85 prevents the user from entering answer more than 12 characters.

Line 111 to 114 displays failedvalidation object

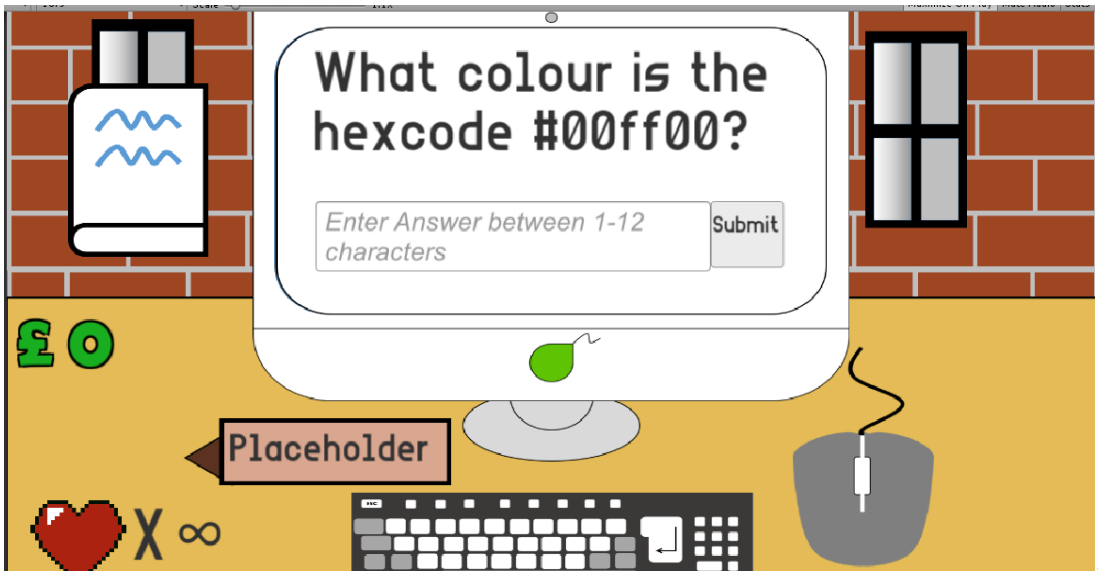
```
58
59              }
60              SetDisplayedQuestion();
61              failedvalidation.gameObject.SetActive(false);
62
63          }
64
```

Stop displaying the game object after every reload of scene.

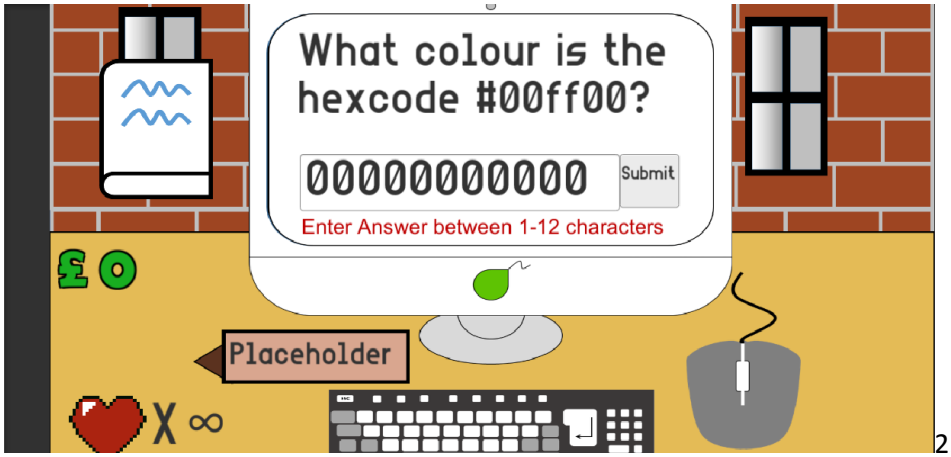
Test:



2.10.1

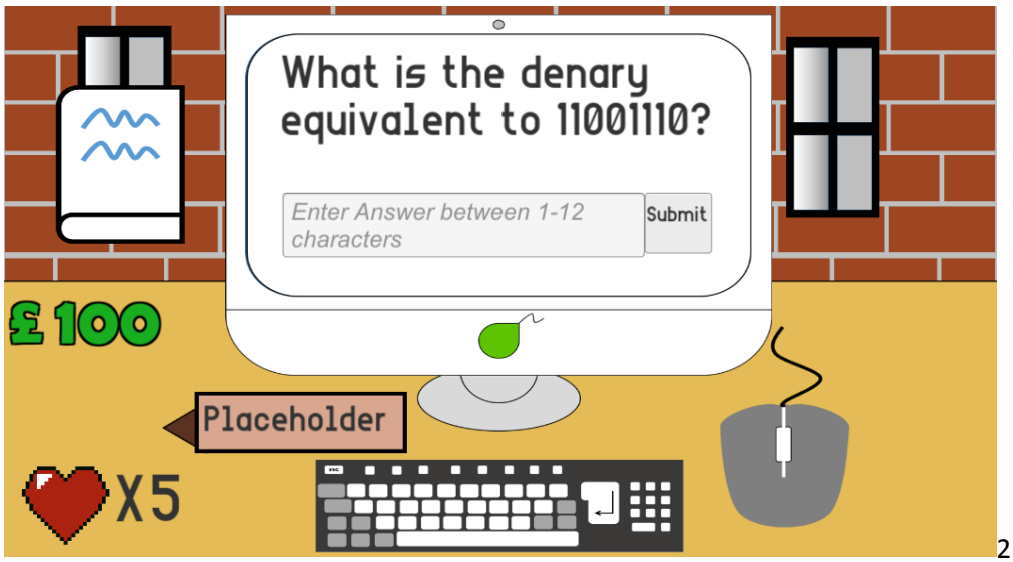


2.10.2



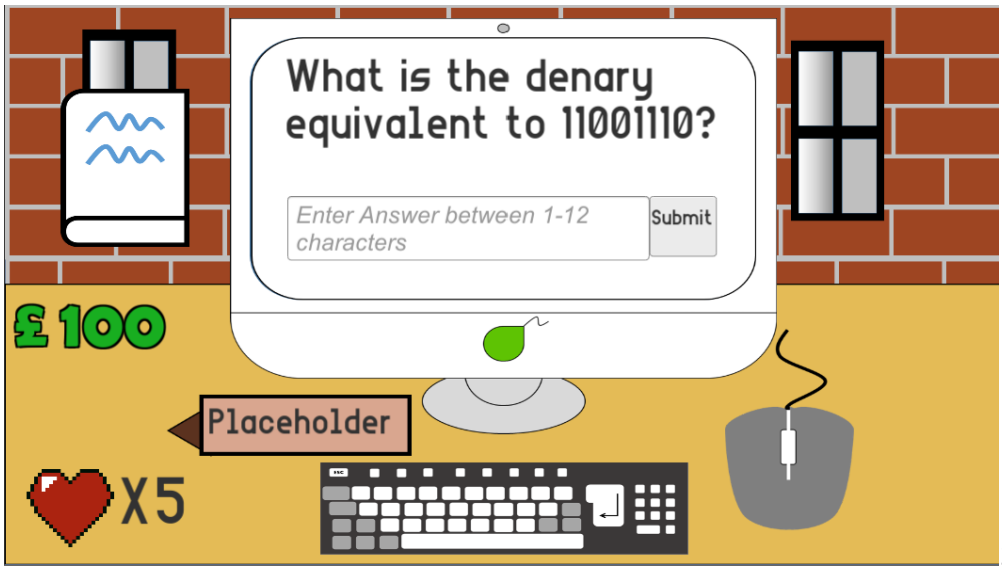
2.10.3

Before click



2.11.1

After click



2.11.2

What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
Validate answer	0000000	Next question	2.10.1	Next question	
	00000000000	Next question	2.10.2	Next question	
	Borderline				
	000000000000	Validation text appears	2.10.3	Validation text appears	
	invalid				
Invalid testing	RC submit (invalid)	Null	2.11.1 to 2.11.2	As expected	

And that is the end of the development of prototype 2.

Stakeholder feedback Fill in the blank: Week Starting 7/01/19

I have finished my second prototype, and showed it to my stakeholder, M.Miah, for her opinion on the prototype.

Success criteria	Justification
Must have a progression system/scheme	Progression is a very crucial feature in a game as it leads the player to believe they are working towards something. I will be using money as a form of progression as my game is an abstraction of a web developers life and in the real world, wealth can show progression of sorts. The observation process justifies this.
If a player gets a question correct, their “balance” should increment by £100	This is an abstraction of getting paid in the real world is like. If a real web developer does their job correctly, they get paid and so if the player completes a challenge then they too shall be paid in the form of in game currency. This will be the form of progression.

If a player gets a question wrong, their "Balance" must decrement by £50	This is an abstraction of consequences in my game of what may happen to a web developer if they create an error in code for the company they work for. This will be the form of progression.
Game must adapt challenges to the selected difficulties	From my interview process, it was clear that people from the variety of age range that my stakeholders are, that different difficulties are a requirement as children will all start from different stages. Some more gifted than others. Thus the game must have challenges of different difficulties to care to that.
Game must deplete lives of user if they get question wrong on harder difficulties	Should the player choose a more difficult level, my game will become harder with the risk of losing from getting too many wrong in a row. This is justified by the questionnaire to make more difficulties.
Game must allow user to attempt challenges infinitely on easy difficulties	Should the player choose an easier difficulty, they will be given infinite chances to answer as they are more likely to be younger. This is justified by my questionnaire as my stakeholders outlined the importance of difficulties.
Medium level must have a 5 lives to begin with.	To make the game more challenging for my player, giving them 5 lives as opposed to the infinite amount easy mode has means that the player is more cautious about the mistakes they make and those who like the challenge will find the game more enjoyable. This is justified by my questionnaire as my stakeholders outlined the importance of difficulties.
Hard Level must have 3 lives to begin with.	This makes the level more difficult than medium mode and will make it more challenging specifically for the older age range of stakeholders. As someone apart of that age range, I think that 3 lives is enough to be challenging on players/users but is fair enough

	based on the difficulty of the questions. This is justified by my questionnaire as my stakeholders outlined the importance of difficulties.
Game must have working input fields to fill in the blank challenges	Very similar to the multiple choice; This means that the player should be able to choose and submit an answer with the game responding accordingly. This is a good choice for challenges as my stakeholders are quite young and I do not expect them to be able to code without aid. This is justified by my questionnaire as my stakeholders outlined the importance of difficulties. It's also a simpler version of solo learns code playground I found in my research to be effective so I have decided I will implement it.
To validate the input fields, I will allow my user to enter no more than 12 characters of the English character set	This is because there will be no need for my user to enter values or answers of over 12 characters but also will require my user to use certain character that are not alphanumeric such as "<" and ">" to answer specific questions.
Game will teach syntax of HTML and CSS	The game is an abstraction of a web developer's job with which they work with HTML and CSS. They also work with JavaScript but I believe my stakeholders will be unable to grasp the concept of JSS and thus I will not be teaching them this syntax unless I have more time than I anticipate
Game must send player to start screen if they run out of lives	This is to stop the player progressing for making too many mistakes. By doing this for the more difficult levels, it increases the difficulty and makes the game more satisfying for completing challenges without failing.

Evidently I have met all of these success criteria.

Upon my stakeholder testing, we stumbled across an issue where if you reached game over and went to play again, you would be stuck in the game over screen because the values weren't reset such as counter so I created a reset function to run whenever difficulty buttons are pressed.

```
52
53      public void resetValues()
54      {
55          gamemechanics.money = (100);
56          gamemechanics.counter = 0;
57
58      }
59  }
```

Altering the code with this function added however, solved this issue as it reset the values from the last time my stakeholder played.

What went well ~

M.Miah said she liked that the game and agreed with me that the success criteria I proposed were met, were also met by her standards. She liked how the UI reflected from the screen designs. She believed the progression system is a bit disappointing as it does not have any value without a shop to use the currency you earn but we discussed how time restraints meant that I may not be able to solve the issue of improving the progression scheme, but as this simulation is more to solve the issue of teaching a younger demographic and she agreed with me that this prototype solved this problem.

She agreed that my game has a form of progression with the balance accumulating and decreasing with answering questions. It is evident that my game increments the balance by 100 when getting a question right and decreases by 50 with getting an answer wrong. My game does change the type of challenges depending on the selected difficulty and thus adapts to the different difficulty by loading in different challenges. We both agreed that there is concrete evidence showing that easy mode has infinite lives and that the harder difficulties have a limited number that decreases with getting an answer incorrect.

We agreed that I met the criteria stating how many lives each difficulty should have and this was agreed upon in the feedback from user section of my design.

She agreed that my game had working input fields that compare the users answer to the actual answer of the question and responds appropriately.

She did say however, whilst my questions were based on HTML and CSS, I have not yet taught the user how to code in HTML or CSS.

It is also evident that the game over screen sends the user back to the main menu.

Improvements ~

She said it needs to be more obvious that the user has got the answer correct or incorrect.

Prototype 3

Improvement Fill in the blank: Week Starting 14/01/19

I altered my script so that it included 2 new gameobjects called correctimg and wrongimg as well as another static gameobject called signal which is named appropriately as it signals to the user whether or not they got the answer correct or not.

I also went back to the last few blocks of code I had written to annotate them in case of further maintenance is done on the system.

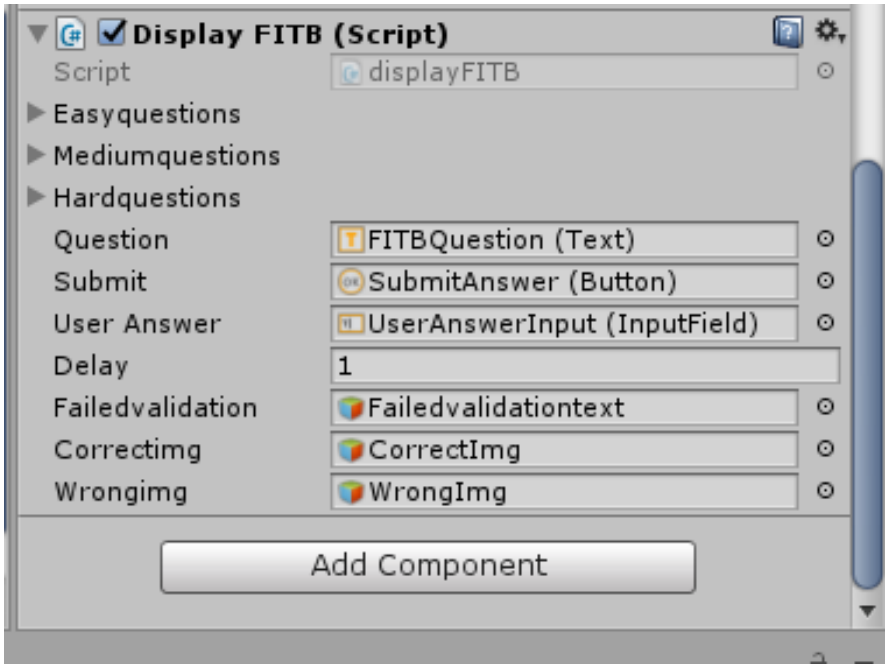
```
38      public GameObject correctimg;
39      public GameObject wrongimg;
40      public static GameObject signal;
41      //Holds correct signal and wrong signal
42      //One of the objects will be assigned to signal which will be displayed during Coroutine
43
```

I will then edit my submitter so that the value of signal is changed depending if user is correct or not (lines 103 and 114)and then will edit the transition coroutine to display the object during the delay.

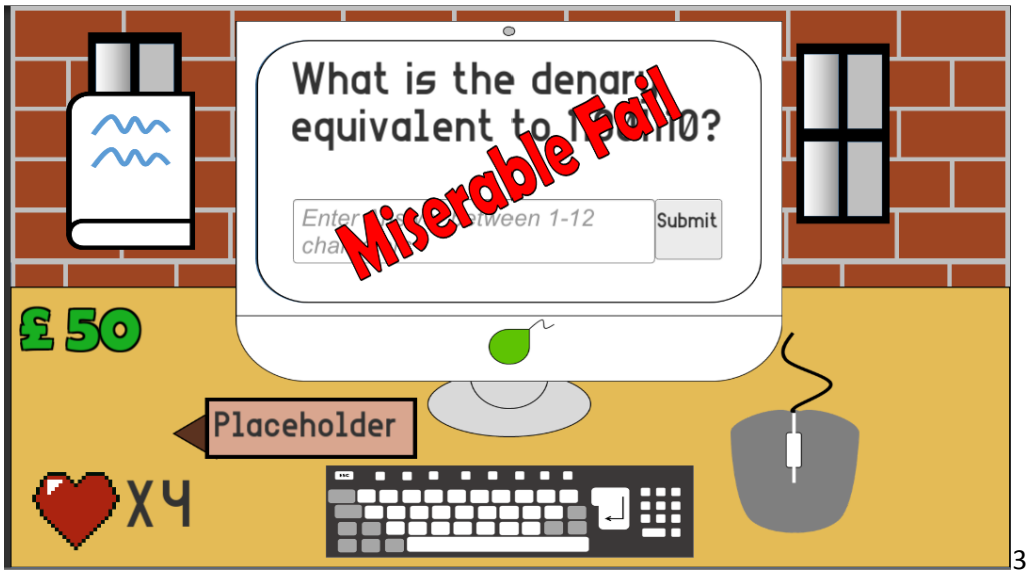
```
98              if ((userAnswer.text).ToLower() == (displayedQuestion.fitbAnsw
99              {
100                 Debug.Log("Correct");
101                 gamemechanics.money += 100;
102                 //increments money if they get question correct
103                 signal = correctimg;
104                 //assigns the correctimg signal to the signal
105
106             }
107             else
108             {
109                 Debug.Log("Wrong");
110                 if (gamemechanics.money > 0)
111                 {
112                     gamemechanics.money -= 50;
113                 }
114                 signal = wrongimg;
115                 //assigns the wrongimg signal to the signal
116
117                 gamemechanics.lives -= 1;
118                 //Decrements life and money if they get answer wrong
119             }
120
121             StartCoroutine(Transition());
122
123
```

```
131
132     IEnumerator Transition()
133     {
134         //This coroutine controls the delay before the game is reloaded
135         unansweredQuestions.Remove(displayedQuestion);
136         //removes the question from the list
137         signal.gameObject.SetActive(true);
138         //signals user that they are correct or incorrect
139         yield return new WaitForSeconds(delay);
140
141         SceneManager.LoadScene("CoreGame");
142         //reloads scene
143     }
144
145
```

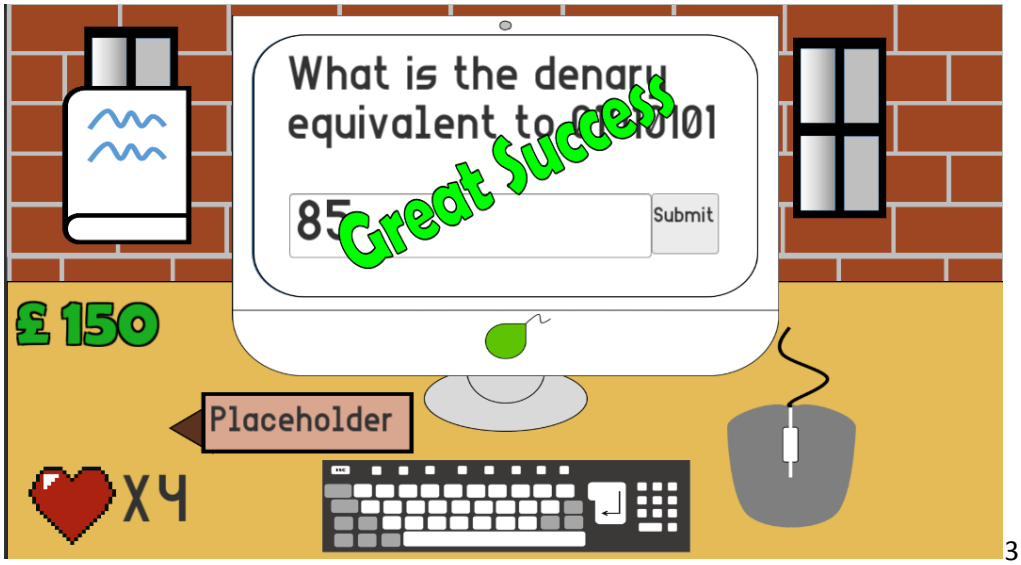
I then created the UI and linked them to their respective elements.



When tested:



3.1.1



3.1.2

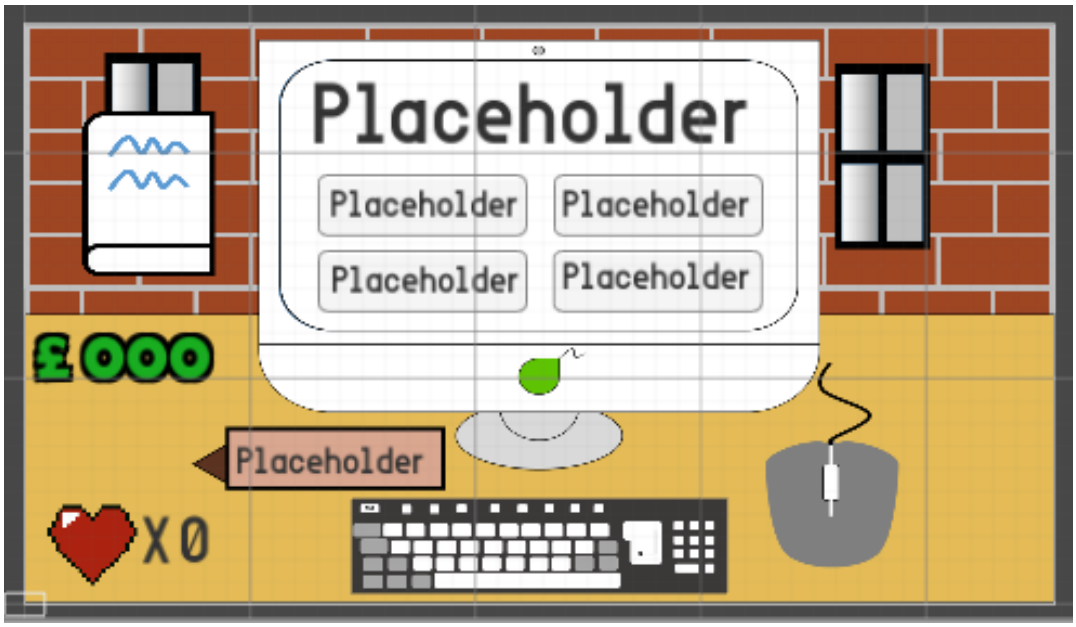
What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
Signal appears		Miserable fail	3.1.1	Next question	Adding game objects displaying during transition phase
	85	Great success	3.1.2	Great success	Adding game objects displaying during

transition phase

I showed this to my stakeholder who agreed that this was a good solution for the problem she had with prototype 2.

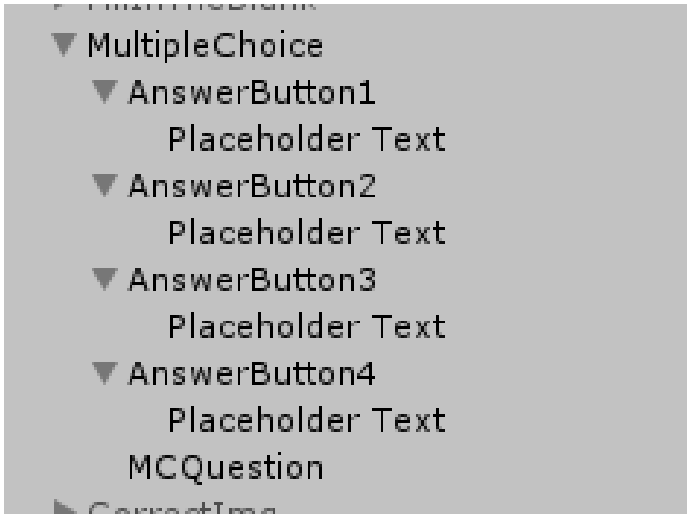
Multiple-choice: Week Starting 21/01/19

First I created the layout for multiple choice questions in my unity editor.



I used this layout as it reflects my screen design that my stakeholder said was effective and made the edits to it that she wished for me to make such as removing the cactus as it “cramped the screen”.

Named appropriately down below.



Then I created a new script called MCQuestions to distinguish them from FITBQuestions and opened it. I edited it in the same way as I created the FITBQuestions but added 4 buttons for the 4 different answers that will be in display.

```
[System.Serializable]
public class MCQuestions
{

    public string theQuestion;
    public string mcAnswer;
    public string firstOption;
    public string secondOption;
    public string thirdOption;
    public string fourthOption;
    // declares MCQuestions class so every object has a question, answer and 4 possible answers.

}

```

This scripts defines the MCQuestions class to have an answer, a question and 4 possible options for the player to choose, one of which will be the answer.

Next, I created another script called displayMC that will have the same function as displayFITB where it will randomly choose a question from a list of unanswered questions and then display it. The script will also compare the user's response to the answer of the question and alter the number of lives and money counter appropriately.

To save time, I copied blocks of code from the displayFITB script and altered them to fit multiple choice questions.

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

public class displayMC : MonoBehaviour {

    public MCQuestions[] easyquestions;
    public MCQuestions[] mediumquestions;
    public MCQuestions[] hardquestions;
    //holds list of questions, options and answers for Multiple choice questions

```

I edited the declaration of the arrays of questions depending on question from being an array of fill in the blank questions to multiple choice questions as the script will be displaying multiple choice questions.

```
14
15      [SerializeField]
16      private Text question;
17      //links the text from the core game to the script
18      [SerializeField]
19      private Text option1;
20      //links the option1 button from the core game to the script
21      [SerializeField]
22      private Text option2;
23      //links the option2 button from the core game to the script
24      [SerializeField]
25      private Text option3;
26      //links the option3 button from the core game to the script
27      [SerializeField]
28      private Text option4;
29      //links the option4 button from the core game to the script
30
31
32
```

I have used text here as I will be comparing the text of the button to the answer rather than the button itself. Not only this, but I will be randomising the order of the options.

I also copied the transition coroutine as I will be using it on this script as well.

```
51      IEnumerator Transition()
52      {
53          //This coroutine controls the delay before the game is reloaded
54          unansweredQuestions.Remove(displayedQuestion);
55          //removes the question from the list
56          signal.gameObject.SetActive(true);
57          //signals user that they are correct or incorrect
58          yield return new WaitForSeconds(delay);
59
60          SceneManager.LoadScene("CoreGame");
61          //reloads scene
62      }
63  }
64
```

The red underline is temporary as I am yet to add these variable and methods. I added the delay prior to this which is why delay is not underlined.

```
35      private static List<MCQuestions> unansweredQuestions;
36      // creates a list of MC objects
37
38      private MCQuestions displayedQuestion;
39      //Will be the displayed question in-game
40
41      public GameObject failedvalidation;
42      //holds text prompting user to enter answer with less than 13 characters
43
44      public GameObject correctimg;
45      public GameObject wrongimg;
46      public static GameObject signal;
47      //Holds correct signal and wrong signal
48      //One of the objects will be assigned to signal which will be displayed during Coroutine
49
```

I have added the correctimg, wrongimg and signal gameobjects once again as these will be displayed if the player gets the answer correct or incorrect.

I then removed the failedvalidation as I realised as the user is not entering answers via input fields, there would be no need for validation. I also altered line 35 and 38 to be a list of MCQuestions rather than FITBQuestions and also changed displayed questions to be a MCQuestions object.

```
38    private MCQuestions displayedQuestion;
39    //Will be the displayed question in-game
40
41
42    public GameObject correctimg;
43    public GameObject wrongimg;
44    public static GameObject signal;
45    //Holds correct signal and wrong signal
46    //One of the objects will be assigned to signal which will be displayed during Coroutine
47
```

I then copied the start function of displayFITB and altered it so it could have functionality for MC questions.

```
50    // Use this for initialization
51    void Start () {
52        if (unansweredQuestions == null || unansweredQuestions.Count == 0)
53        {
54
55            if ((gamemanager.difficulty) == ("easy"))
56            {
57                unansweredQuestions = easyquestions.ToList<MCQuestions>();
58            }
59
60            if ((gamemanager.difficulty) == ("medium"))
61            {
62                unansweredQuestions = mediumquestions.ToList<MCQuestions>();
63            }
64            if ((gamemanager.difficulty) == ("hard"))
65            {
66                unansweredQuestions = hardquestions.ToList<MCQuestions>();
67            }
68            // changes the list of questions are asked depending on their difficulty
69
70        }
71        SetDisplayedQuestion();
72
```

I changed the line 57,62 and 66 to put lists of MCQuestions rather than FITBQuestions.

Line 71 is underlined red as I have not yet added the code for SetDisplayQuestion() in this script but, using thinking ahead, I know that I will be.

I then created the SetDisplayedQuestion function:

```
78  void SetDisplayedQuestion()
79  {
80      int randomQuestionIndex = Random.Range(0, unansweredQuestions.Count);
81      // generates a random question between range of 0 to the number of elements in the list
82      displayedQuestion = unansweredQuestions[randomQuestionIndex];
83      //changes value of displayed question to the randomly picked question
84
85      question.text = displayedQuestion.theQuestion;
86      //value of the text
87
88
89  }
90
```

I completely copied the SetDIsplayedQuestion function from the FITB script as it holds complete functionality to the methods and attributes of this new script.

I then altered the script so that when setting the question, it takes all the options for that question and randomly assigns them to one of the 4 buttons by changing the text value of that button:

```
87
88      List<string> possibleAnswers = new List<string>();
89      possibleAnswers.Add(displayedQuestion.firstOption);
90      possibleAnswers.Add(displayedQuestion.secondOption);
91      possibleAnswers.Add(displayedQuestion.thirdOption);
92      possibleAnswers.Add(displayedQuestion.fourthOption);
93      //Adds all the options for that question into a list called possible answers
94
95      while (possibleAnswers != null)
96          //This while loop randomly assigns a question to the text value of each button
97      {
98          int randomA = Random.Range(0, possibleAnswers.Count);
99          option1.text = possibleAnswers[randomA];
100         possibleAnswers.RemoveAt(randomA);
101
102         int randomB = Random.Range(0, possibleAnswers.Count);
103         option2.text = possibleAnswers[randomB];
104         possibleAnswers.RemoveAt(randomB);
105
106         int randomC = Random.Range(0, possibleAnswers.Count);
107         option3.text = possibleAnswers[randomC];
108         possibleAnswers.RemoveAt(randomC));
109
110         int randomD = Random.Range(0, possibleAnswers.Count);
111         option4.text = possibleAnswers[randomD];
112         possibleAnswers.RemoveAt(randomD);
113     }
114
115
116 }
117
```

Line 88 declares a string called possibleAnswers. Then line 89 to 92 adds all the possible options attributes of the new displayedQuestion to that list.

The while loop states that for so long as the list is not empty, to pick a random number between 0 and the length of the list. It then assigns the text at that element of the possibleAnswers list to one of the buttons and removes the answer from the list so it cannot be reassigned to another button.

Variables randomA , randomB, randomC and randomD are named appropriately as they contain a random integer and are only used once per question so they are not of much significance.

Now I need a function that takes the users answer and compares it to the actual answer and as I have 4 buttons, they can act as 4 submit buttons. So I copied the code for the submit button and

altered it to compare the text of the button with the answer of the question and to then respond accordingly.

I then created this public procedure so it can be used when option1 is clicked:

```
122
123     public void buttonClicked1()
124     {
125         if (displayedQuestion.mcAnswer == option1.text)
126         {
127             Debug.Log("Correct");
128             gamemechanics.money += 100;
129             //increments money if they get question correct
130             signal = correctimg;
131             //assigns the correctimg signal to the signal
132
133         }
134
135         else
136         {
137             Debug.Log("Wrong");
138             if (gamemechanics.money > 0)
139             {
140                 gamemechanics.money -= 50;
141             }
142             signal = wrongimg;
143             //assigns the wrongimg signal to the signal
144
145             gamemechanics.lives -= 1;
146             //Decrements life and money if they get answer wrong
147         }
148         StartCoroutine(Transition());
149     }
150
```

Line 125 is a condition where it takes the text value of button 1 and compares it to the answer and if it is correct then it responds accordingly by incrementing the values and signalling to the user it is correct, otherwise it will decrement the money counter and number of lives. It will then begin the transition coroutine and reload the scene.

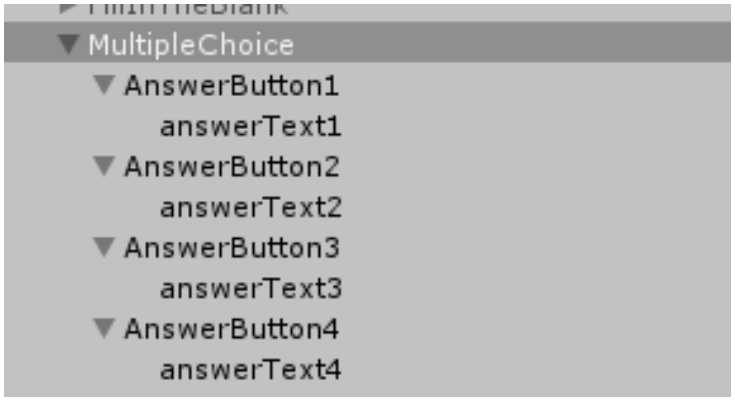
I then copied this code 3 more times and altered them for the other 3 buttons.

```
public void buttonClicked2()
{
    if (displayedQuestion.mcAnswer == option2.text)
    {
        Debug.Log("Correct");
        gamemechanics.money += 100;
        //increments money if they get question correct
        signal = correctimg;
        //assigns the correctimg signal to the signal

    }

    else
    {
        Debug.Log("Wrong");
        if (gamemechanics.money > 0)
        {
            gamemechanics.money -= 50;
        }
        signal = wrongimg;
        //assigns the wrongimg signal to the signal

        gamemechanics.lives -= 1;
        //Decrements life and money if they get answer wrong
    }
    StartCoroutine(Transition());
}
```

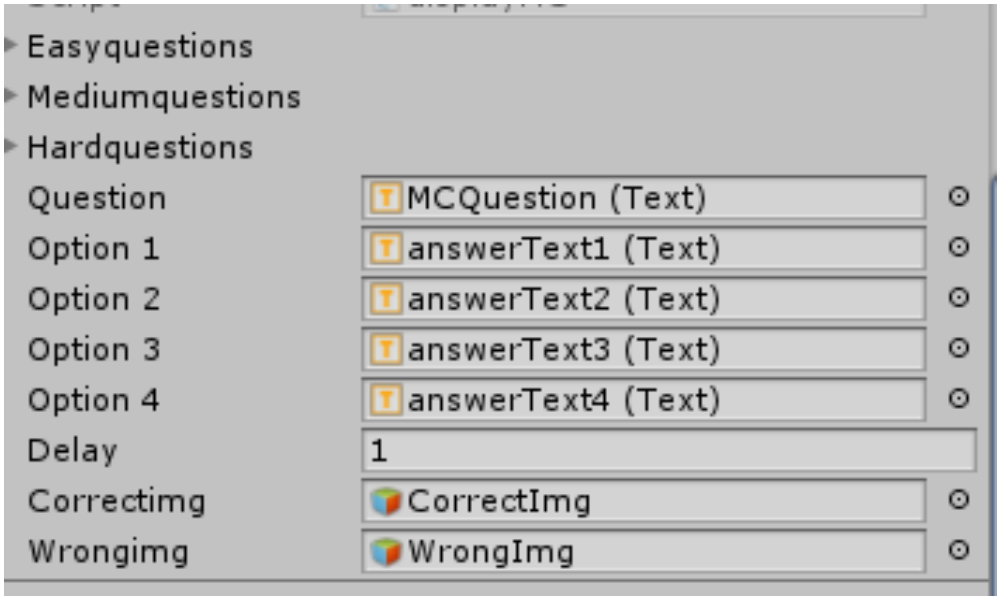
```
if (displayedQuestion.mcAnswer == option3.text)
{
    Debug.Log("Correct");
    gamemechanics.money += 100;
    //increments money if they get question correct
    signal = correctimg;
    //assigns the correctimg signal to the signal

}

else
{
    Debug.Log("Wrong");
    if (gamemechanics.money > 0)
    {
        gamemechanics.money -= 50;
    }
    signal = wrongimg;
    //assigns the wrongimg signal to the signal

    gamemechanics.lives -= 1;
    //Decrements life and money if they get answer wrong
}
StartCoroutine(Transition());
}
```

```
207     public void buttonClicked4()
208     {
209         if (displayedQuestion.mcAnswer == option4.text)
210         {
211             Debug.Log("Correct");
212             gamemechanics.money += 100;
213             //increments money if they get question correct
214             signal = correctimg;
215             //assigns the correctimg signal to the signal
216 
217         }
218 
219         else
220         {
221             Debug.Log("Wrong");
222             if (gamemechanics.money > 0)
223             {
224                 gamemechanics.money -= 50;
225             }
226             signal = wrongimg;
227             //assigns the wrongimg signal to the signal
228 
229             gamemechanics.lives -= 1;
230             //Decrements life and money if they get answer wrong
231         }
232         StartCoroutine(Transition());
233     }
234 
```

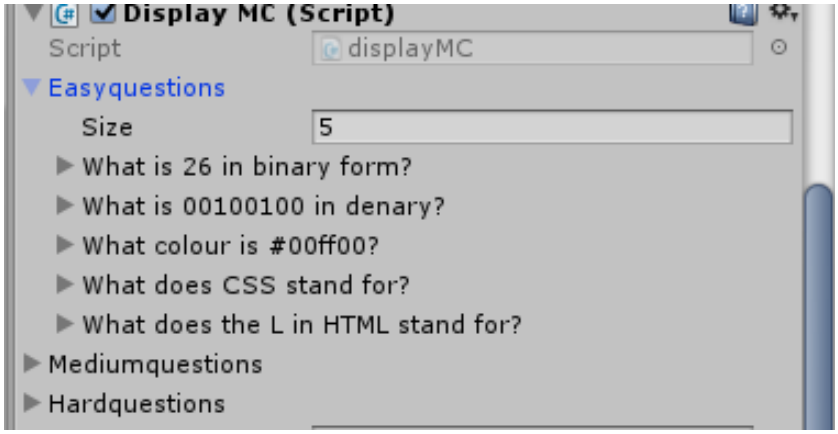
Next, I renamed the placeholder texts so they were more appropriate and could be distinguished from one another.



I then mapped the UI elements to the components of the script so they can begin to have functionality.

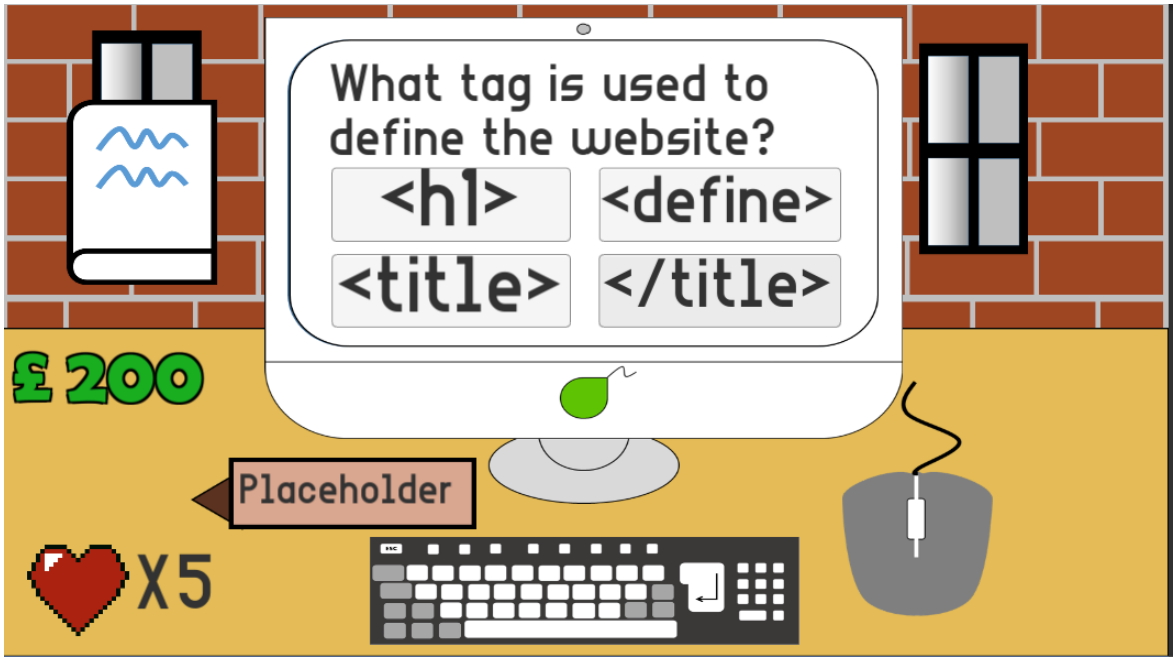


Before testing, I would now add the questions I planned in my design to their respective difficulties



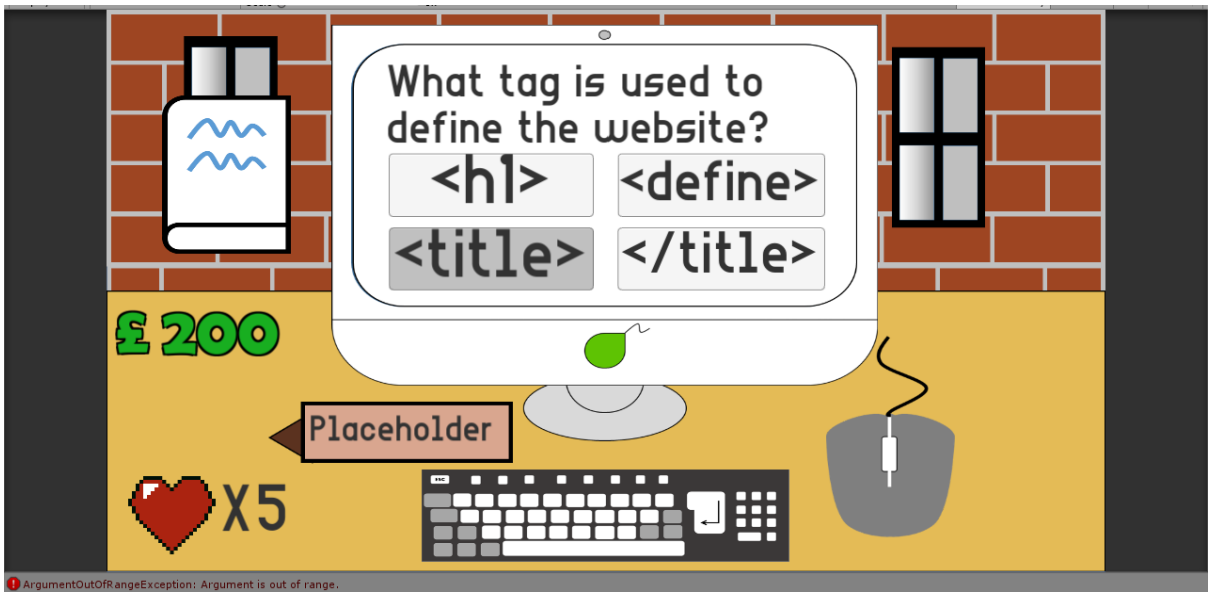
I then tested the prototype:

Before click:



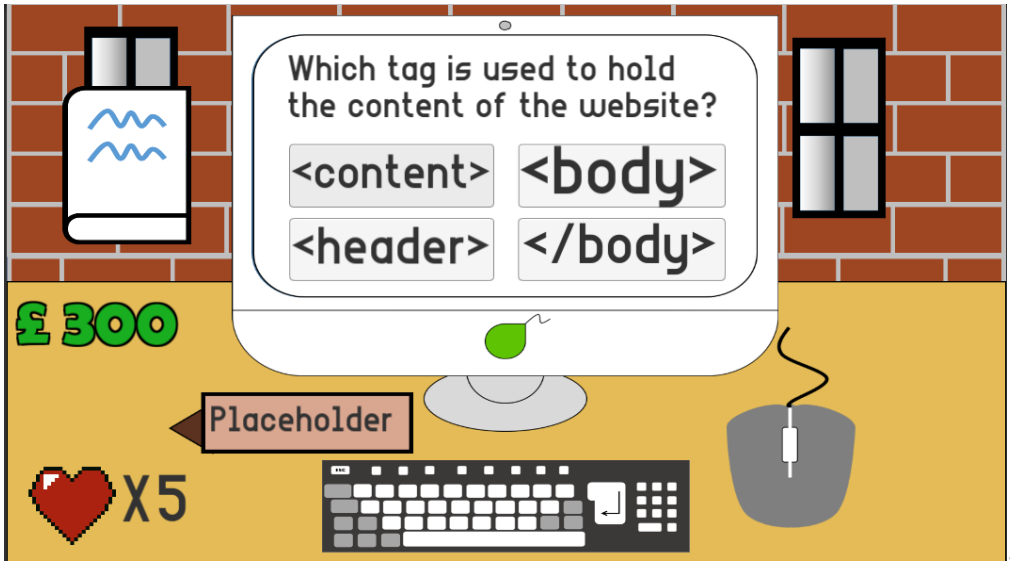
3.2.1

After click

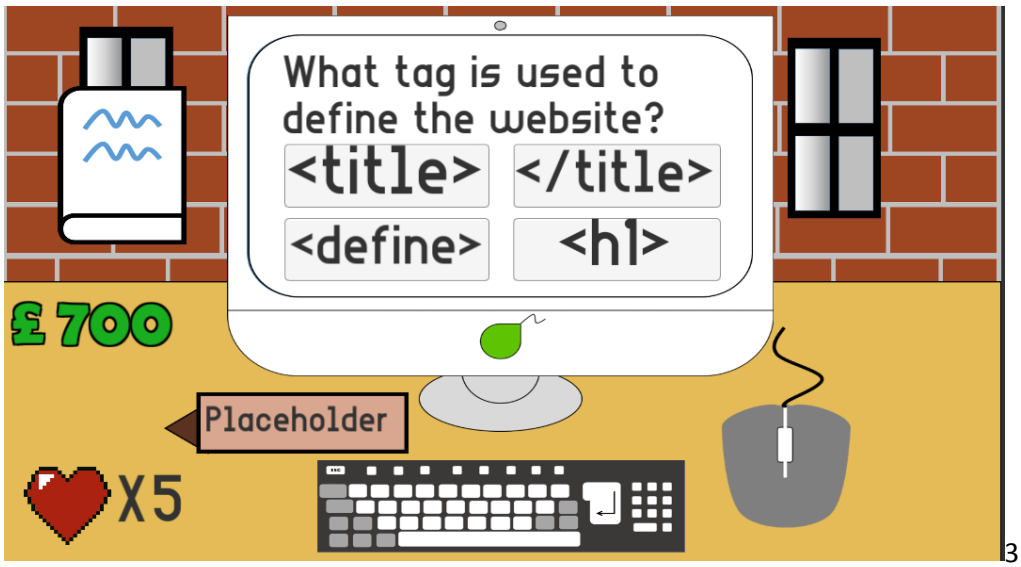


3.2.2

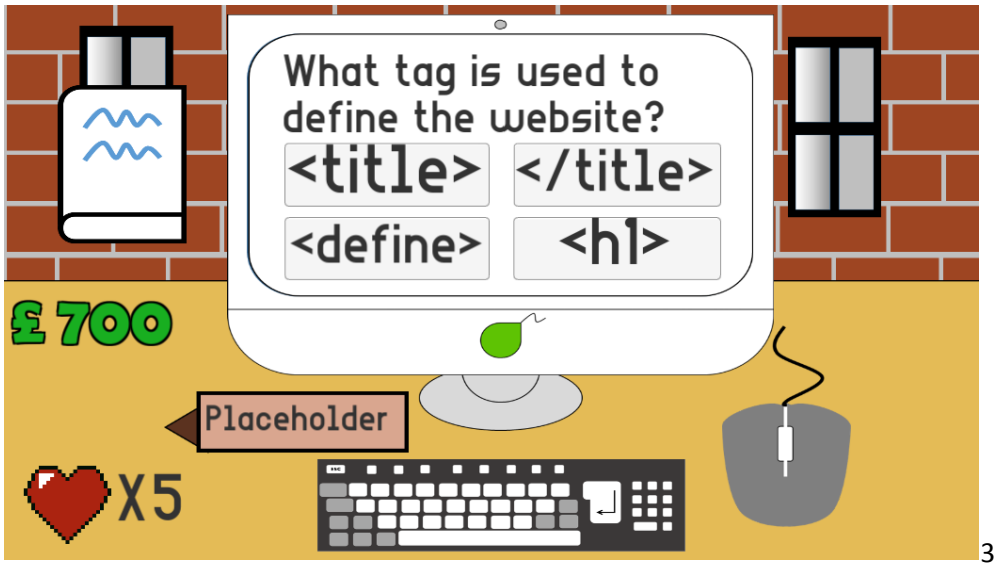
Notice here there is an argument error, this is expected as I remove the element from the list but because it is in a while loop so it will continue to assign the button with one of the options until the list is empty of options to assign.



3.2.3



3.3.1

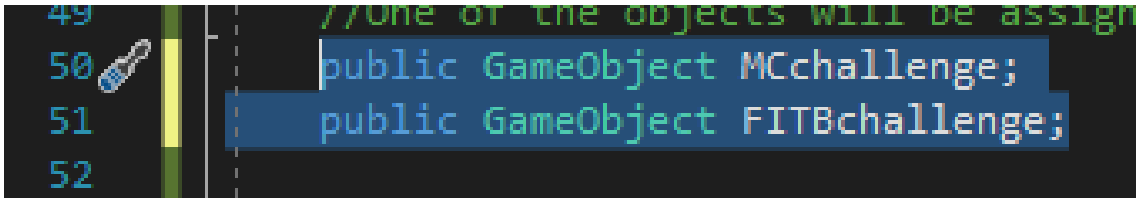


3.3.2

What is	Test data to	What is the	Evidence	Outcome	Any

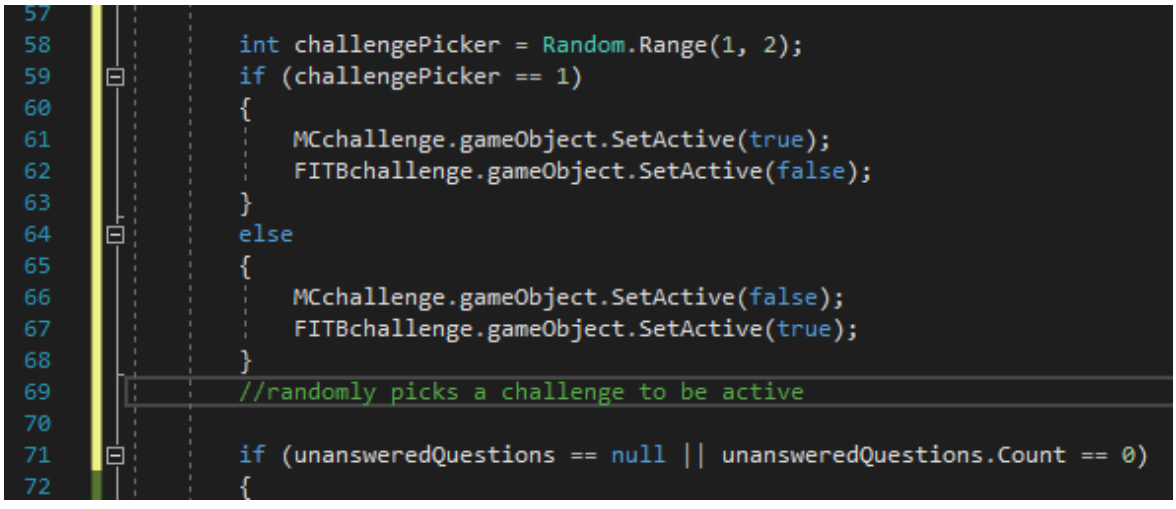
being tested	input	expected result			Modifications
Buttons are functional and loads another question	Clicking answer	Loads another question	3.2.1 3.2.2 3.2.3	Next question Success	
Buttons randomise order	Clicking buttons	Buttons randomise order	3.2.1 and 3.3.1	Randomised buttons Success	
Validation	RC buttons	Null	3.3.1 to 3.3.2	Null Success	

Next I added blocks of code to both the displayFITB and displayMC scripts that would randomly choose which type of question displayed. To do so I would use random as I have done in the past in this project to choose a random number between 1 and 2 and then make that gameobject active.



I added this to both to allow me to control which challenge is active.

Then I created a variable called challengePicker in the start function which will randomly pick a number between 1 and 2 and then make one of the challenges active to the user.



challengePicker is an integer variable that holds a random integer, if that integer = 1 then a multiple choice question will be asked, otherwise a fill in the blank question will be asked.

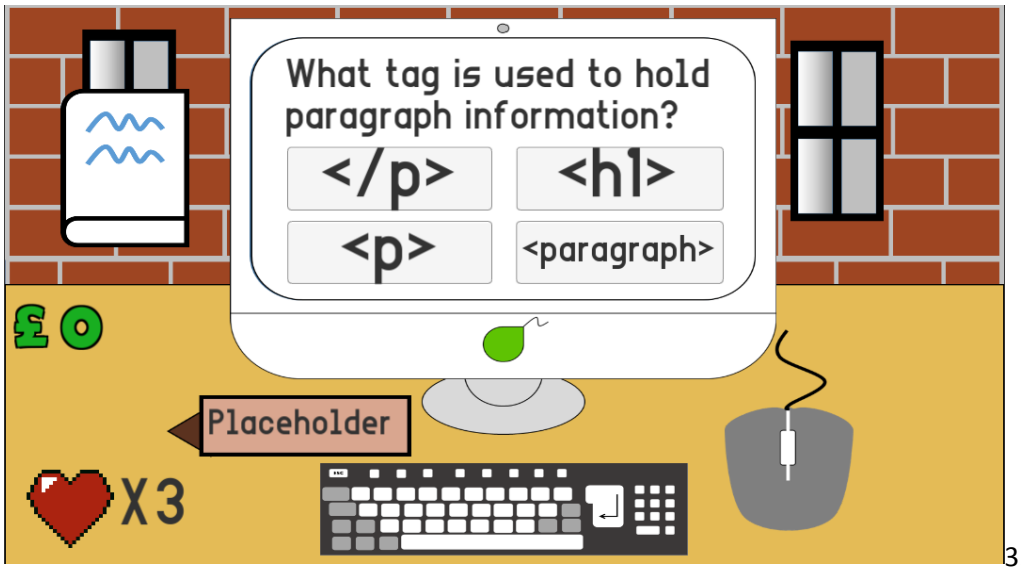
After assigning them I tested out the randomiser.

When tested:

It failed as the issue was that the random has an inclusive first parameter but the second is exclusive so I changed the range to 0 and 2 for both.

After changing this, I re tested this:

3.4.1



3.4.2

What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications

Randomiser	Answering	Loads a different challenge	3.4.1 to 3.4.2	Loads a different challenge	Changing range of randomiser

I believe that this has completed this prototype.

Prototype 4

Feedback Multiple-choice: Week Starting 11/02/19

I presented my stakeholder with the new prototype whilst showing her the success criteria:

Success criteria	Feedback
Game must have working multiple choice challenges	She thought I provided a great solution to this and that my project dealt with this really well. She liked how the answer buttons randomised and how the FITB and MC challenges randomly become active and deactive.

What went well ~ she agreed the criteria was met and thought it was an effective solution.

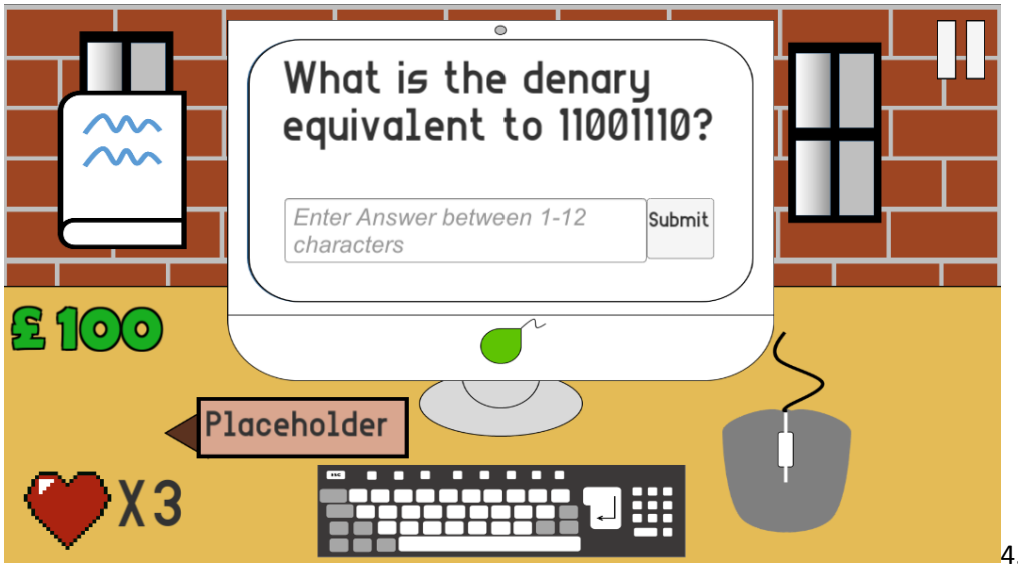
She told me there wasn't a need for improvement at the time being as it met her expectations.

Pause Menu: Week Starting 18/02/19

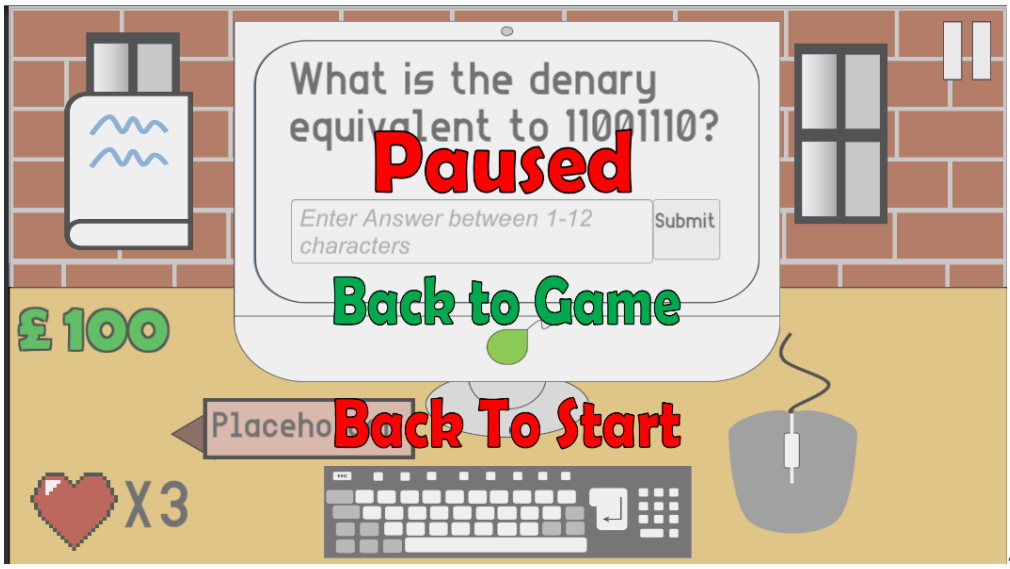
I used the screen design I created in the design section and added a back to game button in my unity editor.

I then tested the buttons and this was the result:

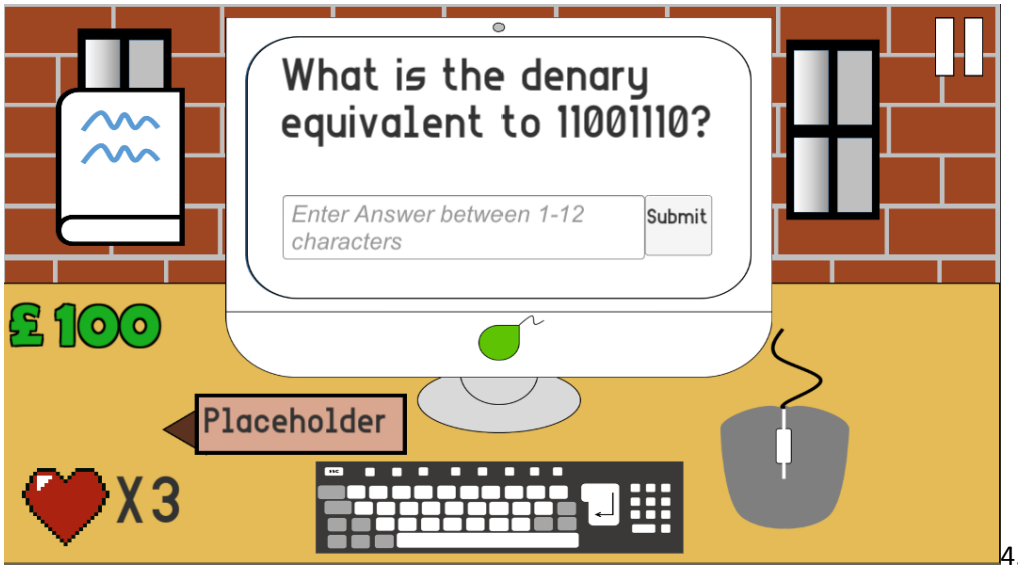
Before click


What is the denary equivalent to 11001110?
Enter Answer between 1-12 characters
Submit
£100
Placeholder
X3

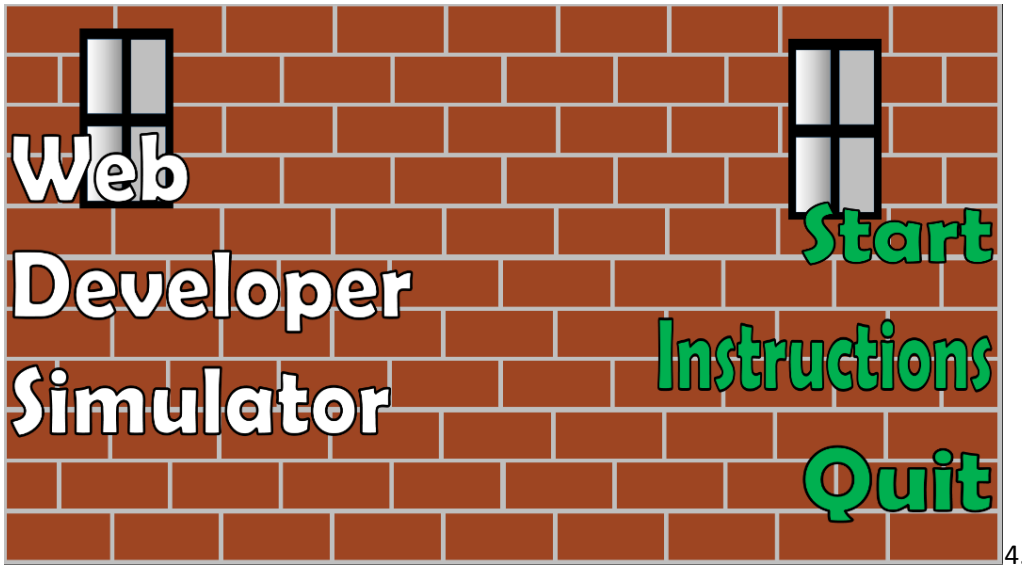
4.1.1


What is the denary equivalent to 11001110?
Paused
Enter Answer between 1-12 characters
Submit
Back to Game
£100
Placeho
Back To Start
X3

4.1.2


What is the denary equivalent to 11001110?
Enter Answer between 1-12 characters
Submit
£100
Placeholder
X3

4.1.3



4.1.4

What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
Pause button opens pause screen	Clicking pause	Pause button opens pause screen	4.1.1 to 4.1.2	Pause menu	
Back to game button goes back to the game	Clicking back to game	Back to game button goes back to the game	4.1.3	Back to game	
Return to title button returns player to title	Clicking Back to main menu	Return to title button returns player to	4.1.4	Main menu	
Validation	RC	Null	N/A	Null	

That completes this prototype.

Prototype 5

Feedback Pause Menu: Week Starting 4/03/19

I showed the prototype to my stakeholder and this was her response:

Success Criteria	Feedback
Game must allow user to pause and have a pause menu	The pause menu was exactly up to her expectations as she agreed it met with her request to add a return to game button that I had left out in my screen designs.

Book feature: Week Starting 4/03/19

First I created a script for the book and used the appropriate namespace.

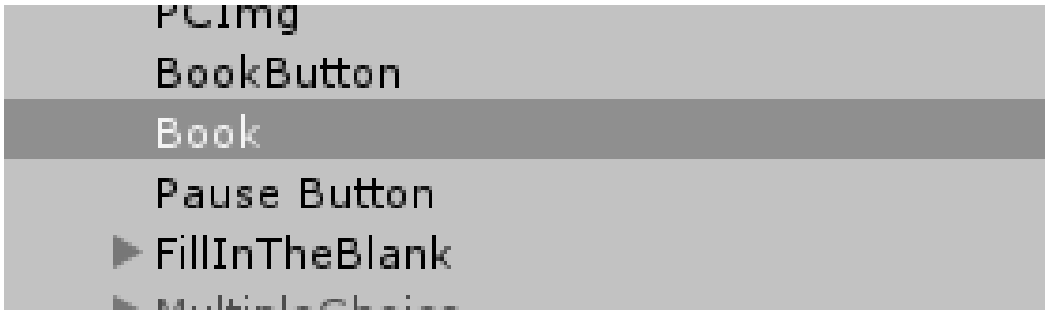
```
1   using System.Collections;
2   using System.Collections.Generic;
3   using UnityEngine;
4   using UnityEngine.UI;
```

As I would be working with pre made images of the book.

```
8       public Image[] pages;
9       public int pagenumber = 0;
10      public GameObject TheBook;
11
```

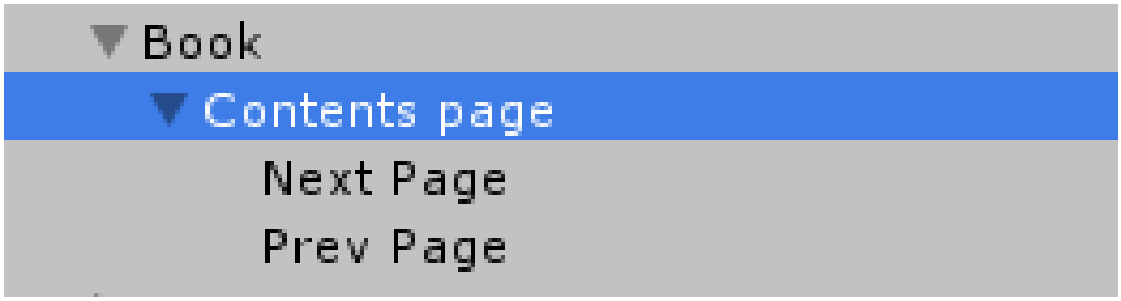
Made a list of images that would hold the pages.

Pagenumber is a variable I am using to show what page the book begins on and gameobject TheBook holds the book object in the game.

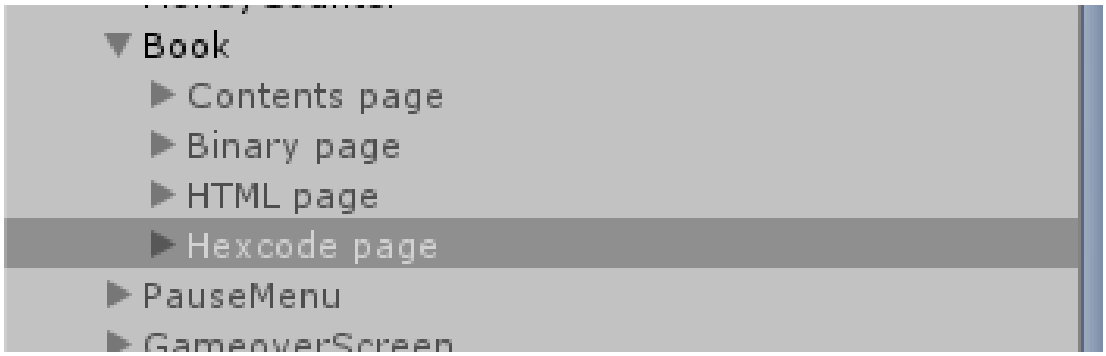


I then created a new object called book that would hold the necessary images and information and would become active when the user clicks the book button.

I added the book script to the book object and added the pages to the list of images after adding to the unity editor.



I duplicated this for all 4 pages and changed them accordingly.



I then added functionality to the next and prev buttons by creating 2 methods that would increment and decrement the page number.

```
12      public void nextpage()
13      {
14          if (pagenumber < pages.GetLength(0) - 1)
15          {
16              pagenumber += 1;
17
18          }
19          //Changes the page to the next page
20      }
21
22      public void prevpage()
23      {
24          if (pagenumber > 0)
25          {
26              pagenumber -= 1;
27
28
29          }
30          //Changes the page to the previous page
31      }
```

Next page would only increment if the user was not on the last page and prevpage would only go back a page if the user was not in the contents page.

I then altered the update function so that only the page the user is on would be active otherwise the pages would overlap one another.

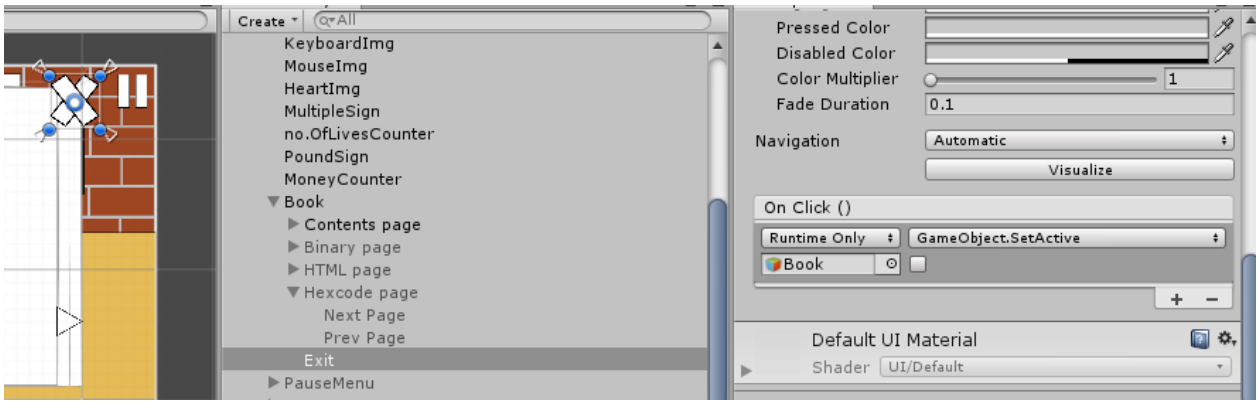
```
void Update()
{

    for (int i = 0; i < pages.GetLength(0); i++)
    {
        pages[i].gameObject.SetActive(false);
    }

    pages[pagenumber].gameObject.SetActive(true);
    //Loop deactivates all pages and then reactivates the page the user is on
}
```

This for loop will go through every page making them un-active and then activate the page the user was actually on to prevent overlapping.

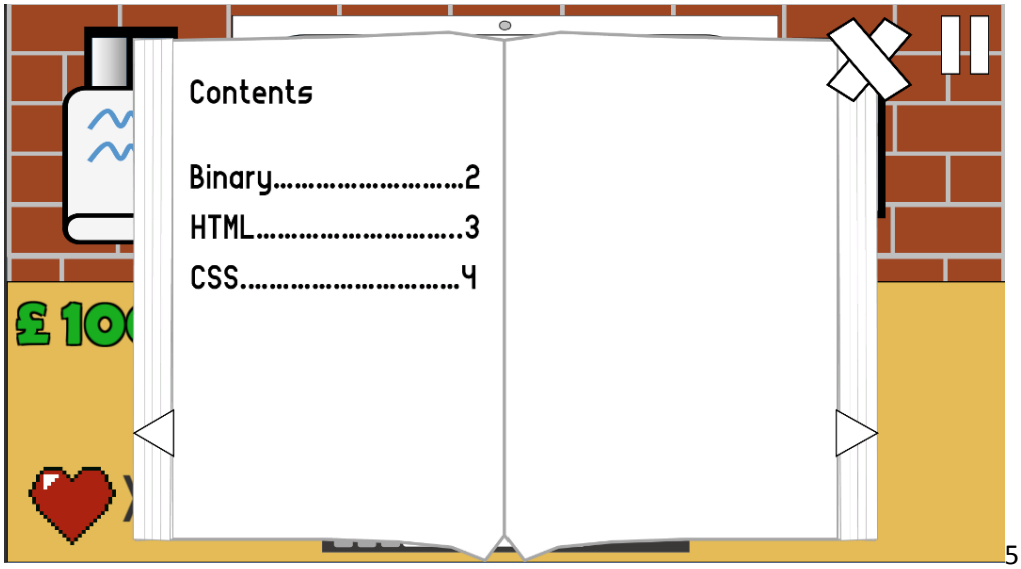
The last part of this was a resetter function to ensure that every time the book closed and reopened, it would reopen on the contents.

```
public void resetter()
{
    pagenumber = 0;
    TheBook.gameObject.SetActive(true);


}
```

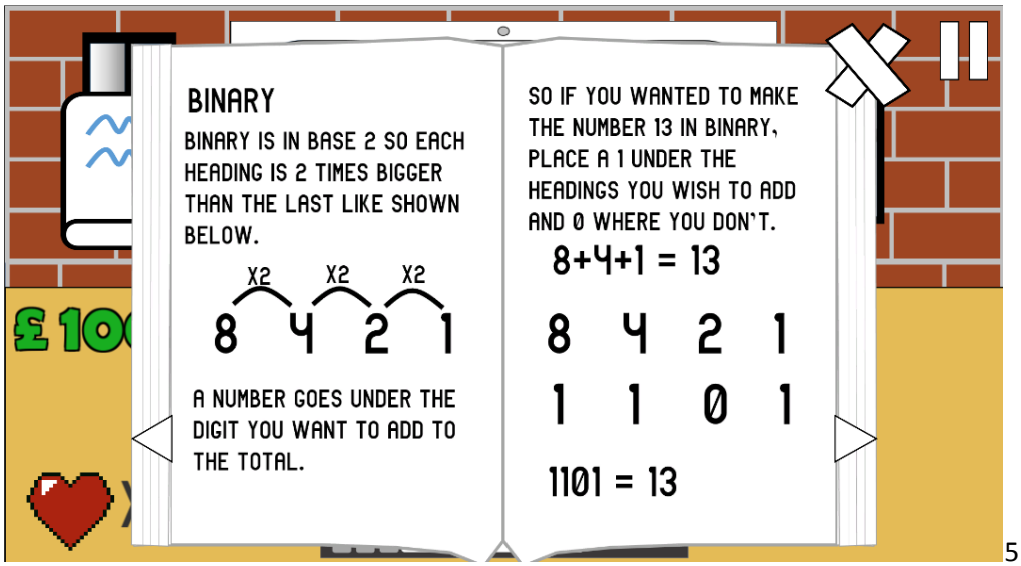
I mapped this to the book button and added an exit button on the book object to close the book.



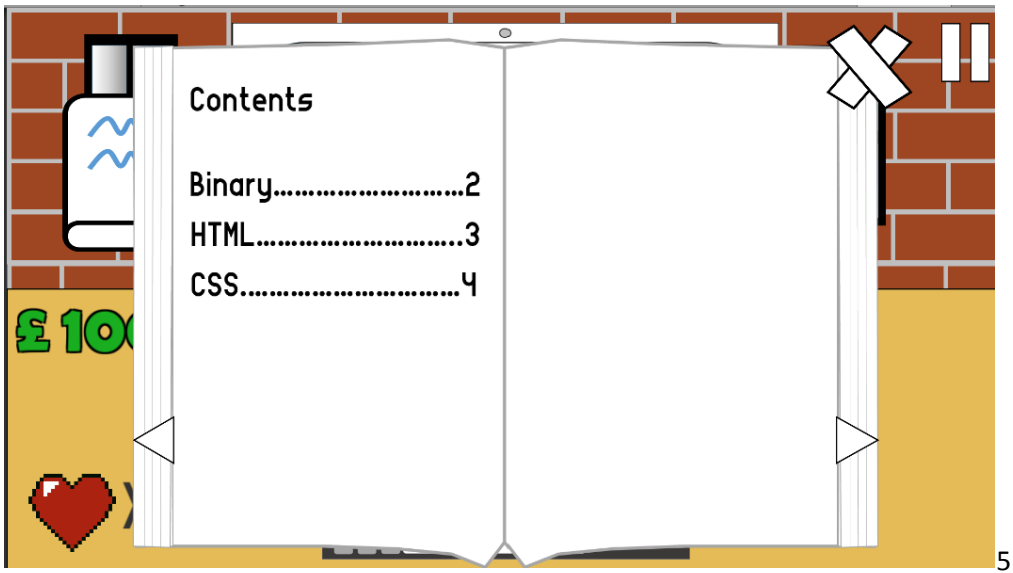
I then tested:



5.1.1

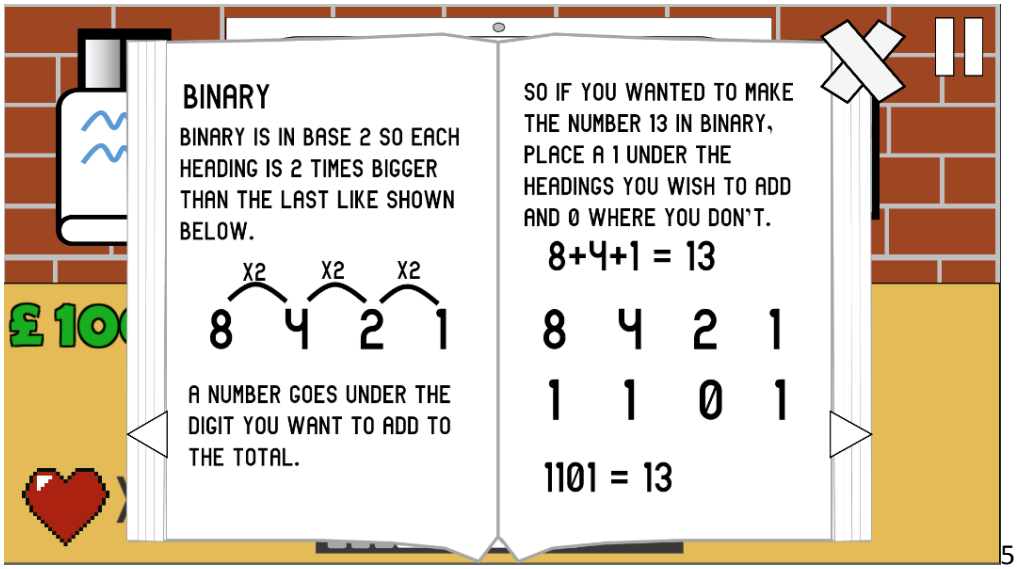


5.1.2



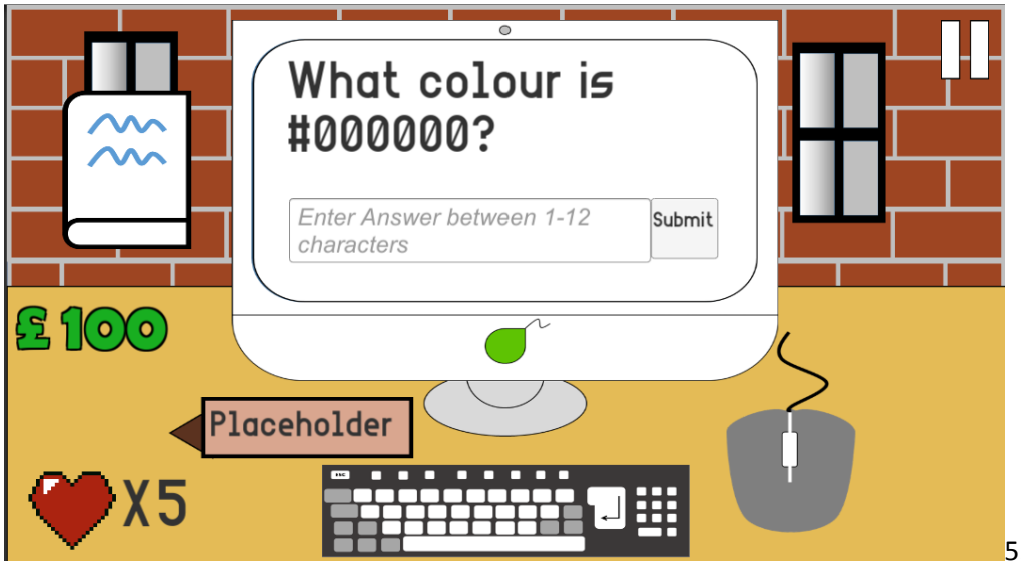
5.1.3

Before click



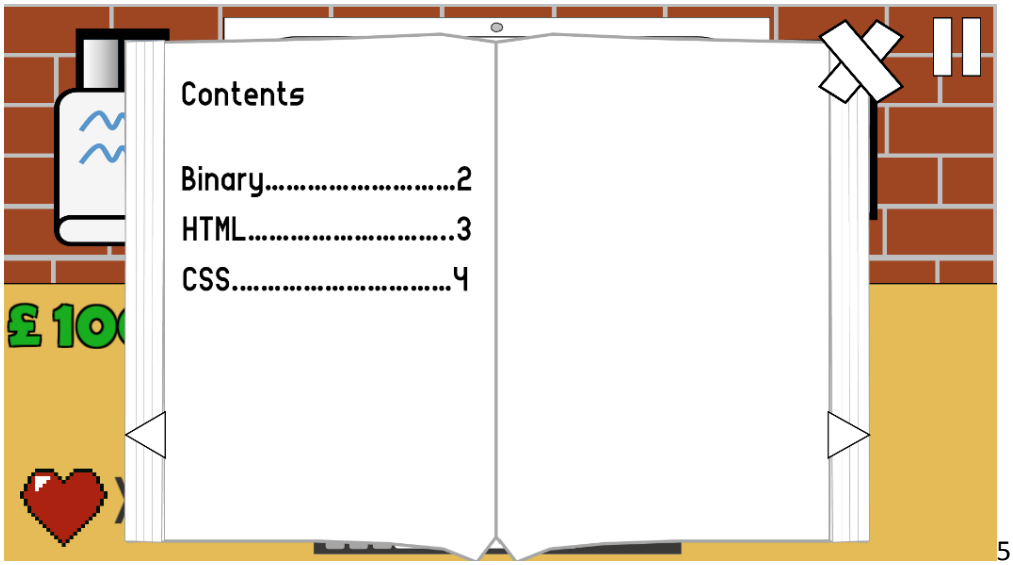
5.2.1

After click



5.2.2

Book click



5.2.3

What is being tested	Test data to input	What is the expected result	Evidence	Outcome	Any Modifications
Book button	Click book	Pause button opens pause screen	5.1.1	Book	
Next page	Click next	Back to game button goes back to the game	5.1.2	Next page	
Prev page	Click prev	Return to title button returns player to	5.1.3	Prev page	
Back to contents	Click X and then book	Null	5.2.1, 5.2.2, 5.2.3	Starts book from the beginning	

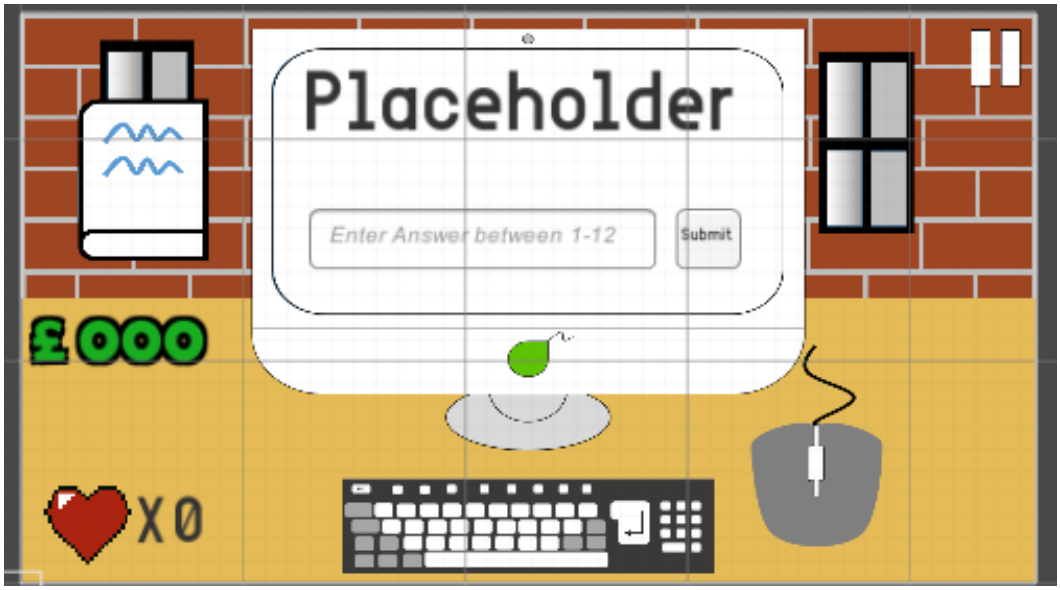
Before showing the prototype, I added the instructions in this prototype as I was running out of time in the project due to the time restraints of this project.

End of prototype 5

Final Interview: 11/03/19

Success criteria	Feedback
Game will teach syntax of HTML and CSS	She thought I did this well and agreed my game did teach her parts of HTML and CSS
The book feature must be working and provide useful tips that would help the player in the challenges	She agreed that the book feature met her expectations and was ascetically pleasing to her
The book feature must have 2 buttons that allow the player to change pages back and forth to their liking.	Evidence shows the buttons feature worked.

I told her that I would be unable to add music or personalisation of any kind due to the difficulty of working with sprites and audio in comparison to my current level of skill with working with unity and c#. She acknowledged that I had completed most of the agreed upon success criteria and that the solution I had provided had solved the initial problem I set out to find a solution for. This is justified as I have a working game that allows the player to learn using the book feature and is able to put the user through challenges that increases their skill in HTML and CSS as well as some other basic computer science topics such as binary to denary conversion. This also led to one final change to remove the nameplate in game as it served no purpose.



Signoff 14/03/19

M.Miah contributed to the development of my project by giving feedback consistently.

……..

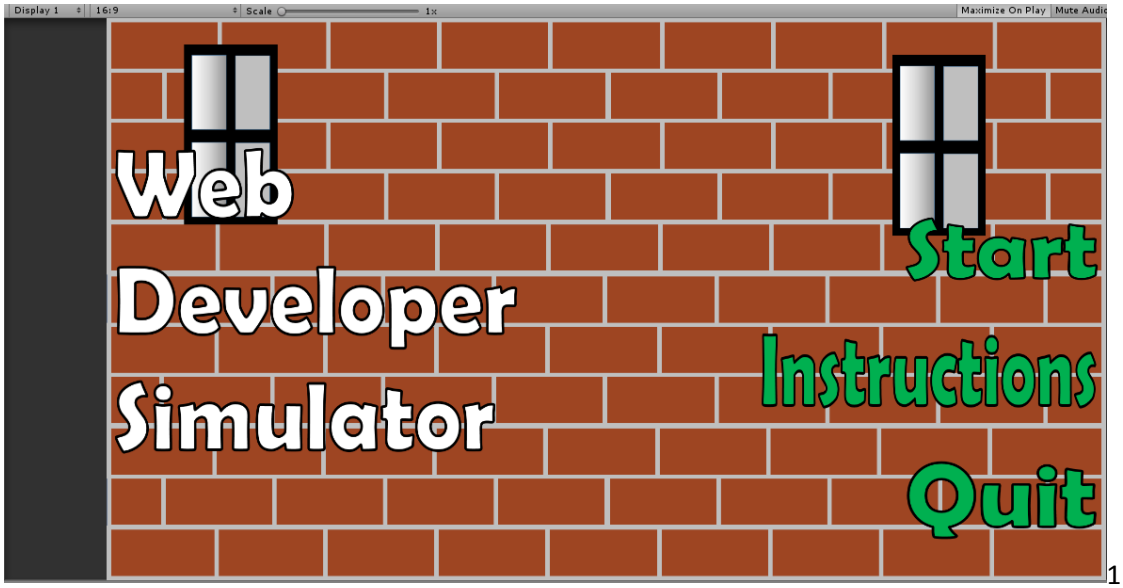
End of development

Testing to inform Evaluation

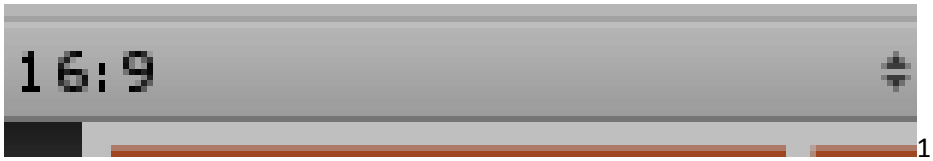
Screenshots:

Main game:

Aspect ratio



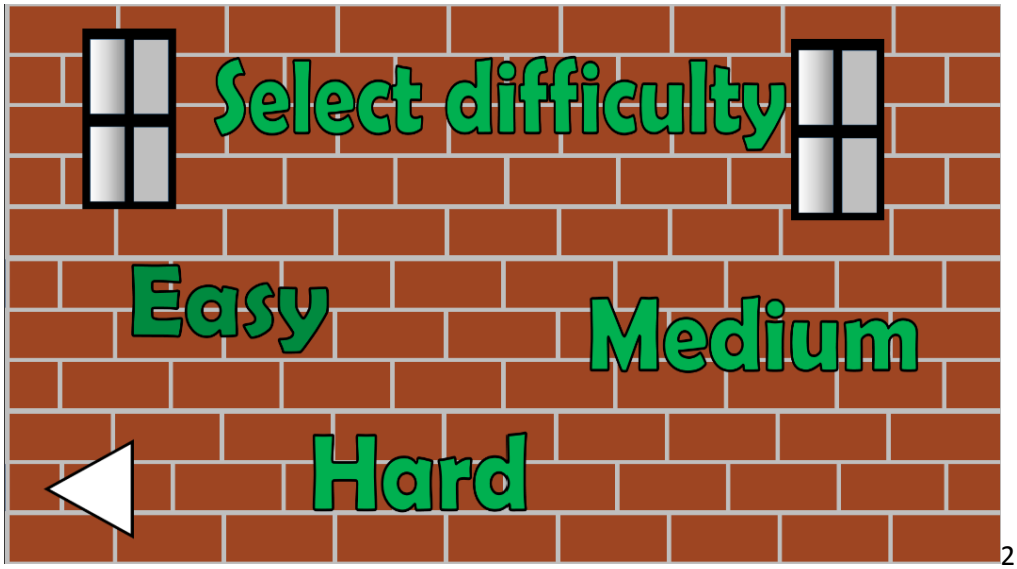
1.1



1.2

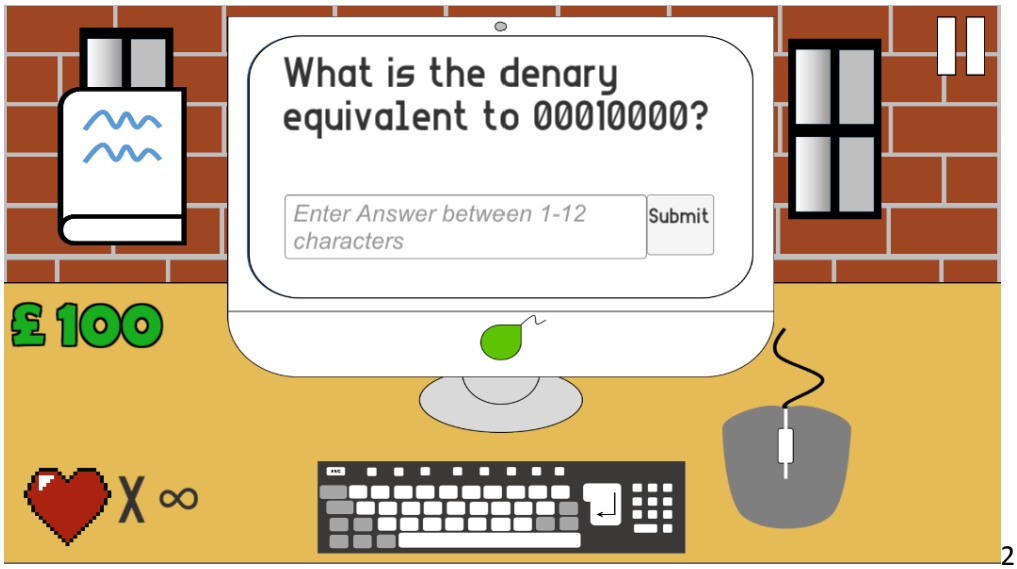
Difficulties

Before click



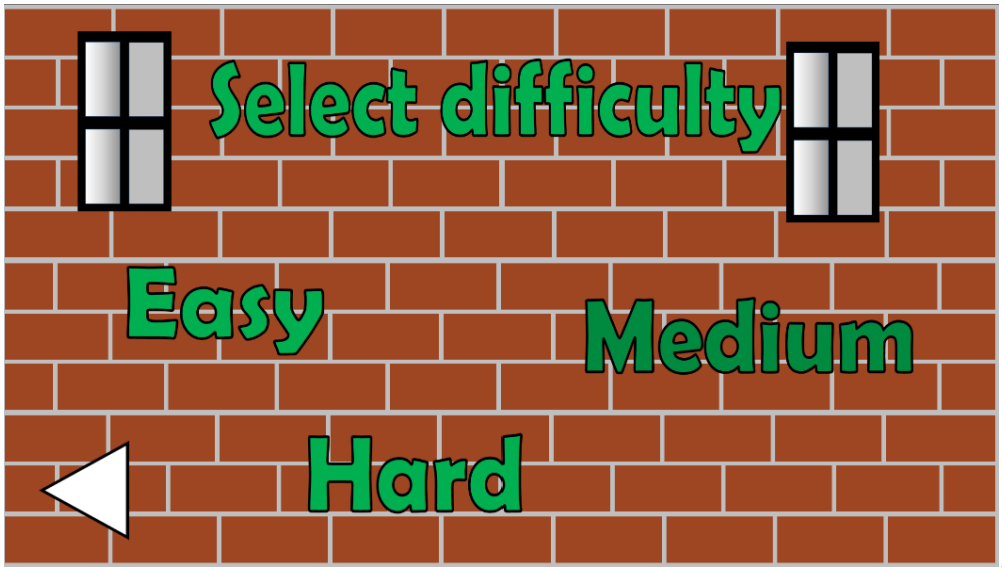
2.1.1

After click



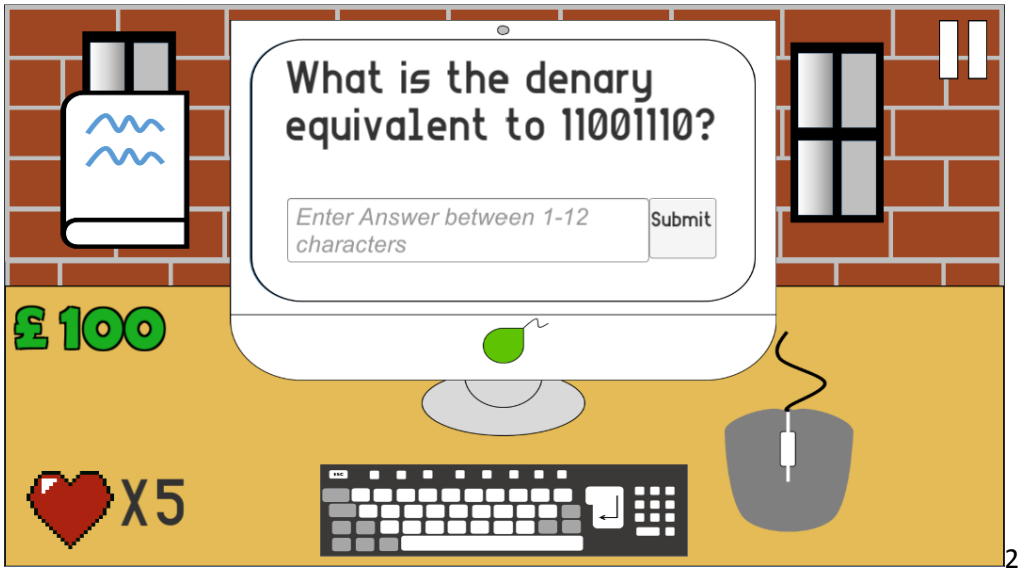
2.1.2

Before Click



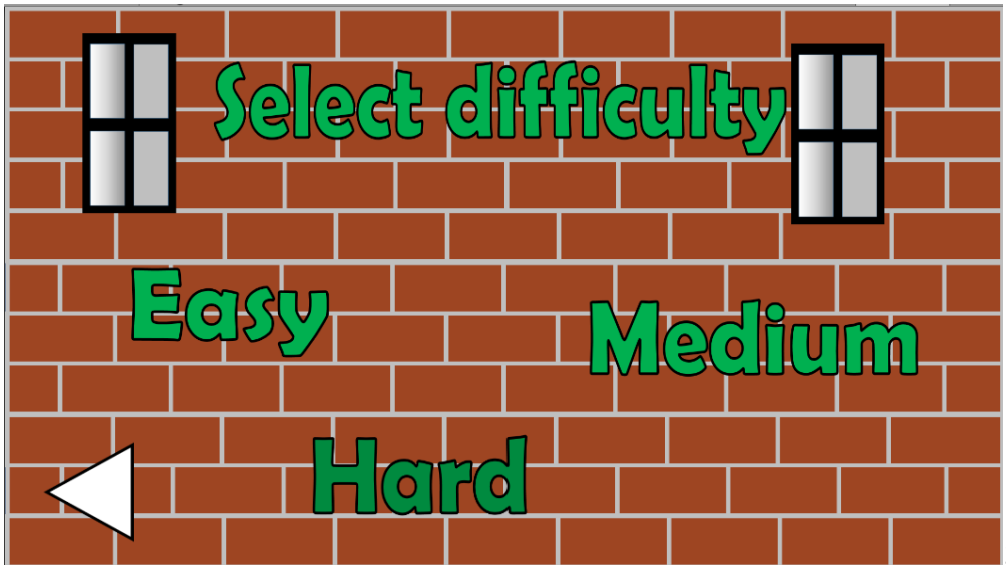
2.2.1

After click



2.2.2

Before click

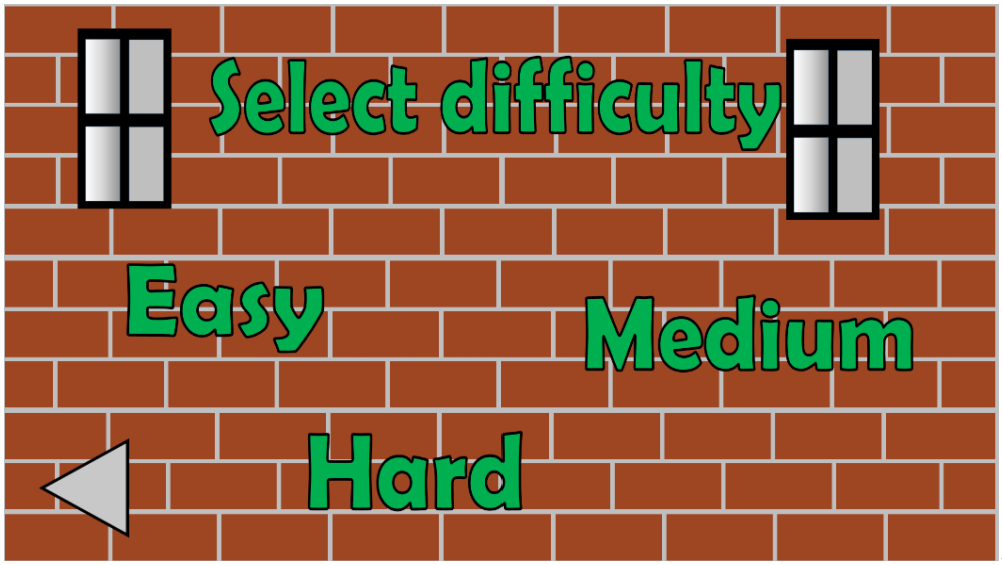


2.3.1

After click

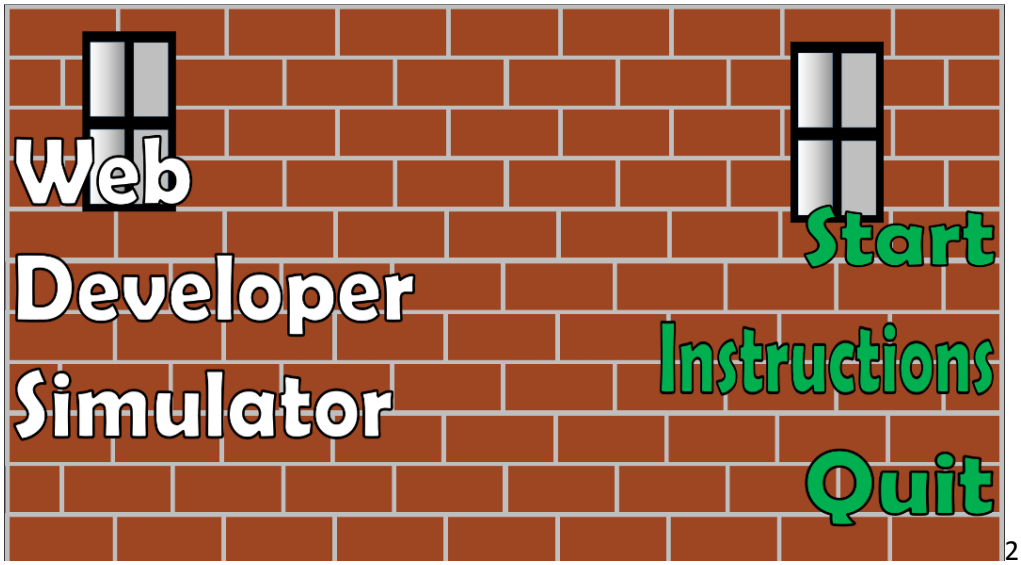
2.3.2

Before click



2.4.1

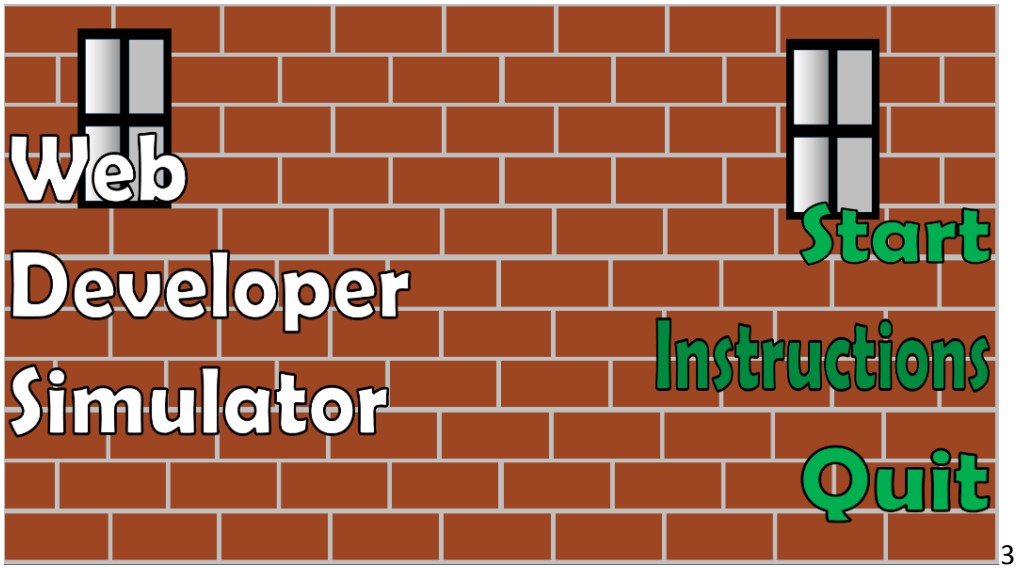
After click



2.4.2

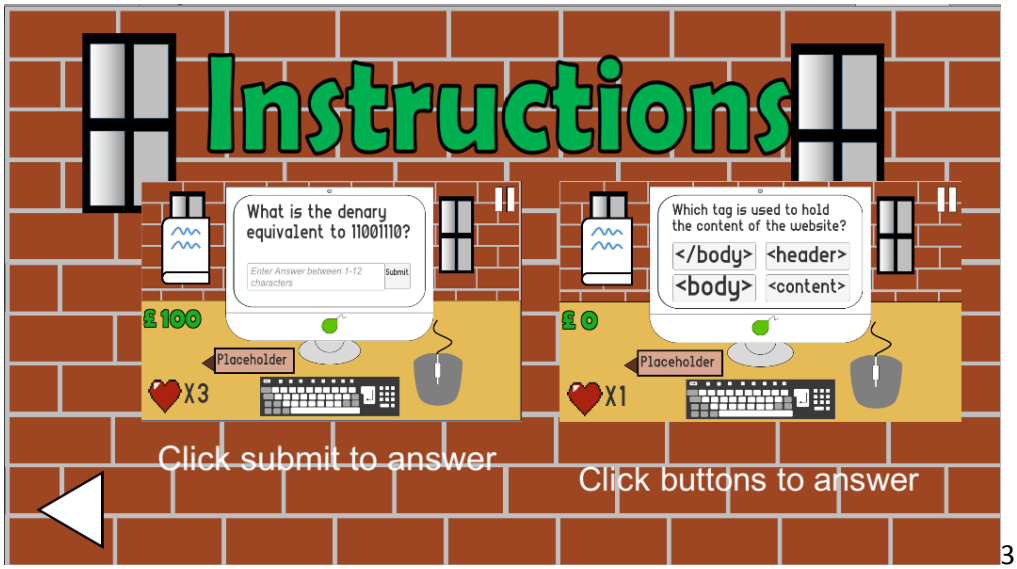
Instructions

Before click



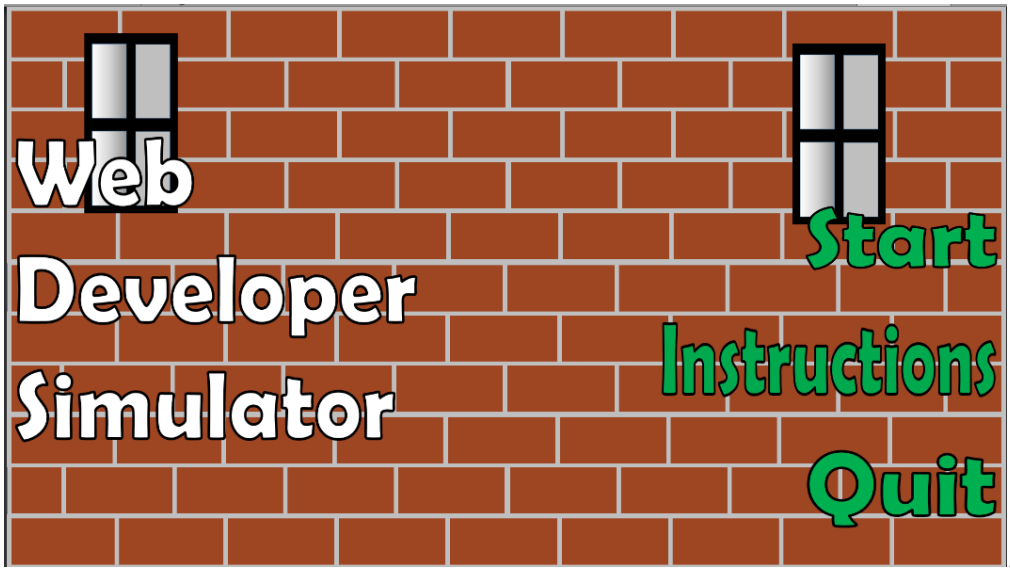
3.1.1

After click



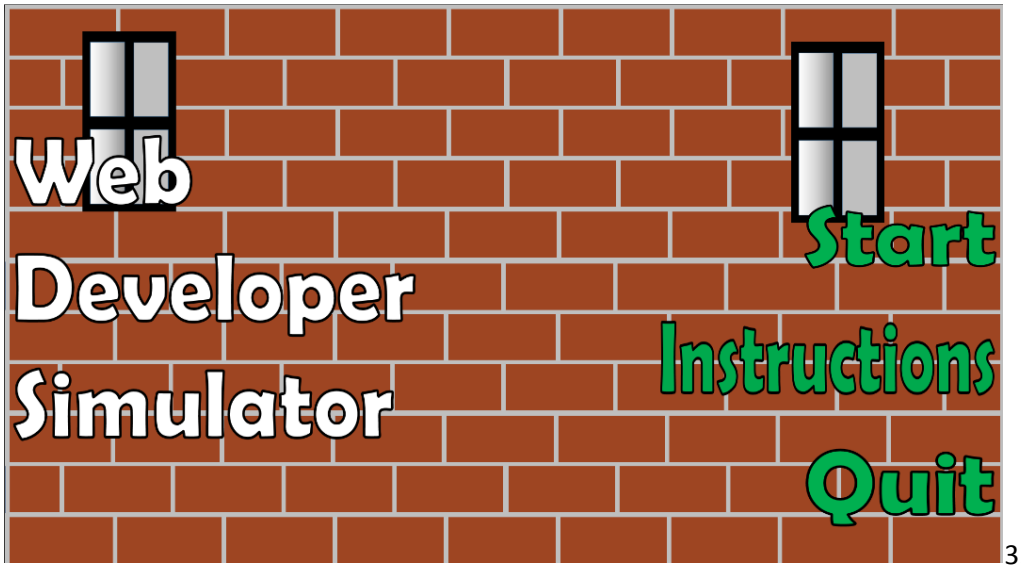
3.1.2

Before click



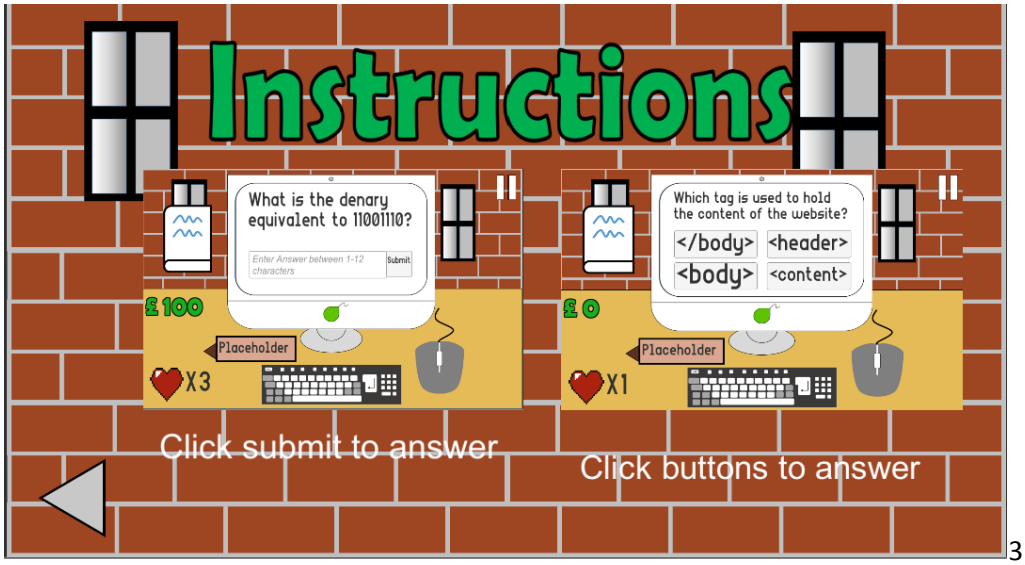
3.2.1

After click



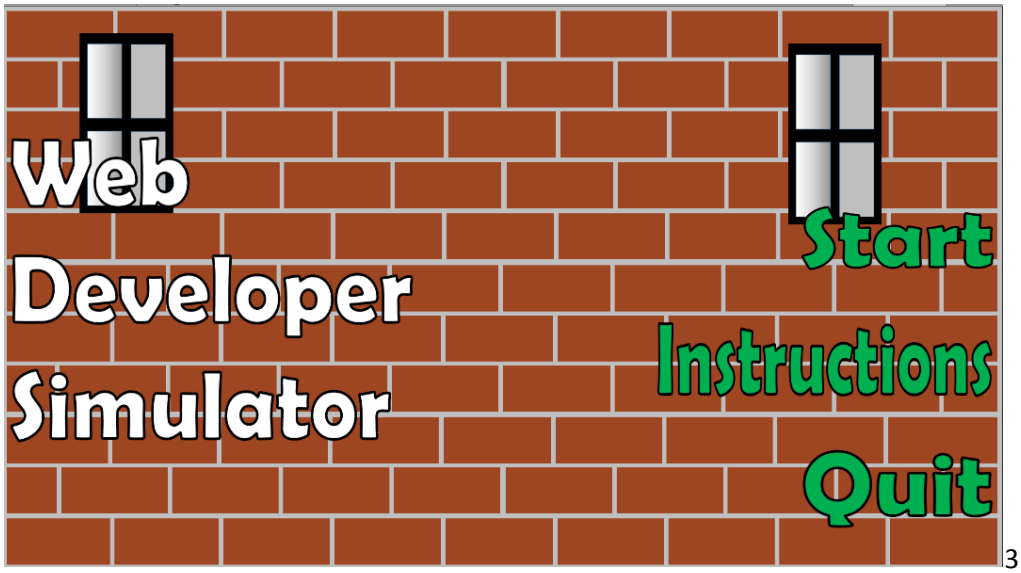
3.2.2

Before click



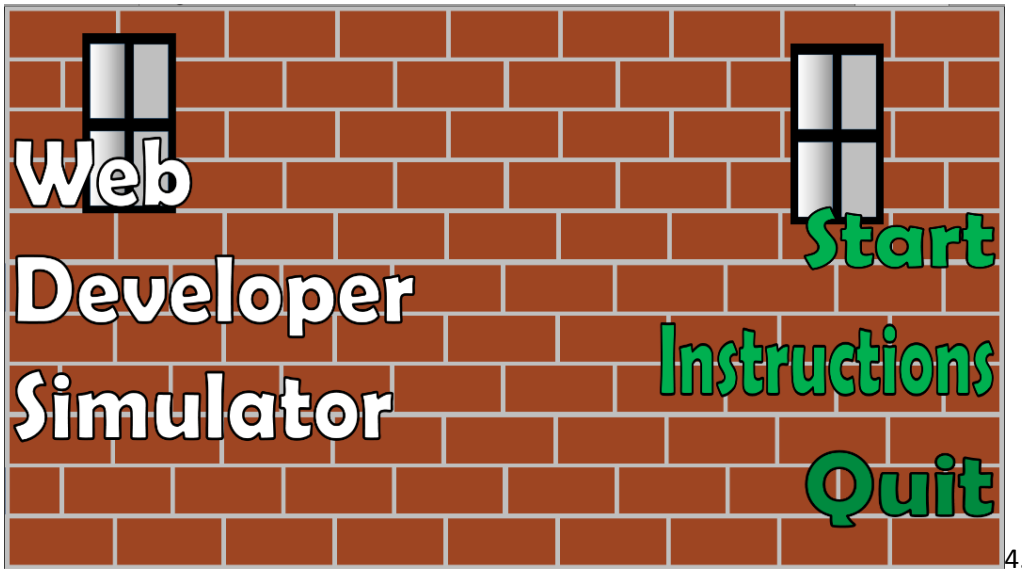
3.3.1

After click



3.3.2

Quit

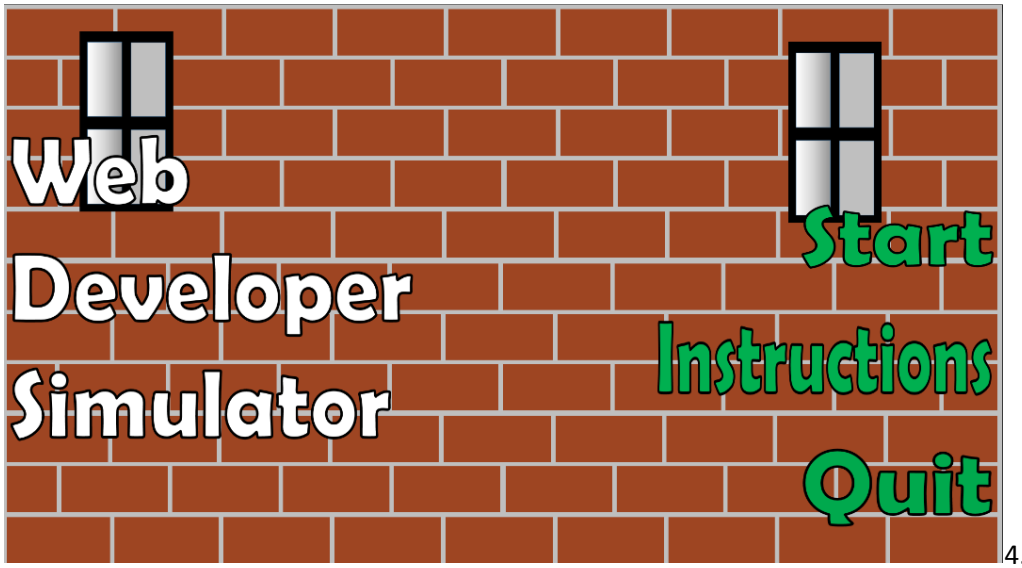


4.1.1

After click

4.1.2

Before click

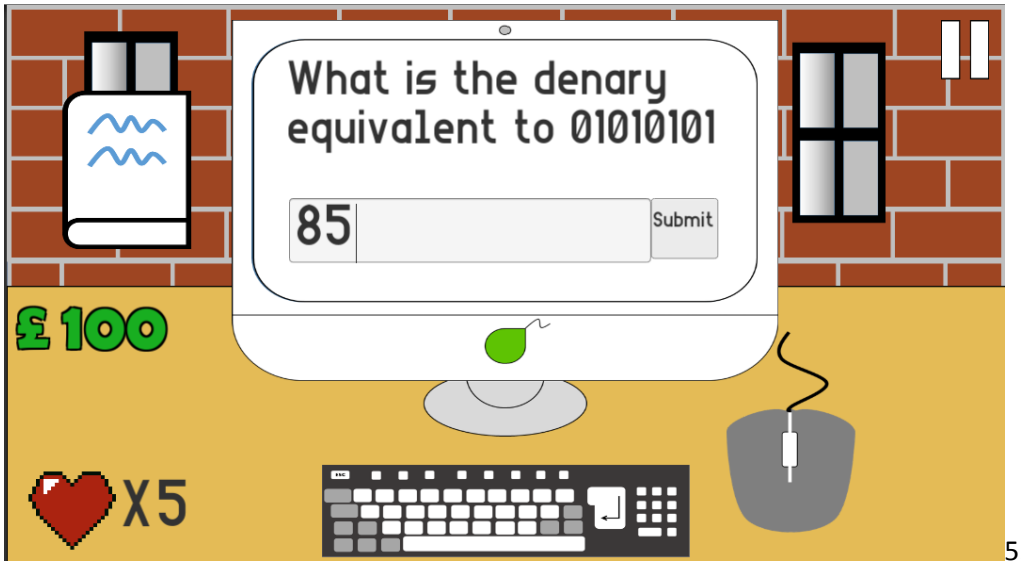


4.2.1


4.2.2

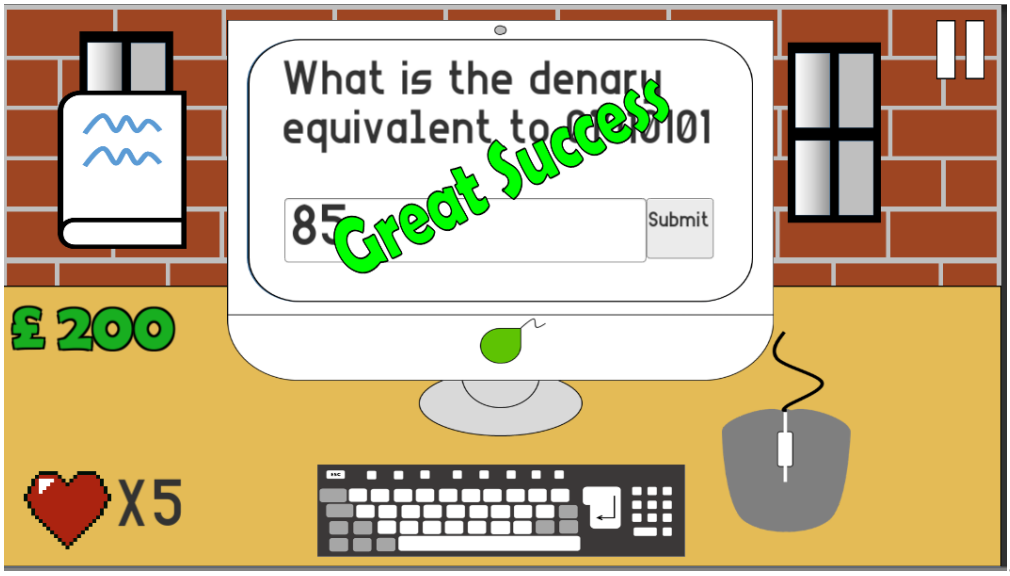
Life system:

Before click



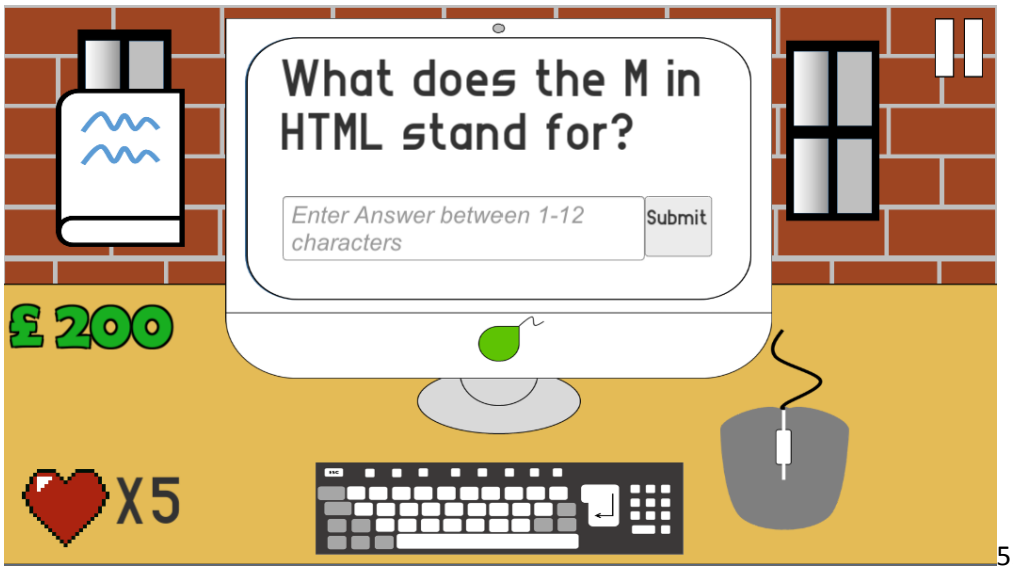
5.1.1

After click



5.1.2

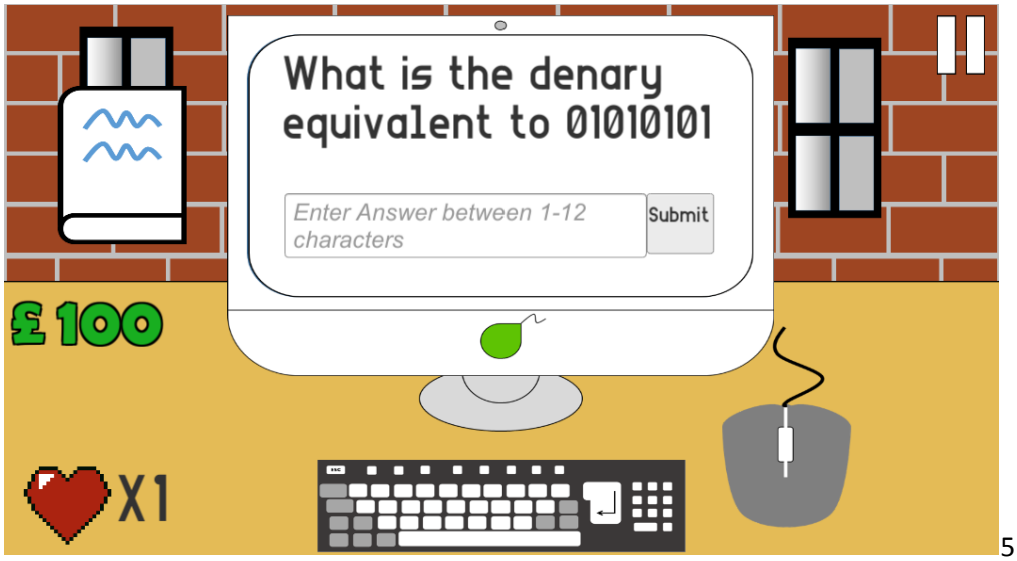
Before click



5.2.1

5.2.2

Before click:



5.4.1

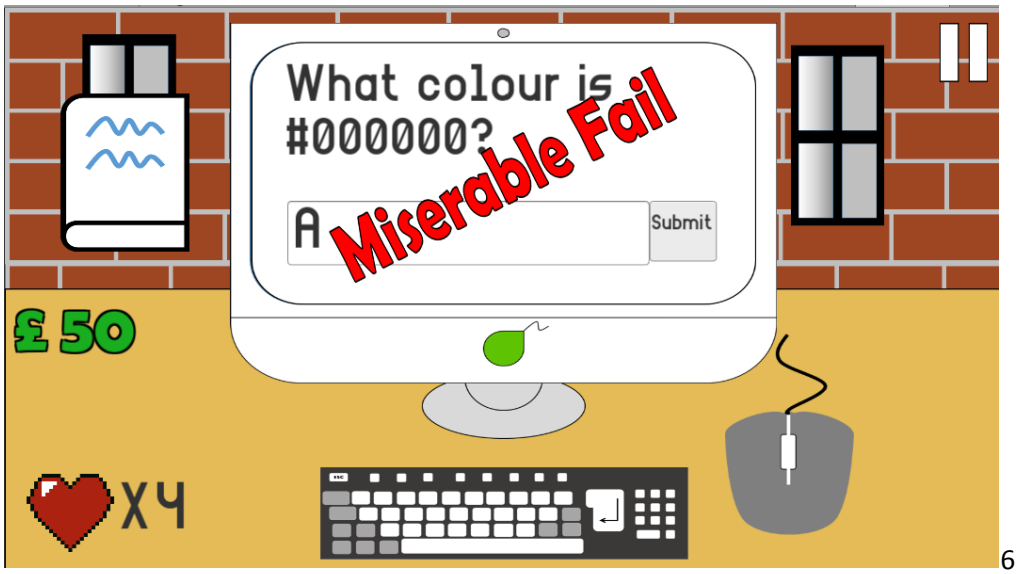
5.4.2

Fill in the blank:

Before click

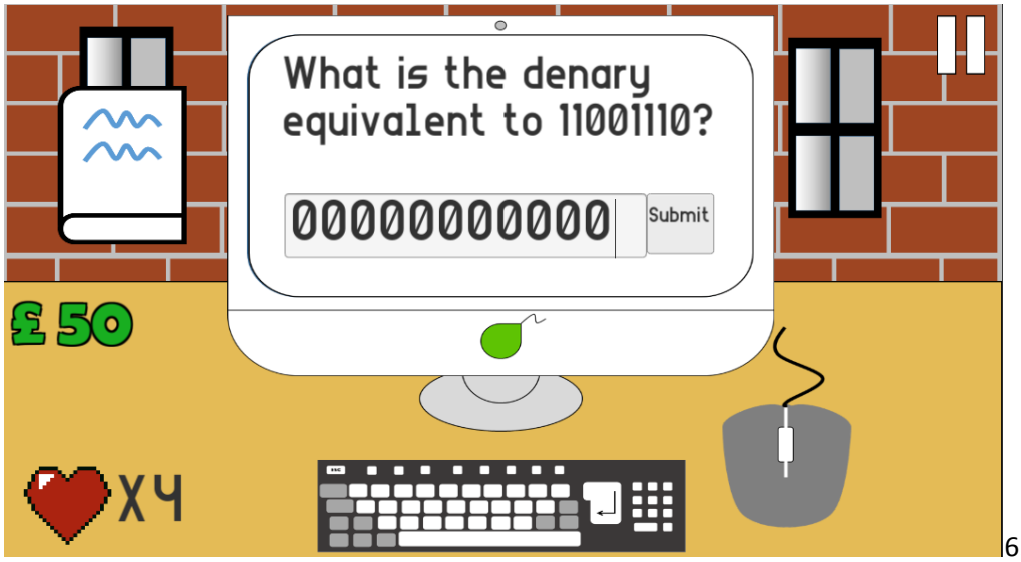
6.1.1

After click



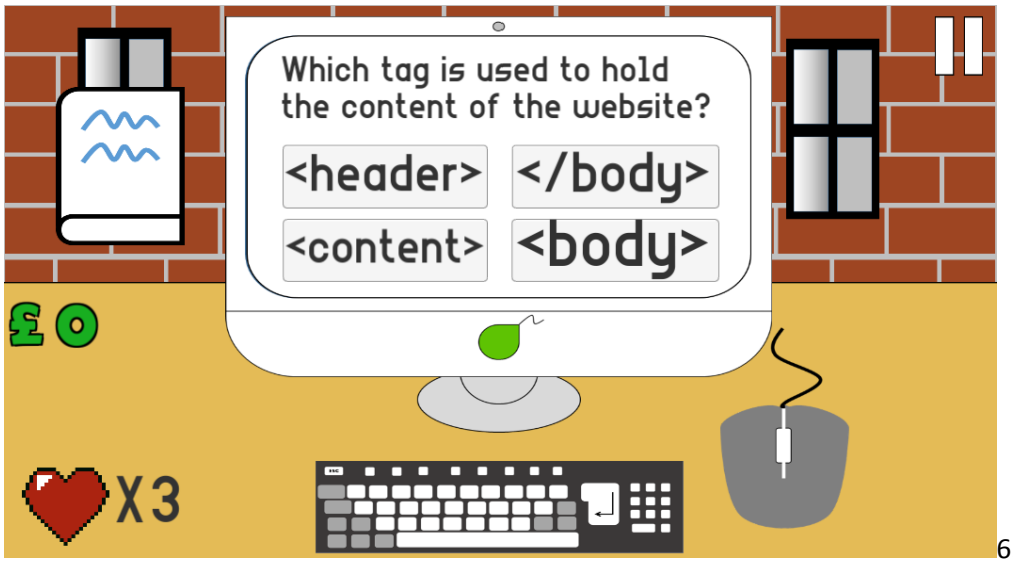
6.1.2

Before click



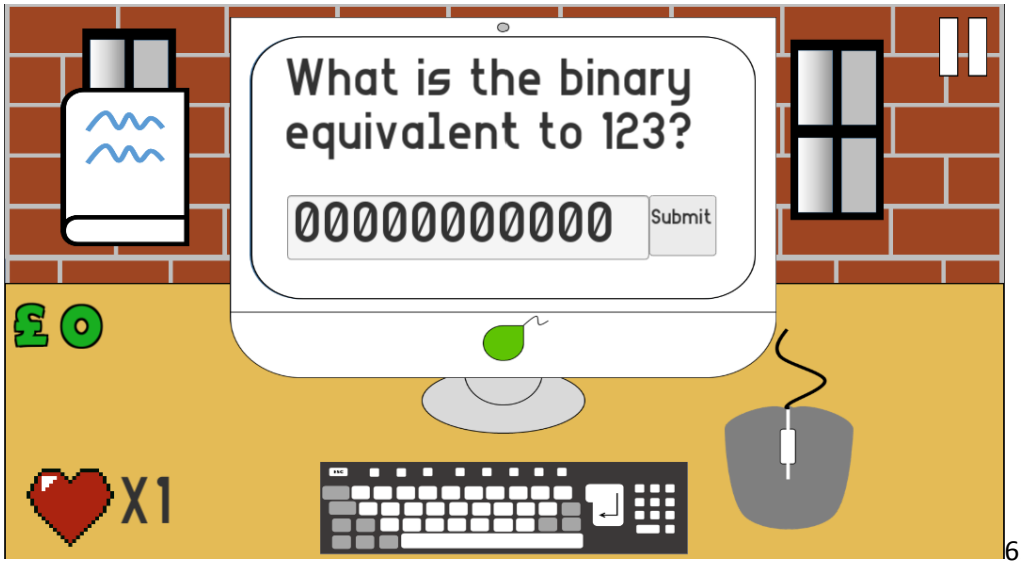
6.2.1

After click



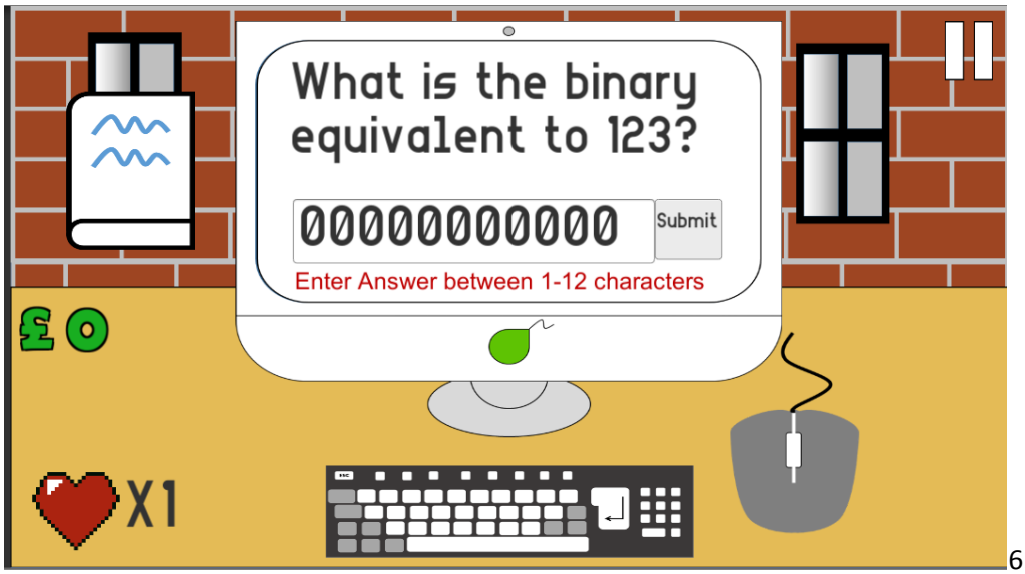
6.2.2

Before click:



6.3.1

After click



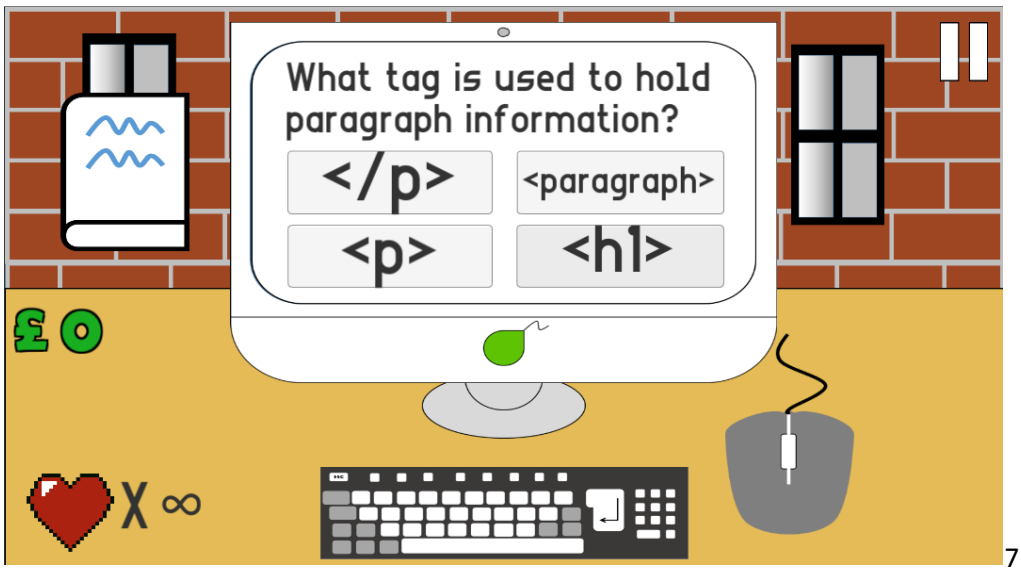
6.3.2

Multiple choice

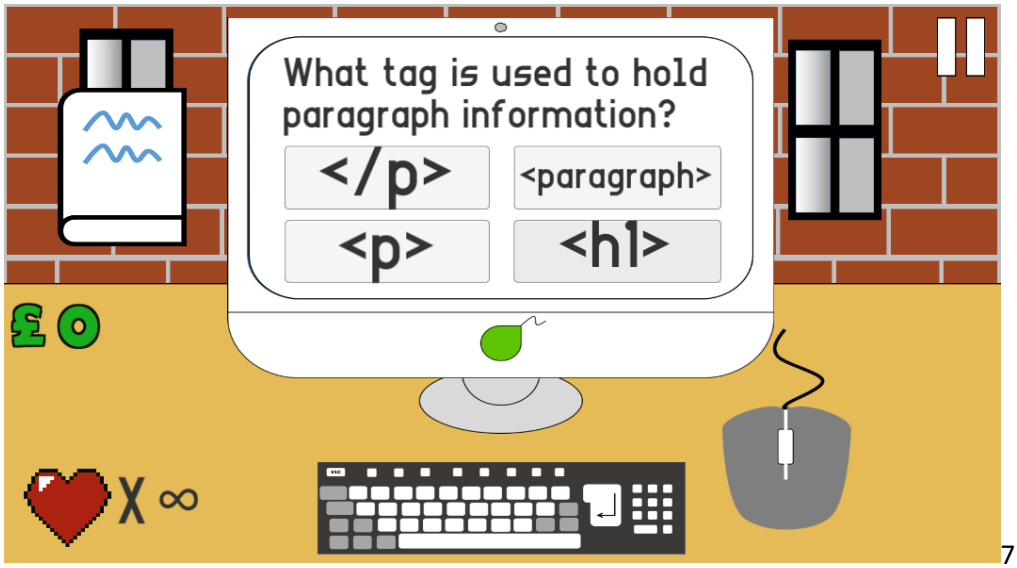
7.1.1

7.1.2

Before click



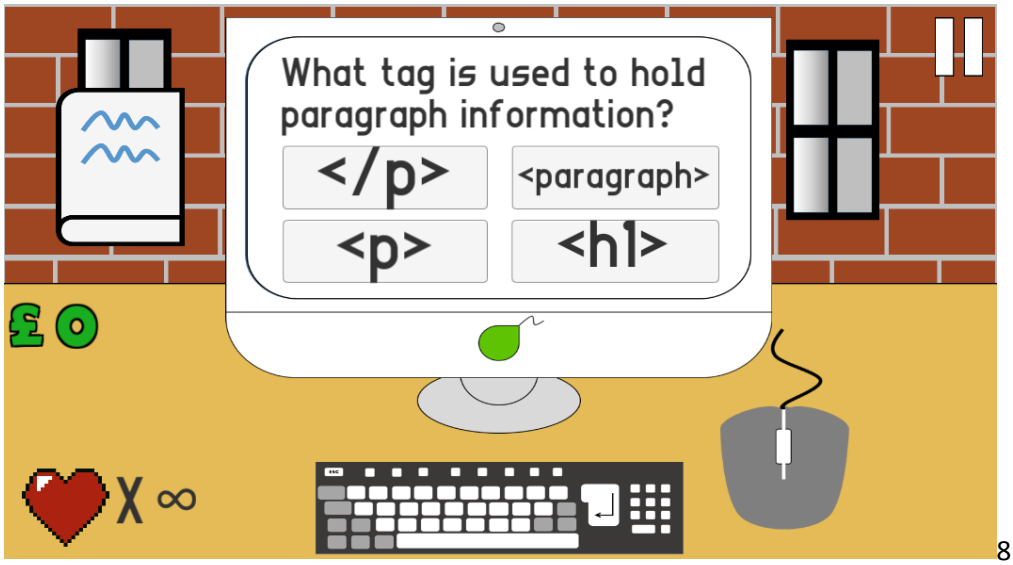
7.2.1



7.2.2

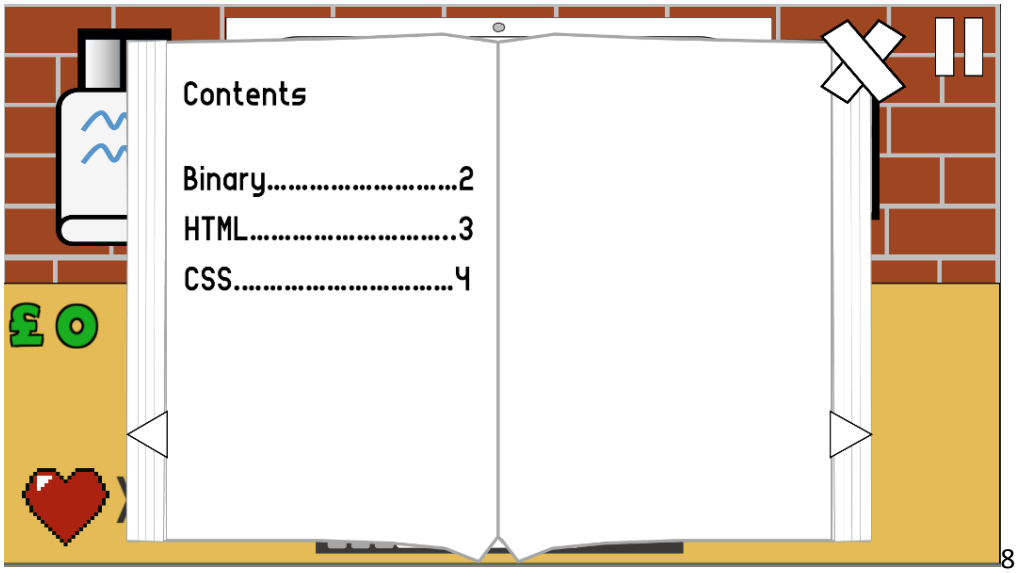
Book feature:

Before click



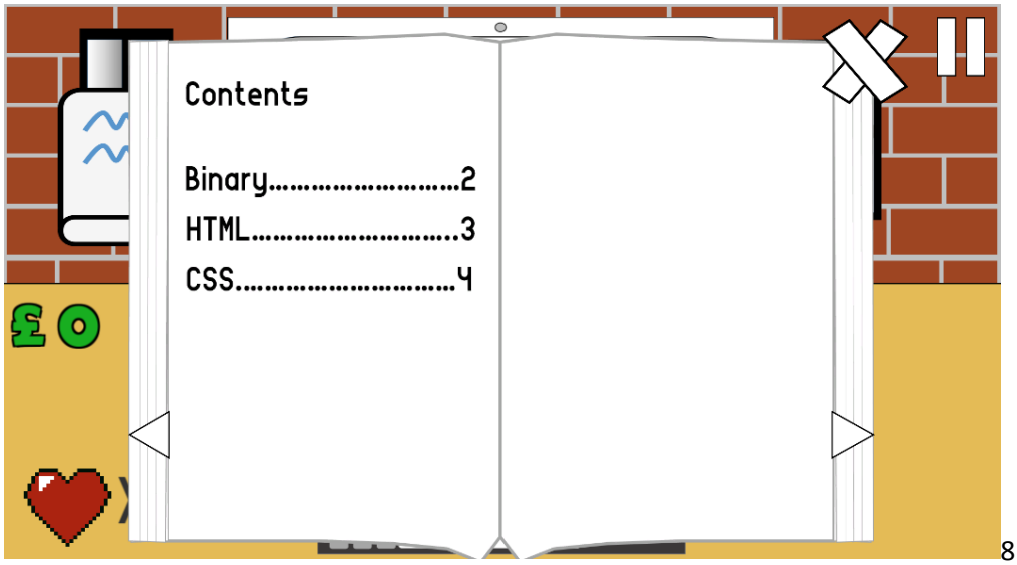
8.1.1

After click



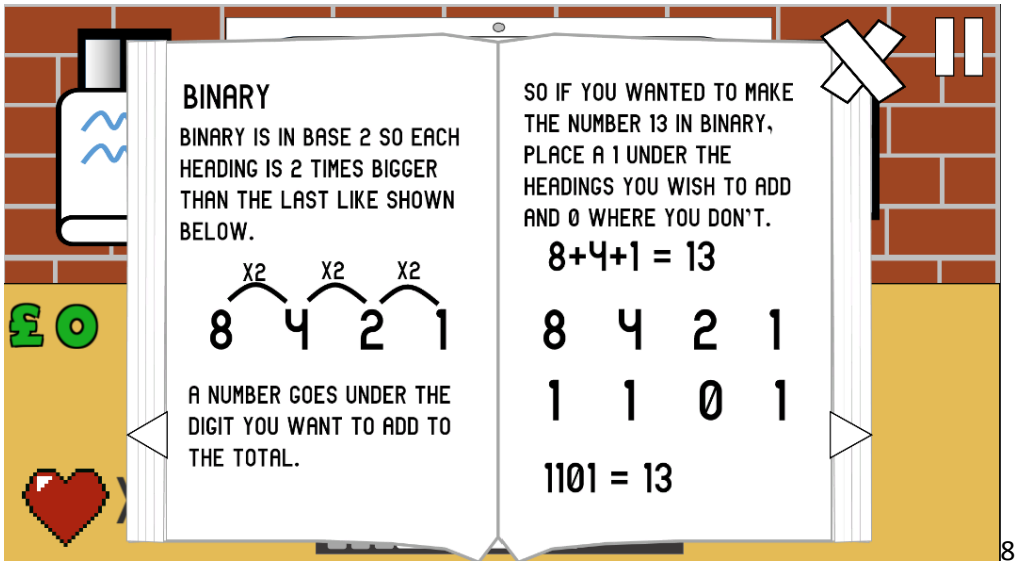
8.1.2

Before click



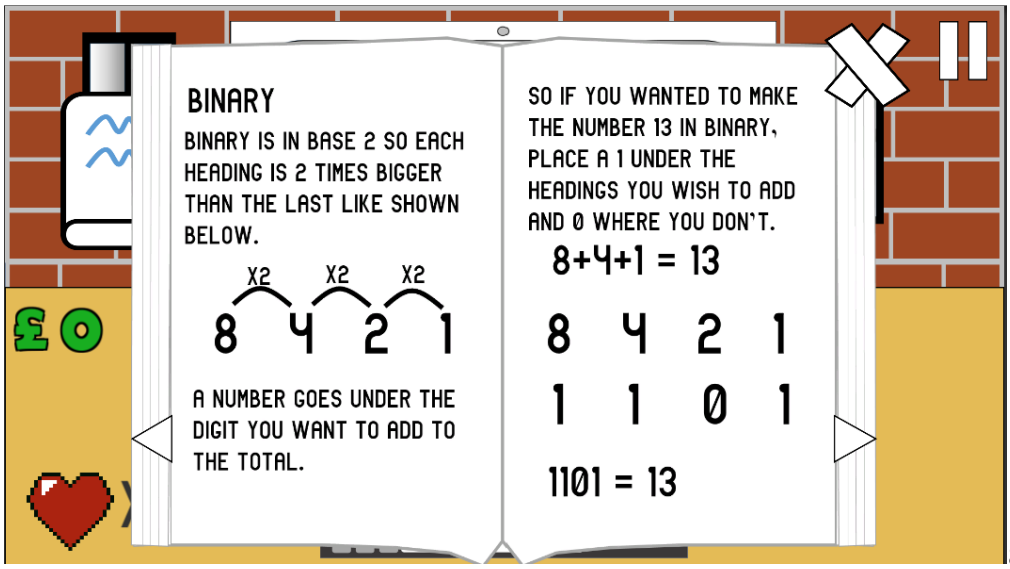
8.2.1

After click



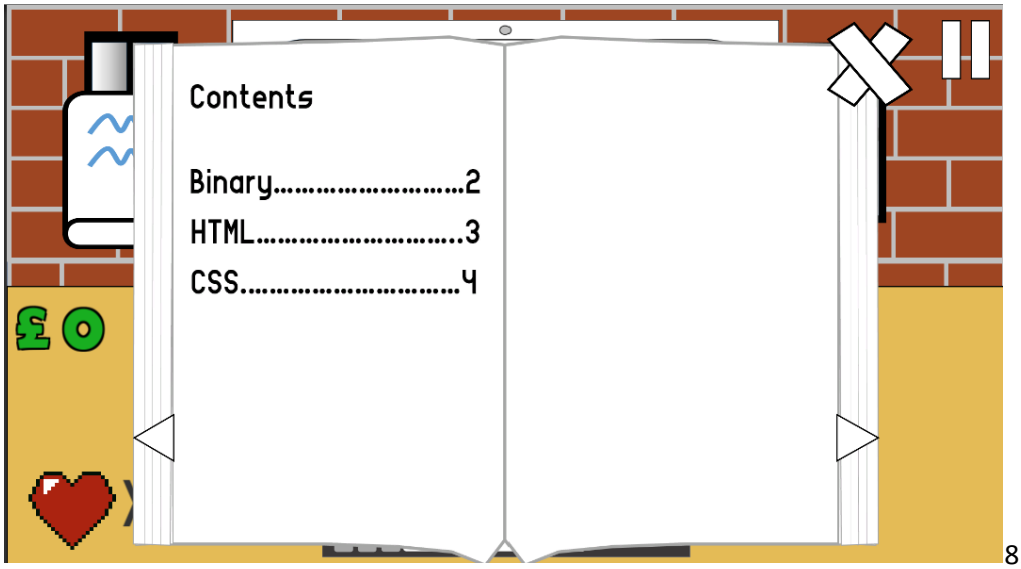
8.2.2

Before click:



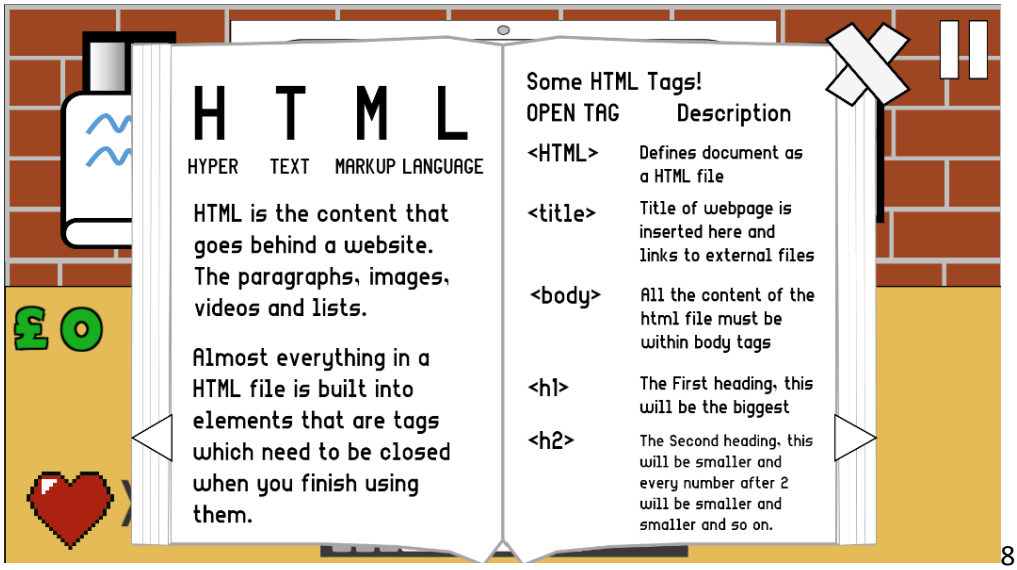
8.3.1

After click



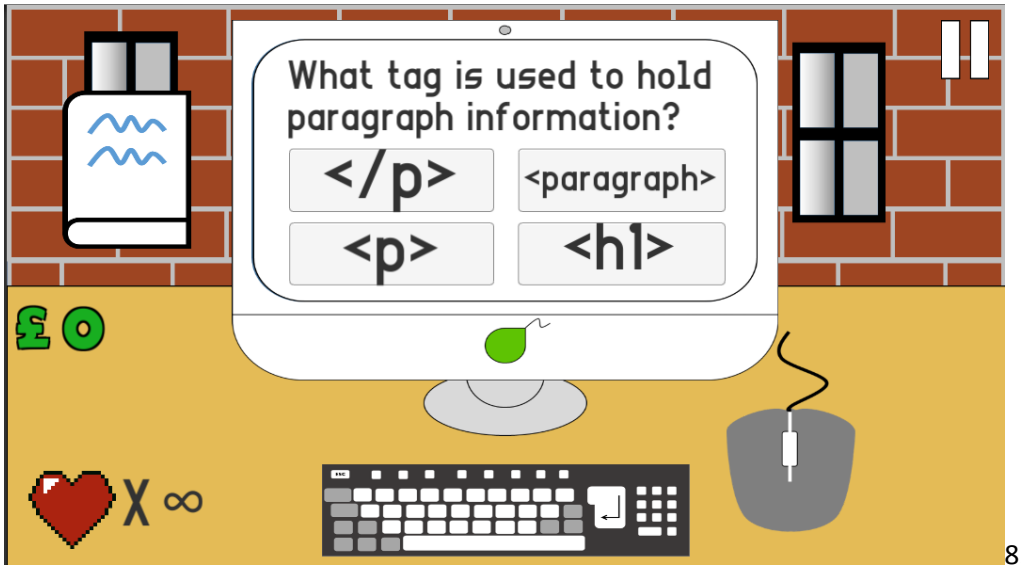
8.3.2

Before click



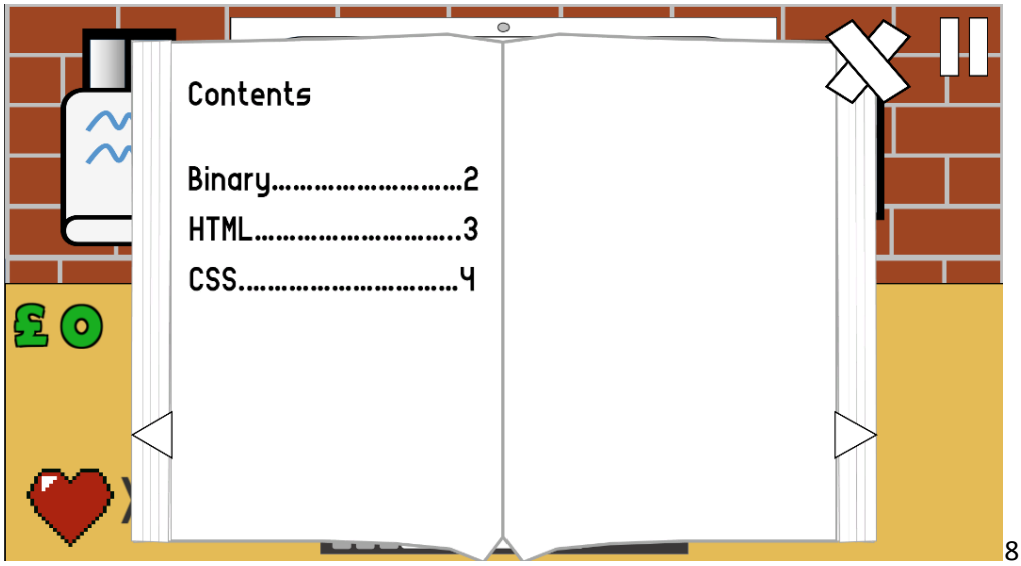
8.4.1

After click



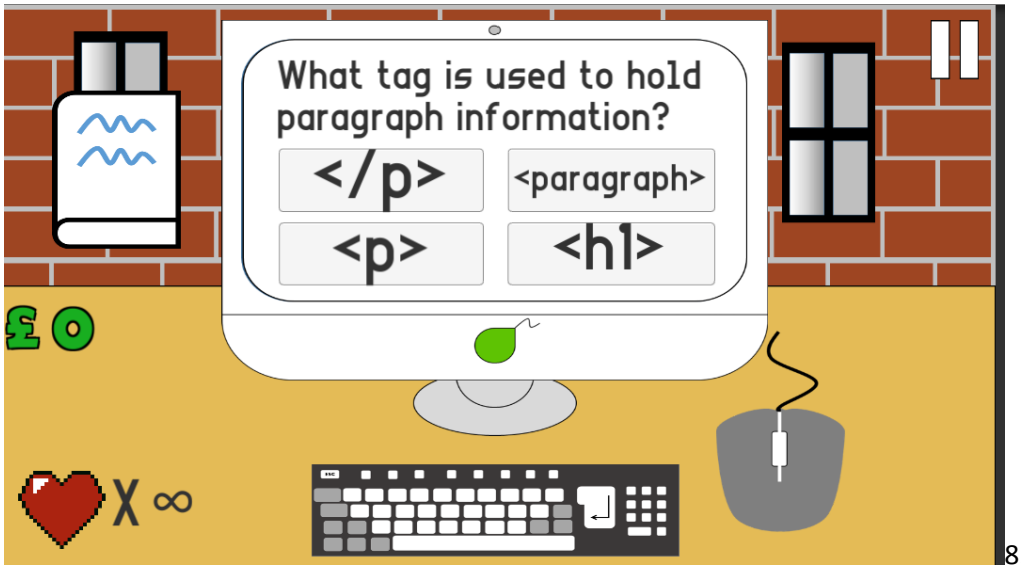
8.4.2

Again when clicking book button



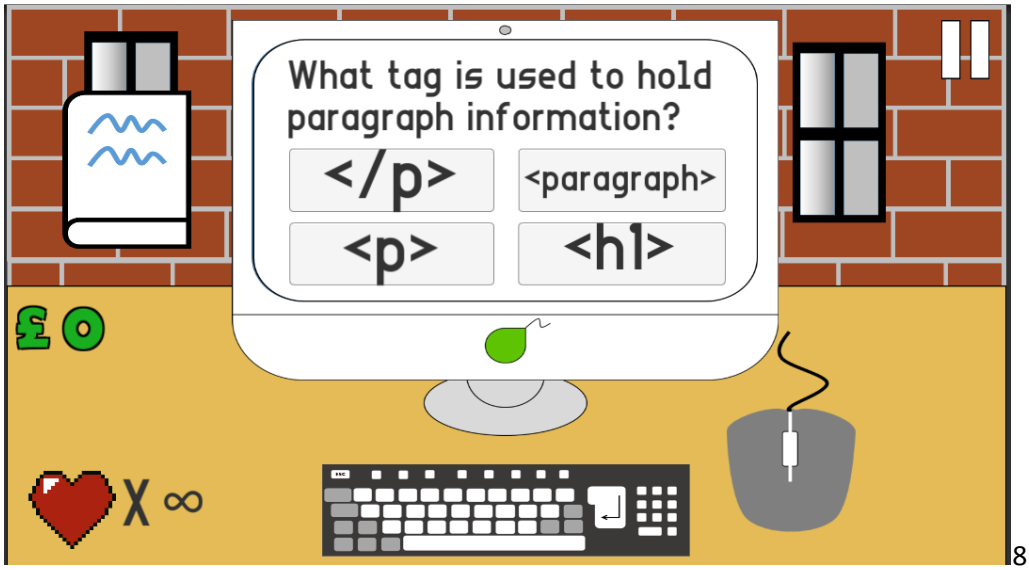
8.4.3

Before click



8.6.1

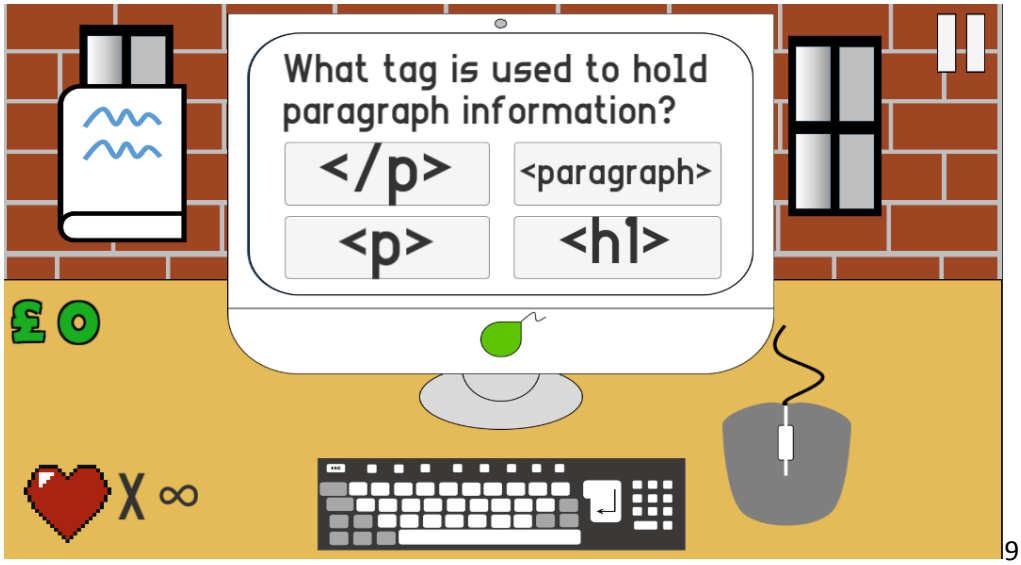
After click



8.6.2

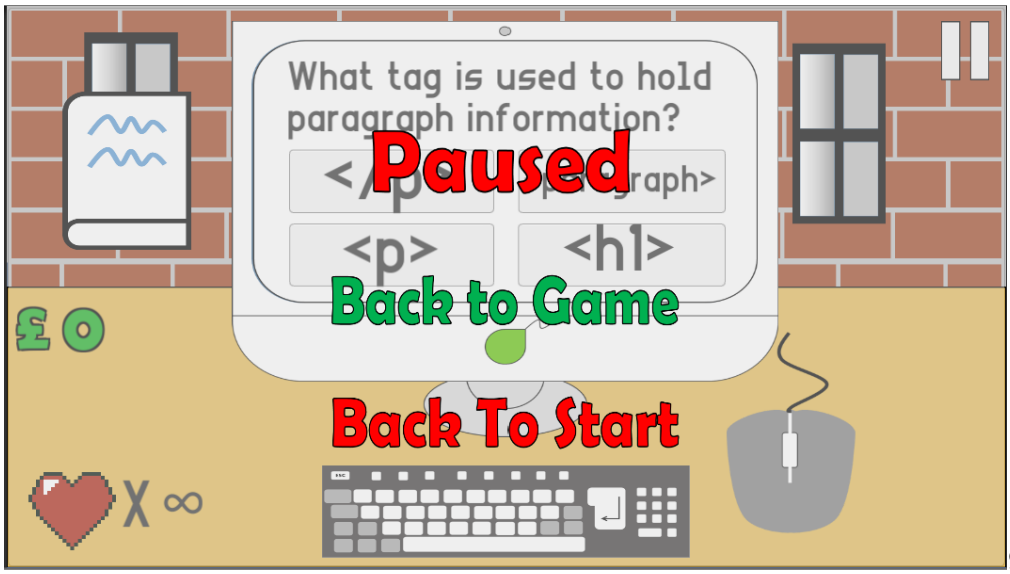
Pause Menu:

Before click



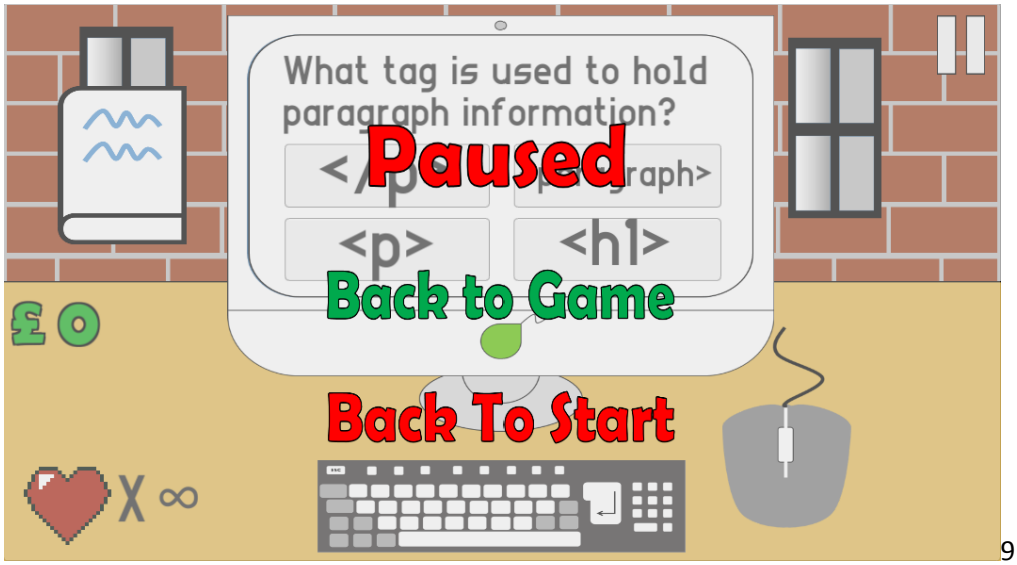
9.1.1

After click



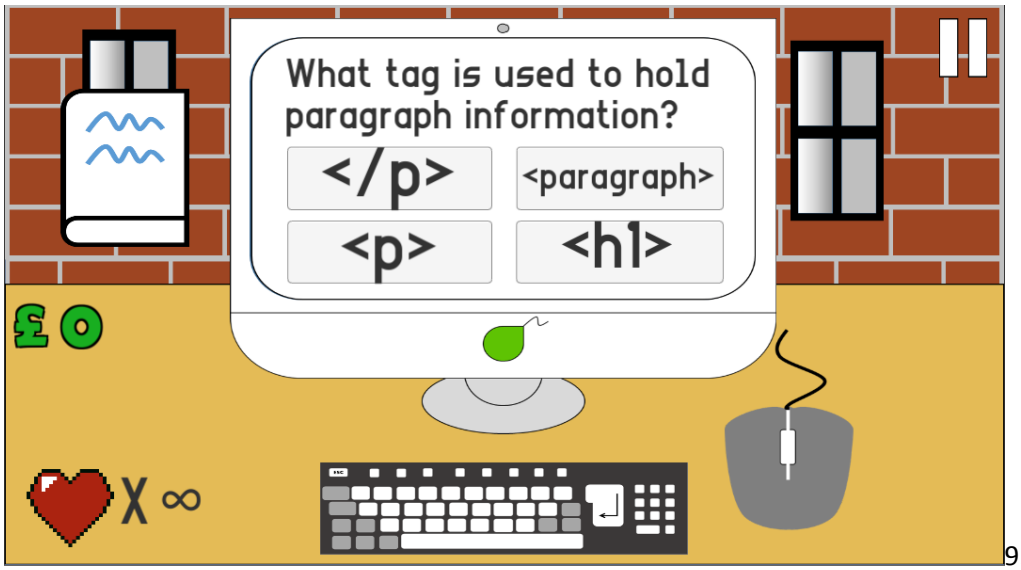
9.1.2

Before click



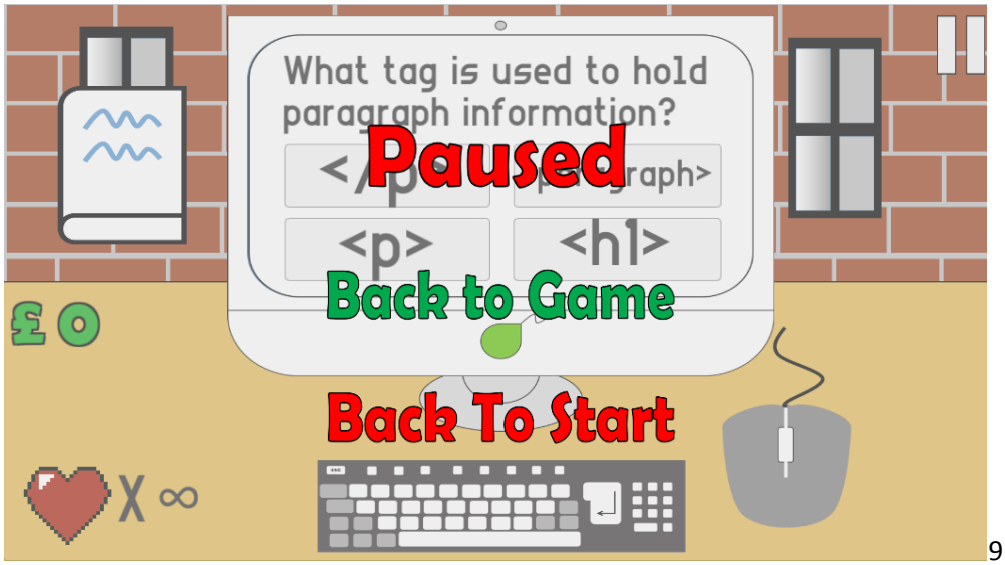
9.2.1

After click



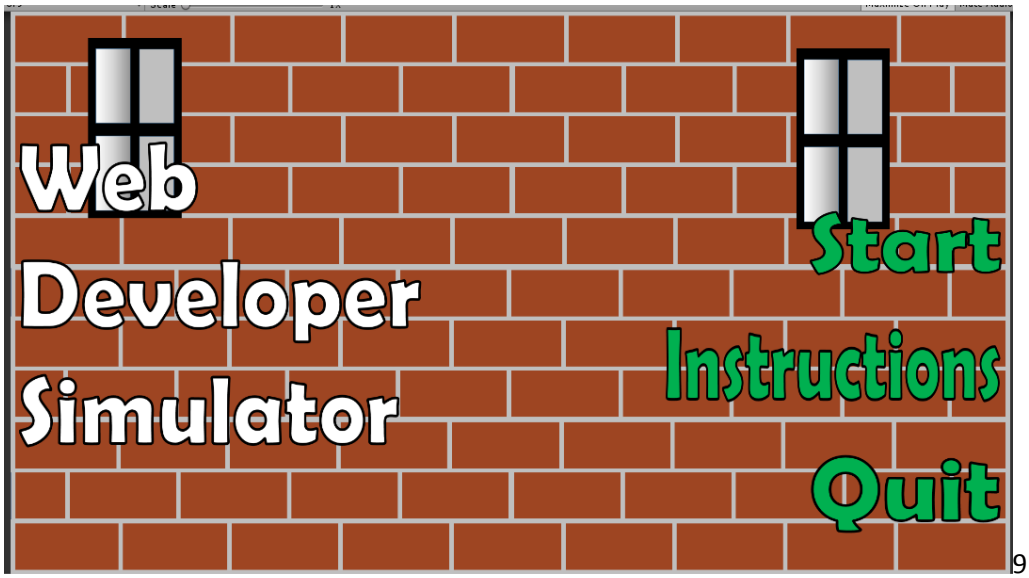
9.2.2

Before click

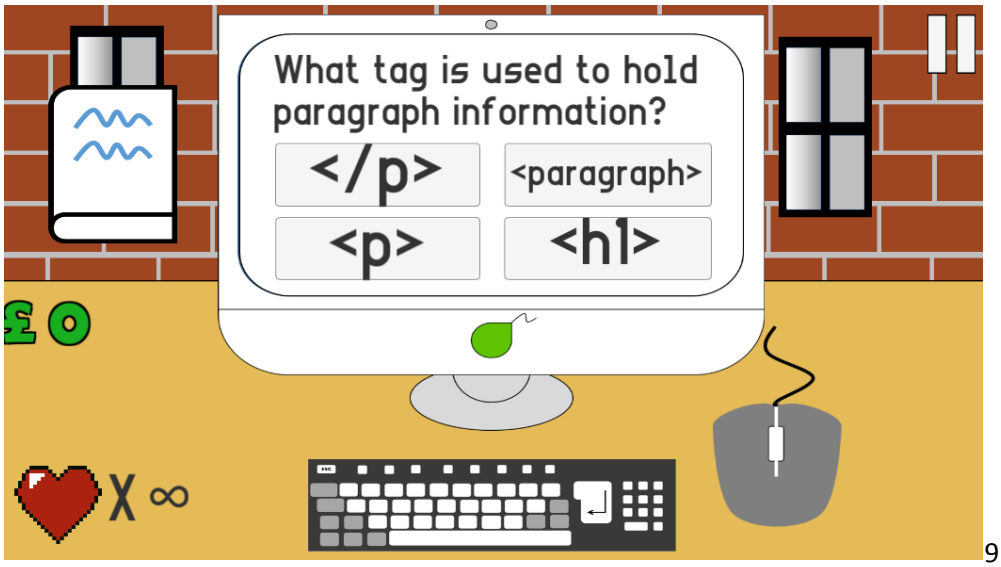


9.3.1

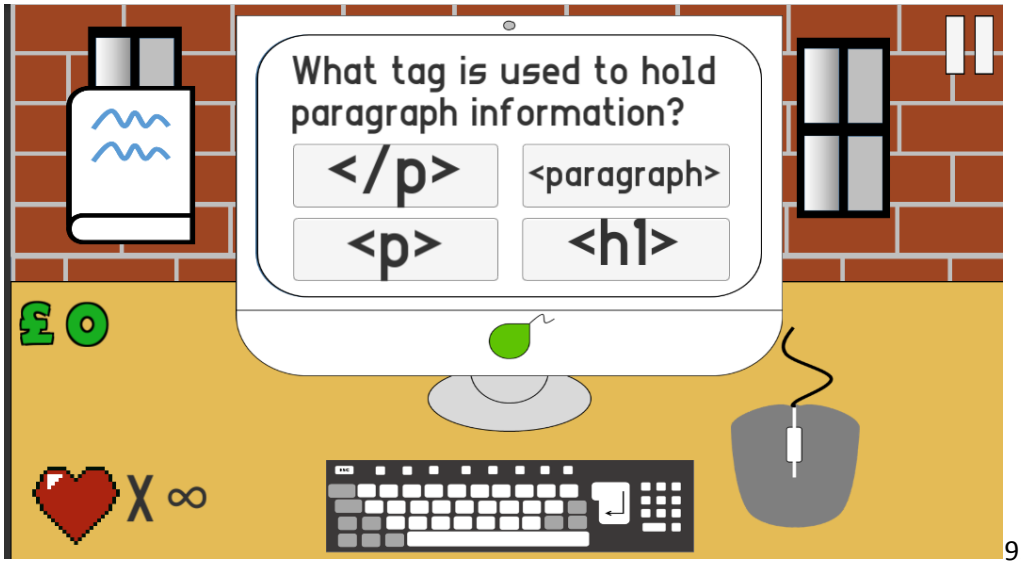
After click



9.3.2



9.4.1



9.4.2

Tests

Main Game:

Test number	What is being tested	Input data	Expected outcome	Evidence	Actual outcome	Criteria Met
1.	Aspect ratio	N/A	16:9	1.1, 1.2	16:9	Yes
2.	Difficulties	1.Choosing Easy 2.Choosing Medium 3.Choosing Hard 4. Click back on selection screen	1.Easy questions loaded 2.Medium Questions loaded 3.Hard questions loaded 4. back to main menu	1) 2.1.1 to 2.1.2 2) 2.2.1 to 2.2.2 3) 2.3.1 to 2.3.2 4) 2.4.1 to 2.4.2	1.Easy questions loaded 2.Medium Questions loaded 3.Hard questions loaded 4. back to main menu	Yes
3.	Instructions	1.LC instructions button (valid) 2.RC (invalid) 3. Go back to menu	1.Instructions load 2.Nothing 3. Go back to menu	1) 3.1.1 to 3.1.2 2)3.2.1 to 3.2.2 3)3.3.1 to 3.3.2	1.Instructions load 2.Nothing 3. Goes back to menu	Yes
4.	Quit	1.LC quit 2.RC	1. Quit is responded on console 2.Quitted	1)4.1.1 to 4.1.2 2) 4.2.1 to 4.2.2	1. Quit is responded on console 2.Quitted	Yes

Personalisation:

I have left out personalisation due to time constraints and lack of the ability to do so. For me to implement personalisation would take more time than I have to complete this project as I discussed with my stakeholder in our final meeting for development.

Music:

I am also unable to test music features as I was unable to implement them due to lack of time to learn how to implement these features. To learn how to implement these features in a sophisticated design would take far too long and may delay my project beyond the deadline as I discussed with my stakeholder.

Life system:

Test number	What is being tested	Input data	Expected outcome	Evidence	Actual outcome	Criteria Met
5	Life system 1.Money increases by 100 2. Money decreases by 50 3.Lives decrease by 1 4.Gameover at 0 lives 5.No gameover for easy mode	1.Getting answer correct 2.Getting answer wrong 3.Getting answer wrong 4.Losing all lives 5.N/A	1.Money increases by 100 2. Money decreases by 50 3.Lives decrease by 1 4.Gameover at 0 lives 5.No gameover for easy mode	1) 5.1.1 to 5.1.2 2)5.2.1 to 5.2.2 3)5.2.1 to 5.2.2 4)5.4.1 to 5.4.2 5)2.1.2	1.Money increases by 100 2. Money decreases by 50 3.Lives decrease by 1 4.Gameover at 0 lives 5.No gameover for easy mode	1.Yes 2.Yes 3.Yes 4.Yes 5.Yes

Fill in the blank:

Test number	What is being tested	Input data	Expected outcome	Evidence	Actual outcome	Criteria Met
6	Fill in the blank challenges	1.A(valid) 2.000000000000 (borderline) 3.0000000000000 (invalid)	1.Accepts answer 2.Accepts answer 3.Denies answer	1)6.1.1 to 6.1.2 2)6.2.1 to 6.2.2 3) 6.3.1 to 6.3.2	1.Accepts answer 2.Accepts answer 3.Denies answer	1.Yes 2.Yes 3.Yes

Multiple choice:

Test number	What is being tested	Input data	Expected outcome	Evidence	Actual outcome	Criteria Met
7	Multiple choice challenges	1.clicking answer 2.Rc answer(invalid)	1.Accepts answer 2.Denies answer	1)7.1.1 to 7.1.2 2)7.2.1 to 7.2.2	1.Accepts answer 2.Denies answer	Yes

Book feature:

Test number	What is being tested	Input data	Expected outcome	Evidence	Actual outcome	Criteria Met
8	Book feature	1.click book button 2. Click next page 3.Click prev page 4. Click X 5. Click back on book 6.RC on buttons (invalid)	1. Book appears 2.Next page is shown 3. Previous page is shown 4.Book closes 5.Book opens on first page 6. Null	1) 8.1.1 to 8.1.2 2)8.2.1 to 8.2.2 3)8.3.1 to 8.3.2 4)8.4.1 to 8.4.2 5) 8.4.1 to 8.4.2 to 8.4.3 6) 8.6.1 to 8.6.2	1. Book appears 2.Next page is shown 3. Previous page is shown 4.Book closes 5.Book opens on first page 6. Null	Yes

Pause menu:

Test number	What is being tested	Input data	Expected outcome	Evidence	Actual outcome	Criteria Met
9	Pause Menu	1.click pause 2. click back to game 3. Click Back to main menu 4. RC buttons	1. Pause menu opens 2. Pause menu closes 3.Takes player to main menu 4. Null	1) 9.1.1 to 9.1.2 2)9.2.1 to 9.2.2 3)9.3.1 to 9.3.2 4)9.4.1 to 9.4.2	1. Pause menu opens 2. Pause menu closes 3.Takes player to main menu 4. Null	Yes

Evaluation:

Testing Success Criteria:

Success Criteria	Justification	Evidence	Criteria met
The game will be created in the 16:9 aspect ratio mode.	This is the most common aspect ratio to use and working with one aspect ratio makes it easier for me to focus on the more important aspects of my game.	Test number 1	Success
Must allow User to choose from different difficulty levels. More specifically, the game must have 3 difficulties. Easy, Medium and Hard	Many children will start on a different level to others. This also allows the player to learn at their own pace.	Test Number 2	Success
Must have a form of personalisation e.g. Game must allow player to create a basic character	From my interview and observation process, it was clear that personalisation in my game would keep children interested in the game as children like to personalise characters making them more emerged in the experience.	N/A	Did not meet
The personalisation should have at least 5 hair variants	I think 5 hair variants gives my user plenty of options to choose from that could appeal to all genders and gives enough opportunity to show I can create a personalisation feature in my game/simulator.	N/A	Did not meet
The personalisation should have 3 or more hair colour tones	I think 3 hair colour tones gives my user plenty of options to choose from that could appeal to my stakeholders for basic customisation. If I have more time I may add more	N/A	Did not meet

	hair colour tones.		
The personalisation should have 5 or more skin tone options	5 skin tones will allow me to generalise the spectrum of skin colour into enough different variants that my players can choose from.	N/A	Did not meet
The personalisation should have 3 or more body shapes	This will give my player the choice to change what shape their body is.	N/A	Did not meet
The personalisation should have 3 or more body colours	This is to further incentivise younger children to play as it add more imagination and personalisation options to my game.	N/A	Did not meet
Must have a progression system/scheme	Progression is a very crucial feature in a game as it leads the player to believe they are working towards something. I will be using money as a form of progression as my game is an abstraction of a web developers life and in the real world, wealth can show progression of sorts.	Test number 5.1, 5.2,5.3,5.4	Success
If a player gets a question correct, their “balance” should increment by £100	This is an abstraction of getting paid in the real world is like. If a real web developer does their job correctly, they get paid and so if the player completes a challenge then they too shall be paid in the form of in game currency.	Test number 5.1	Success
If a player gets a question wrong, their “Balance” must decrement by £50	This is an abstraction of consequences in my game of what may happen to a web developer if they create an error in code for the company they work for.	Test number 5.2	Success

Must have background music	Background Music will not only keep the player interested but may also increase their levels of concentration. My interviewee told me that it is important that a game has background music especially to keep the children wanting to play the game.	N/A	Failed to meet
The book feature must be working and provide useful tips that would help the player in the challenges	The book feature is one of the main features I am adding in the game as it is an abstraction of researching that web developers may do to learn how to code a certain feature into their website.	Test number 8	Success
The book feature must have 2 buttons that allow the player to change pages back and forth to their liking.	This is a simple design that will allow my user to easily skim over the book to find the information they are looking for demonstrating the computational method of abstraction.	Test number 8.2, 8.3	Success
Game must adapt challenges to the selected difficulties	From my interview process, it was clear that people from the variety of age range that my stakeholders are, that different difficulties are a requirement as children will all start from different stages. Some more gifted than others. Thus the game must have challenges of different difficulties to care to that.	Test number 2	Success
Game must allow user to pause and have a pause menu	As my stakeholders are children between 9-14 years old, there may be a need to pause the game for example to go answer	Test number 9	Success

	the door. This will also allow the user to pause the game from timed challenges on harder difficulties.		
Game must deplete lives of user if they get question wrong on harder difficulties	Should the player choose a more difficult level, my game will become harder with the risk of losing from getting too many wrong in a row.	Test number 5.3	Success
Game must allow user to attempt challenges infinitely on easy difficulties	Should the player choose an easier difficulty, they will be given infinite chances to answer as they are more likely to be younger	Test number 5.5 and development show that when user selects easy, infinite lives is displayed	Success
Medium level must have a 5 lives to begin with.	To make the game more challenging for my player, giving them 5 lives as opposed to the infinite amount easy mode has means that the player is more cautious about the mistakes they make and those who like the challenge will find the game more enjoyable.	Test number 2.2 shows 5 lives have been loaded	Success
Hard Level must have 3 lives to begin with.	This makes the level more difficult than medium mode and will make it more challenging specifically for the older age range of stakeholders. As someone apart of that age range, I think that 3 lives is enough to be challenging on players/users but is fair enough based on the difficulty of the questions.	Test number 2.3 shows 3 lives are loaded	Success
Game must have working multiple choice challenges	This means that the player should be able to choose and submit an answer with	Test number 7	Success

	the game responding accordingly. This is a good choice for challenges as my stakeholders are quite young and I do not expect them to be able to code without aid.		
Game must have working input fields to fill in the blank challenges	Very similar to the multiple choice; This means that the player should be able to choose and submit an answer with the game responding accordingly. This is a good choice for challenges as my stakeholders are quite young and I do not expect them to be able to code without aid.	Test number 6	Success
Game will teach syntax of HTML and CSS	The game is an abstraction of a web developer's job with which they work with HTML and CSS. They also work with JavaScript but I believe my stakeholders will be unable to grasp the concept of JSS and thus I will not be teaching them this syntax.	Test number 8	Success
Game must have working Start screen allowing user to begin or quit	This is a requirement in most games to terminate the game itself if in full screen mode	Test number 2, 3 and 4	Success
Game must send player to start screen if they run out of lives	This is to stop the player progressing for making too many mistakes. By doing this for the more harder levels, it makes the game more difficult and more satisfying for completing challenges without failing	Test number 5.4	Success
Game must allow user to write their name and display their name in-	This is to add more and more personalisation features and engage my	N/A	Not met

game	stakeholders into the game more.		
The game must validate the length of the players name so that it only allows a name no longer than 12 letters.	The average US first name is 6 letters long so my game will allow players 12 characters to write names so that it is short enough to fit in the screen but also long enough to allow any longer names if needed.	N/A	Not met
To validate the input fields, I will allow my user to enter no more than 12 characters of the English character set	This is because there will be no need for my user to enter values or answers of over 12 characters but also will require my user to use certain character that are not alphanumeric such as “<” and “>” to answer specific questions.	Test number 6.2 and 6.3	Success
Instructions must use screenshots of the actual design of the game.	This means that the instructions will be clearer and correlate to the final design of the game.	Test number 3	Success
Back button on difficulty selection screen allows user to return to title menu	My stakeholder M.Miah believed this would be a useful feature in case the user wanted to review the instructions once more before playing.	Test number 2.4	Success

Core game

Success Criteria	Justification
The game will be created in the 16:9 aspect ratio mode.	This is the most common aspect ratio to use and working with one aspect ratio makes it easier for me to focus on the more important aspects of my game.

My stakeholder agreed that this was fully met as I developed the game in a 16:9 ratio for a 16:9 game.

Main menu

Success criteria	Justification

Game must have working Start screen allowing user to begin or quit	This is a requirement in most games to terminate the game itself if in full screen mode
Instructions must use screenshots of the actual design of the game.	This means that the instructions will be clearer and correlate to the final design of the game.

I have evidently met this criteria completely as my stakeholder agreed with me in the development process.

Difficulty

Success Criteria	Justification
Must allow User to choose from different difficulty levels. More specifically, the game must have 3 difficulties. Easy, Medium and Hard	Many children will start on a different level to others. This also allows the player to learn at their own pace.
Game must adapt challenges to the selected difficulties	From my interview process, it was clear that people from the variety of age range that my stakeholders are, that different difficulties are a requirement as children will all start from different stages. Some more gifted than others. Thus the game must have challenges of different difficulties to care to that.
Game must deplete lives of user if they get question wrong on harder difficulties	Should the player choose a more difficult level, my game will become harder with the risk of losing from getting too many wrong in a row.
Game must allow user to attempt challenges infinitely on easy difficulties	Should the player choose an easier difficulty, they will be given infinite chances to answer as they are more likely to be younger
Medium level must have a 5 lives to begin with.	To make the game more challenging for my player, giving them 5 lives as opposed to the infinite amount easy mode has means that the player is more cautious about the mistakes they make and those who like the challenge will find the game more enjoyable.
Hard Level must have 3 lives to begin with.	This makes the level more difficult than medium mode and will make it more challenging specifically for the older age range of stakeholders. As someone apart of that age range, I think that 3 lives is enough to be challenging on players/users but is fair enough based on the difficulty of the questions.
Back button on difficulty selection screen	My stakeholder M.Miah believed this would be

allows user to return to title menu	a useful feature in case the user wanted to review the instructions once more before playing.

As evident, my program has met all of these success criteria and has met the original requirements set and the new ones set by M.Miah in the development process. My stakeholder agrees as the game changes the challenge questions depending on the difficulty chosen by my user.

Personalisation

Success criteria	Justification
Must have a form of personalisation e.g. Game must allow player to create a basic character	From my interview and observation process, it was clear that personalisation in my game would keep children interested in the game as children like to personalise characters making them more emerged in the experience.
The personalisation should have at least 5 hair variants	I think 5 hair variants gives my user plenty of options to choose from that could appeal to all genders and gives enough opportunity to show I can create a personalisation feature in my game/simulator.
The personalisation should have 3 or more hair colour tones	I think 3 hair colour tones gives my user plenty of options to choose from that could appeal to my stakeholders for basic customisation. If I have more time I may add more hair colour tones.
The personalisation should have 5 or more skin tone options	5 skin tones will allow me to generalise the spectrum of skin colour into enough different variants that my players can choose from.
The personalisation should have 3 or more body shapes	This will give my player the choice to change what shape their body is.
The personalisation should have 3 or more body colours	This is to further incentivise younger children to play as it add more imagination and personalisation options to my game.
Game must allow user to write their name and display their name in-game	This is to add more and more personalisation features and engage my stakeholders into the game more.
The game must validate the length of the players name so that it only allows a name no longer than 12 letters.	The average US first name is 6 letters long so my game will allow players 12 characters to write names so that it is short enough to fit in the screen but also long enough to allow any longer names if needed.

As outlined before, I have fully failed to meet these requirements due to fears of not completing the project within the time restraints of the projects deadline. This is because to learn how to implement this feature, it will take too long and delay my projects schedule beyond the deadline.

Music

Success criteria	Justification
Must have background music	Background Music will not only keep the player interested but may also increase their levels of concentration. My interviewee told me that it is important that a game has background music especially to keep the children wanting to play the game.

This is similar to personalisation in that the time to learn how to implement this feature would have taken me beyond the projects deadline.

Progression

Success criteria	Justification
Must have a progression system/scheme	Progression is a very crucial feature in a game as it leads the player to believe they are working towards something. I will be using money as a form of progression as my game is an abstraction of a web developers life and in the real world, wealth can show progression of sorts.
If a player gets a question correct, their "balance" should increment by £100	This is an abstraction of getting paid in the real world is like. If a real web developer does their job correctly, they get paid and so if the player completes a challenge then they too shall be paid in the form of in game currency.
If a player gets a question wrong, their "Balance" must decrement by £50	This is an abstraction of consequences in my game of what may happen to a web developer if they create an error in code for the company they work for.

As shown from my tests, my game has a functioning progression system that as my stakeholder said in the development process, was effective but could be better and thus I have met the requirement but it can be improved and I will look to outline how to improve this feature in further maintenance.

My tests show however that getting an answer correct and incorrect respond appropriately.

Book feature

Success Criteria	Justification
The book feature must be working and provide useful tips that would help the player in the	The book feature is one of the main features I am adding in the game as it is an abstraction of

challenges	researching that web developers may do to learn how to code a certain feature into their website.
The book feature must have 2 buttons that allow the player to change pages back and forth to their liking.	This is a simple design that will allow my user to easily skim over the book to find the information they are looking for demonstrating the computational method of abstraction.
Game will teach syntax of HTML and CSS	The game is an abstraction of a web developer's job with which they work with HTML and CSS. They also work with JavaScript but I believe my stakeholders will be unable to grasp the concept of JSS and thus I will not be teaching them this syntax.

As shown in my tests, the book is completely functional and the buttons do as they are expected to. My stakeholder also evidently said that she believed the book feature was very effective showing that the solution met the criteria.

Challenges

Success criteria	Justification
Game must have working multiple choice challenges	This means that the player should be able to choose and submit an answer with the game responding accordingly. This is a good choice for challenges as my stakeholders are quite young and I do not expect them to be able to code without aid.
Game must have working input fields to fill in the blank challenges	Very similar to the multiple choice; This means that the player should be able to choose and submit an answer with the game responding accordingly. This is a good choice for challenges as my stakeholders are quite young and I do not expect them to be able to code without aid.
To validate the input fields, I will allow my user to enter no more than 12 characters of the English character set	This is because there will be no need for my user to enter values or answers of over 12 characters but also will require my user to use certain character that are not alphanumeric such as "<" and ">" to answer specific questions.

As shown by my testing, I have fully met these criteria as my stakeholder agreed my solution was effective and my validation worked as it should.

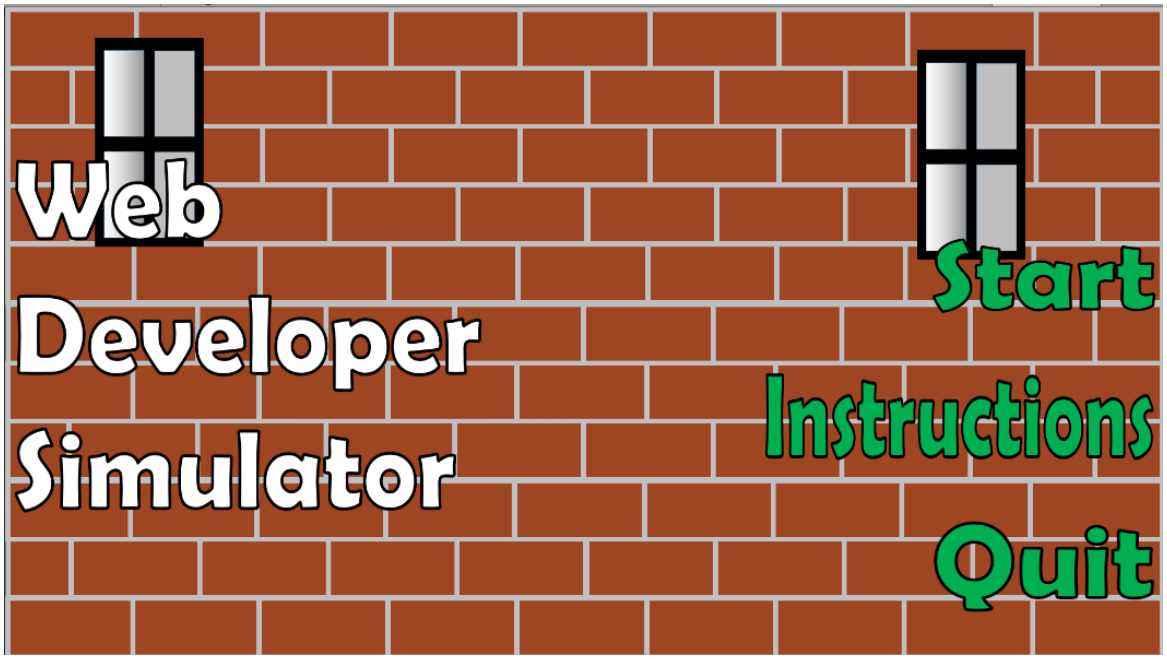
Game over and pause menu

Success criteria	Justification
Game must send player to start screen if they run out of lives	This is to stop the player progressing for making too many mistakes. By doing this for the harder levels, it makes the game more difficult and more satisfying for completing challenges without failing.
Game must allow user to pause and have a pause menu	As my stakeholders are children between 9-14 years old, there may be a need to pause the game for example to go answer the door. This will also allow the user to pause the game from timed challenges on harder difficulties.

As proven by my tests, I have met this criteria fully by having a gameover screen when the user runs out of lives. I also have a button that allows the user to pause the game and return to the title screen or go back to the game showing that I have met this criteria. I was asked by my stakeholder to add a back to game button and thus by adding it, she believed it was a good solution. This was a change from my original screen design.

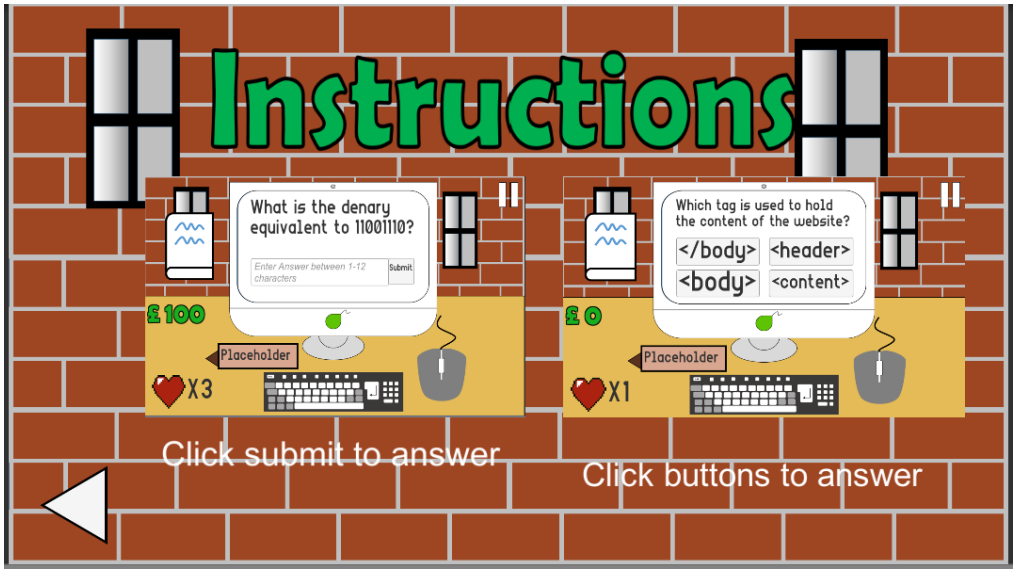
Usability features:

Main Menu:

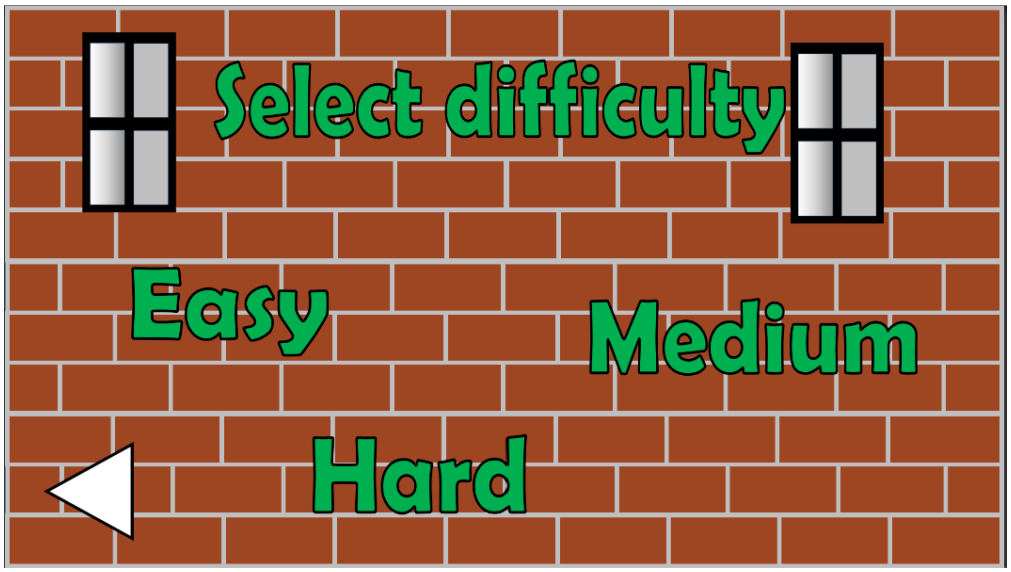


As I laid out in my success criteria, I would have allow my player to start or quit. This shows evidence that of full success in meeting this criteria. My stakeholder said it was simple and easy to traverse through for her showing that it would be effective for my target age range as my stakeholder M.Miah is between the middle of my age range.

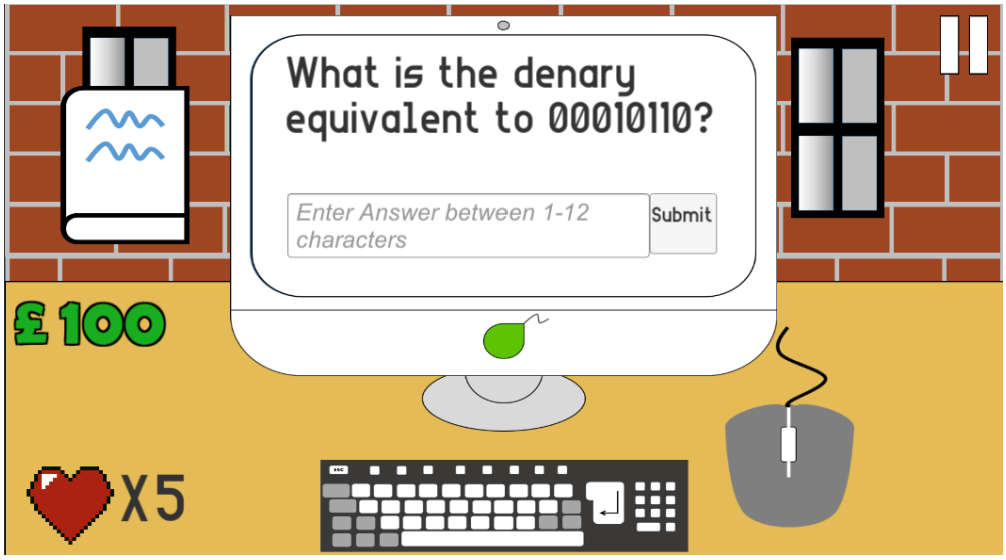
Instructions:



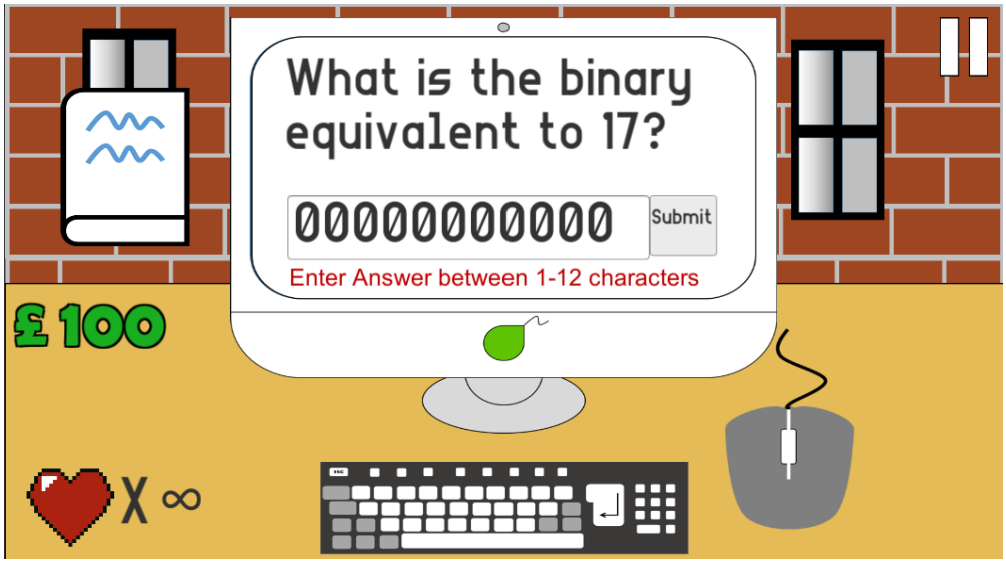
I also provided clear instructions on how to answer questions with a design that was not set out in my screen designs but new success criteria was added in during development which was to use screenshot from my game for the instructions and I have clearly met that criteria. This met my users' needs as she said the game is simple and easy to understand so the instructions did not need to go too in-depth with how to play as most of the game is simple enough to understand.



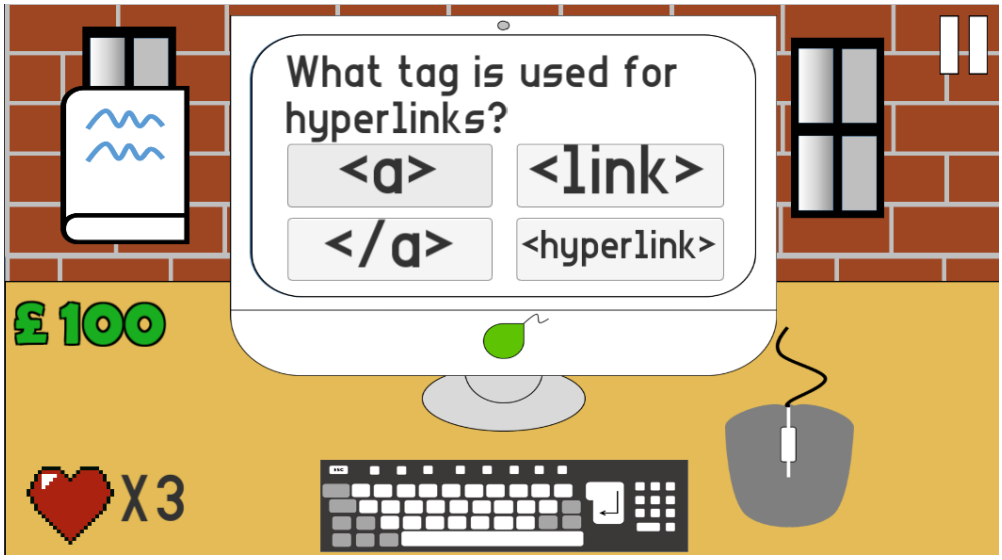
I added in the back button, and choosing a difficulty changes the type of questions as shown in my development and testing to inform evaluation. This was specifically asked for by my user and thus by meeting this, it met their needs.



This shows a working fill in the blank challenge and comes with validation shown below.

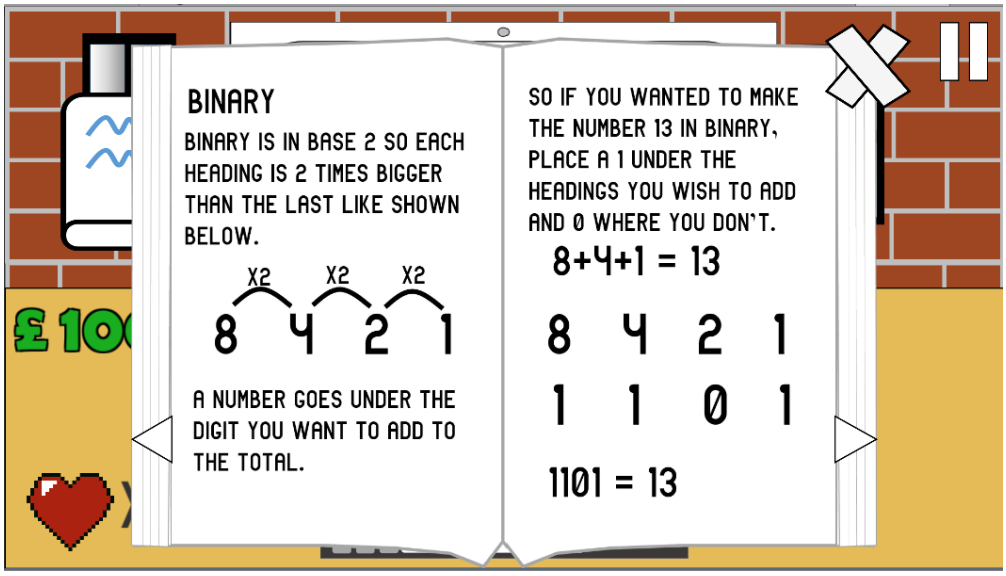


This is effective validation as it does not accept the answer without entering an answer less than 13 characters. My stakeholder said that this was an effective solution to the initial problem and fit all of her expectations



My stakeholder said she enjoyed playing the simulation as a whole because of the challenges and mix up between multiple choice questions and fill in the blank questions. She thought the feature to randomise the buttons was extremely effective as the user would not be able to pick the same button every time the same question re appeared.

My stakeholder said this fit all of her expectations as it stops the player from continuing and displays the users score before the click back to start to retry.



My stakeholder thought the book feature went beyond her expectations in how effective it worked but said that she would prefer if there were more pages to which I responded that there would be if I had more time to design more but for the purpose of displaying a demonstration of what it may look like with further maintenance, we both agreed that it was enough.

Limitations of Usability features and Maintenance:

Firstly, I was unable to implement a music feature due to lack of ability and time to learn how to implement it without delaying the schedule of the project. This was one of the requirements set out in my success criteria and is therefore a limitation of my current system as my stakeholders will currently be going through the simulation without any music or sound. With further maintenance, this can be added as DLC where the user can download background music for the game. This problem can be solved by creating an audio manager that will play background music but that will have to be static in that it must persist between scenes otherwise, in my current system, the music will restart every time a player answers the question as the scene reloads every time the player answer. Furthermore, with further maintenance, using the audio manager, I would add sound effects that plays whenever a user answers to make it more obvious that the user got the answer correct or incorrect.

I was also unable to implement personalisation, which was a part of my success criteria and usability feature, due to the same reasons. If I had more time I would have implemented a system that allows my stakeholder to enter a nickname that would display on the proposed nameplate and for my user to create an avatar in a screen before the game begins. This would, as my interview and questionnaire both showed, increase interest in my target audience and thus solve the initial problem of creating a simulator that interests a younger audience in becoming web developers and as a revision tool for teens learning both CSS and HTML.

With further maintenance, I would create a shop feature in which the user would be able to use the currency they had accumulated to buy in game cosmetic upgrades to the background, the user's PC or the customisation of the character itself. This is to make the progression scheme of the game more satisfying to earn money but also further increase the immersion of the simulation as the shop feature is a form of abstraction of reality where real web developers have the ability to purchase

goods and computers with the money they earn. If I were to carry this out, it could be done by creating a button that opens the shop and then displaying the users balance with a list of the cosmetics they could consume and a buy button to confirm the transaction. Once the transaction was complete, the item would be added to the corresponding list of items the bought item is a part of.

Online leader boards: If I had the time, I would upload the leader board onto a notepad file, after sorting the file so that it went in descending order of high scores, saved as a HTML file which I would then purchase a domain name and host the HTML file onto that domain. This would allow my stakeholders to be more competitive against one another and allow them to see their progress against the rest of the world. This is another way to make the progression scheme better as people can challenge their friend's scores to beat them and have bragging rights.

To maintain the game, more questions should be added for every difficulty to give a wider range of the type of questions asked and increase the range of topics covered by the game to further improve the skills of the user as they get more practice.

In addition to this, I would also maintain the book feature where more is added to cover a greater range of topics, possibly including more information on HTML and CSS, but also Java script so all 3 can be used hand in hand as they should be by the user once they have improved their skills. My stakeholder found the book feature to be effective but was slightly disappointed that more wasn't covered but with further maintenance, it can cover more. To meet future requirements where the book feature may become more and more filled with content, there will be a need for the contents page to allow the user to click buttons that goes straight to that page rather than my user having to traverse through pages and pages before getting to the right page. This would make the experience of learning easier for the user and I anticipate that it would be appreciated by my stakeholders if it was implemented.

Lastly, I would add further maintenance to allow the game to branch into multiple different languages such as python and C# so that younger audiences can also aspire to be software developers and not just create an interest in web developing.

I have placed useful comments throughout all of my scripts and this report to help with maintenance and outlined what many of the methods and attribute do and the information they contain. I have made an attempt to ensure all variables, methods and key constructs are appropriately named so a skilled programmer, for example, will be able to look at the code and is able to understand what type of information they hold.

Final sign-off……..13/04/19

My stakeholder agrees that after discussing limitations and the success of the last prototype, I have fulfilled enough of the success criteria for the system to be dubbed effective enough to be the final answer to the problem solved.

…….

Bibliography

1. Heathcote, P. and Heathcote, R. (n.d.). *OCR AS and A level computer science*. PG Online Limited. ISBN~ 978-1-910523-05-6
2. Rouse, G., Pitt, J. and O'Byrne, S. (2015). *OCR A level computer science*. London: Hodder Education Group. ISBN~ 978-1-471-83976-4
3. YouTube. (2019). *Brackeys*. [online] Available at: https://www.youtube.com/user/Brackeys

Project Appendix

Full Code listing:

```
30      public void HardSelected()
31      {
32          difficulty = ("hard");
33          Debug.Log(difficulty);
34      }
35      // In this method the string "hard" will be assigned difficulty
36
37      public void Startgame()
38      //begins game
39      {
40          SceneManager.LoadScene("CoreGame");
41          //Loads main game
42      }
43
44
45      public void quitter()
46      {
47          Debug.Log("Quitted");
48          Application.Quit();
49          // This method will terminate the game. This will be linked to the quit buttons OnClick()
50          //"Application.Quit();" This is the built-in function that allows the program to terminate
51      }
52
53      public void resetValues()
54      {
55          gamemechanics.money = (100);
56          gamemechanics.counter = 0;
57
58      }
59  }
60
```

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

public class gamemanager : MonoBehaviour {

    public static string difficulty;
    // This is the declaration of a variable called difficulty that will hold a string
    // The purpose of the string is to hold the difficulty my player selects when they click on the respective button
    // I have made it public so I can access the variable outside the script if I need to
    //static so it can be referenced without the need to instantiate

    public void EasySelected()
    {
        difficulty = ("easy");
        Debug.Log(difficulty);
    }
    // This is one of the methods of my class
    // In this method the string "easy" will be assigned difficulty
    // void here means the method will not return a value

    public void MediumSelected()
    {
        difficulty = ("medium");
        Debug.Log(difficulty);
    }
    // In this method the string "medium" will be assigned difficulty
```

Gamemanager script

FITBQuestions:

```
//Allows dev to edit questions in unity inspector
[System.Serializable]
public class FITBQuestions
{

    public string theQuestion;
    //will hold value of FITB question
    public string fitbAnswer;
    //holds value for corresponding answer

}
```

MCQuestions:

```
[System.Serializable]
public class MCQuestions
{

    public string theQuestion;
    public string mcAnswer;
    public string firstOption;
    public string secondOption;
    public string thirdOption;
    public string fourthOption;
    // declares MCQuestions class so every object has a question, answer and 4 possible answers.

}

```

displayFITB:

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

public class displayFITB : MonoBehaviour {

    public FITBQuestions[] easyquestions;
    public FITBQuestions[] mediumquestions;
    public FITBQuestions[] hardquestions;
    //holds list of questions and answers for FITB questions

    [SerializeField]
    private Text question;
    //links the text from the core game to the script
    [SerializeField]
    private Button submit;
    //links the submit button from the core game to the script
    [SerializeField]
    private InputField userAnswer;
    //links the answer input field from core game to script

    [SerializeField]
    private float delay = 1f;
    //Delay between reloading scenes

    private static List<FITBQuestions> unansweredQuestions;
    // creates a list of FITB objects

    private FITBQuestions displayedQuestion;
    //Will be the displayed question in-game
```

```
    public GameObject failedvalidation;
    //holds text prompting user to enter answer with less than 13 characters

    public GameObject correctimg;
    public GameObject wrongimg;
    public static GameObject signal;
    //Holds correct signal and wrong signal
    //One of the objects will be assigned to signal which will be displayed during Coroutine
    public GameObject MCchallenge;
    public GameObject FITBchallenge;
    //allows dev to control which challenge is active


    // Use this for initialization
    void Start () {

        int challengePicker = Random.Range(0, 2);
        if (challengePicker == 1)
        {
            MCchallenge.gameObject.SetActive(true);
            FITBchallenge.gameObject.SetActive(false);
        }
        else
        {
            MCchallenge.gameObject.SetActive(false);
            FITBchallenge.gameObject.SetActive(true);
        }
        //randomly picks a challenge to be active

```

```
        if (unansweredQuestions == null || unansweredQuestions.Count == 0)
        {

            if ((gamemanager.difficulty) == ("easy"))
            {
                unansweredQuestions = easyquestions.ToList<FITBQuestions>();
            }

            if ((gamemanager.difficulty) == ("medium"))
            {
                unansweredQuestions = mediumquestions.ToList<FITBQuestions>();
            }
            if ((gamemanager.difficulty) == ("hard"))
            {
                unansweredQuestions = hardquestions.ToList<FITBQuestions>();
            }
            // changes the list of questions are asked depending on their difficulty

        }
        SetDisplayedQuestion();
        failedvalidation.gameObject.SetActive(false);
        // validates the answer

    }
```

```
90      void SetDisplayedQuestion()
91      {
92          int randomQuestionIndex = Random.Range(0, unansweredQuestions.Count);
93          // generates a random question between range of 0 to the number of elements in the list
94          displayedQuestion = unansweredQuestions[randomQuestionIndex];
95          //changes value of displayed question to the randomly picked question
96
97          question.text = displayedQuestion.theQuestion;
98          //value of the text
99
100     }
101
102
103     // Update is called once per frame
104     void Update () {
105
106     }
107
```

```
103     // Update is called once per frame
104     void Update () {
105
106     }
107
108     public void submitter()
109     {
110         if (userAnswer.text.Length < 13)
111         {
112
113
114             if ((userAnswer.text).ToLower() == (displayedQuestion.fitbAnswer))
115             {
116                 Debug.Log("Correct");
117                 gamemechanics.money += 100;
118                 //increments money if they get question correct
119                 signal = correctimg;
120                 //assigns the correctimg signal to the signal
121
122             }
123             else
124             {
125                 Debug.Log("Wrong");
126                 if (gamemechanics.money > 0)
127                 {
128                     gamemechanics.money -= 50;
129                 }
130                 signal = wrongimg;
131                 //assigns the wrongimg signal to the signal
132
133                 gamemechanics.lives -= 1;
134                 //Decrements life and money if they get answer wrong
135             }
136
```

```
136                 
137                     StartCoroutine(Transition());
138                 
139                 
140                 }
141                 else
142                 {
143                     failedvalidation.gameObject.SetActive(true);
144                     //if length is greater than 12, then it will not accept answer
145                 }
146             }
147             
148             IEnumerator Transition()
149             {
150                 //This coroutine controls the delay before the game is reloaded
151                 unansweredQuestions.Remove(displayedQuestion);
152                 //removes the question from the list
153                 signal.gameObject.SetActive(true);
154                 //signals user that they are correct or incorrect
155                 yield return new WaitForSeconds(delay);
156                 
157                 SceneManager.LoadScene("CoreGame");
158                //reloads scene
159             }
160             
161             
162             
163             
164             
165         }
166
```

displayMC:

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

public class displayMC : MonoBehaviour {

    public MCQuestions[] easyquestions;
    public MCQuestions[] mediumquestions;
    public MCQuestions[] hardquestions;
    //holds list of questions, options and answers for Multiple choice questions

    [SerializeField]
    private Text question;
    //links the text from the core game to the script
    [SerializeField]
    private Text option1;
    //links the option1 button from the core game to the script
    [SerializeField]
    private Text option2;
    //links the option2 button from the core game to the script
    [SerializeField]
    private Text option3;
    //links the option3 button from the core game to the script
    [SerializeField]
    private Text option4;
    //links the option4 button from the core game to the script

    [SerializeField]
    private float delay = 1f;
    //Delay between reloading scenes
```

```
    private static List<MCQuestions> unansweredQuestions;
    // creates a list of MC objects

    private static List<Button> listOfButtons;
    // List of buttons that will be randomised

    private MCQuestions displayedQuestion;
    //Will be the displayed question in-game


    public GameObject correctimg;
    public GameObject wrongimg;
    public static GameObject signal;
    //Holds correct signal and wrong signal
    //One of the objects will be assigned to signal which will be displayed during Coroutine
    public GameObject MCchallenge;
    public GameObject FITBchallenge;
    //allows dev to control which challenge is active
```

```
55      // Use this for initialization
56      void Start () {
57
58          int challengePicker = Random.Range(0, 2);
59          if (challengePicker == 1)
60          {
61              MCchallenge.gameObject.SetActive(true);
62              FITBchallenge.gameObject.SetActive(false);
63          }
64          else
65          {
66              MCchallenge.gameObject.SetActive(false);
67              FITBchallenge.gameObject.SetActive(true);
68          }
69          //randomly picks a challenge to be active
70
71          if (unansweredQuestions == null || unansweredQuestions.Count == 0)
72          {
73
74              if ((gamemanager.difficulty) == ("easy"))
75              {
76                  unansweredQuestions = easyquestions.ToList<MCQuestions>();
77              }
78
79              if ((gamemanager.difficulty) == ("medium"))
80              {
81                  unansweredQuestions = mediumquestions.ToList<MCQuestions>();
82              }
83              if ((gamemanager.difficulty) == ("hard"))
84              {
85                  unansweredQuestions = hardquestions.ToList<MCQuestions>();
86              }
87              // changes the list of questions are asked depending on their difficulty
88
```

```
88              }
89              SetDisplayedQuestion();
90
91
92          }
93
94          void SetDisplayedQuestion()
95          {
96              int randomQuestionIndex = Random.Range(0, unansweredQuestions.Count);
97              // generates a random question between range of 0 to the number of elements in the list
98              displayedQuestion = unansweredQuestions[randomQuestionIndex];
99              //changes value of displayed question to the randomly picked question
100
101             question.text = displayedQuestion.theQuestion;
102             //value of the text
103
104             List<string> possibleAnswers = new List<string>();
105             possibleAnswers.Add(displayedQuestion.firstOption);
106             possibleAnswers.Add(displayedQuestion.secondOption);
107             possibleAnswers.Add(displayedQuestion.thirdOption);
108             possibleAnswers.Add(displayedQuestion.fourthOption);
109             //Adds all the options for that question into a list called possible answers
110
111             while (possibleAnswers != null)
112                 //This while loop randomly assigns a question to the text value of each button
113             {
114                 int randomA = Random.Range(0, possibleAnswers.Count);
115                 option1.text = possibleAnswers[randomA];
116                 possibleAnswers.RemoveAt(randomA);
117
118                 int randomB = Random.Range(0, possibleAnswers.Count);
119                 option2.text = possibleAnswers[randomB];
120                 possibleAnswers.RemoveAt(randomB);
```

```
            option2.text = possibleAnswers[randomB];
            possibleAnswers.RemoveAt(randomB);

            int randomC = Random.Range(0, possibleAnswers.Count);
            option3.text = possibleAnswers[randomC];
            possibleAnswers.RemoveAt(randomC);

            int randomD = Random.Range(0, possibleAnswers.Count);
            option4.text = possibleAnswers[randomD];
            possibleAnswers.RemoveAt(randomD);
        }


    }

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

    }

    public void buttonClicked1()
    {
        if (displayedQuestion.mcAnswer == option1.text)
        {
            Debug.Log("Correct");
            gamemechanics.money += 100;
            //increments money if they get question correct
            signal = correctimg;
            //assigns the correctimg signal to the signal

        }
```

```
        else
        {
            Debug.Log("Wrong");
            if (gamemechanics.money > 0)
            {
                gamemechanics.money -= 50;
            }
            signal = wrongimg;
            //assigns the wrongimg signal to the signal

            gamemechanics.lives -= 1;
            //Decrements life and money if they get answer wrong
        }
        StartCoroutine(Transition());
    }

    public void buttonClicked2()
    {
        if (displayedQuestion.mcAnswer == option2.text)
        {
            Debug.Log("Correct");
            gamemechanics.money += 100;
            //increments money if they get question correct
            signal = correctimg;
            //assigns the correctimg signal to the signal

        }
```

```
179         else
180         {
181             Debug.Log("Wrong");
182             if (gamemechanics.money > 0)
183             {
184                 gamemechanics.money -= 50;
185             }
186             signal = wrongimg;
187             //assigns the wrongimg signal to the signal
188 
189             gamemechanics.lives -= 1;
190             //Decrements life and money if they get answer wrong
191         }
192         StartCoroutine(Transition());
193     }
194 
195     public void buttonClicked3()
196     {
197         if (displayedQuestion.mcAnswer == option3.text)
198         {
199             Debug.Log("Correct");
200             gamemechanics.money += 100;
201             //increments money if they get question correct
202             signal = correctimg;
203             //assigns the correctimg signal to the signal
204 
205         }
206 
```

```
207         else
208         {
209             Debug.Log("Wrong");
210             if (gamemechanics.money > 0)
211             {
212                 gamemechanics.money -= 50;
213             }
214             signal = wrongimg;
215             //assigns the wrongimg signal to the signal
216 
217             gamemechanics.lives -= 1;
218             //Decrements life and money if they get answer wrong
219         }
220         StartCoroutine(Transition());
221     }
222 
223     public void buttonClicked4()
224     {
225         if (displayedQuestion.mcAnswer == option4.text)
226         {
227             Debug.Log("Correct");
228             gamemechanics.money += 100;
229             //increments money if they get question correct
230             signal = correctimg;
231             //assigns the correctimg signal to the signal
232 
233         }
```

```
233             }
234
235             else
236             {
237                 Debug.Log("Wrong");
238                 if (gamemechanics.money > 0)
239                 {
240                     gamemechanics.money -= 50;
241                 }
242                 signal = wrongimg;
243                 //assigns the wrongimg signal to the signal
244
245                 gamemechanics.lives -= 1;
246                 //Decrements life and money if they get answer wrong
247             }
248             StartCoroutine(Transition());
249         }
250
251
252         IEnumerator Transition()
253         {
254             //This coroutine controls the delay before the game is reloaded
255             unansweredQuestions.Remove(displayedQuestion);
256             //removes the question from the list
257             signal.gameObject.SetActive(true);
258             //signals user that they are correct or incorrect
259             yield return new WaitForSeconds(delay);
260
261             SceneManager.LoadScene("CoreGame");
262             //reloads scene
263         }
264     }
265
```

Gamemechanics:

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

public class gamemechanics : MonoBehaviour {

    [SerializeField]
    public static int lives;
    //holds number of lives

    [SerializeField]
    public static int money = 100;
    //holds accumulated money

    [SerializeField]
    private Text displayLives;
    //the lives displayed

    [SerializeField]
    private Text moneyCounter;
    //money displayed

    public static int counter = 0;

    [SerializeField]
    private Text gameoverscore;
    //money displayed

    public GameObject gameover;
    //gameover object displayed

```

```
34
35      // Use this for initialization
36      void Start () {
37          if (counter == 0)
38          {
39
40
41              if ((gamemanager.difficulty) == ("easy"))
42              {
43                  lives = -1;
44              }
45              if ((gamemanager.difficulty) == ("medium"))
46              {
47                  lives = 5;
48              }
49              if ((gamemanager.difficulty) == ("hard"))
50              {
51                  lives = 3;
52              }
53          }
54          counter++;
55          //increases counter so that lives is not changed every time a question is asked or scene reloads
56      }
57
58      // Update is called once per frame
59      void Update () {
60          moneyCounter.text = (money.ToString());
61          gameoverscore.text = (money.ToString());
62
```

```
63
64              if (gamemanager.difficulty == "easy")
65              {
66                  displayLives.text = ("∞");
67              }
68              else
69              {
70                  displayLives.text = (lives.ToString());
71              }
72
73              if (lives == 0)
74              {
75                  gameover.gameObject.SetActive(true);
76              }
77              //displays gameover object
78          }
79
80          public void returntotitle()
81          {
82              SceneManager.LoadScene("Mainmenu");
83          }
84      }
85
```

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

public class Book : MonoBehaviour
{

    public Image[] pages;
    public int pagenumber = 0;
    public GameObject TheBook;

    public void nextpage()
    {
        if (pagenumber < pages.GetLength(0) - 1)
        {
            pagenumber += 1;

        }
        //Changes the page to the next page
    }

    public void prevpage()
    {
        if (pagenumber > 0)
        {
            pagenumber -= 1;


        }
        //Changes the page to the previous page
    }

```

```
34
35      public void resetter()
36      {
37          pagenumber = 0;
38          TheBook.gameObject.SetActive(true);
39
40
41      }
42
43
44      void Update()
45      {
46
47          for (int i = 0; i < pages.GetLength(0); i++)
48          {
49              pages[i].gameObject.SetActive(false);
50          }
51
52          pages[pagenumber].gameObject.SetActive(true);
53          //Loop deactivates all pages and then reactivates the page the user is on
54      }
55
56  }
57
```